Guia do desenvolvedor

AWS IoT Core

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS IoT Core: Guia do desenvolvedor

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

As marcas comerciais e imagens de marcas da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestigie a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

O que é AWS IoT? 1
Como seus dispositivos e aplicativos acessam AWS IoT 2
O que AWS IoT pode fazer 3
IoT na indústria 3
IoT na automação residencial 3
Como AWS IoT funciona 4
O universo IoT 4
AWS IoT visão geral dos serviços 7
AWS IoT Core serviços 12
Saiba mais sobre AWS IoT 16
Recursos de treinamento para AWS IoT 16
AWS IoT recursos e guias 17
AWS IoT nas mídias sociais 17
AWS serviços usados pelo mecanismo de AWS IoT Core regras 18
Protocolos de comunicação suportados por AWS IoT Core 19
Novidades do console AWS IoT 20
Legenda 23
Trabalhando com AWS SDKs 24
Tutoriais de conceitos básicos 26
Conecte seu primeiro dispositivo a AWS IoT Core 26
Configurar Conta da AWS 28
Inscreva-se para um Conta da AWS 28
Criar um usuário com acesso administrativo 29
Abra o AWS IoT console 30
Tutorial interativo 31
Conectar dispositivos de IoT 32
Salvar o estado de um dispositivo offline 33
Rotear dados do dispositivo para serviços 34
Tutorial de conexão rápida 35
Etapa 1. Inicie o tutorial 36
Etapa 2. Criar um objeto 37
Etapa 3. Baixe os arquivos no seu dispositivo 40
Etapa 4: Execute a amostra 43
Etapa 5. Explore mais 46

Teste de conectividade 47
Tutorial avançado de conexão 53
Qual opção de dispositivo é melhor para você? 54
Crie AWS IoT recursos 55
Configurar o dispositivo 59
Exibir mensagens MQTT com o cliente AWS IoT MQTT 99
Visualizar mensagens MQTT no cliente MQTT 99
Publicar mensagens MQTT do cliente MQTT 102
Testar assinaturas compartilhadas no cliente MQTT 104
AWS IoT tutoriais 106
Criando demonstrações com o AWS IoT Device Client 106
Pré-requisitos para criar demonstrações com o Device Client AWS IoT 107
Preparar-se para usar o IoT Device Client 110
Instalar e configurar o IoT Device Client 124
Comunicar-se com o cliente do dispositivo usando MQTT 137
Executar trabalhos de IoT com o Device Client 157
Limpeza 172
Construindo soluções com o AWS IoT dispositivo SDKs 182
Comece a criar soluções com o AWS IoT dispositivo SDKs 182
Conectando um dispositivo AWS IoT Core usando o AWS IoT Device SDK 183
Criação de AWS IoT regras para rotear dados do dispositivo para outros serviços 208
Reter o estado do dispositivo enquanto o dispositivo está off-line com as sombras do dispositivo 252
Criação de um autorizador personalizado para AWS IoT Core 283
Monitoramento da umidade do solo com AWS IoT Raspberry Pi 302
Conecte-se a AWS IoT Core 316
AWS IoT Core - endpoints do ambiente de gerenciamento 316
AWS IoT endpoints do dispositivo 318
AWS IoT Core para dispositivos e gateways de LoRa WAN 319
Conecte-se aos endpoints AWS IoT Core de serviço 320
AWS CLI para AWS IoT Core 321
AWS SDKs 322
AWS Mobilidade SDKs 327
REST APIs dos AWS IoT Core serviços 328
Conecte dispositivos a AWS IoT 329
AWS IoT dados do dispositivo e endpoints de serviço 330

AWS IoT Dispositivo SDKs ... 332
Protocolos de comunicação do dispositivo ... 335
Tópicos do MQTT ... 385
Configurações do domínio ... 413
Conecte-se aos AWS IoT endpoints FIPS ... 442
AWS IoT Core - endpoints do ambiente de gerenciamento ... 442
Endpoints do plano de dados do AWS IoT Core ... 443
AWS IoT Core: endpoints do provedor de credenciais ... 443
Endpoints de dados de trabalhos do AWS IoT Device Management ... 444
Endpoints do Fleet Hub do AWS IoT Device Management ... 444
Endpoints de encapsulamento seguro do AWS IoT Device Management ... 444
AWS IoT Device Management - Endpoints de integrações gerenciadas ... 445
Gerenciar dispositivos ... 446
Registro ... 447
Criar um objeto ... 447
Listar objetos ... 448
Descreva as objetos ... 450
Atualizar um objeto ... 451
Excluir um objeto ... 451
Anexar uma entidade principal a um objeto ... 452
Listar coisas associadas a uma entidade principal ... 453
Listar as entidades principais associadas a uma coisa ... 453
Listar as coisas associadas a uma entidade principal V2 ... 454
Listar as entidades principais associadas a uma coisa V2 ... 455
Desanexar uma entidade principal de um objeto ... 455
Tipos de objeto ... 456
Criar um tipo de objeto ... 457
Listar tipos de objeto ... 457
Descrever um tipo de objeto ... 458
Associar um tipo de objeto a um objeto ... 458
Atualizar um tipo de coisa ... 459
Reprovar um tipo de objeto ... 459
Excluir um tipo de objeto ... 460
Grupos de objetos estáticas ... 461
Criar um grupo de objetos estáticas ... 463
Descrever um grupo de objetos ... 464

Adicionar um objeto a um grupo de objetos estáticas ... 465
Remover um objeto de um grupo de objetos estáticas ... 465
Listar objetos em um grupo de objetos ... 466
Listar grupos de objetos ... 466
Listar grupos para um objeto ... 468
Atualizar um grupo de objetos estáticas ... 469
Excluir um grupo de objetos ... 470
Anexar uma política a um grupo de objetos estáticas ... 471
Desanexar uma política de um grupo de objetos estáticas ... 471
Listar as políticas anexadas a um grupo de objetos estáticas ... 472
Listar os grupos para uma política ... 472
Obter políticas efetivas para um objeto ... 473
Testar a autorização de ações MQTT ... 474
Grupos de objetos dinâmicos ... 475
Casos de uso de grupos objetos dinâmicos ... 476
Criar grupo de objetos dinâmicas ... 478
Descrever um grupo de objetos dinâmicas ... 479
Atualizar um grupo de objetos dinâmicas ... 480
Excluir um grupo de objetos dinâmicas ... 481
Limitações do grupo de objetos estático e dinâmico ... 481
Limitações do grupo de objetos dinâmico ... 481
Associar uma coisa à conexão ... 484
Casos de uso ... 484
Como associar uma coisa a uma conexão ... 485
Adicionar atributos de propagação ... 488
Console de gerenciamento da AWS ... 489
AWS CLI ... 490
Marcar recursos ... 492
Conceitos Básicos de Tags ... 492
Restrições e limitações de tags ... 493
Marcar com políticas do IAM ... 494
Grupos de cobrança ... 496
Visualizar alocação de custos e dados de uso ... 497
Segurança ... 500
Segurança em AWS IoT ... 501
Autenticação ... 502

Visão geral do certificado X.509 ... 502
Autenticação do servidor ... 502
Autenticação de cliente ... 506
Autenticação e autorização personalizadas ... 547
Autorização ... 576
AWS treinamento e certificação ... 580
AWS IoT Core políticas ... 580
Autorizando chamadas diretas para AWS serviços usando o provedor de AWS IoT Core credenciais ... 662
Acesso entre contas com o IAM ... 668
Proteção de dados ... 671
Criptografia de dados em AWS IoT ... 672
Segurança de transporte em AWS IoT Core ... 672
Criptografia de dados ... 678
Gerenciamento de identidade e acesso ... 694
Público ... 695
Como autenticar com identidades do IAM ... 695
Gerenciar o acesso usando políticas ... 696
Como AWS IoT funciona com o IAM ... 698
Exemplos de políticas baseadas em identidade ... 730
AWS políticas gerenciadas ... 735
Solução de problemas ... 750
Registro em log e monitoramento ... 752
Ferramentas de monitoramento ... 753
Validação de conformidade ... 754
Resiliência ... 755
Usando AWS IoT Core com VPC endpoints ... 756
Criação de endpoints VPC para o plano de controle AWS IoT Core ... 756
Criação de endpoints VPC para plano de dados AWS IoT Core ... 757
Criação de endpoints VPC para provedor de credenciais AWS IoT Core ... 758
Criar um endpoint de interface da Amazon VPC ... 759
Configurar uma zona hospedada privada ... 761
Controlando o acesso a AWS IoT Core mais de VPC endpoints ... 763
Limitações ... 764
Escalando endpoints de VPC com AWS IoT Core ... 765
Usar domínios personalizados com endpoints da VPC ... 766

Disponibilidade de VPC endpoints para AWS IoT Core 766
Usando AWS IoT Device Management com VPC endpoints 766
Segurança da infraestrutura 772
Monitoramento de segurança 772
Práticas recomendadas de segurança 773
Protegendo conexões MQTT em AWS IoT 773
Manter o relógio do dispositivo sincronizado 776
Validar o certificado do servidor 777
Usar uma identidade única por dispositivo 777
Use um segundo Região da AWS como backup 778
Usar provisionamento just-in-time 778
Permissões para executar testes AWS IoT do Device Advisor 778
Prevenção do problema do substituto confuso entre serviços para o Device Advisor 780
AWS treinamento e certificação 781
Monitorar AWS IoT 782
Configurar o AWS IoT registro 783
Configurando o login da V2 AWS IoT 784
Configurar a função e a política de registro em log 784
Configurar o login no AWS IoT (console) 787
Configurar login em nível de conta e evento (AWS IoT CLI) 789
Configurar substituições específicas de recursos em (CLI) AWS IoT 792
Níveis de logs 794
Monitore AWS IoT alarmes e métricas usando a Amazon CloudWatch 795
Usando AWS IoT métricas 796
Crie CloudWatch alarmes 796
Métricas e dimensões 801
Monitore AWS IoT usando CloudWatch registros 826
Visualizando AWS IoT registros no CloudWatch console 826
CloudWatch Registra entradas AWS IoT de registro 827
Faça upload de registros do lado do dispositivo para a Amazon CloudWatch 867
Como funciona 867
Carregar registros do lado do dispositivo usando regras AWS IoT 869
Registrar chamadas de AWS IoT API 879
AWS IoT informações em CloudTrail 879
Entendendo as entradas do arquivo de AWS IoT log 880
Regras 883

Concessão de acesso à ... 884
Revogar o acesso ao mecanismo de regras ... 887
Transmitir permissões de função ... 887
Criar uma regra ... 888
Criar uma regra (console) ... 890
Criar uma regra (CLI) ... 891
Gerenciar uma regra ... 895
Marcar uma regra ... 895
Visualizar uma regra ... 897
Como excluir uma regra ... 897
AWS IoT ações de regras ... 897
Apache Kafka ... 901
CloudWatch alarmes ... 914
CloudWatch Registros ... 915
CloudWatch métricas ... 918
DynamoDB ... 920
Dínamo DBv2 ... 923
Elasticsearch ... 926
HTTP ... 928
AWS IoT Events ... 943
AWS IoT SiteWise ... 945
Firehose ... 951
Kinesis Data Streams ... 953
Lambda ... 956
Local ... 959
OpenSearch ... 963
Nova publicação ... 966
S3 ... 969
IoT do Salesforce ... 971
SNS ... 972
SQS ... 975
Step Functions ... 977
Timestream ... 979
Solucionar problemas de uma regra ... 987
Acessar recursos entre contas ... 987
Pré-requisitos ... 988

Configuração entre contas para o Amazon SQS 988
Configuração entre contas para o Amazon SNS 990
Configuração entre contas para o Amazon S3 991
Configuração de várias contas para AWS Lambda 994
Tratamento de erros (ação de erro) 996
Formato da mensagem de ação de erro 997
Exemplo de ação de erro 998
Ingestão Básica 999
Usar a Ingestão básica 1000
AWS IoT Referência SQL 1001
Cláusula SELECT 1003
Cláusula FROM 1005
Cláusula SET 1006
Cláusula WHERE 1010
Tipos de dados 1010
Operadores 1016
Funções 1027
Literais 1103
Declarações de caso 1104
Extensões JSON 1105
Modelos de substituição 1107
Consultas de objeto aninhado 1110
Cargas binárias 1111
Versões do SQL 1118
Shadows 1120
Usar shadows 1120
Optar por usar sombras nomeadas ou sem nome 1121
Acessar sombras 1121
Usar sombras em dispositivos, aplicativos e outros serviços na nuvem 1122
Ordem das mensagens 1123
Reduzir mensagens de shadow 1125
Usar sombras em dispositivos 1126
Inicializando o dispositivo na primeira conexão com AWS IoT 1127
Processando mensagens enquanto o dispositivo está conectado ao AWS IoT 1129
Processando mensagens quando o dispositivo se reconecta ao AWS IoT 1130
Usar sombras em aplicativos e serviços 1130

Inicializando o aplicativo ou serviço em conexão com AWS IoT 1132
O estado de processamento muda enquanto o aplicativo ou serviço está conectado ao AWS IoT 1132
Detectar se um dispositivo está conectado 1132
Simulando comunicações de serviço Sombra do Dispositivo 1134
Configurar a simulação 1135
Inicializar o dispositivo 1135
Enviar uma atualização do aplicativo 1139
Responder à atualização no dispositivo 1142
Observar a atualização no aplicativo 1147
Além da simulação 1148
Interagir com sombras 1148
Suporte ao protocolo 1149
Solicitar e declarar o estado 1150
Atualizar uma shadow 1150
Recuperar um documento de shadow 1155
Excluir dados de sombra 1155
API REST da Sombra do Dispositivo 1159
GetThingShadow 1160
UpdateThingShadow 1161
DeleteThingShadow 1162
ListNamedShadowsForThing 1163
Tópicos MQTT da Sombra do Dispositivo 1165
/get 1166
/get/accepted 1167
/get/rejected 1168
/update 1169
/update/delta 1170
/update/accepted 1171
/update/documents 1172
/update/rejected 1173
/delete 1174
/delete/accepted 1175
/delete/rejected 1176
Documentos do serviço Sombra do Dispositivo 1177
Exemplos de documentos de sombra 1177

Propriedades do documento ... 1184
Estado delta ... 1185
Versionamento de documentos de sombra ... 1187
Tokens de cliente em documentos de sombra ... 1188
Propriedades vazias do documento de sombra ... 1188
Valores de matriz em documentos de sombra ... 1189
Mensagens de erro da Sombra do Dispositivo ... 1190
Catálogo de pacotes de software ... 1192
Preparação para uso do Catálogo de pacotes de software ... 1193
Ciclo de vida da versão do pacote ... 1193
Convenções de nomenclatura de versões de pacotes ... 1195
Versão padrão ... 1195
Atributos de versão ... 1195
Lista de materiais de software ... 1196
Habilitando a indexação de AWS IoT frotas ... 1200
Sombra nomeada reservada ... 1200
Exclusão de um pacote de software ... 1202
Preparação de segurança ... 1202
Autenticação baseada em recurso ... 1203
AWS IoT Job rights para implantar versões de pacotes ... 1204
AWS IoT Job rights para atualizar a sombra nomeada reservada ... 1206
AWS IoT Permissões de trabalho para baixar do Amazon S3 ... 1208
Permissões para atualizar a lista de materiais de software para uma versão do pacote ... 1208
Preparação da indexação de frota ... 1211
Definição da `$package` sombra como fonte de dados ... 1212
Métricas serão exibidos no console ... 1212
Padrões de consulta ... 1213
Coleta da distribuição da versão do pacote por meio de `getBucketsAggregation` ... 1216
Preparando AWS IoT trabalhos ... 1216
Parâmetros de substituição para trabalhos AWS IoT ... 1217
Preparação do documento de trabalho e versão do pacote para implantação ... 1221
Nomeação dos pacotes e das versões durante a implantação ... 1225
Segmentação de empregos por meio de AWS IoT grupos dinâmicos ... 1225
Versões reservadas nomeadas de sombra e pacote ... 1226
Como desinstalar um pacote de software ... 1227
Introdução ... 1227

Criação de um pacote e uma versão 1228
Como implantar uma versão do pacote 1231
Como associar uma versão de pacote 1233
Tarefas 1235
Acessando AWS IoT empregos 1235
AWS IoT Empregos, regiões e endpoints 1235
O que é uma operação remota? 1236
Benefícios de usar o AWS IoT Device Management Jobs para operações remotas 1236
O que é AWS IoT Jobs? 1239
Principais conceitos sobre trabalhos 1240
Trabalhos e estados de execução de trabalhos 1244
Gerenciar trabalhos 1249
Assinatura de código para trabalhos 1250
Documento de trabalho 1250
Pré-assinado URLs 1250
URL pré-assinada para upload de arquivo 1253
URL pré-assinada usando o versionamento do Amazon S3 1254
Crie e gerencie trabalhos usando o console 1256
Crie e gerencie trabalhos usando a CLI 1259
Modelos de trabalho 1271
Modelos personalizados e AWS gerenciados 1271
Use modelos AWS gerenciados 1272
Crie modelos de trabalho personalizados 1292
Configurações de trabalho 1301
Como as configurações de trabalho funcionam 1301
Especificar configurações adicionais 1317
Dispositivos e trabalhos 1327
Programar dispositivos para funcionarem com trabalhos 1329
Fluxo de trabalho do dispositivo 1330
Fluxo de trabalho 1332
Notificações de trabalhos 1336
AWS IoT operações de API de empregos 1345
Tipos de dados e API de controle e gerenciamento de trabalhos 1348
Tipos de dados e operações da API MQTT e HTTPS do dispositivo de trabalhos 1367
Como proteger usuários e dispositivos para trabalhos 1383
Tipo de política necessária para AWS IoT empregos 1383

Como autorizar usuários e serviços em nuvem de trabalhos 1384
Autorizar dispositivos a usar trabalhos 1397
AWS IoT Limites de trabalhos 1402
Limites de execuções de trabalhos 1402
Limites de trabalho ativos e simultâneos 1404
Comandos 1408
Início rápido 1409
Conceitos e status de comandos 1410
Conceitos de chaves de comandos 1410
Estados de comando 1415
Status de execução do comando 1415
Fluxo de trabalho de comandos 1419
Criar e gerenciar comandos 1420
Escolher destinos e assinar tópicos 1420
Iniciar e monitorar execuções de comando 1422
(Opcional) Habilitar notificações para eventos de comandos 1423
Criar e gerenciar comandos 1424
Criar um recurso de comando 1425
Recuperar informações sobre um comando 1430
Liste os comandos em seu Conta da AWS 1432
Atualizar um recurso de comando 1434
Suspender ou restaurar um recurso de comando 1436
Excluir um recurso de comando 1437
Iniciar e monitorar execuções de comando 1439
Iniciar uma execução de comando 1439
Atualizar o resultado de uma execução de comando 1446
Recuperar uma execução de comando 1452
Visualizar atualizações de comandos usando o cliente de teste do MQTT 1456
Liste as execuções de comandos em seu Conta da AWS 1457
Excluir uma execução de comando 1460
Suspender um recurso de comando 1461
Considerações importantes 1462
Suspender um recurso de comando (console) 1462
Suspender um recurso de comando (CLI) 1462
Conferir a hora e o status da suspensão 1463
Restaurar um recurso de comando 1463

Criação Segura de Túneis 1465
O que é encapsulamento seguro? 1465
Conceitos de encapsulamento seguro 1466
Como funciona o encapsulamento seguro 1467
Ciclo de vida do túnel seguro 1468
Tutoriais de encapsulamento seguro 1469
Tutoriais nesta seção 1469
Abra um túnel e inicie a sessão SSH no dispositivo remoto 1470
Abra um túnel para dispositivo remoto e use SSH baseado em navegador 1488
Proxy local 1493
Como usar o proxy local 1493
Configurar o proxy local para dispositivos que usam proxy da web 1500
Multiplexação e conexões TCP simultâneas 1508
Multiplexação de vários fluxos de dados 1509
Como usar conexões TCP simultâneas 1512
Como configurar um dispositivo remoto e usar o atendente de IoT 1515
Snippet de atendente de IoT 1516
Como controlar o acesso aos túneis 1518
Pré-requisitos de acesso ao túnel 1518
Políticas de acesso ao túnel 1518
Como resolver problemas de conectividade de encapsulamento seguro 1525
Erro de token de acesso do cliente inválido 1526
Erro de incompatibilidade do token do cliente 1527
Problemas de conectividade com o dispositivo remoto 1528
Provisionamento de dispositivos 1531
Dispositivos de provisionamento em AWS IoT 1532
Provisionamento de frota APIs 1533
Provisionar dispositivos que não têm certificados de dispositivo usando o provisionamento de frotas 1534
Provisionamento por reivindicação 1535
Provisionamento por usuário confiável 1538
Usando ganchos de pré-provisionamento com a CLI AWS 1540
Provisionamento de dispositivos com certificados de dispositivo 1544
Provisionamento de uma única coisa 1544
Just-in-time provisionamento 1545
Registro em massa 1552

Modelos de provisionamento 1552
Seção de parâmetros 1553
Seção de recursos 1554
Exemplo de modelo para registro em massa 1560
Exemplo de modelo para just-in-time provisionamento (JITP) 1561
Provisionamento de frotas 1563
Ganchos de pré-provisionamento 1567
Entrada de hook de pré-provisão 1567
Valor de retorno do hook de pré-provisão 1568
Exemplo de hook de pré-provisionamento do Lambda 1569
Assinatura de certificado autogerenciada usando provedor AWS IoT Core de certificados 1571
Como a assinatura autogerenciada de certificados funciona no provisionamento de frotas 1572
Entrada da função do Lambda do provedor de certificados 1574
Valor de retorno da função do Lambda do provedor de certificados 1575
Exemplo de função do Lambda 1575
Assinatura de certificado autogerenciada para provisionamento de frota 1577
AWS CLI comandos para provedor de certificados 1578
Criação de políticas e perfis do IAM para um usuário instalando um dispositivo 1581
Criação de uma política do IAM para o usuário que instalará um dispositivo 1582
Criação de um perfil do IAM para o usuário que instalará um dispositivo 1583
Atualização de uma política existente para autorizar um novo modelo 1584
API MQTT de provisionamento de dispositivos 1585
CreateCertificateFromCsr 1586
CreateKeysAndCertificate 1588
RegisterThing 1590
Indexação de frotas 1594
Gerenciamento de atualizações de índice 1594
Consultar o status de conectividade de um dispositivo específico 1594
Pesquisa em várias fontes dos dados 1594
Consulta de dados agregados 1595
Monitoramento de dados agregados e criação de alarmes usando as métricas da frota 1595
Gerenciamento da indexação de frotas 1595
Indexação de objetos 1595
Indexação de grupo de objetos 1597
Campos gerenciados 1597
Campos personalizados 1599

Gerenciar a indexação de objetos ... 1600
Gerenciamento da indexação de grupos de objetos ... 1616
Status de conectividade do dispositivo ... 1618
Como funciona ... 1619
Recursos ... 1619
Benefícios ... 1619
Pré-requisitos ... 1620
Exemplos ... 1620
Consulta de dados agregados ... 1622
GetStatistics ... 1622
GetCardinality ... 1625
GetPercentiles ... 1626
GetBucketsAggregation ... 1629
Autorização ... 1630
Sintaxe de consulta ... 1630
Recursos compatíveis ... 1630
Atributos não compatíveis ... 1631
Observações ... 1631
Exemplo de consultas de objetos ... 1632
Exemplo de consultas de grupos de objetos ... 1637
Indexação de dados de localização ... 1638
Formatos de dados compatíveis ... 1639
Como indexar dados de localização ... 1640
Atualizar a configuração da indexação de objeto ... 1641
Exemplo de consultas geográficas ... 1643
Tutorial de inicialização ... 1644
Métricas da frota ... 1649
Tutorial de inicialização ... 1650
Gerenciar métricas de frota ... 1657
Entrega de arquivos baseada em MQTT ... 1664
O que é um stream? ... 1664
Gerenciar um fluxo ... 1665
Conceda permissões aos seus dispositivos ... 1666
Conecte seus dispositivos a AWS IoT ... 1667
TagResource Uso ... 1668
Use a entrega de arquivos AWS IoT baseada em MQTT em dispositivos ... 1669

Use DescribeStream para obter dados de stream ... 1669
Obter blocos de dados de um arquivo de stream ... 1671
Tratamento de erros da entrega de arquivos AWS IoT baseada em MQTT ... 1677
Um exemplo de caso de uso no FreeRTOS OTA ... 1679
Device Advisor ... 1680
Configurar ... 1682
Criar um objeto do IoT ... 1682
Criar um perfil do IAM a ser usado como perfil de dispositivo ... 1682
Crie uma política gerenciada personalizada para que um usuário do IAM use o Device Advisor ... 1685
Criar um usuário do IAM para usar o Device Advisor ... 1686
Configurar o dispositivo ... 1689
Conceitos básicos do Device Advisor no console ... 1690
Fluxo de trabalho do Device Advisor ... 1700
Pré-requisitos ... 1700
Criar uma definição de conjunto de teste ... 1700
Obtenha uma definição de conjunto de teste ... 1703
Obtenha um endpoint de teste ... 1704
Inicie a execução de um conjunto de testes ... 1704
Obtenha a execução de um conjunto de testes ... 1705
Interromper a execução de um conjunto de testes ... 1705
Obtenha um relatório de qualificação para uma execução bem-sucedida do conjunto de testes de qualificação ... 1706
Fluxo de trabalho detalhado do console do Device Advisor ... 1706
Pré-requisitos ... 1707
Criar uma definição de conjunto de teste ... 1707
Inicie a execução de um conjunto de testes ... 1714
Interromper a execução de um conjunto de testes (opcional) ... 1716
Exibir detalhes e logs da execução do conjunto de testes ... 1717
Baixe um relatório AWS IoT de qualificação ... 1719
Fluxo de trabalho do console de testes de longa duração ... 1719
Endpoints da VPC do Device Advisor (AWS PrivateLink) ... 1728
Considerações sobre AWS IoT Core Device Advisor VPC endpoints ... 1728
Crie uma interface VPC endpoint para AWS IoT Core Device Advisor ... 1729
Controle do acesso a AWS IoT Core Device Advisor mais de VPC endpoints ... 1730
Casos de teste do Device Advisor ... 1731

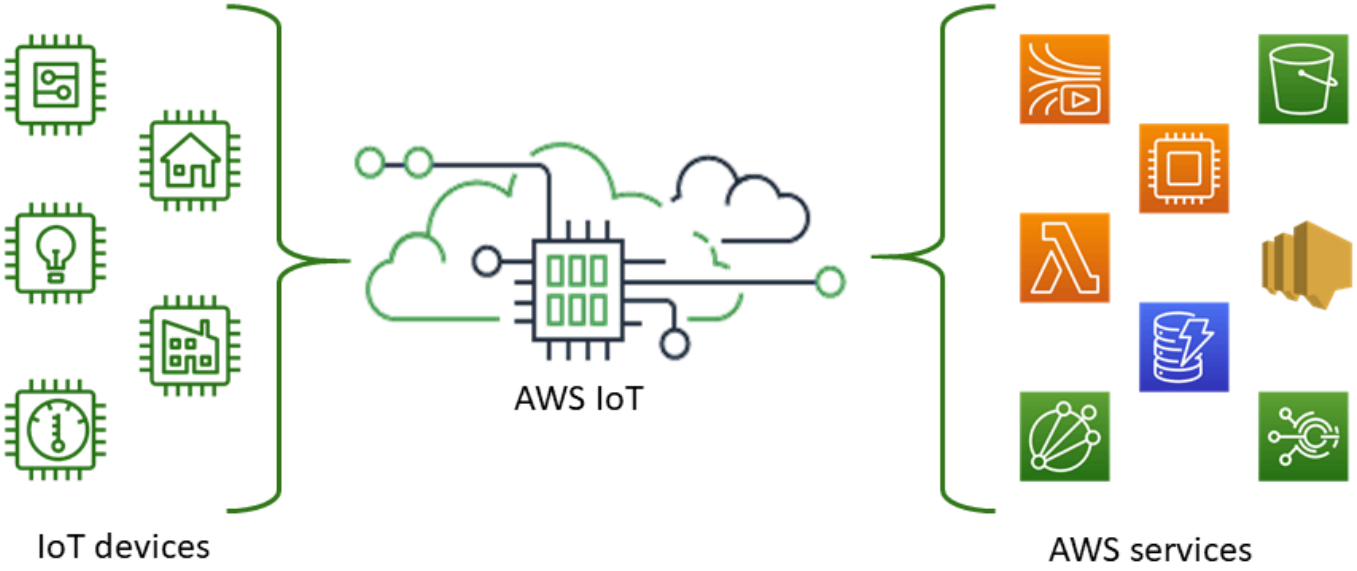
Casos de teste do Device Advisor para se qualificar para o Programa de Qualificação de AWS Dispositivos. ... 1732
TLS ... 1732
MQTT ... 1739
Shadow ... 1754
Execução de trabalho ... 1756
Políticas e permissões ... 1758
Testes de longa duração ... 1759
Local do dispositivo ... 1777
Tipos de medição e solucionadores ... 1778
Como funciona a localização do AWS IoT Core dispositivo ... 1779
Como usar a localização AWS IoT Core do dispositivo ... 1780
Como resolver a localização dos dispositivos de IoT ... 1781
Como resolver o local do dispositivo (console) ... 1782
Como resolver o local do dispositivo (API) ... 1785
Como solucionar problemas com erros ao resolver a localização ... 1787
Como resolver o local do dispositivo usando os tópicos MQTT ... 1788
Formato dos tópicos MQTT de local do dispositivo ... 1788
Política para tópicos MQTT de local do dispositivo ... 1790
Tópicos de local do dispositivo e carga ... 1791
Solucionadores de localização e carga do dispositivo ... 1796
Solucionador baseado em Wi-Fi ... 1796
Solucionador baseado em celular ... 1797
Solucionador de pesquisa reversa de IP ... 1802
Solucionador GNSS ... 1803
Solucionador BLE para dispositivos habilitados para Amazon Sidewalk ... 1805
Mensagens de eventos ... 1806
Como as mensagens de eventos são geradas ... 1806
Política para receber mensagens de eventos ... 1806
Habilitar eventos para AWS IoT ... 1807
Eventos de registro ... 1812
Eventos de objetos ... 1812
Eventos de tipos de objeto ... 1814
Eventos de grupos de objetos ... 1817
Eventos de trabalho ... 1823
Eventos de execução de comandos ... 1828

Eventos de ciclo de vida 1829
Eventos de conexão/desconexão 1830
Evento de falha na tentativa de conexão 1835
Eventos de assinatura/cancelamento de assinatura 1836
Solução de problemas 1839
AWS IoT Core guia de solução de problemas 1839
Diagnóstico de problemas de conectividade 1840
Diagnosticar problemas de regras 1843
Diagnosticar problemas com shadows 1846
Diagnosticar problemas de ação da Salesforce 1848
Diagnosticando limites de fluxo 1849
Solução de problemas de desconexões da frota de dispositivos 1850
AWS IoT Device Management guia de solução de problemas 1851
AWS IoT Solução de problemas de trabalhos 1851
Solução de problemas de indexação de frota 1856
AWS IoT Solução de problemas do Device Management Software Package Catalog 1859
AWS IoT Guia de solução de problemas do Device Advisor 1866
AWS IoT erros 1869
AWS IoT Device SDKs, Mobile SDKs e AWS IoT Device Client 1871
AWS IoTSDKs de dispositivo da 1871
SDK do dispositivo de AWS IoT para C incorporado 1874
AWSSDKs móveis do 1874
AWS IoT Device Client 1875
Versões anteriores de AWS IoT Device SDKs 1876
Exemplos de código 1877
Conceitos básicos 1878
Olá AWS IoT 1878
Conheça os conceitos básicos 1887
Ações 1998
Cotas do AWS IoT 2095
Preços do AWS IoT Core 2096
..... mmxcvii

O que é AWS IoT?

AWS IoT fornece os serviços em nuvem que conectam seus dispositivos de IoT a outros dispositivos e serviços AWS em nuvem. AWS IoT fornece software de dispositivo que pode ajudá-lo a integrar seus dispositivos de IoT em soluções AWS IoT baseadas. Se seus dispositivos puderem se conectar AWS IoT, AWS IoT poderá conectá-los aos serviços de nuvem que AWS fornece.

Para uma introdução prática AWS IoT, visite. Tutoriais de conceitos básicos



Com AWS IoT, você pode selecionar as up-to-date tecnologias mais adequadas para sua solução. Para ajudar você a gerenciar e dar suporte aos seus dispositivos de IoT em campo, AWS IoT Core oferece suporte aos seguintes protocolos:

- MQTT (enfileiramento de mensagens e transporte de telemetria)
- MQTT sobre WSS (Websockets Secure)
- HTTPS (Protocolo de transferência de hipertexto - seguro)
- LoRaWAN (rede de longa distância de longo alcance)

O agente de AWS IoT Core mensagens oferece suporte a dispositivos e clientes que usam os protocolos MQTT e MQTT sobre WSS para publicar e assinar mensagens. Ele também oferece suporte a dispositivos e clientes que usam o protocolo HTTPS para publicar mensagens.

AWS IoT Core for LoRa WAN ajuda você a conectar e gerenciar dispositivos LoRa WAN sem fio (rede de longa distância de longo alcance de baixa potência). AWS IoT Core para LoRa WAN substitui a necessidade de você desenvolver e operar um servidor de rede LoRa WAN (LNS).

Se você não precisar de AWS IoT recursos como comunicações, regras ou trabalhos do dispositivo, consulte AWS Mensagens para obter informações sobre outros serviços AWS IoT de mensagens que podem atender melhor às suas necessidades.

Como seus dispositivos e aplicativos acessam AWS IoT

AWS IoT fornece as seguintes interfaces paraAWS IoT tutoriais:

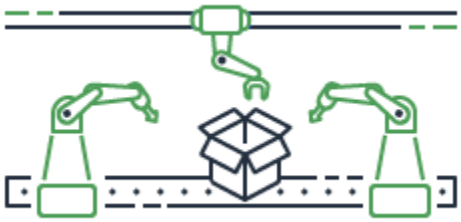
- AWS IoT Dispositivo SDKs — Crie aplicativos em seus dispositivos que enviam e recebem mensagens de AWS IoT. Para obter mais informações, consulte AWS IoT Device SDKs, Mobile SDKs e AWS IoT Device Client.
- AWS IoT Core para LoRa WAN — Conecte e gerencie seus dispositivos e gateways LoRa WAN (WAN) de longo alcance usando AWS IoT Core para LoRa WAN.
- AWS Command Line Interface (AWS CLI) — Execute comandos para AWS IoT Windows, macOS e Linux. Esses comandos permitem criar e gerenciar objetos, certificados, regras, trabalhos e políticas. Para começar a usar, consulte o Guia do usuário da AWS Command Line Interface. Para obter mais informações sobre os comandos para AWS IoT, consulte iot na Referência de AWS CLI Comandos.
- AWS IoT API — Crie seus aplicativos de IoT usando solicitações HTTP ou HTTPS. Essas ações da API permitem que você crie e gerencie de modo programático de objetos, certificados, regras e políticas. Para obter mais informações sobre as ações da API para AWS IoT, consulte Ações na Referência AWS IoT da API.
- AWS SDKs—Crie seus aplicativos de IoT usando linguagens específicas. APIs Eles SDKs envolvem a HTTP/HTTPS API e permitem que você programe em qualquer uma das linguagens suportadas. Para obter mais informações, consulte AWS SDKs e Ferramentas.

Você também pode acessar AWS IoT por meio do AWS IoT console, que fornece uma interface gráfica de usuário (GUI) por meio da qual você pode configurar e gerenciar objetos, certificados, regras, trabalhos, políticas e outros elementos de suas soluções de IoT.

O que AWS IoT pode fazer

Este tópico descreve algumas das soluções que você pode precisar e que ofereçam suporte a AWS IoT .

IoT na indústria



Esses são alguns exemplos de AWS IoT soluções para casos de uso industrial que aplicam tecnologias de IoT para melhorar o desempenho e a produtividade dos processos industriais.

Soluções para casos de uso industrial

- Use AWS IoT para criar modelos preditivos de qualidade em operações industriais

  Veja como AWS IoT coletar e analisar dados de operações industriais para criar modelos preditivos de qualidade. Saiba mais
- Use AWS IoT para apoiar a manutenção preditiva em operações industriais

  Veja como AWS IoT você pode ajudar a planejar a manutenção preventiva para reduzir o tempo de inatividade não planejado. Saiba mais

IoT na automação residencial

Esses são alguns exemplos de AWS IoT soluções para casos de uso de automação residencial que aplicam tecnologias de IoT para criar aplicativos de IoT escaláveis que automatizam as atividades domésticas usando dispositivos domésticos conectados.

Soluções para automação residencial

- Use AWS IoT em sua casa conectada

  Veja como AWS IoT podemos fornecer soluções integradas de automação residencial.

- Use AWS IoT para fornecer segurança e monitoramento doméstico

  Veja como AWS IoT você pode aplicar o aprendizado de máquina e a computação de ponta à sua solução de automação residencial.

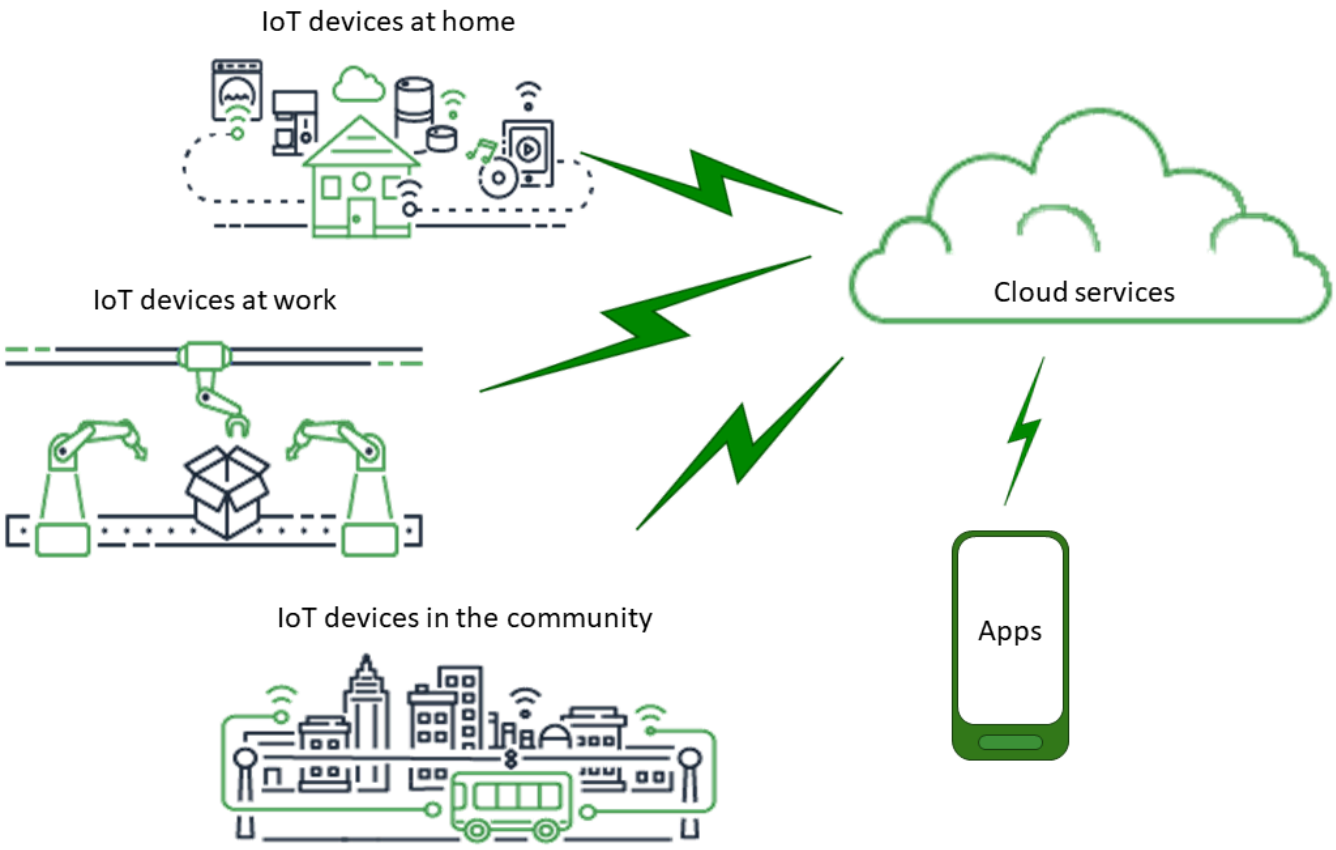
Para obter uma lista de soluções para casos de uso industrial, de consumo e comercial, consulte o AWS IoT repositório de soluções.

Como AWS IoT funciona

AWS IoT fornece serviços de nuvem e suporte a dispositivos que você pode usar para implementar soluções de IoT. AWS fornece muitos serviços em nuvem para oferecer suporte a aplicativos baseados em IoT. Portanto, para ajudá-lo a entender por onde começar, esta seção fornece um diagrama e uma definição de conceitos essenciais para apresentá-lo ao universo IoT.

O universo IoT

Em geral, a Internet das Coisas (IoT) consiste nos principais componentes mostrados neste diagrama.



Apps

Os aplicativos fornecem aos usuários finais acesso a dispositivos IoT e aos atributos fornecidos pelos serviços em nuvem aos quais esses dispositivos estão conectados.

Serviços em nuvem

Os serviços em nuvem são serviços distribuídos de armazenamento e processamento de dados em grande escala conectados à Internet. Os exemplos incluem:

- Serviços de conexão e gerenciamento de IoT

  AWS IoT é um exemplo de um serviço de gerenciamento e conexão de IoT.
- Serviços de computação, como Amazon Elastic Compute Cloud e AWS Lambda
- Serviços de banco de dados, como o Amazon DynamoDB

Comunicações

Os dispositivos se comunicam com os serviços em nuvem usando várias tecnologias e protocolos. Os exemplos incluem:

- Wi-Fi/Internet de banda larga
- Dados celulares de banda larga
- Dados celulares de banda estreita
- Rede de longa distância (LoRaWAN) de longo alcance
- Comunicações de RF proprietárias

Dispositivos

Um dispositivo é um tipo de hardware que gerencia interfaces e comunicações. Os dispositivos geralmente estão localizados próximos às interfaces do mundo real que eles monitoram e controlam. Os dispositivos podem incluir recursos de computação e armazenamento, como microcontroladores, CPU e memória. Os exemplos incluem:

- Raspberry Pi
- Arduino
- Assistentes de interface de voz
- LoRaWAN e dispositivos
- Dispositivos Amazon Sidewalk
- Dispositivos de IoT personalizados

Interfaces

Uma interface é um componente que conecta um dispositivo ao mundo físico.

- Interfaces do usuário

  Componentes que permitem que os dispositivos e os usuários se comuniquem entre si.

  - Interfaces de entrada

    Permitir que um usuário se comunique com um dispositivo

Exemplos: teclado, botão

- Interfaces de entrada

  Permitir que um dispositivo se comunique com um usuário

  Exemplos: display alfanumérico, display gráfico, luz indicadora, campainha de alarme

- Sensores

  Componentes de entrada que medem ou detectam algo no mundo externo de uma forma que um dispositivo entenda. Os exemplos incluem:

  - Sensor de temperatura (converte a temperatura em um sinal analógico ou digital)
  - Sensor de umidade (converte a umidade relativa em um sinal analógico ou digital)
  - Conversor analógico para digital (converte uma tensão analógica em um valor numérico)
  - Unidade ultrassônica de medição de distância (converte uma distância em um valor numérico)
  - Sensor óptico (converte um nível de luz em um valor numérico)
  - Câmera (converte dados de imagem em dados digitais)

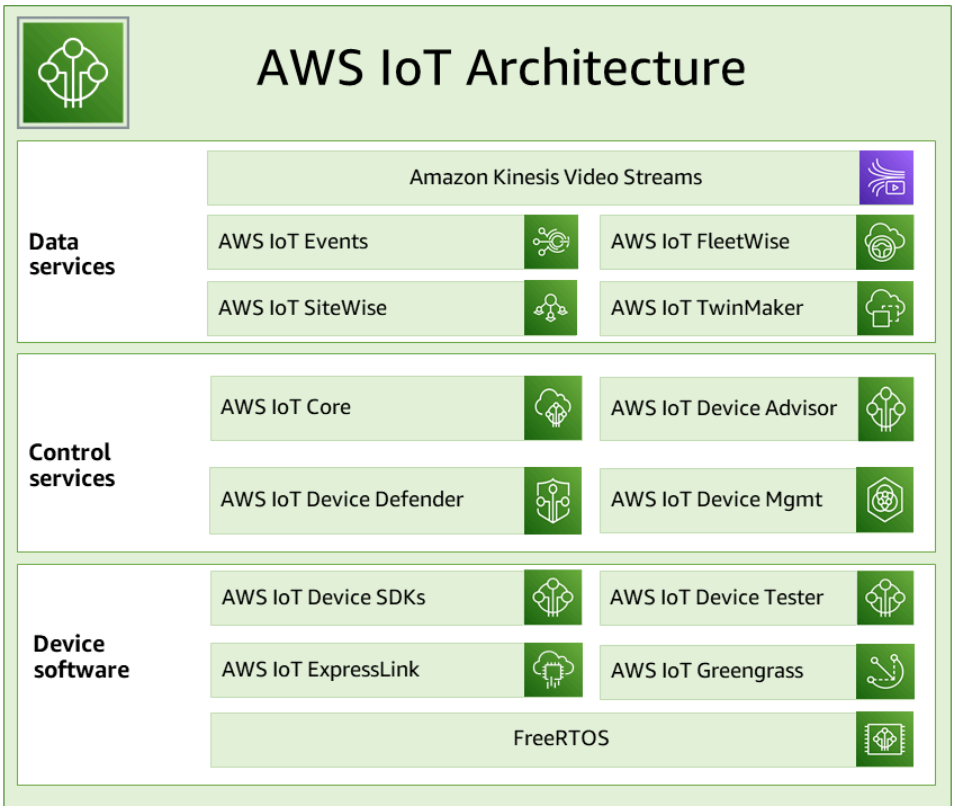
- Atuadores

  Componentes de saída que o dispositivo pode usar para controlar algo no mundo externo. Os exemplos incluem:

  - Motores de passo (convertem sinais elétricos em movimento)
  - Relés (controlam altas tensões e correntes elétricas)

AWS IoT visão geral dos serviços

No universo da IoT, AWS IoT fornece os serviços que suportam os dispositivos que interagem com o mundo e os dados que passam entre eles e. AWS IoT AWS IoT é composto pelos serviços mostrados nesta ilustração para dar suporte à sua solução de IoT.



AWS IoT software do dispositivo

AWS IoT fornece esse software para oferecer suporte aos seus dispositivos de IoT.

AWS IoT Dispositivo SDKs

O AWS IoT dispositivo e o celular SDKs ajudam você a conectar seus dispositivos com eficiência AWS IoT a. O AWS IoT Device and Mobile SDKs inclui bibliotecas de código aberto, guias para desenvolvedores com amostras e guias de portabilidade para que você possa criar produtos ou soluções de IoT inovadores nas plataformas de hardware de sua escolha.

AWS IoT Device Tester

AWS IoT Device Testerpara FreeRTOS AWS IoT Greengrass e é uma ferramenta de automação de testes para microcontroladores. AWS IoT Device Tester testa seu dispositivo para determinar se ele executará FreeRTOS AWS IoT Greengrass ou interoperará com os serviços. AWS IoT

AWS IoT ExpressLink

AWS IoT ExpressLink alimenta uma variedade de módulos de hardware desenvolvidos e oferecidos pelos AWS parceiros. Os módulos AWS de conectividade incluem software validado, tornando mais rápido e fácil conectar dispositivos à nuvem com segurança e integrar-se

perfeitamente a uma variedade de serviços. AWS Para obter mais informações, visite a página de AWS IoT ExpressLinkvisão geral ou consulte o Guia do AWS IoT ExpressLink programador.

AWS IoT Greengrass

AWS IoT Greengrassse estende AWS IoT aos dispositivos de ponta para que eles possam agir localmente com base nos dados que geram, executar previsões com base em modelos de aprendizado de máquina e filtrar e agregar dados do dispositivo. AWS IoT Greengrass permite que seus dispositivos coletem e analisem dados mais perto de onde esses dados são gerados, reajam de forma autônoma a eventos locais e se comuniquem com segurança com outros dispositivos na rede local. Você pode usar AWS IoT Greengrass para criar aplicativos de ponta usando módulos de software pré-criados, chamados de componentes, que podem conectar seus dispositivos de borda a AWS serviços ou serviços de terceiros.

FreeRTOS

O FreeRTOS é um sistema operacional de código aberto em tempo real para microcontroladores que permite incluir dispositivos de ponta pequenos e de baixo consumo de energia em sua solução de IoT. O FreeRTOS inclui um kernel e um conjunto crescente de bibliotecas de software que oferecem suporte a muitos aplicativos. Os sistemas FreeRTOS podem conectar com segurança seus dispositivos pequenos e de baixo consumo de energia a AWS IoT e oferecer suporte a dispositivos de borda mais potentes executando AWS IoT Greengrass.

AWS IoT serviços de controle

Conecte-se aos seguintes AWS IoT serviços para gerenciar os dispositivos em sua solução de IoT.

AWS IoT Core

AWS IoT Coreé um serviço de nuvem gerenciado que permite que dispositivos conectados interajam com segurança com aplicativos em nuvem e outros dispositivos. AWS IoT Core pode suportar muitos dispositivos e mensagens e pode processar e rotear essas mensagens para AWS IoT endpoints e outros dispositivos. Com AWS IoT Core, seus aplicativos podem interagir com todos os seus dispositivos, mesmo quando não estão conectados.

AWS IoT Core Consultor de dispositivos

O AWS IoT Core Device Advisor é um recurso de teste totalmente gerenciado baseado em nuvem para validar dispositivos de IoT durante o desenvolvimento do software do dispositivo. O Device Advisor fornece testes pré-criados que você pode usar para validar dispositivos de IoT para conectividade confiável e segura antes de implantá-los na produção. AWS IoT Core

AWS IoT Defensor do dispositivo

AWS IoT O Device Defender ajuda você a proteger sua frota de dispositivos de IoT. AWS IoT O Device Defender audita continuamente suas configurações de IoT para garantir que elas não estejam se desviando das melhores práticas de segurança. AWS IoT O Device Defender envia um alerta quando detecta qualquer lacuna na sua configuração de IoT que possa criar um risco de segurança, como certificados de identidade compartilhados em vários dispositivos ou um dispositivo com um certificado de identidade revogado tentando se conectar. AWS IoT Core

AWS IoT Gerenciamento de dispositivos

AWS IoT Os serviços de gerenciamento de dispositivos ajudam você a rastrear, monitorar e gerenciar a infinidade de dispositivos conectados que compõem suas frotas de dispositivos. AWS IoT Os serviços de gerenciamento de dispositivos ajudam você a garantir que seus dispositivos de IoT funcionem de forma adequada e segura após a implantação. Eles também fornecem tunelamento seguro para acessar seus dispositivos, monitorar sua integridade, detectar e solucionar problemas remotamente, bem como serviços para gerenciar atualizações de software e firmware do dispositivo.

AWS IoT serviços de dados

Analise os dados dos dispositivos em sua solução de IoT e tome as medidas apropriadas usando os seguintes AWS IoT serviços.

Amazon Kinesis Video Streams

O Amazon Kinesis Video Streams permite que você transmita vídeo ao vivo de dispositivos AWS para a nuvem, onde é armazenado, criptografado e indexado de forma durável, permitindo que você acesse seus dados por meio de. easy-to-use APIs Você pode usar o Amazon Kinesis Video Streams para capturar enormes quantidades de dados de vídeo ao vivo de milhões de fontes, incluindo smartphones, câmeras de segurança, webcams, câmeras integradas em carros, drones e muito mais. O Amazon Kinesis Video Streams permite que você reproduza vídeo para visualização ao vivo e sob demanda, crie rapidamente aplicativos que aproveitam a visão computacional e a análise de vídeo por meio da integração com o Amazon Rekognition Video e frameworks do ML. Você também pode enviar dados diferentes de vídeo e não serializados em tempo, como dados de áudio, imagens térmicas, dados de profundidade, dados RADAR e muito mais.

Amazon Kinesis Video Streams com WebRTC

O Amazon Kinesis Video Streams com WebRTC fornece uma implementação de WebRTC compatível com os padrões como recurso totalmente gerenciado. Você pode usar o Amazon Kinesis Video Streams com WebRTC para transmitir mídia ao vivo com segurança ou realizar interação bidirecional de áudio ou vídeo entre qualquer câmera, dispositivo de IoT e players móveis ou Web compatíveis com WebRTC. Como recurso totalmente gerenciado, não é preciso criar, operar ou escalar nenhuma infraestrutura de nuvem relacionada ao WebRTC, como servidores de sinalização ou retransmissão de mídia para transmitir mídia com segurança entre aplicativos e dispositivos. Usando o Amazon Kinesis Video Streams com WebRTC, você pode criar facilmente peer-to-peer aplicativos para streaming de mídia ao vivo ou interatividade de áudio ou vídeo em tempo real entre dispositivos IoT de câmera, navegadores da web e dispositivos móveis para uma variedade de casos de uso.

AWS IoT Eventos

AWS IoT O Events detecta e responde a eventos de sensores e aplicativos de IoT. Eventos são padrões de dados que identificam circunstâncias mais complicadas do que o esperado, como detectores de movimento usando sinais de movimento para ativar luzes e câmeras de segurança. AWS IoT O Events monitora continuamente os dados de vários sensores e aplicativos de IoT e se integra a outros serviços, como AWS IoT Core IoT, SiteWise DynamoDB e outros, para permitir a detecção precoce e insights exclusivos.

AWS IoT FleetWise

AWS IoT FleetWiseé um serviço gerenciado que você pode usar para coletar e transferir dados do veículo para a nuvem quase em tempo real. Com AWS IoT FleetWise, você pode coletar e organizar facilmente dados de veículos que usam protocolos e formatos de dados diferentes. AWS IoT FleetWise ajuda a transformar mensagens de baixo nível em valores legíveis por humanos e a padronizar o formato de dados na nuvem para análises de dados. Você também pode definir esquemas de coleta de dados para controlar quais dados coletar em veículos e quando transferi-los para a nuvem.

AWS IoT SiteWise

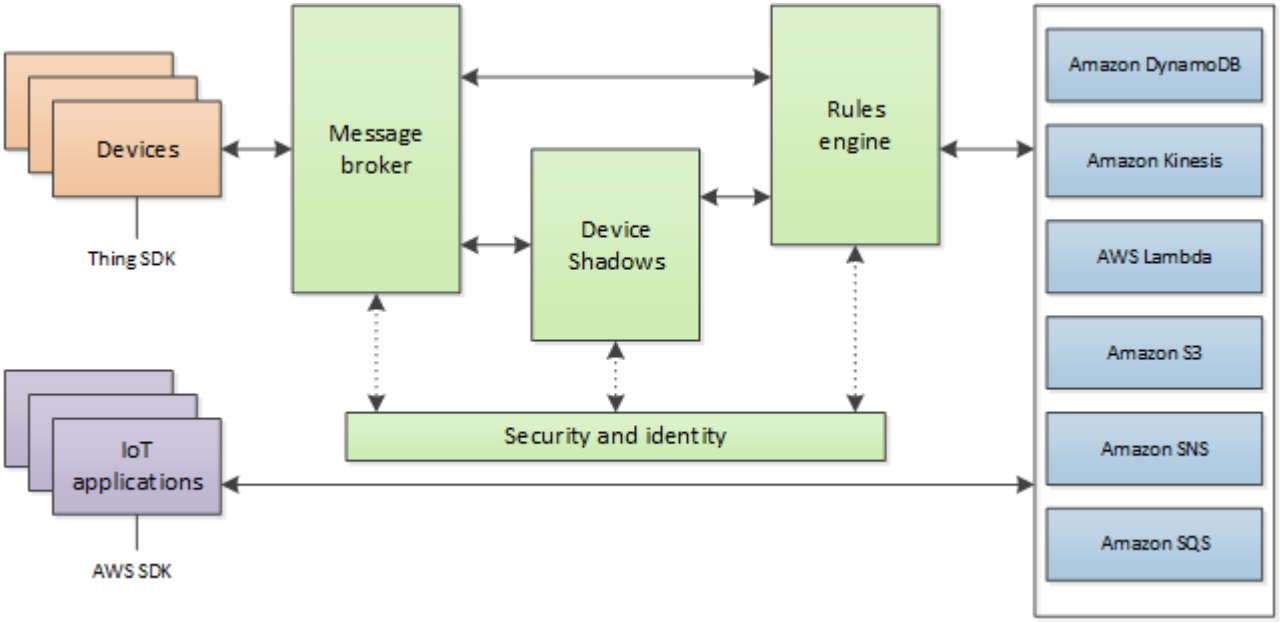
AWS IoT SiteWisecoleta, armazena, organiza e monitora dados transmitidos de equipamentos industriais por mensagens MQTT ou APIs em grande escala, fornecendo software executado em um gateway em suas instalações. O gateway se conecta com segurança aos seus servidores de dados locais e automatiza o processo de coleta e organização dos dados e envio para a nuvem. AWS

AWS IoT TwinMaker

AWS IoT TwinMakercria gêmeos digitais operacionais de sistemas físicos e digitais. AWS IoT TwinMaker cria visualizações digitais usando medições e análises de uma variedade de sensores, câmeras e aplicativos corporativos do mundo real para ajudá-lo a acompanhar sua fábrica física, prédio ou planta industrial. Você pode usar dados do mundo real para monitorar operações, diagnosticar e corrigir erros, e otimizar as operações.

AWS IoT Core serviços

AWS IoT Core fornece os serviços que conectam seus dispositivos de IoT à AWS nuvem para que outros serviços e aplicativos em nuvem possam interagir com seus dispositivos conectados à Internet.



A próxima seção descreve cada um dos AWS IoT Core serviços mostrados na ilustração.

AWS IoT Core serviços de mensagens

Os serviços de AWS IoT Core conectividade fornecem comunicação segura com os dispositivos de IoT e gerenciam as mensagens que passam entre eles e. AWS IoT

Gateway do dispositivo

Permite que dispositivos se comuniquem com a AWS IoT com segurança e eficiência. A comunicação do dispositivo é protegida por protocolos seguros que utilizam certificados X.509.

Operador de mensagens

Fornece um mecanismo seguro para dispositivos e AWS IoT aplicativos publicarem e receberem mensagens uns dos outros. Você pode usar o protocolo MQTT diretamente ou o MQTT over WebSocket para publicar e assinar. Para obter mais informações sobre protocolos compatíveis com o AWS IoT , consulte the section called “Protocolos de comunicação do dispositivo”. Dispositivos e clientes também podem usar a interface HTTP REST para publicar dados no agente de mensagens.

O agente de mensagens distribui os dados do dispositivo para os dispositivos que se inscreveram nele e para outros AWS IoT Core serviços, como o serviço Device Shadow e o mecanismo de regras.

AWS IoT Core para LoRa WAN

AWS IoT Core for LoRa WAN possibilita a configuração de uma rede LoRa WAN privada conectando seus dispositivos e gateways de LoRa WAN AWS sem a necessidade de desenvolver e operar um servidor de rede LoRa WAN (LNS). As mensagens recebidas dos dispositivos LoRa WAN são enviadas para o mecanismo de regras, onde podem ser formatadas e enviadas para outros AWS IoT serviços.

Mecanismo de regras

O mecanismo de regras conecta dados do agente de mensagens a outros serviços AWS IoT para armazenamento e processamento adicional. Por exemplo, você pode inserir, atualizar ou consultar uma tabela do DynamoDB ou invocar uma função do Lambda com base em uma expressão definida no mecanismo de regras. Você pode usar uma linguagem baseada em SQL para selecionar dados de cargas de mensagem, processar e enviar os dados para outros serviços, como Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB e AWS Lambda. Também é possível criar regras que republicam mensagens para o agente de mensagens e para outros assinantes. Para obter mais informações, consulte Regras para AWS IoT.

AWS IoT Core serviços de controle

Os serviços AWS IoT Core de controle fornecem recursos de segurança, gerenciamento e registro de dispositivos.

Serviço de autenticação personalizada

Você pode definir autorizadores personalizados que permitem que você gerencie sua própria estratégia de autenticação e autorização usando um serviço de autenticação personalizada e uma função do Lambda. Os autorizadores personalizados permitem AWS IoT autenticar seus dispositivos e autorizar operações usando estratégias de autenticação e autorização do token do portador.

Os autorizadores personalizados podem implementar várias estratégias de autenticação; por exemplo, verificação do JSON Web Token ou chamada do OAuth provedor. Eles devem retornar documentos de política usados pelo gateway do dispositivo para autorizar operações MQTT. Para obter mais informações, consulte Autenticação e autorização personalizadas.

Serviço de provisionamento de dispositivos

Permite provisionar dispositivos usando um modelo que descreve os recursos necessários para seu dispositivo: um objeto, um certificado e uma ou mais políticas. Um objeto é uma entrada no registro que contém atributos que descrevem um dispositivo. Os dispositivos usam certificados para se autenticar AWS IoT. As políticas determinam quais operações um dispositivo pode executar na AWS IoT.

Os modelos contêm variáveis que são substituídas por valores em um dicionário (mapa). Você pode usar o mesmo modelo para provisionar vários dispositivos bastando passar diferentes valores para as variáveis do modelo no dicionário. Para obter mais informações, consulte Provisionamento de dispositivos.

Registro de grupo

Os grupos permitem gerenciar vários dispositivos simultaneamente, categorizando-os em grupos. Os grupos também podem conter grupos — você pode criar uma hierarquia de grupos. Qualquer ação executada em um grupo pai será aplicada aos grupos filho. A mesma ação também se aplica a todos os dispositivos do grupo principal e a todos os dispositivos dos grupos filho. As permissões concedidas a um grupo serão aplicadas a todos os dispositivos no grupo e a todos os seus grupos filho. Para obter mais informações, consulte Gerenciando dispositivos com AWS IoT.

Serviço Jobs

Permite definir um conjunto de operações remotas que são enviadas e executadas em um ou mais dispositivos conectados à AWS IoT. Por exemplo, você pode definir um trabalho que instrui um conjunto de dispositivos a baixar e instalar um aplicativo ou atualizações de firmware, reinicializar, alternar certificados ou executar operações de solução de problemas remotamente.

Para criar um trabalho, você especifica uma descrição das operações remotas a serem executadas e uma lista de destinos que devem executá-las. Os destinos podem ser dispositivos individuais, grupos ou ambos. Para obter mais informações, consulte AWS IoT Empregos.

Registro

Organiza os recursos associados a cada dispositivo na nuvem AWS . É possível registrar seus dispositivos e associar até três atributos personalizados a cada um. Você também pode associar certificados e clientes IDs MQTT a cada dispositivo para melhorar sua capacidade de gerenciá-los e solucioná-los. Para obter mais informações, consulte Gerenciando dispositivos com AWS IoT.

Serviço de segurança e identidade

Fornece responsabilidade compartilhada pela segurança na AWS nuvem. Seus dispositivos devem manter as credenciais seguras para enviar dados com segurança ao agente de mensagens. O agente de mensagens e o mecanismo de regras usam os recursos de segurança da AWS para enviar dados com segurança para dispositivos ou outros serviços da AWS . Para obter mais informações, consulte Autenticação.

AWS IoT Core serviços de dados

AWS IoT Core Os serviços de dados ajudam suas soluções de IoT a fornecer uma experiência de aplicação confiável, mesmo com dispositivos que nem sempre estão conectados.

Sombra do Dispositivo

Um documento JSON usado para armazenar e recuperar as informações do estado atual de um dispositivo.

Serviço Sombra do Dispositivo

O serviço Sombra do Dispositivo mantém o estado de um dispositivo para que os aplicativos possam se comunicar com um dispositivo, esteja ele online ou não. Quando um dispositivo está offline, o serviço Sombra do Dispositivo gerencia seus dados para aplicativos conectados. Quando o dispositivo se reconecta, ele sincroniza seu estado com o de sua sombra no serviço Sombra do Dispositivo. Seus dispositivos também podem publicar o estado atual deles em uma sombra para ser usado por aplicativos ou outros dispositivos que podem não estar conectados o tempo todo. Para obter mais informações, consulte AWS IoT Serviço Device Shadow.

AWS IoT Core serviço de suporte

Integração com o Amazon Sidewalk para AWS IoT Core

O Amazon Sidewalk é uma rede compartilhada que aprimora as opções de conectividade para ajudar os dispositivos a funcionarem melhor juntos. O Amazon Sidewalk oferece suporte a uma ampla variedade de dispositivos de clientes, como aqueles que localizam animais de estimação ou objetos de valor, aqueles que fornecem segurança residencial inteligente e controle de iluminação e aqueles que fornecem diagnóstico remoto para eletrodomésticos e ferramentas. A integração com o Amazon Sidewalk AWS IoT Core possibilita que os fabricantes de dispositivos adicionem sua frota de dispositivos Sidewalk à AWS IoT nuvem.

Para obter mais informações, consulte AWS IoT Core para Amazon Sidewalk.

Saiba mais sobre AWS IoT

Este tópico ajuda você a se familiarizar com o mundo do AWS IoT. Você pode obter informações gerais sobre como as soluções de IoT são aplicadas em vários casos de uso, recursos de treinamento, links para mídias sociais AWS IoT e todos os outros AWS serviços e uma lista dos serviços e protocolos de comunicação que AWS IoT usa.

Recursos de treinamento para AWS IoT

Oferecemos esses cursos de treinamento para ajudar você a aprender sobre eles AWS IoT e como aplicá-los ao design de sua solução.

- Introdução ao AWS IoT

  Uma visão geral em vídeo AWS IoT e de seus principais serviços.
- Mergulhe profundamente na AWS IoT autenticação e autorização

  Um curso avançado que explora os conceitos de AWS IoT autenticação e autorização. Você aprenderá como autenticar e autorizar clientes a acessar o plano de AWS IoT controle e o plano de dados. APIs
- Série de apresentação sobre a Internet das Coisas

  Um caminho de aprendizado dos módulos de eLearning de IoT sobre diferentes tecnologias e recursos de IoT.

AWS IoT recursos e guias

Esses são recursos técnicos aprofundados sobre aspectos específicos do AWS IoT.

- IoT Lens — AWS IoT Well-Architected Framework

  Um documento que descreve as melhores práticas para arquitetar seus aplicativos de IoT no. AWS
- Criando tópicos do MQTT para AWS IoT Core

  Um whitepaper que descreve as melhores práticas para criar tópicos do MQTT AWS IoT Core e aproveitar recursos AWS IoT Core com o MQTT.
- Resumo e introdução

  Um documento PDF que descreve as diferentes formas de AWS IoT provisionar grandes frotas de dispositivos.
- AWS IoT Core Consultor de dispositivos

  AWS IoT Core O Device Advisor fornece testes pré-criados que você pode usar para validar dispositivos de IoT para obter as melhores práticas de conectividade confiáveis e seguras antes de implantar dispositivos na produção. AWS IoT Core
- AWS IoT Recursos

  Recursos específicos de IoT, como guias técnicos, arquiteturas de referência, e-books e postagens de blog selecionadas, apresentados em um índice pesquisável.
- Atlas de IoT

  Visão geral sobre como resolver problemas comuns de design de IoT. O Atlas de IoT fornece uma análise aprofundada dos desafios de design que você provavelmente vai enfrentar ao desenvolver sua solução de IoT.
- AWS Documentos técnicos e guias

  Nossa coleção atual de documentos técnicos e guias sobre AWS IoT e outras AWS tecnologias.

AWS IoT nas mídias sociais

Esses canais de mídia social fornecem informações AWS IoT e tópicos AWS relacionados.

- A Internet das Coisas em AWS IoT — Blog oficial

- AWS IoT vídeos no canal da Amazon Web Services em YouTube

Essas contas de mídia social cobrem todos os AWS serviços, incluindo AWS IoT

- O canal da Amazon Web Services em YouTube
- Amazon Web Services no Twitter
- Amazon Web Services no Facebook
- Amazon Web Services no Instagram
- Amazon Web Services em LinkedIn

AWS serviços usados pelo mecanismo de AWS IoT Core regras

O mecanismo de AWS IoT Core regras pode se conectar a esses AWS serviços.

- Amazon DynamoDB

  O Amazon DynamoDB é um serviço de banco de dados NoSQL escalável que fornece desempenho de banco de dados rápido e previsível.

- Amazon Kinesis

  O Amazon Kinesis facilita a coleta, o processamento e a análise de dados de streaming em tempo real para que você possa obter insights oportunos e reagir rapidamente a novas informações. O Amazon Kinesis pode ingerir dados em tempo real, como vídeo, áudio, logs de aplicativos, fluxos de cliques de sites e dados de telemetria de IoT para machine learning, análises e outros aplicativos.

- AWS Lambda

  AWS Lambda permite que você execute código sem provisionar ou gerenciar servidores. Você pode configurar seu código para ser acionado automaticamente a partir de AWS IoT dados e eventos ou chamá-lo diretamente de um aplicativo web ou móvel.

- Amazon Simple Storage Service

  O Amazon Simple Storage Service (Amazon S3) pode armazenar e recuperar qualquer quantidade de dados a qualquer momento, de qualquer lugar na web. AWS IoT as regras podem enviar dados para o Amazon S3 para armazenamento.

- Amazon Simple Notification Service

O Amazon Simple Notification Service (Amazon SNS) é um serviço web que permite que aplicativos, usuários finais e dispositivos enviem e recebam notificações da nuvem.

- Amazon Simple Queue Service

  O Amazon Simple Queue Service (Amazon SQS) é um serviço de enfileiramento de mensagens que desacopla e dimensiona microsserviços, sistemas distribuídos e aplicativos de tecnologia sem servidor.

- OpenSearch Serviço Amazon

  O Amazon OpenSearch Service (OpenSearch Service) é um serviço gerenciado que facilita a implantação, a operação e a escalabilidade OpenSearch, um popular mecanismo de pesquisa e análise de código aberto.

- SageMaker IA da Amazon

  A Amazon SageMaker AI pode criar modelos de aprendizado de máquina (ML) encontrando padrões em seus dados de IoT. O serviço usa esses modelos para processar novos dados e gerar previsões para seu aplicativo.

- Amazon CloudWatch

  CloudWatch A Amazon fornece uma solução de monitoramento confiável, escalável e flexível para ajudar a configurar, gerenciar e escalar seus próprios sistemas e infraestrutura de monitoramento.

Protocolos de comunicação suportados por AWS IoT Core

Esses tópicos fornecem mais informações sobre os protocolos de comunicação usados pelo AWS IoT. Para obter mais informações sobre os protocolos usados AWS IoT e os que conectam dispositivos e serviços aos AWS IoT, consulteConecte-se a AWS IoT Core.

- MQTT (Transporte de telemetria de enfileiramento de mensagens)

  Na página inicial do site MQTT.org, onde você pode encontrar as especificações do protocolo MQTT. Para obter mais informações sobre como AWS IoT suporta o MQTT, consulteMQTT.

- HTTPS (Protocolo de transferência de hipertexto - seguro)

  Dispositivos e aplicativos podem acessar AWS IoT serviços usando HTTPS.

- LoRaWAN (rede de longa distância de longo alcance)

LoRaDispositivos e gateways de WAN podem se conectar AWS IoT Core usando AWS IoT Core para LoRa WAN.

- Transport Layer Security (TLS) v1.3

  A especificação do TLS v1.3 (RFC 5246). AWS IoT usa TLS v1.3 para estabelecer conexões seguras entre dispositivos e. AWS IoT

Novidades do console AWS IoT

Estamos atualizando a interface do usuário do console AWS IoT para uma nova experiência. Estamos atualizando a interface do usuário em etapas, portanto, algumas páginas no console terão uma nova experiência, algumas poderão ter a experiência original e a nova e algumas poderão ter apenas a experiência original.

Esta tabela exibe o estado das áreas individuais da interface do usuário do console AWS IoT em 27 de janeiro de 2022.

status da interface do usuário do console AWS IoT

Página do console	Experiência original	Nova experiência	Comentários
Monitorar	Indisponível	Available (Disponível)	
Atividade	Indisponível	Available (Disponível)	
Onboard - Comece agora	Indisponível	Available (Disponível)	Não disponível nas regiões CN
Onboard – Modelos de provisionamento de frota	Available (Disponível)	Available (Disponível)	
Gerenciar - Objetos	Available (Disponível)	Available (Disponível)	
Gerenciar - Tipos	Available (Disponível)	Available (Disponível)	
Gerenciar - Grupos de objeto	Available (Disponível)	Available (Disponível)	

Página do console	Experiência original	Nova experiência	Comentários
Gerenciar - Grupos de cobrança	Available (Disponível)	Available (Disponível)	
Gerenciar - trabalhos	Available (Disponível)	Available (Disponível)	
Gerenciar - Modelos de trabalho	Indisponível	Available (Disponível)	
Gerenciar - Túneis	Indisponível	Available (Disponível)	
Fleet Hub - Comece agora	Indisponível	Available (Disponível)	Não disponível em todas as Regiões da AWS
Fleet Hub - Aplicações	Indisponível	Available (Disponível)	Não disponível em todas as Regiões da AWS
Greengrass - Introdução	Indisponível	Available (Disponível)	Não disponível em todas as Regiões da AWS
Greengrass - Dispositivos principais	Indisponível	Available (Disponível)	Não disponível em todas as Regiões da AWS
Greengrass - Componentes	Indisponível	Available (Disponível)	Não disponível em todas as Regiões da AWS
Greengrass - Implantações	Indisponível	Available (Disponível)	Não disponível em todas as Regiões da AWS
Greengrass - Clássico (V1)	Available (Disponível)	Available (Disponível)	

Página do console	Experiência original	Nova experiência	Comentários
Conectividade sem fio - Introdução	Indisponível	Available (Disponível)	Não disponível em todas as Regiões da AWS
Conectividade sem fio - Gateways	Indisponível	Available (Disponível)	Não disponível em todas as Regiões da AWS
Conectividade sem fio - Dispositivos	Indisponível	Available (Disponível)	Não disponível em todas as Regiões da AWS
Conectividade sem fio - Perfis	Indisponível	Available (Disponível)	Não disponível em todas as Regiões da AWS
Conectividade sem fio - Destinos	Indisponível	Available (Disponível)	Não disponível em todas as Regiões da AWS
Seguro - Certificados	Available (Disponível)	Available (Disponível)	
Seguro - Políticas	Available (Disponível)	Available (Disponível)	
Seguro - CAs	Available (Disponível)	Available (Disponível)	
Seguro - Aliases de funções	Available (Disponível)	Available (Disponível)	
Seguro - Autorizadores	Available (Disponível)	Available (Disponível)	
Defender - Introdução	Indisponível	Available (Disponível)	
Defender - Auditar	Indisponível	Available (Disponível)	
Defender - Detectar	Indisponível	Available (Disponível)	

Página do console	Experiência original	Nova experiência	Comentários
Defender - Ações de mitigação	Indisponível	Available (Disponível)	
Defender - Configurações	Indisponível	Available (Disponível)	
Ato - Regras	Available (Disponível)	Available (Disponível)	
Ato - Destinos	Available (Disponível)	Available (Disponível)	
Teste - Consultor de dispositivos	Available (Disponível)	Available (Disponível)	Não disponível em todas as Regiões da AWS
Teste - cliente de teste MQTT	Available (Disponível)	Available (Disponível)	
Software	Available (Disponível)	Available (Disponível)	
Configurações	Indisponível	Available (Disponível)	
Saiba mais	Available (Disponível)	Ainda não disponível	

Legenda

Valores do status

- Available (Disponível)

  Essa experiência de interface de usuário pode ser usada.
- Indisponível

  Essa experiência de interface de usuário não pode ser usada.
- Ainda não disponível

  A nova experiência da interface do usuário está sendo aprimorada, mas ainda não está pronta.
- In progress (Em andamento)

A nova experiência da interface do cliente está em processo de atualização. No entanto, algumas páginas ainda podem ter a experiência original do usuário.

Usando AWS IoT com um AWS SDK

AWS kits de desenvolvimento de software (SDKs) estão disponíveis para muitas linguagens de programação populares. Cada SDK fornece uma API, exemplos de código e documentação que permitem que os desenvolvedores criem facilmente aplicações em seu idioma de preferência.

Documentação do SDK	Exemplos de código
AWS SDK para C++	AWS SDK para C++ exemplos de código
AWS CLI	AWS CLI exemplos de código
AWS SDK para Go	AWS SDK para Go exemplos de código
AWS SDK para Java	AWS SDK para Java exemplos de código
AWS SDK para JavaScript	AWS SDK para JavaScript exemplos de código
AWS SDK para Kotlin	AWS SDK para Kotlin exemplos de código
AWS SDK para .NET	AWS SDK para .NET exemplos de código
AWS SDK para PHP	AWS SDK para PHP exemplos de código
Ferramentas da AWS para PowerShell	Ferramentas da AWS para PowerShell exemplos de código
AWS SDK para Python (Boto3)	AWS SDK para Python (Boto3) exemplos de código
AWS SDK para Ruby	AWS SDK para Ruby exemplos de código
AWS SDK para Rust	AWS SDK para Rust exemplos de código
AWS SDK para SAP ABAP	AWS SDK para SAP ABAP exemplos de código

Documentação do SDK	Exemplos de código
AWS SDK para Swift	AWS SDK para Swift exemplos de código

Exemplo de disponibilidade

Não consegue encontrar o que precisa? Solicite um exemplo de código usando o link Fornecer feedback na parte inferior desta página.

Introdução aos AWS IoT Core tutoriais

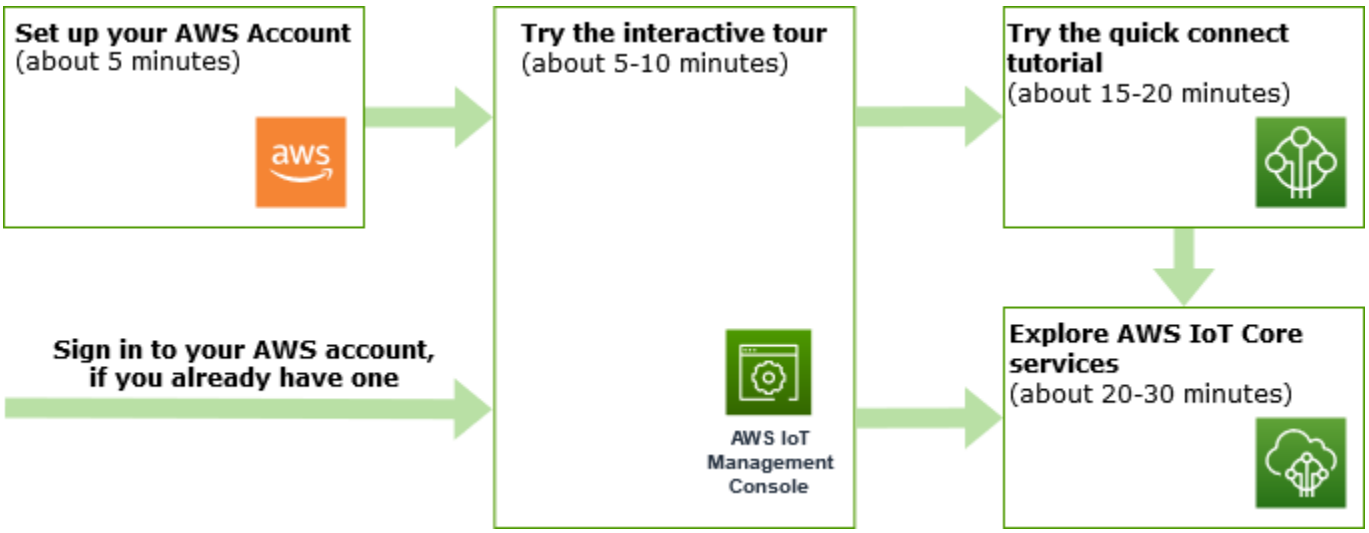
Se você é novato na IoT ou tem anos de experiência, esses recursos apresentam os AWS IoT conceitos e os termos que ajudarão você a começar a usar. AWS IoT

- Olhe para dentro AWS IoT e seus componentes para dentroComo AWS IoT funciona.
- Saiba mais sobre o AWS IoT com nossa coleção de materiais e vídeos de treinamento. Este tópico também inclui uma lista de serviços aos quais o AWS IoT pode se conectar, links de mídias sociais e links para especificações de protocolos de comunicação.
- the section called “Conecte seu primeiro dispositivo a AWS IoT Core”.
- Desenvolva suas soluções de IoT com Conecte-se a AWS IoT Core e explorando os AWS IoT tutoriais.
- Teste e valide seus dispositivos de IoT para uma comunicação segura e confiável com o Device Advisor.
- Gerencie sua solução com serviços de gerenciamento do AWS IoT Core , como Indexação de frotas, AWS IoT Empregos e AWS IoT Device Defender.
- Analise os dados de seus dispositivos com os AWS IoT serviços de dados.

Conecte seu primeiro dispositivo a AWS IoT Core

AWS IoT Core os serviços conectam dispositivos de IoT a AWS IoT serviços e outros AWS serviços. AWS IoT Core inclui o gateway do dispositivo e o agente de mensagens, que conectam e processam mensagens entre seus dispositivos de IoT e a nuvem.

Veja como você pode começar com AWS IoT Core AWS IoT e.



Esta seção apresenta um tour do AWS IoT Core para apresentar seus principais serviços e fornece vários exemplos de como conectar um dispositivo AWS IoT Core e transmitir mensagens entre eles. Passar mensagens entre dispositivos e a nuvem é fundamental para todas as soluções de IoT e é como seus dispositivos podem interagir com outros AWS serviços.

- Configurar Conta da AWS

  Antes de poder usar AWS IoT os serviços, você deve configurar um Conta da AWS. Se você já tem um Conta da AWS e um usuário do IAM para si mesmo, você pode usá-los e pular esta etapa.

- Experimente o tutorial de conexão rápida

  Este tutorial é melhor se você quiser começar rapidamente AWS IoT e ver como ele funciona em um cenário limitado. Neste tutorial, você precisará de um dispositivo e instalará alguns AWS IoT softwares nele. Para este tutorial, se não tiver um dispositivo de IoT, é possível usar seu computador pessoal Windows, Linux ou macOS como um dispositivo. Se você quiser experimentar AWS IoT, mas não tiver um dispositivo, tente a próxima opção.

- Experimente o tutorial interativo

  Esta demonstração é melhor se você quiser ver o que uma AWS IoT solução básica pode fazer sem conectar um dispositivo ou baixar nenhum software. O tutorial interativo apresenta uma solução simulada baseada em AWS IoT Core serviços que ilustra como eles interagem.

- Explore AWS IoT Core os serviços com um tutorial prático

  Este tutorial é ideal para desenvolvedores que desejam começar AWS IoT a explorar outros AWS IoT Core recursos, como o mecanismo de regras e as sombras. Este tutorial segue um processo

semelhante ao tutorial de conexão rápida, mas oferece mais detalhes a respeito de cada etapa para permitir uma transição mais harmoniosa para os tutoriais mais avançados.

- Exibir mensagens MQTT com o cliente AWS IoT MQTT

  Aprenda a usar o cliente de teste MQTT para ver seu primeiro dispositivo publicar mensagens MQTT no AWS IoT. O cliente de teste MQTT é uma ferramenta útil para monitorar e solucionar problemas de conexão dos dispositivos.

> **Note**
>
> Caso queira concluir mais de um dos tutoriais de introdução ou repetir o mesmo tutorial, será preciso excluir o objeto criado em um tutorial anterior antes de dar início ao outro. Caso não exclua o objeto de um tutorial anterior, será preciso usar um nome de objeto diferente para os tutoriais posteriores. Isso ocorre porque o nome de objeto deve ser único à sua conta e Região da AWS.

Para obter mais informações sobre AWS IoT Core, consulte O que é AWS IoT Core?

Configurar Conta da AWS

Antes de usar AWS IoT Core pela primeira vez, conclua as seguintes tarefas:

Tópicos

- Inscreva-se para um Conta da AWS
- Criar um usuário com acesso administrativo
- Abra o AWS IoT console

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

1. Abra a https://portal.aws.amazon.com/billing/inscrição.
2. Siga as instruções online.

Parte do procedimento de inscrição envolve receber uma chamada telefônica ou uma mensagem de texto e inserir um código de verificação pelo teclado do telefone.

Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWSé criado. O usuário-raiz tem acesso a todos os Serviços da AWS e recursos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. A qualquer momento, você pode visualizar a atividade atual da sua conta e gerenciar sua conta acessando https://aws.amazon.com/e escolhendo Minha conta.

Criar um usuário com acesso administrativo

Depois de se inscrever em um Conta da AWS, proteja seu Usuário raiz da conta da AWS Centro de Identidade do AWS IAM, habilite e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu Usuário raiz da conta da AWS

1. Faça login Console de gerenciamento da AWScomo proprietário da conta escolhendo Usuário raiz e inserindo seu endereço de Conta da AWS e-mail. Na próxima página, insira a senha.

   Para obter ajuda ao fazer login usando o usuário-raiz, consulte Fazer login como usuário-raiz no Guia do usuário do Início de Sessão da AWS .
2. Habilite a autenticação multifator (MFA) para o usuário-raiz.

   Para obter instruções, consulte Habilitar um dispositivo de MFA virtual para seu usuário Conta da AWS raiz (console) no Guia do usuário do IAM.

Criar um usuário com acesso administrativo

1. Habilita o Centro de Identidade do IAM.

   Para obter instruções, consulte Habilitar o Centro de Identidade do AWS IAM no Guia do usuário do Centro de Identidade do AWS IAM .
2. No Centro de Identidade do IAM, conceda o acesso administrativo a um usuário.

Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM no Guia Centro de Identidade do AWS IAM do usuário.

Iniciar sessão como o usuário com acesso administrativo

- Para fazer login com o seu usuário do Centro de Identidade do IAM, use o URL de login enviado ao seu endereço de e-mail quando o usuário do Centro de Identidade do IAM foi criado.

  Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte Como fazer login no portal de AWS acesso no Guia Início de Sessão da AWS do usuário.

Atribuir acesso a usuários adicionais

1. No Centro de Identidade do IAM, crie um conjunto de permissões que siga as práticas recomendadas de aplicação de permissões com privilégio mínimo.

   Para obter instruções, consulte Criar um conjunto de permissões no Guia do usuário do Centro de Identidade do AWS IAM .
2. Atribua usuários a um grupo e, em seguida, atribua o acesso de logon único ao grupo.

   Para obter instruções, consulte Adicionar grupos no Guia do usuário do Centro de Identidade do AWS IAM .

- Abra o AWS IoT console

Se você já tem um usuário Conta da AWS e um usuário para si mesmo, pode usá-los e seguir em frente. the section called “Abra o AWS IoT console”

Abra o AWS IoT console

A maioria dos tópicos orientados ao console nesta seção começa no console. AWS IoT Se você ainda não estiver conectado ao seu Conta da AWS, entre, abra o AWS IoT console e vá para a próxima seção para continuar começando AWS IoT.

Tutorial interativo

O tutorial interativo exibe os componentes de uma solução simples de IoT baseada no AWS IoT. O tutorial mostra como os dispositivos de IoT interagem com AWS IoT Core os serviços. Este tópico fornece uma prévia do tutorial AWS IoT Core interativo.

> **Note**
>
> As imagens do console incluem animações que não estão incluídas nas imagens deste tutorial.

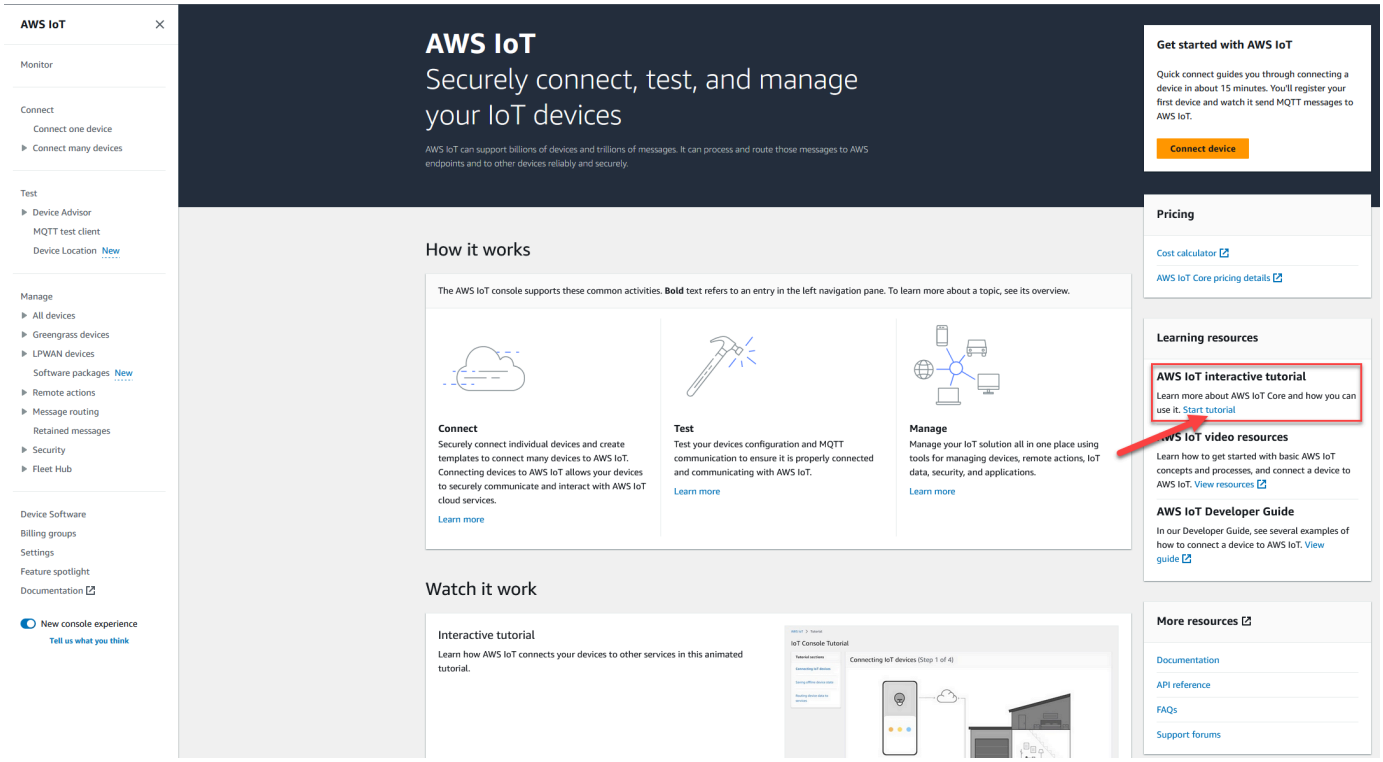
Para executar a demonstração, você deve primeiro the section called “Configurar Conta da AWS”. O tutorial, no entanto, não requer nenhum AWS IoT recurso, software adicional ou qualquer codificação.

Espere passar cerca de 5 a 10 minutos para a conclusão desta demonstração. Reservar 10 minutos permitirá mais tempo para compreender todas as etapas.

Para executar o tutorial AWS IoT Core interativo

1. Abra a página AWS IoT inicial no AWS IoT console.

   Na página inicial do AWS IoT , no painel da janela Recursos de aprendizagem, selecione Iniciar tutorial.



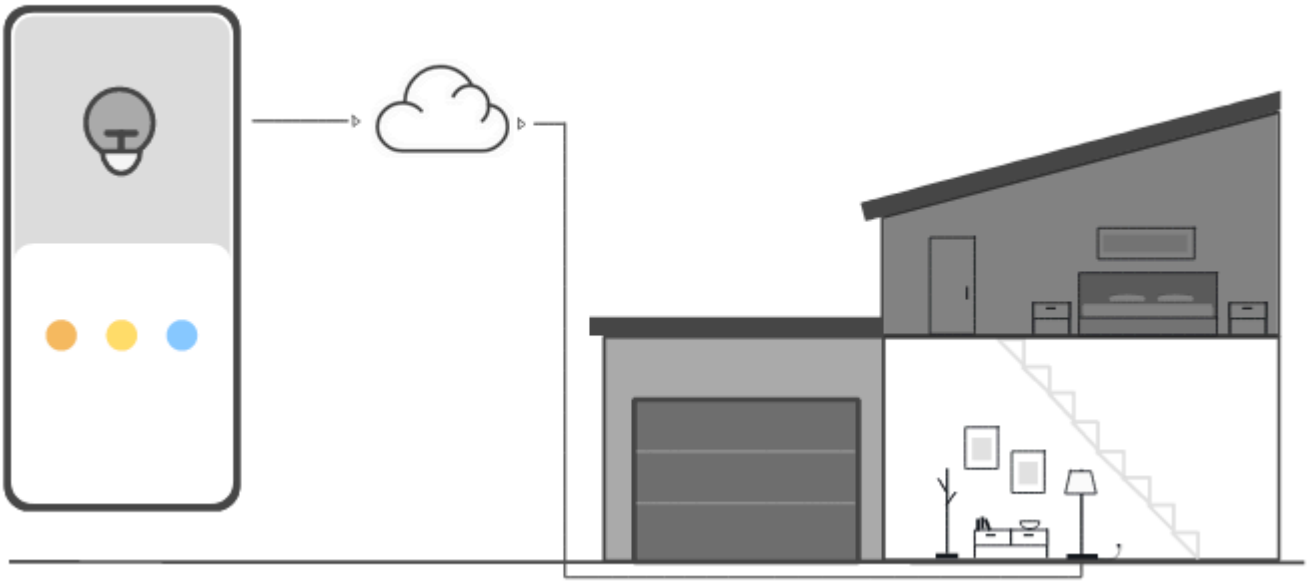
2. Na página Tutorial do console do AWS IoT , examine as seções do tutorial e selecione a seção Iniciar quando estiver pronto para prosseguir.

As seções a seguir descrevem como o tutorial do AWS IoT console apresenta esses AWS IoT Core recursos:

- Conectar dispositivos de IoT
- Salvar o estado de um dispositivo offline
- Rotear dados do dispositivo para serviços

Conectar dispositivos de IoT

Saiba como os dispositivos de IoT se comunicam com. AWS IoT Core

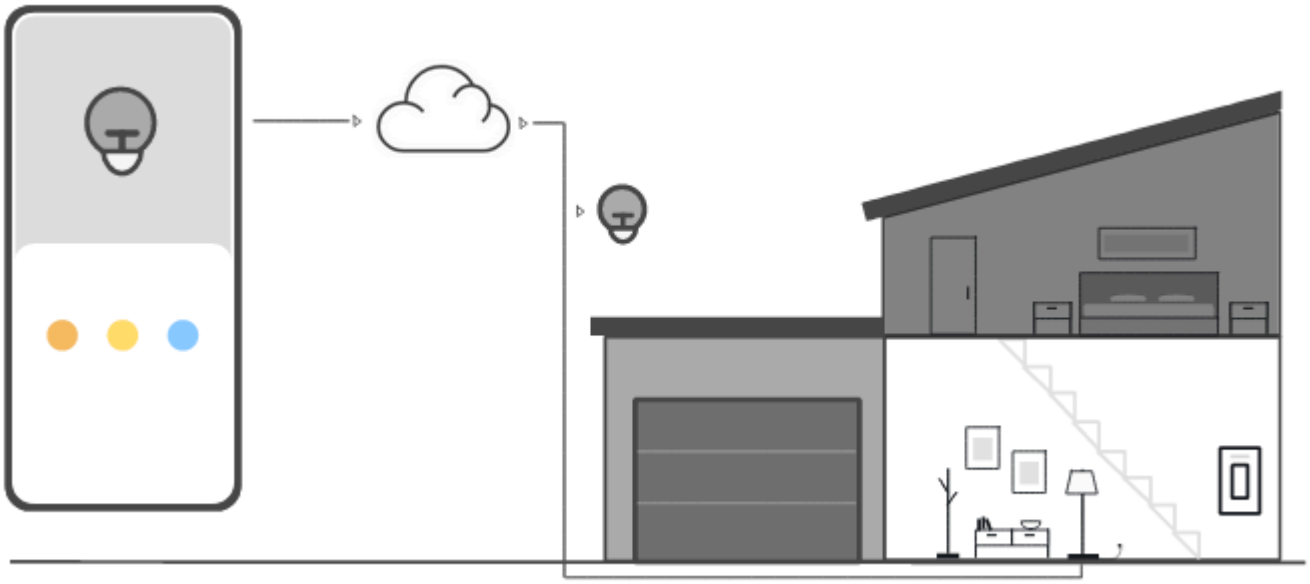


A animação desta etapa exibe como dois dispositivos, o dispositivo de controle à esquerda e uma lâmpada inteligente na casa à direita, se conectam e se comunicam com o AWS IoT Core na nuvem. A animação mostra os dispositivos se comunicando AWS IoT Core e reagindo às mensagens que recebem.

Para obter mais informações sobre como conectar dispositivos a AWS IoT Core, consulteConecte-se a AWS IoT Core.

Salvar o estado de um dispositivo offline

Saiba como AWS IoT Core salva o estado do dispositivo enquanto um dispositivo ou aplicativo está off-line.



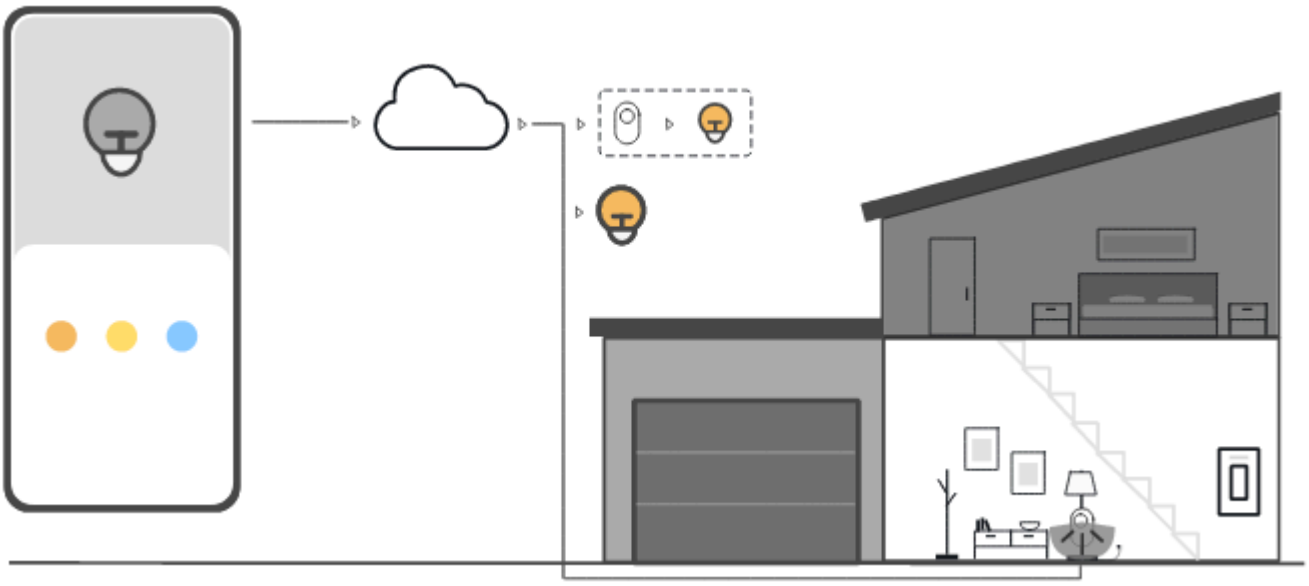
A animação desta etapa mostra como o serviço Device Shadow AWS IoT Core salva as informações de estado do dispositivo para o dispositivo de controle e a lâmpada inteligente. Enquanto a lâmpada inteligente está offline, a sombra do dispositivo salva os comandos do dispositivo de controle.

Quando a lâmpada inteligente se reconecta AWS IoT Core, ela recupera esses comandos. Quando o dispositivo de controle está offline, a sombra do dispositivo salva as informações de estado da lâmpada inteligente. Quando o dispositivo de controle se reconecta, ele resgata o estado atual da lâmpada inteligente para atualizar sua tela.

Para acessar mais informações sobre as sombras de dispositivos, consulte AWS IoT Serviço Device Shadow.

Rotear dados do dispositivo para serviços

Saiba como AWS IoT Core envia o estado do dispositivo para outros AWS serviços.



A animação nesta etapa mostra como AWS IoT Core envia dados dos dispositivos para outros AWS serviços usando AWS IoT regras. AWS IoT as regras assinam mensagens específicas dos dispositivos, interpretam os dados nessas mensagens e encaminham os dados interpretados para outros serviços. Neste exemplo, uma AWS IoT regra interpreta dados de um sensor de movimento e envia comandos para um Device Shadow, que os envia para a lâmpada inteligente. Assim como no exemplo anterior, a sombra do dispositivo armazena as informações de estado do dispositivo para o dispositivo de controle.

Para obter mais informações sobre AWS IoT regras, consulteRegras para AWS IoT.

Experimente o tutorial de conexão AWS IoT Core rápida

Neste tutorial, você criará seu primeiro objeto, conectará um dispositivo a ele e observará como ele envia mensagens MQTT.

Você deve levar de 15 a 20 minutos para concluir este tutorial.

Este tutorial é ideal para pessoas que desejam começar rapidamente AWS IoT para ver como ele funciona em um cenário limitado. Caso esteja procurando por um exemplo que o ajude a começar a explorar mais atributos e serviços, consulte Explore AWS IoT Core em tutoriais práticos.

Neste tutorial, você baixará e executará um software em um dispositivo que se conecta a um recurso físico AWS IoT Core como parte de uma solução de IoT muito pequena. O dispositivo pode ser um dispositivo de IoT, como um Raspberry Pi, como também pode ser um computador que execute

Linux, OS e OSX ou Windows. Se você deseja conectar um dispositivo WAN (LoRaWAN) de longo alcance AWS IoT, consulte o tutorial >Conectando dispositivos e gateways AWS IoT Core à LoRa WAN.

Caso o seu dispositivo seja compatível com um navegador que execute o console do AWS IoT, recomendamos que você conclua este tutorial nesse dispositivo.

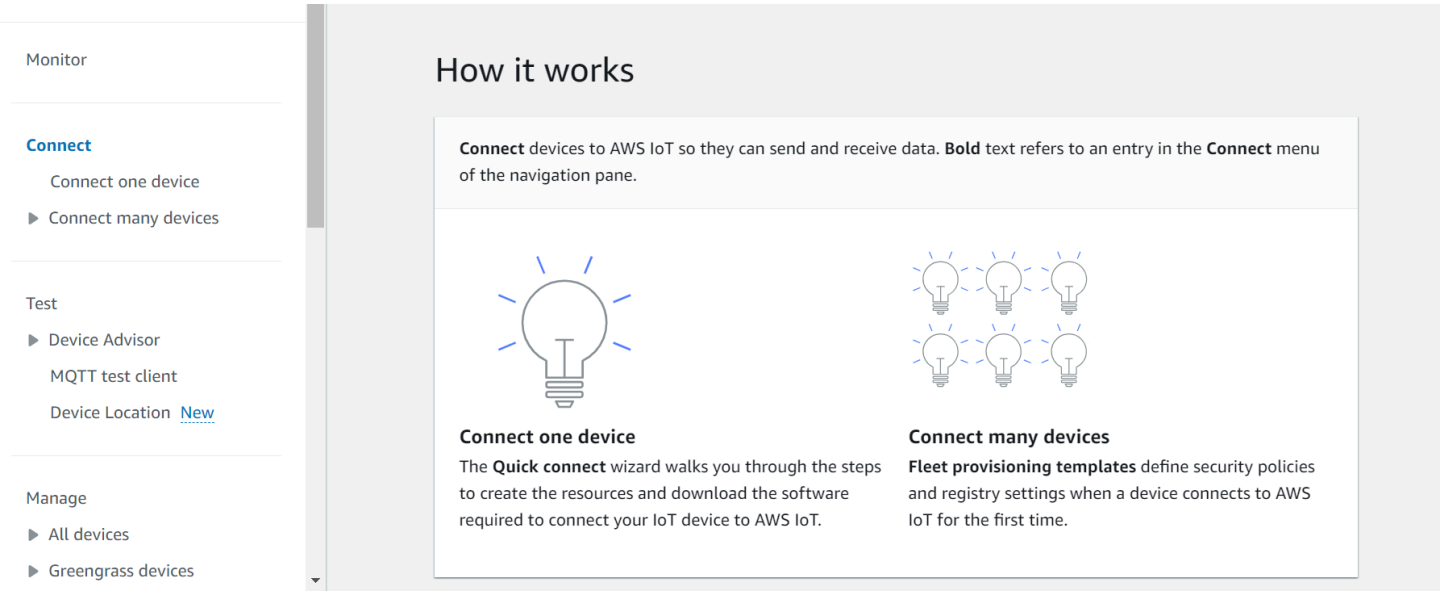
> **Note**
>
> Se o seu dispositivo não tiver um navegador compatível, conclua o tutorial em um computador. Quando o procedimento solicitar que você baixe o arquivo, baixe-o em seu computador e, depois, transfira o arquivo baixado para seu dispositivo através do Secure Copy (SCP) ou de um processo similar.

O tutorial requer que o seu dispositivo de IoT se comunique com a porta 8443 no endpoint de dados de dispositivo da sua Conta da AWS. Para testar se ele consegue acessar essa porta, conclua os procedimentos de Testar a conectividade com o endpoint de dados de um dispositivo.

Etapa 1. Inicie o tutorial

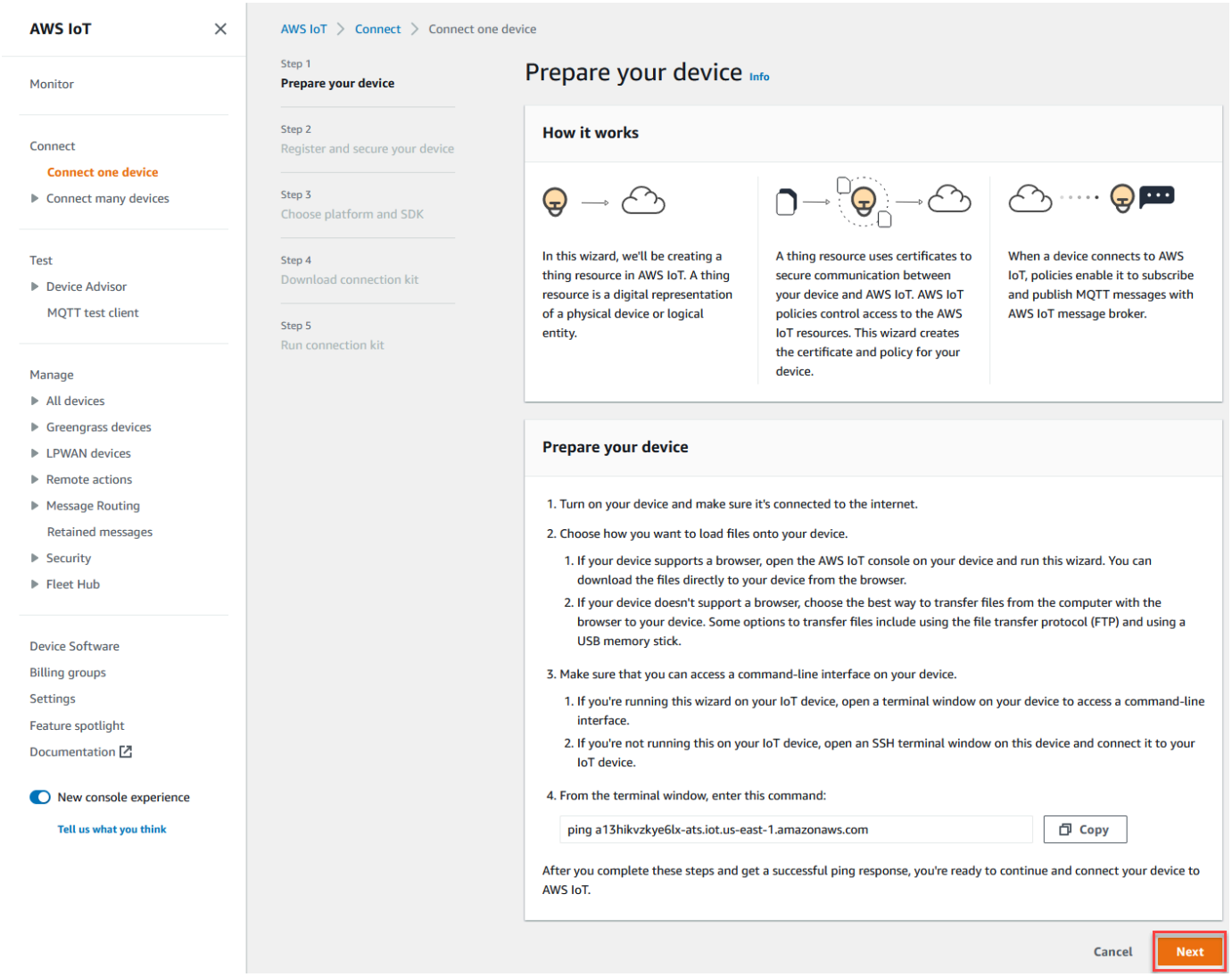
Se possível, realize este procedimento em seu dispositivo; caso contrário, esteja pronto para transferir um arquivo para seu dispositivo mais adiante neste procedimento.

Para iniciar o tutorial, inicie uma sessão no console do AWS IoT. Na página inicial do AWS IoT console, à esquerda, escolha Conectar e, em seguida, escolha Conectar um dispositivo.



Etapa 2. Criar um objeto

1. Na seção Preparar seu dispositivo, siga as instruções na tela para preparar seu dispositivo para conexão com o AWS IoT.



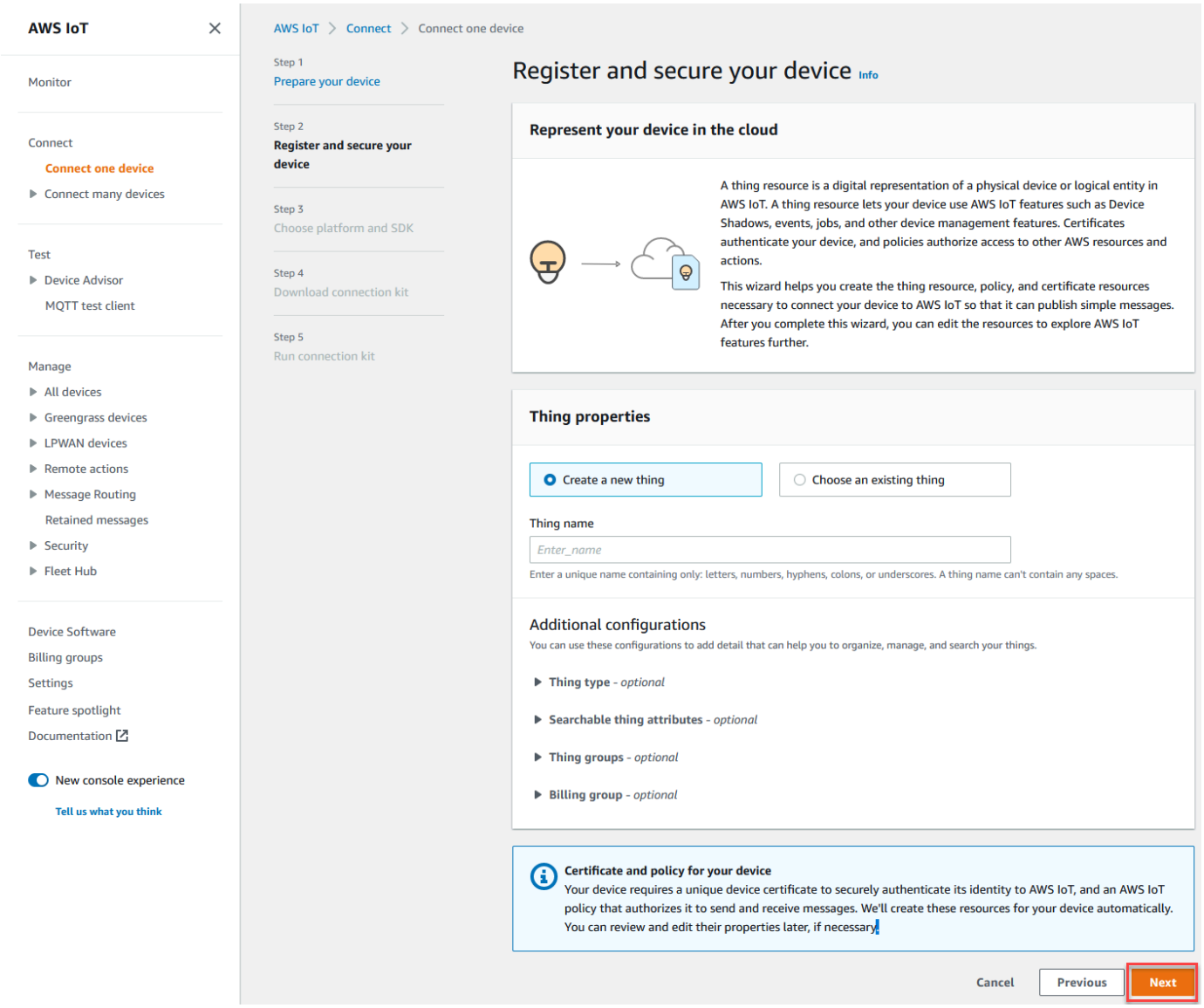
2. Na seção Registrar e proteger seu dispositivo, selecione Criar um objeto nova ou Selecionar um objeto existente. No campo Nome do objeto, digite o nome do objeto. O nome do objeto usado no exemplo é **`TutorialTestThing`**

> **Important**
>
> Verifique o nome de objeto antes de continuar.

> Não é possível alterar o nome de um objeto depois da criação do objeto. Se você quiser alterar o nome de um objeto, será preciso criar um novo objeto com o nome correto e, em seguida, excluir o objeto com o nome incorreto.

Na seção Configurações adicionais, personalize ainda mais seu recurso de objeto com as configurações opcionais listadas.

Depois de fornecer um nome ao objeto e selecionar as configurações adicionais, selecione Próximo.

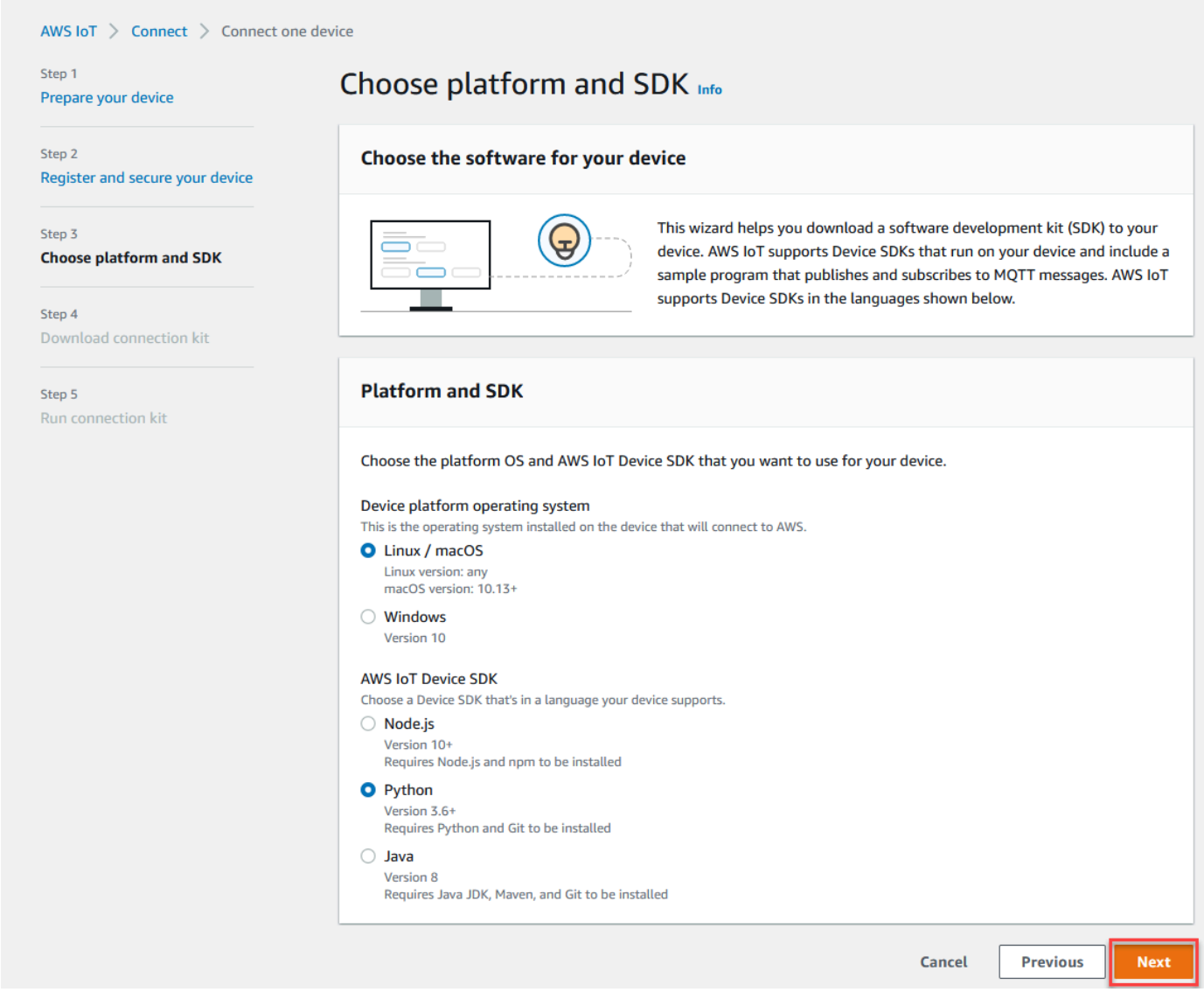


3. Na seção Escolher plataforma e SDK, escolha a plataforma e o idioma do SDK do AWS IoT dispositivo que você deseja usar. Este exemplo usa a Linux/OSX plataforma e o SDK do Python.

Assegure-se de ter o python3 e o pip3 instalados no dispositivo de destino antes de avançar para a próxima etapa.

> **Note**
>
> Não deixe de verificar a lista de pré-requisitos de software exigidos pelo SDK escolhido na parte inferior da página do console.
> É preciso ter o software necessário instalado no computador de destino antes de prosseguir para a próxima etapa.

Depois de escolher a plataforma e linguagem do SDK do dispositivo, selecione Próximo.

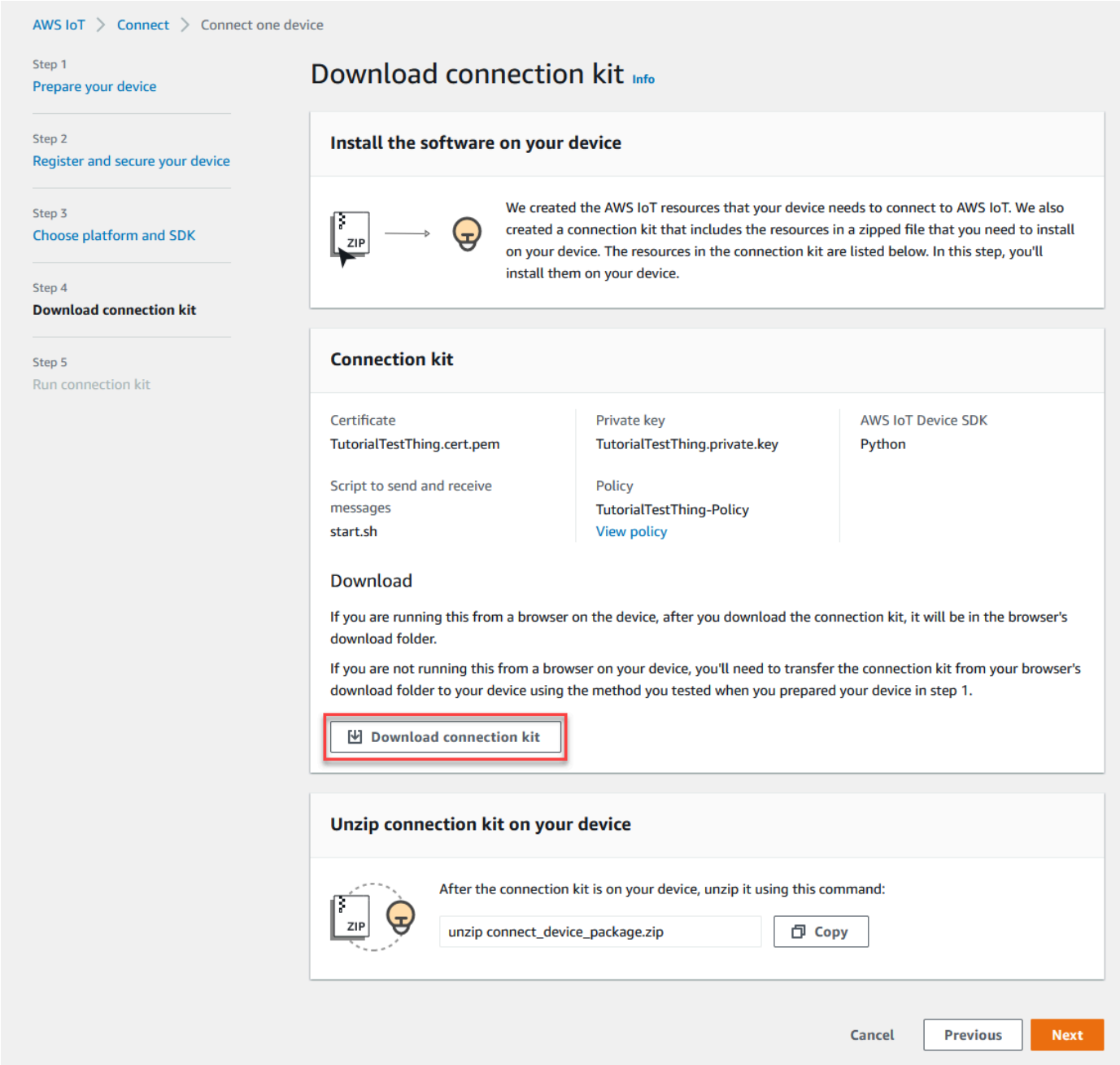


Etapa 3. Baixe os arquivos no seu dispositivo

Essa página aparece após AWS IoT a criação do kit de conexão, que inclui os seguintes arquivos e recursos necessários para seu dispositivo:

- Os arquivos de certificado do objeto, para autenticar o dispositivo
- Um recurso de política, para autorizar seu objeto a interagir com o AWS IoT
- O script para baixar o AWS Device SDK e executar o programa de amostra em seu dispositivo

1. Quando estiver pronto para continuar, selecione o botão Baixar kit de conexão para baixar o kit de conexão para a plataforma selecionada anteriormente.



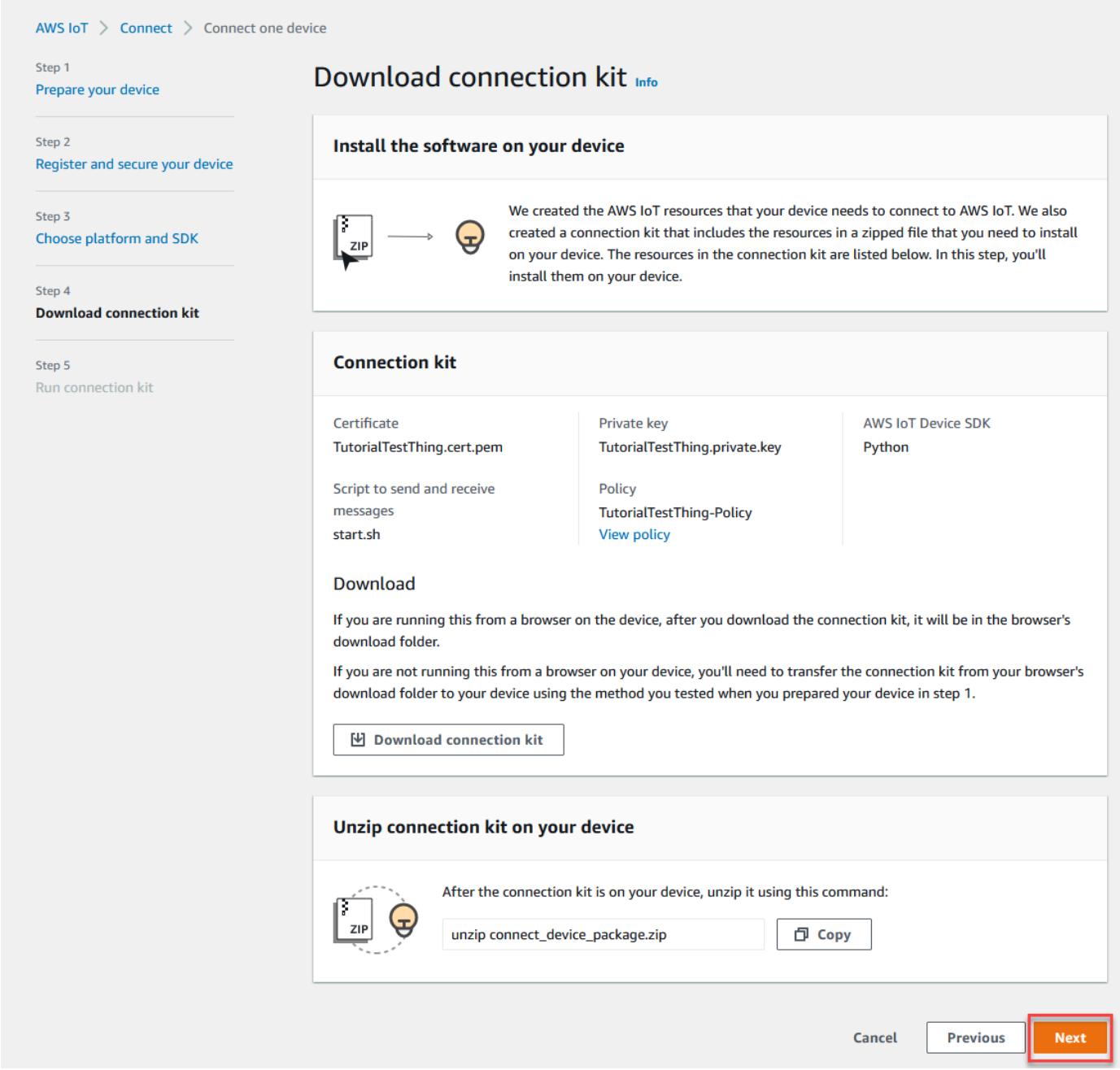
2. Se estiver executando este procedimento no seu dispositivo, salve o arquivo do kit de conexão em um diretório a partir do qual você consiga executar comandos de linha de comando.

   Se não estiver executando este procedimento no seu dispositivo, salve o arquivo do kit de conexão em um diretório local e, em seguida, transfira o arquivo para seu dispositivo.

3. Na seção Descompactar o kit de conexão no seu dispositivo, digite unzip connect_device_package.zip no diretório onde estão os arquivos do kit de conexão.

Se você estiver usando uma janela de PowerShell comando do Windows e o unzip comando não funcionarexpand-archive, unzip substitua por e tente usar a linha de comando novamente.

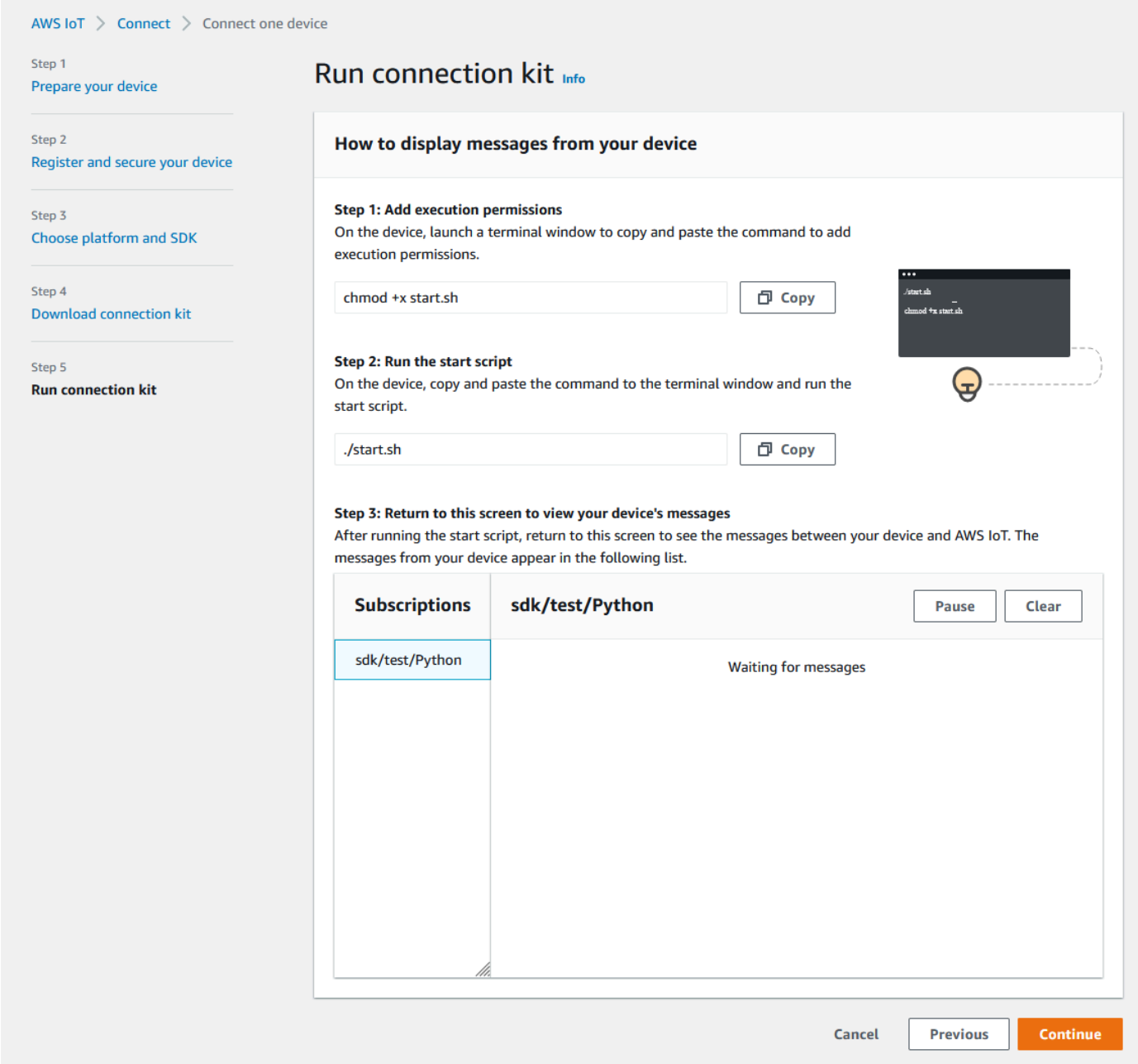
4. Depois de obter o arquivo do kit de conexão em seu dispositivo, prossiga com o tutorial selecionando Próximo.



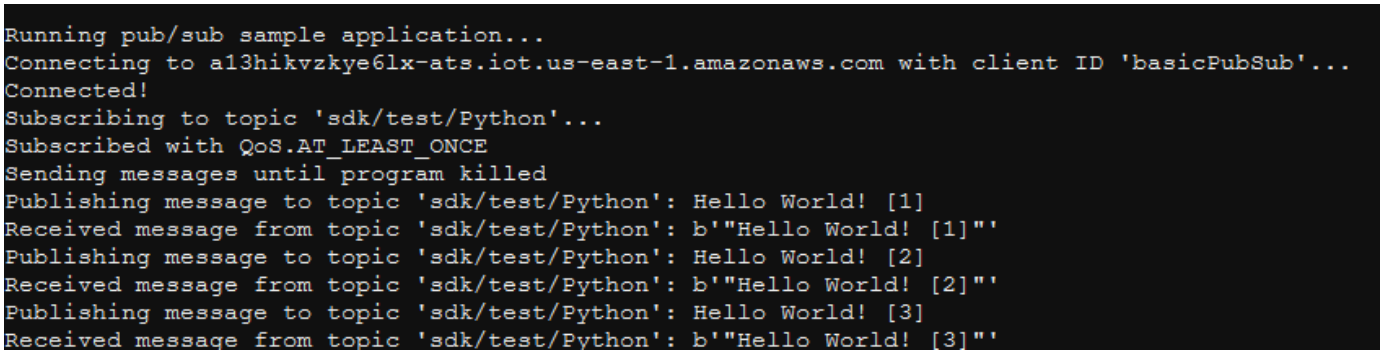
Etapa 4: Execute a amostra

Este procedimento deve ser realizado em um terminal ou janela de comando no seu dispositivo enquanto você segue as instruções exibidas no console. Os comandos do console são orientados para o sistema operacional selecionado em the section called “Etapa 2. Criar um objeto”. As mostradas aqui são para os sistemas Linux/OSX operacionais.

1. Em um terminal ou janela de comando em seu dispositivo, no diretório com o arquivo do kit de conexão, execute as etapas mostradas no AWS IoT console.



2. Após inserir o comando da Etapa 2 no console, você deve poder observar uma saída no terminal ou na janela de comando do dispositivo semelhante à seguinte. Essa saída é oriunda das mensagens que o programa está enviando e das quais recebe de volta do AWS IoT Core.



   Enquanto o programa de exemplo estiver em execução, a mensagem de teste `Hello World!` também será exibida. A mensagem de teste é exibida no terminal ou na janela de comando do seu dispositivo.

   > **Note**
   >
   > Para acessar mais informações sobre a assinatura e publicação de tópicos, consulte o código de exemplo do seu SDK escolhido.

3. Para executar o programa de exemplo novamente, repita os comandos da Etapa 2 no console deste procedimento.

4. (Opcional) Se você quiser ver as mensagens do seu cliente de IoT no AWS IoT console, abra o cliente de teste MQTT na página Teste do console. AWS IoT Se você selecionou o SDK de Python, no cliente de teste MQTT, em Filtro de tópicos, insira o tópico, como por exemplo, **`sdk/test/python`**, para assinar as mensagens do seu dispositivo. Os filtros de tópicos diferenciam maiúsculas de minúsculas e dependem da linguagem de programação do SDK selecionada na Etapa 1. Para acessar mais informações sobre a assinatura e publicação de tópicos, consulte o exemplo de código do seu SDK escolhido.

5. Depois de assinar no tópico de teste, execute ./start.sh no seu dispositivo. Para obter mais informações, consulte the section called “Exibir mensagens MQTT com o cliente AWS IoT MQTT”.

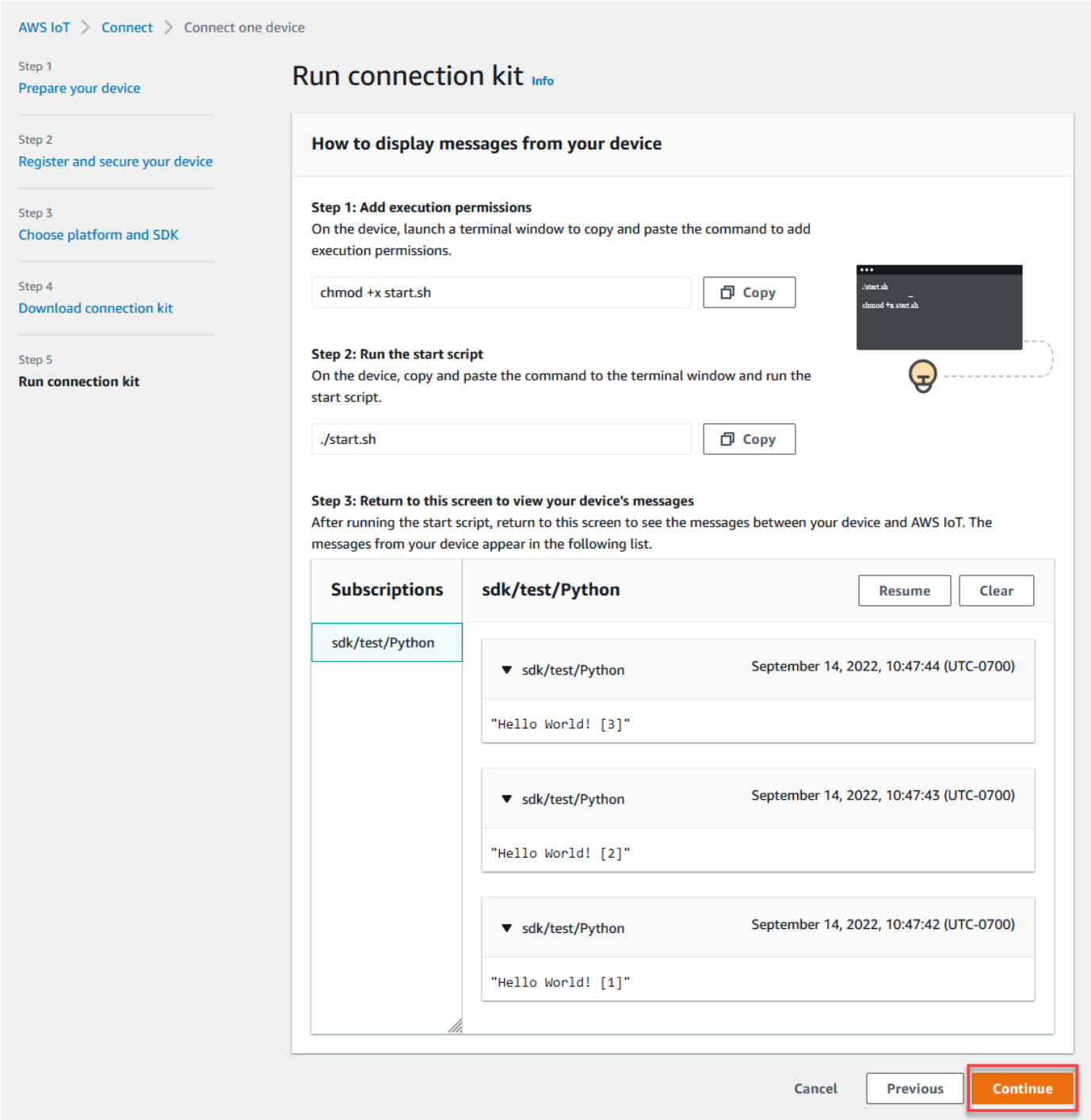
   Depois de executar ./start.sh, aparecerão mensagens no cliente MQTT, semelhantes à seguinte:

```
{
  "message": "Hello World!" [1]
```

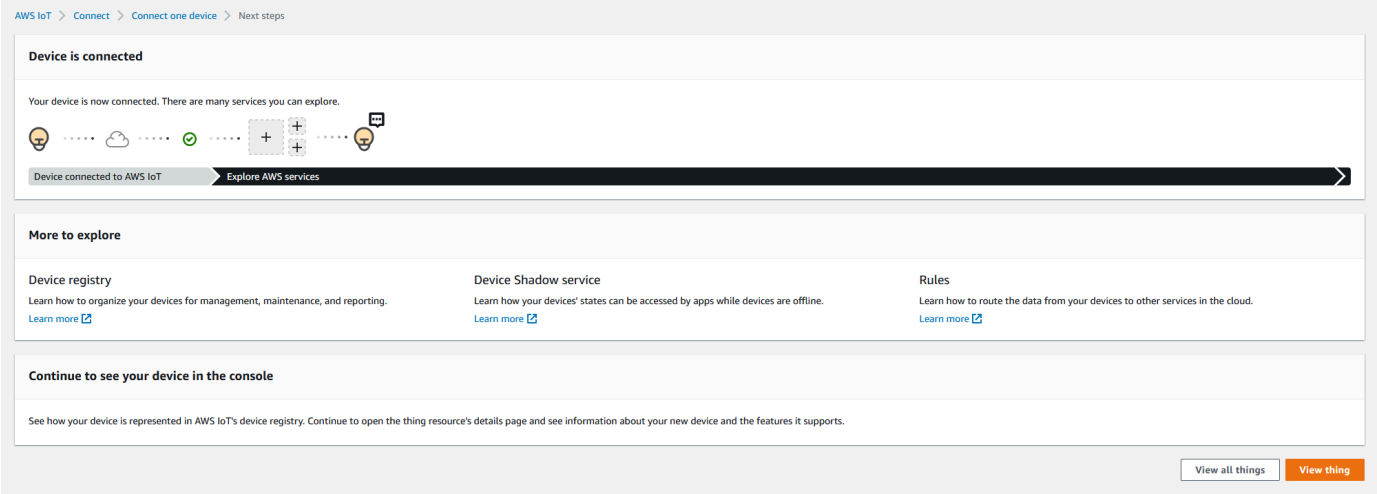
```
}
```

O número de `sequence` incluído entre `[]` aumenta em um cada vez que uma nova mensagem `Hello World!` é recebida e é interrompido quando você encerra o programa.

6. Para concluir o tutorial e ver um resumo, no AWS IoT console, escolha Continuar.



7. Um resumo do seu tutorial de conexão AWS IoT rápida aparecerá agora.



Etapa 5. Explore mais

Aqui estão algumas ideias para explorar AWS IoT mais depois de concluir o início rápido.

- Visualize mensagens MQTT no cliente de teste MQTT

  No console da AWS IoT, é possível abrir o cliente do MQTT na página Teste do console da AWS IoT . No cliente de teste MQTT, assine **#** e, em seguida, execute o programa ./start.sh no seu dispositivo conforme descrito na etapa anterior. Para obter mais informações, consulte the section called “Exibir mensagens MQTT com o cliente AWS IoT MQTT”.

- Execute testes em seus dispositivos com o Device Advisor

  Use o Device Advisor para testar se seus dispositivos podem se conectar e interagir com, de forma segura e confiável. AWS IoT

- the section called “Tutorial interativo”

  Para iniciar o tutorial interativo, na página Learn do AWS IoT console, no quadro Veja como AWS IoT funciona, escolha Iniciar o tutorial.

- Prepare-se para explorar mais tutoriais

  Esse início rápido fornece apenas uma amostra de AWS IoT. Se você quiser explorar AWS IoT mais e aprender sobre os recursos que a tornam uma poderosa plataforma de soluções de IoT, comece a preparar sua plataforma de desenvolvimento. Explore AWS IoT Core em tutoriais práticos

Testar a conectividade com o endpoint de dados de um dispositivo

Este tópico descreve como testar a conexão de um dispositivo com o endpoint de dados do dispositivo da sua conta, o endpoint que seus dispositivos de IoT usam para se conectar ao AWS IoT.

Execute estes procedimentos no dispositivo que você deseja testar ou com uma sessão de terminal SSH conectada ao dispositivo que você deseja testar.

Para testar a conectividade de um dispositivo com o endpoint de dados de um dispositivo.

- Localize o endpoint de dados do seu dispositivo
- Teste a conexão rapidamente
- Obtenha o aplicativo para testar a conexão com o endpoint de dados e a porta do seu dispositivo
- Teste a conexão com o endpoint de dados e a porta do seu dispositivo

Localize o endpoint de dados do seu dispositivo

Este procedimento explica como encontrar o endpoint de dados do dispositivo no console de AWS IoT para testar a conexão com seu dispositivo de IoT.

Para localizar o endpoint de dados do seu dispositivo

1. No console da AWS IoT, na seção Conectar, acesse Configurações de domínio.
2. Na página Configurações de domínio, acesse o contêiner Configurações de domínio e copie o nome do domínio. O valor do seu endpoint é exclusivo para você Conta da AWS e é semelhante a este exemplo:`a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com`.
3. Salve o endpoint de dados do seu dispositivo para usá-lo nos procedimentos seguintes.

Teste a conexão rapidamente

Este procedimento testa a conectividade geral com o endpoint de dados do seu dispositivo, mas não testa a porta específica a serem usadas pelos seus dispositivos. Este teste usa um programa comum e geralmente é suficiente para descobrir se os dispositivos podem se conectar ao AWS IoT.

Se quiser testar a conectividade com a porta específica que seus dispositivos usarão, pule este procedimento e avance para Obtenha o aplicativo para testar a conexão com o endpoint de dados e a porta do seu dispositivo.

Para testar o endpoint de dados do dispositivo rapidamente

1. Em uma janela de terminal ou linha de comando no seu dispositivo, substitua o endpoint de dados do dispositivo de exemplo (*a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com*) pelo endpoint de dados do dispositivo da sua conta e, depois, digite este comando.

   Linux

   ```
   ping -c 5 a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
   ```

   Windows

   ```
   ping -n 5 a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
   ```

2. Se `ping` exibir uma saída semelhante à seguinte, ele se conectou ao endpoint de dados do seu dispositivo com êxito. Embora não tenha se comunicado AWS IoT diretamente, ele encontrou o servidor e é provável que ele AWS IoT esteja disponível por meio desse endpoint.

   ```
   PING a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com (xx.xx.xxx.xxx) 56(84) bytes of
    data.
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=1 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=2 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=3 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=4 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=5 ttl=231 time=127 ms
   ```

   Se você estiver satisfeito com esse resultado, pode interromper o teste neste ponto.

   Se você quiser testar a conectividade com a porta específica usada por AWS IoT, avance para Obtenha o aplicativo para testar a conexão com o endpoint de dados e a porta do seu dispositivo.

3. Se `ping` não retornou uma saída bem-sucedida, verifique o valor do endpoint para se assegurar de que você tenha o endpoint correto e verifique a conexão do dispositivo com a Internet.

Obtenha o aplicativo para testar a conexão com o endpoint de dados e a porta do seu dispositivo

Um teste de conectividade mais minucioso pode ser realizado com `nmap`. Este procedimento realiza testes para averiguar se `nmap` está instalado no seu dispositivo.

Para buscar **`nmap`** no dispositivo

1. Em uma janela de terminal ou linha de comando no dispositivo que você quer testar, insira este comando para averiguar se `nmap` está instalado.

   ```
   nmap --version
   ```

2. Se uma saída semelhante à seguinte for exibida, `nmap` está instalado e você pode prosseguir para the section called “Teste a conexão com o endpoint de dados e a porta do seu dispositivo”.

   ```
   Nmap version 6.40 ( http://nmap.org )
   Platform: x86_64-koji-linux-gnu
   Compiled with: nmap-liblua-5.2.2 openssl-1.0.2k libpcre-8.32 libpcap-1.5.3 nmap-
   libdnet-1.12 ipv6
   Compiled without:
   Available nsock engines: epoll poll select
   ```

3. Caso não veja uma resposta semelhante à mostrada na etapa anterior, será preciso instalar `nmap` no dispositivo. Escolha o procedimento com base no sistema operacional do dispositivo.

Linux

Para este procedimento, é preciso que você tenha permissão para instalar softwares no computador.

Para instalar o nmap em seu computador Linux

1. Em uma janela de terminal ou linha de comando no dispositivo, insira o comando correspondente à versão do Linux em execução.

   a. Debian ou Ubuntu:

      ```
      sudo apt install nmap
      ```

   b. CentOS ou RHEL:

```
sudo yum install nmap
```

2. Teste a instalação com o seguinte comando:

```
nmap --version
```

3. Se uma saída semelhante à seguinte for exibida, nmap está instalado e você pode prosseguir para the section called “Teste a conexão com o endpoint de dados e a porta do seu dispositivo”.

```
Nmap version 6.40 ( http://nmap.org )
Platform: x86_64-koji-linux-gnu
Compiled with: nmap-liblua-5.2.2 openssl-1.0.2k libpcre-8.32 libpcap-1.5.3 nmap-
libdnet-1.12 ipv6
Compiled without:
Available nsock engines: epoll poll select
```

macOS

Para este procedimento, é preciso que você tenha permissão para instalar softwares no computador.

Para instalar o nmap em seu computador macOS

1. Em um navegador, abra https://nmap.org/download#macosx e baixe o instalador estável mais recente.

   Quando solicitado, selecione Abrir com DiskImageInstaller.

2. Na janela de instalação, transfira o pacote para a pasta Aplicações.
3. No Localizador, localize o pacote `nmap-xxxx-mpkg` na pasta Aplicações. Pressione Ctrl-click no pacote e selecione Abrir para abrir o pacote.
4. Examine a caixa de diálogo de segurança. Se estiver pronto para instalar nmap, selecione Abrir para instalar nmap.
5. No Terminal, teste a instalação com o seguinte comando:

```
nmap --version
```

6. Se uma saída semelhante à seguinte for exibida, nmap está instalado e você pode prosseguir para the section called “Teste a conexão com o endpoint de dados e a porta do seu dispositivo”.

```
Nmap version 7.92 ( https://nmap.org )
Platform: x86_64-apple-darwin17.7.0
Compiled with: nmap-liblua-5.3.5 openssl-1.1.1k nmap-libssh2-1.9.0 libz-1.2.11
 nmap-libpcre-7.6 nmap-libpcap-1.9.1 nmap-libdnet-1.12 ipv6 Compiled without:
Available nsock engines: kqueue poll select
```

Windows

Para este procedimento, é preciso que você tenha permissão para instalar softwares no computador.

Para instalar o nmap em seu computador Windows

1. Em um navegador, abra https://nmap.org/download#windows e baixe a versão estável mais recente do programa de instalação.

   Se solicitado, selecione Salvar arquivo. Quando o arquivo for baixado, abra-o na pasta de downloads.
2. Quando o download do arquivo de configuração for concluído, abra o nmap-xxxx-setup.exe baixado para instalar o aplicativo.
3. Aceite as configurações padrão conforme o programa é instalado.

   O aplicativo Npcap não é necessário para este teste. É possível desmarcar essa opção se não quiser instalá-la.
4. Em Command, teste a instalação com o seguinte comando:

```
nmap --version
```

5. Se uma saída semelhante à seguinte for exibida, nmap está instalado e você pode prosseguir para the section called “Teste a conexão com o endpoint de dados e a porta do seu dispositivo”.

```
Nmap version 7.92 ( https://nmap.org )
Platform: i686-pc-windows-windows
```

```
Compiled with: nmap-liblua-5.3.5 openssl-1.1.1k nmap-libssh2-1.9.0 nmap-
libz-1.2.11 nmap-libpcre-7.6 Npcap-1.50 nmap-libdnet-1.12 ipv6
Compiled without:
Available nsock engines: iocp poll select
```

Teste a conexão com o endpoint de dados e a porta do seu dispositivo

Esse procedimento testa a conexão do seu dispositivo de IoT com o endpoint de dados do dispositivo usando a porta selecionada.

Para testar o endpoint de dados e a porta do seu dispositivo

1. Em uma janela de terminal ou linha de comando no seu dispositivo, substitua o endpoint de dados do dispositivo de exemplo (*a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com*) pelo endpoint de dados do dispositivo da sua conta e, depois, digite este comando.

   ```
   nmap -p 8443 a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
   ```

2. Se nmap exibir uma saída semelhante à seguinte, nmap obteve êxito em se conectar ao endpoint de dados do seu dispositivo na porta selecionada.

   ```
   Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-18 16:23 Pacific Standard Time
   Nmap scan report for a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
    (xx.xxx.147.160)
   Host is up (0.036s latency).
   Other addresses for a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com (not scanned):
    xx.xxx.134.144 xx.xxx.55.139 xx.xxx.110.235 xx.xxx.174.233 xx.xxx.74.65
    xx.xxx.122.179 xx.xxx.127.126
   rDNS record for xx.xxx.147.160: ec2-EXAMPLE-160.eu-west-1.compute.amazonaws.com

   PORT     STATE SERVICE
   8443/tcp open  https-alt
   MAC Address: 00:11:22:33:44:55 (Cimsys)

   Nmap done: 1 IP address (1 host up) scanned in 0.91 seconds
   ```

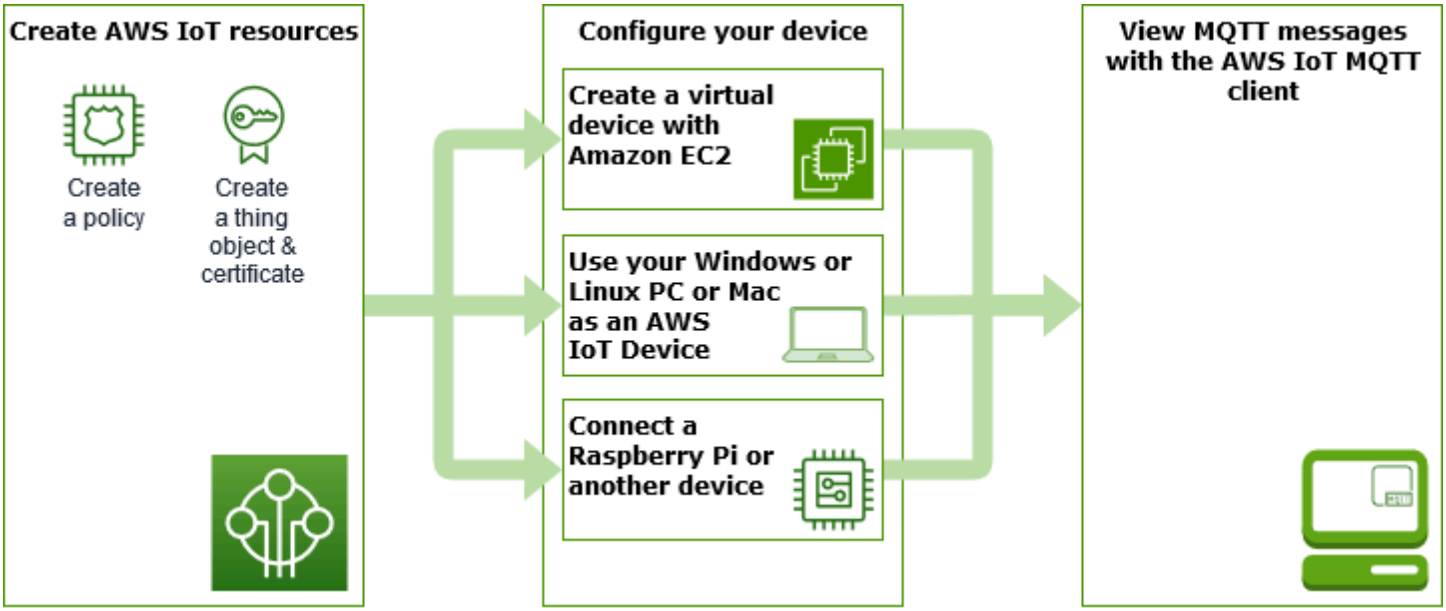
3. Se nmap não retornou uma saída bem-sucedida, verifique o valor do endpoint para se assegurar de que você tenha o endpoint correto e verifique a conexão do seu dispositivo com a Internet.

É possível testar outras portas no endpoint de dados do dispositivo, como a porta 443, a porta HTTPS primária, substituindo a porta usada na etapa 1, **8443**, pela porta que você quer testar.

Explore AWS IoT Core em tutoriais práticos

Neste tutorial, você instalará o software e criará os AWS IoT recursos necessários para conectar um dispositivo para AWS IoT Core que ele possa enviar e receber mensagens MQTT com AWS IoT Core. Você verá as mensagens no cliente MQTT no AWS IoT console.

Você deve levar de 20 a 30 minutos para concluir este tutorial. Caso esteja usando um dispositivo de IoT ou um Raspberry Pi, este tutorial pode levar mais tempo se, por exemplo, for preciso instalar o sistema operacional e configurar o dispositivo.



Este tutorial é ideal para desenvolvedores que desejam começar AWS IoT Core a explorar recursos mais avançados, como o mecanismo de regras e as sombras. Este tutorial prepara você para continuar aprendendo sobre AWS IoT Core e como ele interage com outros AWS serviços, explicando as etapas com mais detalhes do que o tutorial de início rápido. Se estiver procurando apenas uma experiência rápida, aos modos de Olá, mundo, teste Experimente o tutorial de conexão AWS IoT Core rápida.

Depois de configurar seu Conta da AWS e o AWS IoT console, você seguirá estas etapas para ver como conectar um dispositivo e fazer com que ele envie mensagens para AWS IoT Core.

Próximas etapas

- Escolha qual opção de dispositivo é a melhor para você

- the section called “Crie AWS IoT recursos” se não for criar um dispositivo virtual com o Amazon EC2
- the section called “Configurar o dispositivo”
- the section called “Exibir mensagens MQTT com o cliente AWS IoT MQTT”

Para obter mais informações sobre AWS IoT Core, consulte O que é AWS IoT Core?

Qual opção de dispositivo é melhor para você?

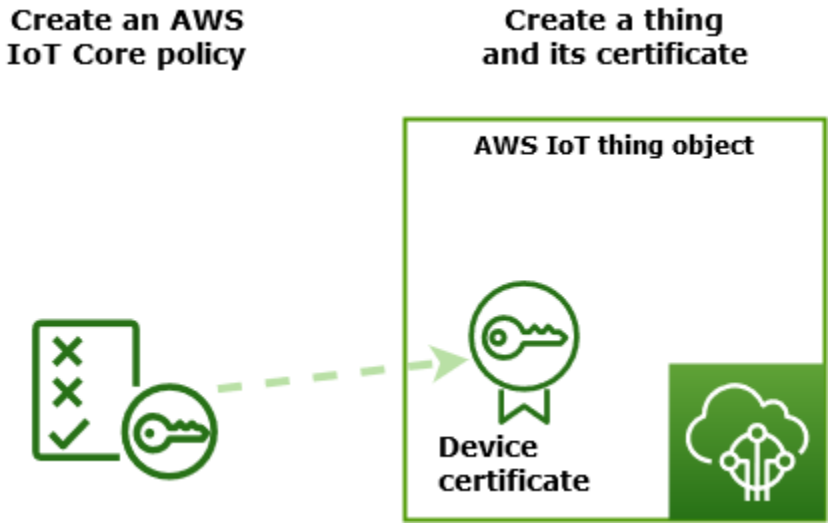
Se não estiver certo sobre qual opção escolher, use esta lista de vantagens e desvantagens de cada opção para ajudá-lo a decidir a melhor opção para você.

Opção	Esta pode ser uma boa opção se:	Esta pode ser uma boa opção se:
the section called “Criar um dispositivo virtual com o Amazon EC2”	• Você não tem seu próprio dispositivo para realizar o teste. • Você não deseja instalar nenhum software em seu sistema. • Você deseja testar usando um sistema operacional Linux.	• Você não se sente confortáv el usando comandos de linha de comando. • Você não quer incorrer em cobranças adicionais da AWS . • Você não deseja testar usando um sistema operacional Linux.
the section called “Use seu PC Windows ou Linux ou Mac como um AWS IoT dispositivo”	• Você não quer incorrer em cobranças adicionais da AWS . • Você não deseja configurar dispositivos adicionais.	• Você não deseja instalar softwares em seu computador pessoal. • Você deseja uma plataform a de teste mais represent ativa.
the section called “Conectar um Raspberry Pi ou outro dispositivo”	• Você quer testar AWS IoT com um dispositivo real. • Você já possui um dispositi vo para testes.	• Você não deseja adquirir ou configurar um dispositivo só para testá-lo.

Opção	Esta pode ser uma boa opção se:	Esta pode ser uma boa opção se:
	• Você tem experiência em integrar hardware em sistemas.	• Você quer testar da forma mais simples AWS IoT possível, por enquanto.

Crie AWS IoT recursos

Neste tutorial, você criará os AWS IoT recursos necessários para um dispositivo se conectar AWS IoT Core e trocar mensagens.



1. Crie um documento AWS IoT de política que autorizará seu dispositivo a interagir com os AWS IoT serviços.
2. Crie um objeto de coisa AWS IoT e seu certificado de dispositivo X.509 e, em seguida, anexe o documento de política. O objeto da coisa é a representação virtual do seu dispositivo no AWS IoT registro. O certificado autentica seu dispositivo em AWS IoT Core, e o documento de política autoriza seu dispositivo a interagir com. AWS IoT

> **Note**
>
> Se estiver planejando the section called “Criar um dispositivo virtual com o Amazon EC2”, pule esta página e avance para the section called “Configurar o dispositivo”. Você criará estes recursos quando criar suo objeto virtual.

Este tutorial usa o AWS IoT console para criar os AWS IoT recursos. Se o dispositivo for compatível com um navegador da Web, pode ser mais fácil realizar este procedimento no navegador da Web do dispositivo, pois você poderá baixar os arquivos do certificado diretamente para o dispositivo. Se executar esse procedimento em outro computador, será preciso copiar os arquivos do certificado para o dispositivo antes que eles possam ser usados pelo aplicativo de exemplo.

Crie uma AWS IoT política

Os dispositivos usam um certificado X.509 para se autenticar. AWS IoT Core O certificado tem AWS IoT políticas anexadas a ele. Essas políticas determinam quais operações do AWS IoT o dispositivo tem permissão para realizar, como assinatura ou publicação de tópicos do MQTT. Seu dispositivo apresenta seu certificado quando se conecta e envia mensagens para AWS IoT Core o.

Siga as etapas para criar uma política que permita que o dispositivo execute as operações do AWS IoT que são necessárias para executar o programa de exemplo. É preciso criar a política do AWS IoT antes que possa anexá-la ao certificado do dispositivo, que você criará depois.

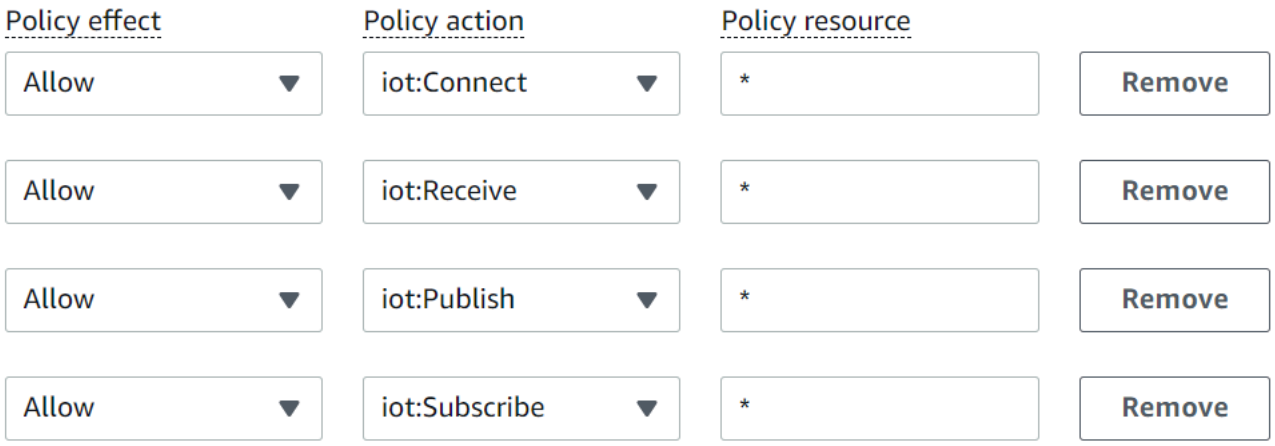
Para criar uma AWS IoT política

1. No console do AWS IoT, no menu esquerdo, selecione Segurança e, depois, selecione Políticas.
2. Na página Você ainda não possui uma política, selecione Criar uma política.

   Se a sua conta possuir políticas existentes, selecione Criar política.
3. Na página Criar uma política:

   1. Na seção Propriedades da política, no campo Nome da política, digite um nome para a política (como, por exemplo, **`My_Iot_Policy`**). Não use informações de identificação pessoal em nomes de políticas.
   2. Na seção Documento de política, crie as declarações de política que concedem ou negam o acesso dos recursos às operações do AWS IoT Core . Para criar uma declaração de política que permita que todos os clientes executem **`iot:Connect`**, siga as etapas a seguir:
      - No campo Efeito da política, selecione Permitir. Isso permite que os clientes que tenham essa política anexada ao certificado executem a ação listada no campo Ação da política.
      - No campo Ação da política, selecione uma ação da política, como **`iot:Connect`**. As ações da política são as ações para as quais o seu dispositivo precisa de permissão para realizar ao executar o programa de exemplo a partir do SDK do dispositivo.
      - No campo Recurso da política, digite um nome do recurso da Amazon (ARN) ou **`*`**. Um **`*`** para selecionar qualquer cliente (dispositivo).

Para criar as declarações de política para **`iot:Receive`**, **`iot:Publish`** e **`iot:Subscribe`**, selecione Adicionar nova declaração e repita as etapas.



> **Note**
>
> Neste início rápido, o caractere curinga (*) é usado para simplificação. Para maior segurança, restrinja quais clientes (dispositivos) podem se conectar e publicar mensagens especificando um ARN de cliente ao invés do caractere curinga como o recurso. O cliente ARNs segue este formato:`arn:aws:iot:`*`your-region`*`:`*`your-aws-account`*`:client/`*`my-client-id`*.
> Entretanto, é preciso primeiro criar o recurso (como um dispositivo cliente ou uma sombra) antes de atribuir o ARN a uma política. Para acessar mais informações, consulte Recursos de ação do AWS IoT Core.

4. Depois de inserir as informações da sua política, selecione Criar.

Para obter mais informações, consulte Como AWS IoT funciona com o IAM.

Criar um objeto

Os dispositivos conectados AWS IoT Core são representados por objetos de coisas no AWS IoT registro. Um objeto representa um dispositivo específico ou uma entidade lógica. Ele pode ser um dispositivo físico ou um sensor (por exemplo, uma lâmpada ou um interruptor de parede). Também pode ser uma entidade lógica, como uma instância de um aplicativo ou entidade física que não se conecta AWS IoT, mas está relacionada a outros dispositivos que o fazem (por exemplo, um carro com sensores de motor ou um painel de controle).

Para criar algo no AWS IoT console

1. No console do AWS IoT, no menu esquerdo, selecione Todos os dispositivos e, depois, selecione Objetos.
2. Na página Objetos, selecione Criar objetos.
3. Na página Criar objetos, selecione Criar uma único objeto e, depois, selecione Próximo.
4. Na página Especificar propriedades do objeto, em Nome do objeto, digite um nome para o objeto, como **`MyIotThing`**.

   Tenha cuidado ao escolher os nomes das objetos, pois não é possível alterar o nome de um objeto mais adiante.

   Para alterar o nome de um objeto, é necessário criar um objeto, fornecer o novo nome e, depois, excluir o objeto antiga.

   > **Note**
   >
   > Não use informações de identificação pessoal nos nomes de objetos. O nome do objeto pode surgir em comunicações e relatórios não criptografados.

5. Mantenha e restante dos campos desta página vazios. Escolha Próximo.
6. Na página Configurar certificado de dispositivo - opcional, selecione Gerar um novo certificado automaticamente (recomendado). Escolha Próximo.
7. Na página Anexar políticas ao certificado - opcional, escolha a política criada na seção anterior. Nessa seção, a política foi nomeada **`My_Iot_Policy`**. Selecione Criar grupo.
8. Na página Baixar certificados e chaves:

   1. Baixe cada um dos arquivos de certificado e chave e salve-os para uso posterior. Será preciso instalar esses arquivos no seu dispositivo.

      Ao salvar os arquivos de certificado, nomeie-os conforme a tabela a seguir. Estes são os nomes dos arquivos usados em exemplos posteriores.

      Nomes de arquivos de certificado

Arquivo	Caminho do arquivo
Chave privada	`private.pem.key`

Arquivo	Caminho do arquivo
Chave pública	(não utilizada nestes exemplos)
Certificado de dispositivo	`device.pem.crt`
Certificado CA raiz	`Amazon-root-CA-1.pem`

2. Para baixar o arquivo CA raiz dos arquivos, selecione o link Baixar do arquivo de certificado CA raiz correspondente ao tipo de endpoint de dados e pacote de criptografia sendo usado. Neste tutorial, selecione Baixar à direita da Chave RSA de 2048 bits: Amazon Root CA 1 e baixe o arquivo de certificado Chave RSA de 2048 bits: Amazon Root CA 1.

   > **Important**
   >
   > É preciso salvar os arquivos de certificado antes de sair da página. Após sair desta página no console, você não terá mais acesso aos arquivos de certificado.
   > Caso tenha se esquecido de baixar os arquivos de certificado criados nesta etapa, será preciso sair desta tela do console, acessar a lista de objetos no console, excluir o objeto criado e reiniciar o procedimento do início.

3. Escolha Concluído.

Após concluir esse procedimento, você deve ser capaz de encontrar o novo objeto na lista de objetos.

Configurar o dispositivo

Esta seção descreve como configurar seu dispositivo para se conectar ao AWS IoT Core. Se quiser começar, AWS IoT Core mas ainda não tiver um dispositivo, você pode criar um dispositivo virtual usando o Amazon EC2 ou usar seu PC Windows ou Mac como um dispositivo de IoT.

Selecione a melhor opção de dispositivo para você experimentar AWS IoT Core. Evidente, é possível testar todos eles, mas o faça com apenas um de cada vez. Se não tiver certeza sobre qual a melhor opção de dispositivo para você, leia sobre como decidir qual opção de dispositivo é a melhor e volte para esta página.

Opções do dispositivo

- Criar um dispositivo virtual com o Amazon EC2
- Use seu PC Windows ou Linux ou Mac como um AWS IoT dispositivo
- Conectar um Raspberry Pi ou outro dispositivo

Criar um dispositivo virtual com o Amazon EC2

Neste tutorial, você criará uma instância do Amazon EC2 para atuar como seu dispositivo virtual na nuvem.

Para concluir este tutorial, você precisa de um Conta da AWS. Se você não possuir uma, conclua as etapas descritas em Configurar Conta da AWS antes de continuar.

Neste tutorial, você vai:

- Configurar uma instância do Amazon EC2
- Instale o Git, Node.js e configure o AWS CLI
- Crie AWS IoT recursos para seu dispositivo virtual
- Instale o AWS IoT Device SDK para JavaScript
- Executar os aplicativos de exemplo
- Exibir mensagens do aplicativo de amostra no AWS IoT console

Configurar uma instância do Amazon EC2

As etapas a seguir demonstram como criar uma instância do Amazon EC2 que atuará como seu dispositivo virtual no lugar de um dispositivo físico.

Se esta é a primeira vez que você cria uma instância do Amazon EC2, você pode achar que as instruções em Comece a usar as instâncias do Amazon EC2 Linux são mais úteis.

Como iniciar uma instância

1. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. A partir do menu do console à esquerda, expanda a seção Instâncias e selecione Instâncias. A partir do painel Instâncias, selecione Iniciar instâncias à direita para exibir uma lista de configurações básicas.

3. Na seção Nome e tags, digite um nome para a instância e, opcionalmente, adicione tags.
4. Na seção Imagens de aplicações e sistemas operacionais (imagem de máquina da Amazon), selecione um modelo de AMI para a sua instância, como AMI do Amazon Linux 2 (HVM). Observe que essas AMIs estão marcadas como “Qualificáveis para nível gratuito”.
5. Na página Selecione um tipo de instância, você pode selecionar a configuração de hardware da instância. Selecione o tipo `t2.micro`, que é selecionado por padrão. Observe que este tipo de instância está qualificado para o nível gratuito.
6. Na seção Par de chaves (login), selecione um nome de par de chaves na lista suspensa ou selecione Criar um novo par de chaves para criar um novo. Quando criar um novo par de chaves, certifique-se de baixar o arquivo de chave privada e salvá-lo em um local seguro, pois essa é sua única oportunidade de baixá-lo e salvá-lo. Você precisará fornecer o nome do par de chaves ao iniciar uma instância e a chave privada correspondente sempre que se conectar à instância.

   > **Warning**
   >
   > Não selecione a opção Prosseguir sem um par de chaves. Se você executar sua instância sem um par de chaves, você não poderá conectá-la.

7. Você pode manter as configurações padrão da seção Configurações de rede e da seção Configurar armazenamento. Quando tudo estiver pronto, selecione Iniciar instância.
8. Uma página de confirmação informa que sua instância está sendo executada. Selecione Visualizar instâncias para fechar a página de confirmação e voltar ao console.
9. Na tela Instances, é possível visualizar o status da execução. Demora um pouco para executar uma instância. Ao executar uma instância, seu estado inicial é `pending`. Após a inicialização da instância, seu estado muda para `running` e ela recebe um nome DNS público. (Se a coluna DNS público (IPv4) estiver oculta, escolha Mostrar/ocultar colunas (o ícone em forma de engrenagem) no canto superior direito da página e selecione DNS público ().) IPv4
10. Pode levar alguns minutos até que a instância esteja pronta para que você possa se conectar a ela. Certifique-se de que sua instância tenha sido aprovada nas verificações de status. É possível visualizar essas informações na coluna Status Checks.

    Quando a nova instância passar pelas verificações de status, prossiga para o próximo procedimento e conecte-se a ela.

Para se conectar à sua instância

Você pode se conectar a uma instância usando o cliente baseado em navegador selecionando a instância no console do Amazon EC2 e optando por se conectar usando o Amazon EC2 Instance Connect. O Instance Connect lida com as permissões e fornece uma conexão bem-sucedida.

1. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. No menu esquerdo, selecione Instâncias.
3. Selecione a instância e escolha Conectar.
4. Selecione Amazon EC2 Instance Connect, Connect.

Agora você deve poder ver uma janela do Amazon EC2 Instance Connect conectada à nova instância do Amazon EC2.

Instale o Git, Node.js e configure o AWS CLI

Nesta seção, você instalará o Git e o Node.js em sua instância do Linux.

Para instalar o Git

1. Na janela do Amazon EC2 Instance Connect, atualize sua instância com o seguinte comando.

```
sudo yum update -y
```

2. Na janela do Amazon EC2 Instance Connect, instale o Git com o seguinte comando.

```
sudo yum install git -y
```

3. Para verificar se o Git foi instalado e a versão atual do Git, execute o comando a seguir:

```
git --version
```

Para instalar o Node.js

1. Na janela do Amazon EC2 Instance Connect, instale o gerenciador de versão do node (nvm) com o seguinte comando.

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash
```

Usaremos o nvm para instalar o Node.js, pois o nvm pode instalar várias versões do Node.js e permitir que você alterne entre elas.

2. Na janela do Amazon EC2 Instance Connect, ative o nvm com o seguinte comando.

```
. ~/.nvm/nvm.sh
```

3. Na janela do Amazon EC2 Instance Connect, use o nvm para instalar a versão mais recente do Node.js com o seguinte comando.

```
nvm install 16
```

> **Note**
>
> Isso instalará a versão LTS mais recente do Node.js.

Instalar Node.js também instala o gerenciador de pacotes do nó (npm, do inglês "node package manager") para que você possa instalar módulos adicionais, conforme necessário.

4. Na janela do Amazon EC2 Instance Connect, teste se o Node.js está instalado e funcionando corretamente com o seguinte comando.

```
node -e "console.log('Running Node.js ' + process.version)"
```

Este tutorial requer o Node v10.0 ou uma versão posterior. Para acessar mais informações, consulte Tutorial: configuração do Node.js em uma instância do Amazon EC2.

Para configurar AWS CLI

Sua instância do Amazon EC2 vem pré-carregada com a AWS CLI. No entanto, você deve preencher seu AWS CLI perfil. Para acessar mais informações sobre como configurar a CLI, consulte Configurar a AWS CLI.

1. O exemplo a seguir mostra valores de exemplo. Substitua os valores pelos seus próprios valores. É possível encontrar esses valores no AWS console nas informações da sua conta em Credenciais de segurança.

   Na janela do Amazon EC2 Instance Connect, insira este comando:

```
aws configure
```

Depois, insira os valores da sua conta nas solicitações exibidas.

```
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
```

2. Você pode testar sua AWS CLI configuração com este comando:

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

Se o seu AWS CLI estiver configurado corretamente, o comando deverá retornar um endereço de endpoint do seu Conta da AWS.

Crie AWS IoT recursos para seu dispositivo virtual

Esta seção descreve como usar o AWS CLI para criar o objeto objeto e seus arquivos de certificado diretamente no dispositivo virtual. Isso é realizado diretamente no dispositivo para evitar possíveis complicações que possam surgir ao copiá-los para o dispositivo de outro computador. Nesta seção, você criará os recursos a seguir para o dispositivo virtual:

- Um objeto em que representar seu dispositivo virtual AWS IoT.
- Um certificado para autenticar um dispositivo virtual.
- Um documento de política autorizando seu dispositivo virtual a se conectar ao AWS IoT e publicar, receber e assinar mensagens.

Para criar um objeto de AWS IoT coisa na sua instância Linux

Os dispositivos conectados AWS IoT são representados por objetos de coisas no AWS IoT registro. Um objeto representa um dispositivo específico ou uma entidade lógica. Neste caso, seu objeto representará seu dispositivo virtual, esta instância do Amazon EC2.

1. Na janela do Amazon EC2 Instance Connect, execute o seguinte comando para criar seu objeto.

```
aws iot create-thing --thing-name "MyIotThing"
```

2. O resultado do JSON deve ser semelhante a esta:

```
{
    "thingArn": "arn:aws:iot:your-region:your-aws-account:thing/MyIotThing",
    "thingName": "MyIotThing",
    "thingId": "6cf922a8-d8ea-4136-f3401EXAMPLE"
}
```

Para criar e anexar AWS IoT chaves e certificados em sua instância Linux

O comando create-keys-and-certificate cria certificados de cliente assinados pela autoridade de certificação raiz da Amazon. Esse certificado é usado para autenticar a identidade do dispositivo virtual.

1. Na janela do Amazon EC2 Instance Connect, crie um diretório para armazenar os arquivos de certificado e chave.

```
mkdir ~/certs
```

2. Na janela do Amazon EC2 Instance Connect, baixe uma cópia do certificado da autoridade de certificação (CA) da Amazon com o comando a seguir.

```
curl -o ~/certs/Amazon-root-CA-1.pem \
    https://www.amazontrust.com/repository/AmazonRootCA1.pem
```

3. Na janela do Amazon EC2 Instance Connect, execute o seguinte comando para criar sua chave privada, chave pública e arquivos de certificado X.509. Esse comando também registra e ativa o certificado com. AWS IoT

```
aws iot create-keys-and-certificate \
    --set-as-active \
    --certificate-pem-outfile "~/certs/device.pem.crt" \
    --public-key-outfile "~/certs/public.pem.key" \
    --private-key-outfile "~/certs/private.pem.key"
```

A resposta é semelhante à seguinte. Salve o `certificateArn` para usá-lo em comandos subsequentes. Você precisará dele para anexar seu certificado à suo objeto e anexar a política ao certificado em etapas futuras.

```
{
    "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
    "certificateId":
 "9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
    "certificatePem": "
-----BEGIN CERTIFICATE-----
MIICiTCCEXAMPLE6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgEXAMPLEAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSEXAMPLE2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYEXAMPLEb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCEXAMPLEJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAEXAMPLEsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEXAMPLE25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+aEXAMPLE
EXAMPLEfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZEXAMPLELG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAEXAMPLEWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9qEXAMPLEyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDEXAMPLEBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----\n",
    "keyPair": {
        "PublicKey": "-----BEGIN PUBLIC
 KEY-----\nMIIBIjANBgkqhkEXAMPLEQEFAAOCAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h
\nMMEXAMPLEuuN/dMAS3fyce8DW/4+EXAMPLEyjmoF/YVF/
gHr99VEEXAMPLE5VF13\n59VK7cEXAMPLE67GK+y+jikqXOgHh/xJTwo
+sGpWEXAMPLEDz18xOd2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkR7qkEXAMPLE1DR6sx2HocliOOLtu6Fkw91swQWE
\GB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEvp9mQOUXP6plfgxwKRX2fEXAMPLEDa
\nhJLXkX3rHU2xbxJSq7D+XEXAMPLEcw+LyFhI5mgFRl88eGdsAEXAMPLElnI9EesG\nFQIDAQAB\n-----
END PUBLIC KEY-----\n",
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\nkey omitted for security
 reasons\n-----END RSA PRIVATE KEY-----\n"
    }
}
```

4. Na janela do Amazon EC2 Instance Connect, anexe seu objeto ao certificado que você acabou de criar usando o comando a seguir e a *certificateArn* resposta do comando anterior.

```
aws iot attach-thing-principal \
    --thing-name "MyIotThing" \
    --principal "certificateArn"
```

Se obtiver êxito, esse comando não retornará nenhuma saída.

Para criar e anexar uma política

1. Na janela do Amazon EC2 Instance Connect, crie o arquivo de política copiando e colando este documento de política em um arquivo nomeado **~/policy.json**.

   Se você não tiver um editor do Linux de preferência, pode abrir nano usando o seguinte comando.

   ```
   nano ~/policy.json
   ```

   Cole o documento de política para `policy.json` nele. Pressione ctrl+x para sair do editor nano e salve o arquivo.

   Conteúdo do documento de política para `policy.json`.

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "iot:Publish",
                   "iot:Subscribe",
                   "iot:Receive",
                   "iot:Connect"
               ],
               "Resource": [
                   "*"
               ]
           }
       ]
   }
   ```

2. Na janela do Amazon EC2 Instance Connect, crie sua política com o comando a seguir.

   ```
   aws iot create-policy \
       --policy-name "MyIotThingPolicy" \
   ```

```
    --policy-document "file://~/policy.json"
```

Saída:

```
{
    "policyName": "MyIotThingPolicy",
    "policyArn": "arn:aws:iot:us-east-1:your-aws-account:policy/
MyIotThingPolicy",
    "policyDocument": "{
        \"Version\": \"2012-10-17\",
        \"Statement\": [
            {
                \"Effect\": \"Allow\",
                \"Action\": [
                    \"iot:Publish\",
                    \"iot:Receive\",
                    \"iot:Subscribe\",
                    \"iot:Connect\"
                ],
                \"Resource\": [
                    \"*\"
                ]
        }
        ]
    }",
    "policyVersionId": "1"
}
```

3. Na janela do Amazon EC2 Instance Connect, anexe a política ao certificado do dispositivo virtual com o comando a seguir.

```
aws iot attach-policy \
    --policy-name "MyIotThingPolicy" \
    --target "certificateArn"
```

Se obtiver êxito, esse comando não retornará nenhuma saída.

Instale o AWS IoT Device SDK para JavaScript

Nesta seção, você instalará o AWS IoT Device SDK for JavaScript, que contém o código que os aplicativos podem usar para se comunicar AWS IoT e os programas de amostra. Para obter mais informações, consulte o AWS IoT Device SDK para JavaScript GitHub repositório.

Para instalar o AWS IoT Device SDK JavaScript em sua instância Linux

1. Na janela do Amazon EC2 Instance Connect, clone o AWS IoT Device SDK para JavaScript repositório no `aws-iot-device-sdk-js-v2` diretório do seu diretório inicial usando este comando.

   ```
   cd ~
   git clone https://github.com/aws/aws-iot-device-sdk-js-v2.git
   ```

2. Navegue até o diretório `aws-iot-device-sdk-js-v2` criado na etapa anterior.

   ```
   cd aws-iot-device-sdk-js-v2
   ```

3. Use npm para instalar o SDK.

   ```
   npm install
   ```

Executar os aplicativos de exemplo

Os comandos das próximas seções pressupõem que seus arquivos de chave e certificado estão armazenados em seu dispositivo virtual, conforme mostrado na seguinte tabela.

Nomes de arquivos de certificado

Arquivo	Caminho do arquivo
Chave privada	`~/certs/private.pem.key`
Certificado de dispositivo	`~/certs/device.pem.crt`
Certificado CA raiz	`~/certs/Amazon-root-CA-1.pem`

Nesta seção, você instalará e executará o aplicativo de `pub-sub.js` amostra encontrado no `aws-iot-device-sdk-js-v2/samples/node` diretório do AWS IoT Device SDK for JavaScript. Este

aplicativo mostra a maneira como um dispositivo, sua instância do Amazon EC2, usa a biblioteca MQTT para publicar e assinar mensagens MQTT. O aplicativo de exemplo `pub-sub.js` assina um tópico, `topic_1`, publica 10 mensagens no tópico e exibe as mensagens à medida que elas são recebidas do agente de mensagens.

Para instalar e executar o aplicativo de exemplo

1. Na janela do Amazon EC2 Instance Connect, navegue até o diretório `aws-iot-device-sdk-js-v2/samples/node/pub_sub` criado pelo SDK e instale o aplicativo de exemplo com os seguintes comandos.

   ```
   cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
   npm install
   ```

2. Em sua janela do Amazon EC2 Instance Connect, obtenha *`your-iot-endpoint`* AWS IoT usando este comando.

   ```
   aws iot describe-endpoint --endpoint-type iot:Data-ATS
   ```

3. Na janela do Amazon EC2 Instance Connect, insira *`your-iot-endpoint`* conforme indicado e execute esse comando.

   ```
   node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/
   certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
   ```

O aplicativo de exemplo:

1. Conecta-se AWS IoT Core à sua conta.
2. Assina o tópico de mensagens topic_1 e exibe as mensagens recebidas sobre esse tópico.
3. Publica 10 mensagens no tópico topic_1.
4. Exibe uma saída semelhante à seguinte:

```
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":1}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":2}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":3}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
```

```
{"message":"Hello world!","sequence":4}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":5}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":6}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":7}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":8}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":9}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":10}
```

Se você estiver com problemas para executar a aplicação de exemplo, examine the section called “Solucionar problemas com a aplicação de amostra”.

Também é possível adicionar o parâmetro `--verbosity debug` à linha de comando para que o aplicativo de exemplo exiba mensagens detalhadas sobre sua operação. Essas informações podem fornecer a ajuda necessária para você corrigir o problema.

Exibir mensagens do aplicativo de amostra no AWS IoT console

É possível ver as mensagens do aplicativo de exemplo à medida que elas passam pelo agente de mensagens usando o cliente de teste MQTT no console do AWS IoT .

Para visualizar as mensagens MQTT publicadas pelo aplicativo de exemplo

1. Consulte Exibir mensagens MQTT com o cliente AWS IoT MQTT. Isso ajudará você a aprender a usar o cliente de teste MQTT no console do AWS IoT para visualizar as mensagens MQTT à medida que elas passam pelo agente de mensagens.
2. Abra o cliente de teste MQTT no console do AWS IoT .
3. Em Assinar um tópico, assine o tópico topic_1.
4. Na janela do Amazon EC2 Instance Connect, execute o aplicativo de exemplo novamente e observe as mensagens no cliente de teste MQTT no console do AWS IoT .

```
cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/
certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
```

Para obter mais informações sobre o MQTT e como é AWS IoT Core compatível com o protocolo, consulte MQTT.

Use seu PC Windows ou Linux ou Mac como um AWS IoT dispositivo

Neste tutorial, você configurará um computador pessoal para uso com AWS IoT. Essas instruções são compatíveis com Windows, Linux PCs e Macs. Para concluir este tutorial, você precisará instalar alguns softwares no seu computador. Se não quiser instalar software no seu computador, você pode tentar Criar um dispositivo virtual com o Amazon EC2 e instalar todos os softwares em uma máquina virtual.

Neste tutorial, você vai:

- Configurar seu computador pessoal
- Instale o Git, o Python e o Device SDK para Python AWS IoT
- Configurar a política e executar o aplicativo de exemplo
- Exibir mensagens do aplicativo de amostra no AWS IoT console
- Execute o exemplo de assinatura compartilhada em Python

Configurar seu computador pessoal

Para concluir este tutorial, será preciso um PC com Windows ou Linux ou de um Mac com conexão à Internet.

Antes de avançar para a próxima etapa, verifique se você pode abrir uma janela de linha de comando no computador. Em um PC com Windows, use cmd.exe. Em um PC com Linux ou em um Mac, use Terminal.

Instale o Git, o Python e o Device SDK para Python AWS IoT

Nesta seção, você instalará o Python e o AWS IoT Device SDK for Python em seu computador.

Instalar a versão mais recente do Git e do Python

Este procedimento explica como instalar a versão mais recente do Git e do Python em seu computador pessoal.

Para baixar e instalar o Git e o Python em seu computador

1. Verifique se você possui o Git já instalado no computador. Insira este comando na linha de comando.

```
git --version
```

Se o comando exibir a versão do Git, o Git está instalado e você pode avançar para a próxima etapa.

Caso o comando exiba um erro, abra https://git-scm.com/download e instale o Git em seu computador.

2. Verifique se você já possui o Python instalado. Insira o seguinte comando na linha de comando.

```
python -V
```

> **Note**
>
> Se esse comando exibir um erro: `Python was not found`, pode ser que seu sistema operacional chame o executável Python v3.x como `Python3`. Nesse caso, substitua todas as instâncias de `python` com `python3` e siga com o resto deste tutorial.

Se o comando exibir a versão do Python, então o Python já está instalado. Este tutorial requer o Python v3.7 ou uma versão posterior.

3. Se o Python já estiver instalado, você pode ignorar o restante das etapas desta seção. Se não estiver, então continue.
4. Abra https://www.python.org/downloads/ e baixe o instalador para o seu computador.
5. Se o download não começar a instalação automaticamente, execute o programa baixado para instalar o Python.
6. Verifique a instalação do Python.

```
python -V
```

Confirme se o comando exibe a versão do Python. Se a versão do Python não for exibida, tente baixar e instalar novamente.

Instale o AWS IoT Device SDK para Python

Para instalar o AWS IoT Device SDK para Python em seu computador

1. Instale a v2 do AWS IoT Device SDK para Python.

   ```
   python3 -m pip install awsiotsdk
   ```

2. Clone o repositório AWS IoT Device SDK for Python no diretório aws-iot-device-sdk -python-v2 do seu diretório inicial. Esse procedimento se refere ao diretório base dos arquivos como os quais você está instalando*home*.

   A localização real do *home* diretório depende do seu sistema operacional.

   Linux/macOS

   No macOS e no Linux, o *home* diretório é. ~

   ```
   cd ~
   git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
   ```

   Windows

   No Windows, você pode encontrar o caminho do *home* diretório executando esse comando na cmd janela.

   ```
   echo %USERPROFILE%
   cd %USERPROFILE%
   git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
   ```

   > **Note**
   >
   > Se você estiver usando PowerShell o Windows em vez decmd.exe, use o comando a seguir.
   >
   > ```
   > echo $home
   > ```

Para obter mais informações, consulte o repositório AWS IoT Device SDK for GitHub Python.

Preparar-se para executar os aplicativos de exemplo

Para preparar seu sistema para executar o aplicativo de exemplo

- Crie o diretório `certs`. No diretório `certs`, copie a chave privada, o certificado do dispositivo e os arquivos do certificado CA raiz salvos ao criar e registrar o objeto em the section called "Crie AWS IoT recursos". Os nomes de cada arquivo no diretório de destino devem corresponder àqueles na tabela.

  Os comandos na próxima seção pressupõem que seus arquivos de chave e certificado estão armazenados em seu dispositivo, conforme mostrado nesta tabela.

  Linux/macOS

  Execute o seguinte comando para criar o subdiretório `certs` que você usará ao executar os aplicativos de exemplo.

  ```
  mkdir ~/certs
  ```

  No novo subdiretório, copie os arquivos para os caminhos de arquivo de destino exibidos na seguinte tabela.

  Nomes de arquivos de certificado

Arquivo	Caminho do arquivo
Chave privada	`~/certs/private.pem.key`
Certificado de dispositivo	`~/certs/device.pem.crt`
Certificado CA raiz	`~/certs/Amazon-root-CA-1.pem`

  Execute o seguinte comando para listar os arquivos no diretório `certs` e compare-os com aqueles listados na tabela.

  ```
  ls -l ~/certs
  ```

Windows

Execute o seguinte comando para criar o subdiretório `certs` que você usará ao executar os aplicativos de exemplo.

```
mkdir %USERPROFILE%\certs
```

No novo subdiretório, copie os arquivos para os caminhos de arquivo de destino exibidos na seguinte tabela.

Nomes de arquivos de certificado

Arquivo	Caminho do arquivo
Chave privada	`%USERPROFILE%\certs\private.pem.key`
Certificado de dispositivo	`%USERPROFILE%\certs\device.pem.crt`
Certificado CA raiz	`%USERPROFILE%\certs\Amazon-root-CA-1.pem`

Execute o seguinte comando para listar os arquivos no diretório `certs` e compare-os com aqueles listados na tabela.

```
dir %USERPROFILE%\certs
```

Configurar a política e executar o aplicativo de exemplo

Nesta seção, você configurará sua política e executará o script de exemplo `pubsub.py` encontrado no diretório `aws-iot-device-sdk-python-v2/samples` do AWS IoT Device SDK for Python. Este script mostra a maneira como seu dispositivo usa a biblioteca MQTT para publicar e assinar mensagens MQTT.

O aplicativo de exemplo `pubsub.py` assina um tópico, `test/topic`, publica 10 mensagens no tópico e exibe as mensagens à medida que elas são recebidas do agente de mensagens.

Para executar o script de exemplo `pubsub.py`, você precisa das seguintes informações:

Valores de parâmetros de aplicação

Parâmetro	Onde encontrar o valor
`your-iot-endpoint`	1. No console do AWS IoT, no menu esquerdo, selecione Configurações. 2. Na página Configurações, seu endpoint estará exibido na seção Endpoint de dados do dispositivo.

O *`your-iot-endpoint`* valor tem um formato de:*`endpoint_id`*`-ats.iot.`*`region`*`.amazonaws.com`, por exemplo,`a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com`.

Antes de executar o script, certifique-se de que a política da suo objeto forneça permissões para o script de exemplo se conectar, assinar, publicar e receber.

Para encontrar e examinar o documento de política de um recurso de objeto

1. No console do AWS IoT, na lista Objetos, localize o recurso de objeto que representa o seu dispositivo.
2. Escolha o link Nome do recurso de objeto que representa o seu dispositivo para abrir a página de Detalhes do objeto.
3. Na página Detalhes do objeto, na guia Certificados, selecione o certificado anexado ao recurso de objeto. Deve haver apenas um certificado na lista. Se houver mais de um, selecione o certificado cujos arquivos estão instalados no seu dispositivo e que serão usados para se conectar ao AWS IoT Core.

   Na página de detalhes do Certificado, na guia Políticas, selecione a política anexada ao certificado. Deve haver apenas uma. Caso haja mais de uma, repita a próxima etapa para cada uma delas para garantir que ao menos uma política conceda o acesso necessário.
4. Na página de visão geral da Política, localize o editor JSON e selecione Editar documento de política para revisar e editar o documento de política conforme necessário.

5. O JSON da política é exibido no exemplo a seguir. No `"Resource"` elemento, *`region:account`* substitua por seu Região da AWS e Conta da AWS em cada um dos `Resource` valores.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Receive"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/test/topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topicfilter/test/topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:client/test-*"
            ]
        }
    ]
}
```

Linux/macOS

Para executar o script de exemplo no Linux/macOS

1. Na janela de linha de comando, navegue até o diretório `~/aws-iot-device-sdk-python-v2/samples/node/pub_sub` criado pelo SDK usando os seguintes comandos.

```
cd ~/aws-iot-device-sdk-python-v2/samples
```

2. Na janela da linha de comando, substitua *your-iot-endpoint* conforme indicado e execute esse comando.

```
python3 pubsub.py --endpoint your-iot-endpoint --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/private.pem.key
```

Windows

Para executar o aplicativo de exemplo em um PC com Windows

1. Na janela de linha de comando, navegue até o diretório `%USERPROFILE%\aws-iot-device-sdk-python-v2\samples` criado pelo SDK e instale o aplicativo de exemplo usando os seguintes comandos.

```
cd %USERPROFILE%\aws-iot-device-sdk-python-v2\samples
```

2. Na janela da linha de comando, substitua *your-iot-endpoint* conforme indicado e execute esse comando.

```
python3 pubsub.py --endpoint your-iot-endpoint --ca_file %USERPROFILE%
\certs\Amazon-root-CA-1.pem --cert %USERPROFILE%\certs\device.pem.crt --key
 %USERPROFILE%\certs\private.pem.key
```

O script de exemplo:

1. Conecta-se ao AWS IoT Core para sua conta.
2. Assina o tópico de mensagens test/topic e exibe as mensagens recebidas sobre esse tópico.
3. Publica 10 mensagens no tópico test/topic.
4. Exibe uma saída semelhante à seguinte:

```
Connected!
Subscribing to topic 'test/topic'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 10 message(s)
Publishing message to topic 'test/topic': Hello World! [1]
Received message from topic 'test/topic': b'"Hello World! [1]"'
Publishing message to topic 'test/topic': Hello World! [2]
Received message from topic 'test/topic': b'"Hello World! [2]"'
Publishing message to topic 'test/topic': Hello World! [3]
Received message from topic 'test/topic': b'"Hello World! [3]"'
Publishing message to topic 'test/topic': Hello World! [4]
Received message from topic 'test/topic': b'"Hello World! [4]"'
Publishing message to topic 'test/topic': Hello World! [5]
Received message from topic 'test/topic': b'"Hello World! [5]"'
Publishing message to topic 'test/topic': Hello World! [6]
Received message from topic 'test/topic': b'"Hello World! [6]"'
Publishing message to topic 'test/topic': Hello World! [7]
Received message from topic 'test/topic': b'"Hello World! [7]"'
Publishing message to topic 'test/topic': Hello World! [8]
Received message from topic 'test/topic': b'"Hello World! [8]"'
Publishing message to topic 'test/topic': Hello World! [9]
Received message from topic 'test/topic': b'"Hello World! [9]"'
Publishing message to topic 'test/topic': Hello World! [10]
Received message from topic 'test/topic': b'"Hello World! [10]"'
10 message(s) received.
Disconnecting...
Disconnected!
```

Se você estiver com problemas para executar a aplicação de exemplo, examine the section called “Solucionar problemas com a aplicação de amostra”.

Também é possível adicionar o parâmetro `--verbosity Debug` à linha de comando para que o aplicativo de exemplo exiba mensagens detalhadas sobre sua operação. Essas informações podem ajudar você a corrigir o problema.

Exibir mensagens do aplicativo de amostra no AWS IoT console

É possível ver as mensagens do aplicativo de exemplo à medida que elas passam pelo agente de mensagens usando o cliente de teste MQTT no console do AWS IoT .

Para visualizar as mensagens MQTT publicadas pelo aplicativo de exemplo

1. Consulte Exibir mensagens MQTT com o cliente AWS IoT MQTT. Isso ajudará você a aprender a usar o cliente de teste MQTT no console do AWS IoT para visualizar as mensagens MQTT à medida que elas passam pelo agente de mensagens.
2. Abra o cliente de teste MQTT no console do AWS IoT .
3. Em Assinar um tópico, assine o tópico test/topic.
4. Na janela de linha de comando, execute o aplicativo de exemplo novamente e observe as mensagens no cliente MQTT no console do AWS IoT .

   Linux/macOS

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples
   python3 pubsub.py --topic test/topic --ca_file ~/certs/Amazon-root-CA-1.pem --
   cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
   endpoint
   ```

   Windows

   ```
   cd %USERPROFILE%\aws-iot-device-sdk-python-v2\samples
   python3 pubsub.py --topic test/topic --ca_file %USERPROFILE%\certs\Amazon-root-
   CA-1.pem --cert %USERPROFILE%\certs\device.pem.crt --key %USERPROFILE%\certs
   \private.pem.key --endpoint your-iot-endpoint
   ```

Para obter mais informações sobre o MQTT e como é AWS IoT Core compatível com o protocolo, consulte MQTT.

Execute o exemplo de assinatura compartilhada em Python

AWS IoT Core suporta assinaturas compartilhadas para MQTT 3 e MQTT 5. As assinaturas compartilhadas permitem que vários clientes compartilhem uma assinatura de um tópico e somente um cliente receberá mensagens publicadas nesse tópico usando uma distribuição randomizada. Para usar assinaturas compartilhadas, os clientes assinam o filtro de tópicos de uma assinatura compartilhada: `$share/{ShareName}/{TopicFilter}`.

Para configurar a política e executar o exemplo de assinatura compartilhada

1. Para executar o exemplo de assinatura compartilhada, é preciso configurar a política da suo objeto conforme documentado em Assinatura compartilhada do MQTT 5.

2. Para executar o exemplo de assinatura compartilhada, execute os comandos a seguir.

   Linux/macOS

   Para executar o script de exemplo no Linux/macOS

   1. Na janela de linha de comando, navegue até o diretório `~/aws-iot-device-sdk-python-v2/samples` criado pelo SDK usando os seguintes comandos.

      ```
      cd ~/aws-iot-device-sdk-python-v2/samples
      ```

   2. Na janela da linha de comando, substitua *`your-iot-endpoint`* conforme indicado e execute esse comando.

      ```
      python3 mqtt5_shared_subscription.py --endpoint your-iot-endpoint --ca_file
       ~/certs/Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/
      private.pem.key --group_identifier consumer
      ```

   Windows

   Para executar o aplicativo de exemplo em um PC com Windows

   1. Na janela de linha de comando, navegue até o diretório `%USERPROFILE%\aws-iot-device-sdk-python-v2\samples` criado pelo SDK e instale o aplicativo de exemplo usando os seguintes comandos.

      ```
      cd %USERPROFILE%\aws-iot-device-sdk-python-v2\samples
      ```

   2. Na janela da linha de comando, substitua *`your-iot-endpoint`* conforme indicado e execute esse comando.

      ```
      python3 mqtt5_shared_subscription.py --endpoint your-iot-endpoint --ca_file
       %USERPROFILE%\certs\Amazon-root-CA-1.pem --cert %USERPROFILE%\certs
      \device.pem.crt --key %USERPROFILE%\certs\private.pem.key --group_identifier
       consumer
      ```

> **Note**
>
> Você pode, opcionalmente, especificar um identificador de grupo com base nas suas necessidades ao executar o exemplo (por exemplo, `--group_identifier consumer`). `python-sample` será o identificador de grupo padrão se você não especificar nenhum.

3. A saída na linha de comando pode se assemelhar ao seguinte:

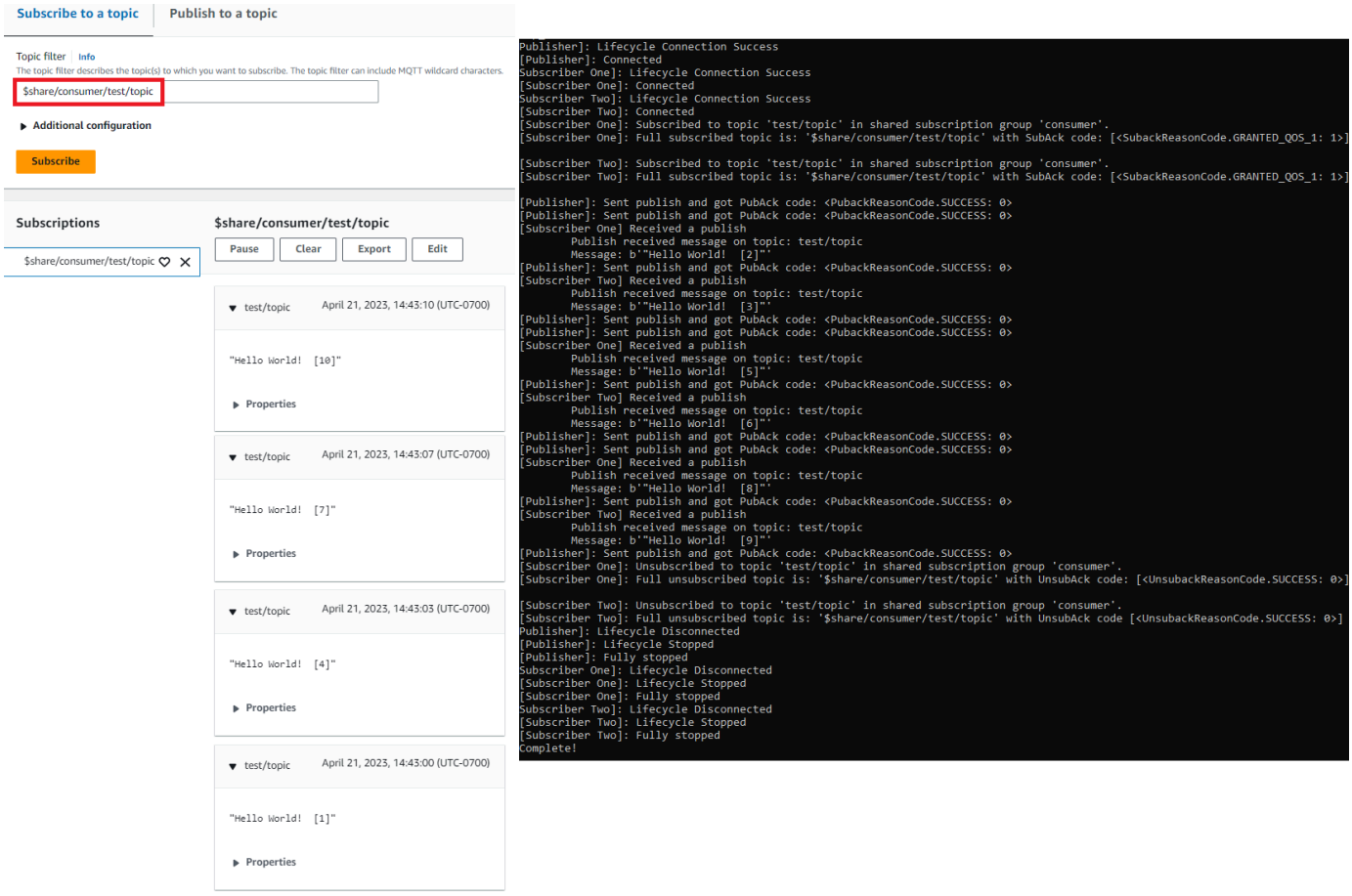
```
Publisher]: Lifecycle Connection Success
[Publisher]: Connected
Subscriber One]: Lifecycle Connection Success
[Subscriber One]: Connected
Subscriber Two]: Lifecycle Connection Success
[Subscriber Two]: Connected
[Subscriber One]: Subscribed to topic 'test/topic' in shared subscription group
 'consumer'.
[Subscriber One]: Full subscribed topic is: '$share/consumer/test/topic' with
 SubAck code: [<SubackReasonCode.GRANTED_QOS_1: 1>]
[Subscriber Two]: Subscribed to topic 'test/topic' in shared subscription group
 'consumer'.
[Subscriber Two]: Full subscribed topic is: '$share/consumer/test/topic' with
 SubAck code: [<SubackReasonCode.GRANTED_QOS_1: 1>]
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [1]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [2]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [3]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [4]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
```

```
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [5]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [6]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [7]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [8]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [9]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [10]"'
[Subscriber One]: Unsubscribed to topic 'test/topic' in shared subscription group
 'consumer'.
[Subscriber One]: Full unsubscribed topic is: '$share/consumer/test/topic' with
 UnsubAck code: [<UnsubackReasonCode.SUCCESS: 0>]
[Subscriber Two]: Unsubscribed to topic 'test/topic' in shared subscription group
 'consumer'.
[Subscriber Two]: Full unsubscribed topic is: '$share/consumer/test/topic' with
 UnsubAck code [<UnsubackReasonCode.SUCCESS: 0>]
Publisher]: Lifecycle Disconnected
[Publisher]: Lifecycle Stopped
[Publisher]: Fully stopped
Subscriber One]: Lifecycle Disconnected
[Subscriber One]: Lifecycle Stopped
[Subscriber One]: Fully stopped
Subscriber Two]: Lifecycle Disconnected
[Subscriber Two]: Lifecycle Stopped
[Subscriber Two]: Fully stopped
Complete!
```

4. Abra o cliente de teste MQTT no console do AWS IoT . Em Assinar um tópico, assine o tópico da assinatura compartilhada, como: `$share/consumer/test/topic`. Você pode especificar um

identificador de grupo com base nas suas necessidades ao executar o exemplo (por exemplo, `--group_identifier consumer`). Se você não especificar um identificador de grupo, o valor padrão será `python-sample`. Para obter mais informações, consulte o exemplo em Assinatura compartilhada do MQTT 5 em Python e Assinaturas compartilhadas do Guia do desenvolvedor do AWS IoT Core .

Na janela de linha de comando, execute o aplicativo de exemplo novamente e observe a distribuição de mensagens no cliente de teste MQTT do console do AWS IoT e na linha de comando.



Conectar um Raspberry Pi ou outro dispositivo

Nesta seção, configuraremos um Raspberry Pi para uso com AWS IoT. Caso tenha outro dispositivo que gostaria de conectar, as instruções do Raspberry Pi incluem referências para ajudá-lo a adaptar essas instruções ao seu dispositivo.

Em geral, isso leva cerca de 20 minutos, mas pode levar mais tempo se você tiver muitas atualizações de software do sistema a serem instaladas.

Neste tutorial, você vai:

- Configure seu dispositivo
- Instale as ferramentas e bibliotecas necessárias para o AWS IoT Device SDK
- Instale o AWS IoT Device SDK
- Instalar e executar o aplicativo de exemplo
- Exibir mensagens do aplicativo de amostra no AWS IoT console

> **Important**
>
> Adaptar estas instruções a outros dispositivos e sistemas operacionais pode ser um desafio. Você precisará conhecer seu dispositivo o bastante para interpretar estas instruções e aplicá-las ao seu dispositivo.
>
> Se você encontrar dificuldades ao configurar seu dispositivo para AWS IoT, tente uma das outras opções do dispositivo como alternativa, como Criar um dispositivo virtual com o Amazon EC2 ouUse seu PC Windows ou Linux ou Mac como um AWS IoT dispositivo.

Configure seu dispositivo

O objetivo desta etapa é coletar o que é preciso para configurar seu dispositivo para que ele possa iniciar o sistema operacional (SO), conectar-se à Internet e permitir que você interaja com ele em uma interface de linha de comando.

Para concluir este tutorial, você precisará do seguinte:

- Um Conta da AWS. Se você não possuir uma, conclua as etapas descritas em Configurar Conta da AWS antes de continuar.
- Um Modelo B do Raspberry Pi 3 ou modelo mais recente. Pode ser que isso funcione em versões anteriores do Raspberry Pi, mas elas não foram testadas.
- Raspberry Pi OS (32 bits) ou uma versão posterior. Recomendamos usar a versão mais recente do SO do Raspberry Pi. Versões anteriores do SO podem funcionar, mas não foram testadas.

  Para executar este exemplo, não é preciso instalar a área de trabalho com a interface gráfica do usuário (GUI); entretanto, se você é novo ao Raspberry Pi e seu hardware do Raspberry Pi for compatível, usar a área de trabalho com a GUI pode ser mais fácil.
- Uma Ethernet ou WiFi conexão.

- Teclado, mouse, monitor, cabos, fontes de alimentação e outros itens de hardware necessários ao seu dispositivo.

> **Important**
>
> Antes de avançar para a próxima etapa, seu dispositivo deve ter o sistema operacional instalado, configurado e em execução. O dispositivo deve estar conectado à Internet e você precisa ser capaz de acessá-lo usando sua interface de linha de comando. O acesso à linha de comando pode ser realizado através de um teclado, mouse e monitor conectados diretamente ou usando uma interface remota do terminal SSH.

Se estiver executando um sistema operacional em seu Raspberry Pi que possua uma interface gráfica de usuário (GUI), abra uma janela de terminal no dispositivo e conclua as instruções a seguir nessa janela. Caso contrário, se estiver se conectando ao seu dispositivo usando um terminal remoto, como o PuTTY, abra um terminal remoto no seu dispositivo e use-o.

Instale as ferramentas e bibliotecas necessárias para o AWS IoT Device SDK

Antes de instalar o AWS IoT Device SDK e o código de amostra, verifique se o sistema está atualizado e tem as ferramentas e bibliotecas necessárias para instalar o. SDKs

1. Atualize o sistema operacional e instale as bibliotecas necessárias

   Antes de instalar um AWS IoT Device SDK, execute esses comandos em uma janela de terminal em seu dispositivo para atualizar o sistema operacional e instalar as bibliotecas necessárias.

   ```
   sudo apt-get update
   ```

   ```
   sudo apt-get upgrade
   ```

   ```
   sudo apt-get install cmake
   ```

   ```
   sudo apt-get install libssl-dev
   ```

2. Instale o Git

Se o sistema operacional do seu dispositivo não vem com o Git instalado, você deve instalá-lo para instalar o AWS IoT Device SDK for. JavaScript

a. Teste para verificar se o Git já está instalado executando o seguinte comando.

```
git --version
```

b. Se o comando anterior retornar a versão do Git, então o Git já está instalado e você pode avançar para a Etapa 3.

c. Se for exibido um erro ao executar o comando git, instale o Git executando o seguinte comando.

```
sudo apt-get install git
```

d. Teste novamente para verificar se o Git está instalado executando o seguinte comando.

```
git --version
```

e. Se o Git estiver instalado, avance para a próxima seção. Caso contrário, solucione o problema e corrija o erro antes de prosseguir. Você precisa do Git para instalar o AWS IoT Device SDK para. JavaScript

Instale o AWS IoT Device SDK

Instale o AWS IoT Device SDK.

Python

Nesta seção, você instalará o Python, suas ferramentas de desenvolvimento e o AWS IoT Device SDK for Python em seu dispositivo. Estas instruções são para um Raspberry Pi executando o sistema operacional mais recente do Raspberry Pi. Caso tenha outro dispositivo ou esteja usando outro sistema operacional, pode ser preciso adaptar estas instruções para o dispositivo.

1. Instale o Python e suas ferramentas de desenvolvimento

   O AWS IoT Device SDK para Python exige que o Python v3.5 ou posterior esteja instalado no seu Raspberry Pi.

   Em uma janela de terminal do seu dispositivo, execute os comandos a seguir.

1. Execute o seguinte comando para determinar a versão do Python que está instalada no dispositivo.

   ```
   python3 --version
   ```

   Se o Python estiver instalado, ele exibirá a versão.
2. Se a versão exibida for `Python 3.5` ou superior, você pode avançar para a Etapa 2.
3. Se a versão exibida for inferior a `Python 3.5`, você pode instalar a versão correta executando o comando a seguir.

   ```
   sudo apt install python3
   ```
4. Execute o seguinte comando para confirmar que a versão correta do Python foi instalada.

   ```
   python3 --version
   ```

2. Teste para pip3

   Em uma janela de terminal do seu dispositivo, execute os comandos a seguir.

   1. Execute o seguinte comando para verificar se pip3 está instalado.

      ```
      pip3 --version
      ```
   2. Se o comando retornar um número de versão, então o pip3 está instalado e você pode avançar para a Etapa 3.
   3. Se o comando anterior retornar um erro, execute o seguinte comando para instalar o pip3.

      ```
      sudo apt install python3-pip
      ```
   4. Execute o seguinte comando para verificar se pip3 está instalado.

      ```
      pip3 --version
      ```

3. Instale o AWS IoT Device SDK atual para Python

   Instale o AWS IoT Device SDK para Python e baixe os aplicativos de amostra em seu dispositivo.

No seu dispositivo, execute os comandos a seguir.

```
cd ~
python3 -m pip install awsiotsdk
```

```
git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
```

JavaScript

Nesta seção, você instalará o Node.js, o gerenciador de pacotes npm e o AWS IoT Device SDK for JavaScript em seu dispositivo. Estas instruções são para um Raspberry Pi executando o sistema operacional do Raspberry Pi. Caso tenha outro dispositivo ou esteja usando outro sistema operacional, pode ser preciso adaptar estas instruções para o dispositivo.

1. Instalar a versão mais recente do Node.js

   O AWS IoT Device SDK for JavaScript requer que o Node.js e o gerenciador de pacotes npm estejam instalados no seu Raspberry Pi.

   a. Baixe a versão mais recente do repositório do Node digitando o seguinte comando.

   ```
   cd ~
   curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
   ```

   b. Instale o Node e o npm.

   ```
   sudo apt-get install -y nodejs
   ```

   c. Verifique a instalação do Node.

   ```
   node -v
   ```

   Confirme se o comando exibe a versão do Node. Este tutorial requer o Node v10.0 ou uma versão posterior. Se a versão do Node não for exibida, tente baixar o repositório do Node mais uma vez.

   d. Verifique a instalação do npm.

```
npm -v
```

Confirme se o comando exibe a versão do npm. Se a versão do npm não for exibida, tente instalar o Node e o npm mais uma vez.

e. Reinicie o dispositivo.

```
sudo shutdown -r 0
```

Continue depois que o dispositivo for reiniciado.

2. Instale o AWS IoT Device SDK para JavaScript

Instale o AWS IoT Device SDK para JavaScript em seu Raspberry Pi.

a. Clone o AWS IoT Device SDK for JavaScript repository no `aws-iot-device-sdk-js-v2` diretório do seu diretório. *home* No Raspberry Pi, o *home* diretório é`~/`, que é usado como *home* diretório nos comandos a seguir. Se o seu dispositivo usa um caminho diferente para o *home* diretório, você deve `~/` substituir pelo caminho correto para o seu dispositivo nos comandos a seguir.

Estes comandos criam o diretório `~/aws-iot-device-sdk-js-v2` e copiam o código do SDK para ele.

```
cd ~
git clone https://github.com/aws/aws-iot-device-sdk-js-v2.git
```

b. Vá para o diretório `aws-iot-device-sdk-js-v2` criado na etapa anterior e execute `npm install` para instalar o SDK. O comando `npm install` invocará a criação da biblioteca `aws-crt`, que pode levar alguns minutos para ser concluída.

```
cd ~/aws-iot-device-sdk-js-v2
npm install
```

Instalar e executar o aplicativo de exemplo

Nesta seção, você instalará e executará o aplicativo de `pubsub` amostra encontrado no AWS IoT Device SDK. Este aplicativo mostra a maneira como seu dispositivo usa a biblioteca MQTT para publicar e assinar mensagens MQTT. O aplicativo de exemplo assina um tópico, `topic_1`, publica

10 mensagens no tópico e exibe as mensagens à medida que elas são recebidas do agente de mensagens.

Instale os arquivos de certificado

O aplicativo de exemplo exige que os arquivos de certificado que autenticam o dispositivo estejam instalados no dispositivo.

Para instalar os arquivos de certificado do dispositivo para o aplicativo de exemplo

1. Crie um `certs` subdiretório em seu *home* diretório executando esses comandos.

   ```
   cd ~
   mkdir certs
   ```

2. No diretório `~/certs`, copie a chave privada, certificado do dispositivo e certificado CA raiz criados anteriormente em the section called “Crie AWS IoT recursos”.

   A forma de copiar os arquivos do certificado para o dispositivo depende do dispositivo e do sistema operacional e não está descrita aqui. Entretanto, se o dispositivo for compatível com uma interface gráfica de usuário (GUI) e tiver um navegador da Web, será possível executar o procedimento descrito em the section called “Crie AWS IoT recursos” a partir do navegador da Web do dispositivo para baixar os arquivos resultantes diretamente no dispositivo.

   Os comandos na próxima seção pressupõem que seus arquivos de chave e certificado estão armazenados no dispositivo, conforme mostrado nesta tabela.

   Nomes de arquivos de certificado

Arquivo	Caminho do arquivo
Certificado CA raiz	`~/certs/Amazon-root-CA-1.pem`
Certificado de dispositivo	`~/certs/device.pem.crt`
Chave privada	`~/certs/private.pem.key`

Para executar o aplicativo de exemplo, você precisa das seguintes informações:

Valores de parâmetros de aplicação

Parâmetro	Onde encontrar o valor
your-iot-endpoint	No console do AWS IoT, selecione Todos os dispositivos e, depois, selecione Objetos. Na página Configurações no menu AWS IoT . Seu endpoint é exibido na seção Endpoint de dados do dispositivo.

O *your-iot-endpoint* valor tem um formato de:*endpoint_id*-ats.iot.*region*.amazonaws.com, por exemplo,a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com.

Python

Para instalar e executar o aplicativo de exemplo

1. Navegue até o diretório do aplicativo de exemplo.

```
cd ~/aws-iot-device-sdk-python-v2/samples
```

2. Na janela da linha de comando, substitua *your-iot-endpoint* conforme indicado e execute esse comando.

```
python3 pubsub.py --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --
cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
endpoint
```

3. Observe que o aplicativo de exemplo:

    1. Conecta-se ao AWS IoT serviço da sua conta.
    2. Assina o tópico de mensagens topic_1 e exibe as mensagens recebidas sobre esse tópico.
    3. Publica 10 mensagens no tópico topic_1.
    4. Exibe uma saída semelhante à seguinte:

```
Connecting to a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-0c8ae2ff-cc87-49d2-a82a-ae7ba1d0ca5a'...
Connected!
Subscribing to topic 'topic_1'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 10 message(s)
Publishing message to topic 'topic_1': Hello World! [1]
Received message from topic 'topic_1': b'Hello World! [1]'
Publishing message to topic 'topic_1': Hello World! [2]
Received message from topic 'topic_1': b'Hello World! [2]'
Publishing message to topic 'topic_1': Hello World! [3]
Received message from topic 'topic_1': b'Hello World! [3]'
Publishing message to topic 'topic_1': Hello World! [4]
Received message from topic 'topic_1': b'Hello World! [4]'
Publishing message to topic 'topic_1': Hello World! [5]
Received message from topic 'topic_1': b'Hello World! [5]'
Publishing message to topic 'topic_1': Hello World! [6]
Received message from topic 'topic_1': b'Hello World! [6]'
Publishing message to topic 'topic_1': Hello World! [7]
Received message from topic 'topic_1': b'Hello World! [7]'
Publishing message to topic 'topic_1': Hello World! [8]
Received message from topic 'topic_1': b'Hello World! [8]'
Publishing message to topic 'topic_1': Hello World! [9]
Received message from topic 'topic_1': b'Hello World! [9]'
Publishing message to topic 'topic_1': Hello World! [10]
Received message from topic 'topic_1': b'Hello World! [10]'
10 message(s) received.
Disconnecting...
Disconnected!
```

Se você estiver com problemas para executar a aplicação de exemplo, examine the section called “Solucionar problemas com a aplicação de amostra”.

Também é possível adicionar o parâmetro `--verbosity Debug` à linha de comando para que o aplicativo de exemplo exiba mensagens detalhadas sobre sua operação. Essas informações podem fornecer a ajuda necessária para você corrigir o problema.

JavaScript

Para instalar e executar o aplicativo de exemplo

1. Na janela de linha de comando, navegue até o diretório `~/aws-iot-device-sdk-js-v2/samples/node/pub_sub` criado pelo SDK e instale o aplicativo de exemplo usando os seguintes comandos. O comando `npm install` invocará a criação da biblioteca `aws-crt`, que pode levar alguns minutos para ser concluída.

```
cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
npm install
```

2. Na janela da linha de comando, substitua *your-iot-endpoint* conforme indicado e execute esse comando.

```
node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --
cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
endpoint
```

3. Observe que o aplicativo de exemplo:

   1. Conecta-se ao AWS IoT serviço da sua conta.
   2. Assina o tópico de mensagens topic_1 e exibe as mensagens recebidas sobre esse tópico.
   3. Publica 10 mensagens no tópico topic_1.
   4. Exibe uma saída semelhante à seguinte:

```
Publish received on topic topic_1
{"message":"Hello world!","sequence":1}
Publish received on topic topic_1
{"message":"Hello world!","sequence":2}
Publish received on topic topic_1
{"message":"Hello world!","sequence":3}
Publish received on topic topic_1
{"message":"Hello world!","sequence":4}
Publish received on topic topic_1
{"message":"Hello world!","sequence":5}
Publish received on topic topic_1
{"message":"Hello world!","sequence":6}
Publish received on topic topic_1
```

```
{"message":"Hello world!","sequence":7}
Publish received on topic topic_1
{"message":"Hello world!","sequence":8}
Publish received on topic topic_1
{"message":"Hello world!","sequence":9}
Publish received on topic topic_1
{"message":"Hello world!","sequence":10}
```

Se você estiver com problemas para executar a aplicação de exemplo, examine the section called “Solucionar problemas com a aplicação de amostra”.

Também é possível adicionar o parâmetro `--verbosity Debug` à linha de comando para que o aplicativo de exemplo exiba mensagens detalhadas sobre sua operação. Essas informações podem fornecer a ajuda necessária para você corrigir o problema.

Exibir mensagens do aplicativo de amostra no AWS IoT console

É possível ver as mensagens do aplicativo de exemplo à medida que elas passam pelo agente de mensagens usando o cliente de teste MQTT no console do AWS IoT .

Para visualizar as mensagens MQTT publicadas pelo aplicativo de exemplo

1. Consulte Exibir mensagens MQTT com o cliente AWS IoT MQTT. Isso ajudará você a aprender a usar o cliente de teste MQTT no console do AWS IoT para visualizar as mensagens MQTT à medida que elas passam pelo agente de mensagens.
2. Abra o cliente de teste MQTT no console do AWS IoT .
3. Assine o tópico topic_1.
4. Na janela de linha de comando, execute o aplicativo de exemplo novamente e observe as mensagens no cliente MQTT no console do AWS IoT .

   Python

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples
   python3 pubsub.py --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --
   cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
   endpoint
   ```

JavaScript

```
cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --
cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
endpoint
```

Solucionar problemas com a aplicação de amostra

Se encontrar um erro ao tentar executar o aplicativo de exemplo, veja a seguir algumas objetos a serem verificadas.

Verifique o certificado

Se o certificado não estiver ativo, não AWS IoT aceitará nenhuma tentativa de conexão que o use para autorização. Na criação do certificado, é fácil ignorar o botão Ativar. Felizmente, você pode ativar seu certificado no console do AWS IoT.

Para verificar a ativação do certificado

1. No console do AWS IoT, no menu esquerdo, selecione Segurança e, depois, selecione Certificados.
2. Na lista de certificados, localize o certificado criado para o exercício e verifique seu status na coluna Status.

   Caso você não se lembre do nome do certificado, verifique se há algum que esteja Inativo para ver se pode ser o que você está usando.

   Selecione o certificado na lista para abrir sua página de detalhes. Na página de detalhes, é possível ver a Data de criação para ajudá-lo a identificar o certificado.
3. Para ativar um certificado inativo, na página de detalhes do certificado, selecione Ações e, depois, selecione Ativar.

Se você encontrou o certificado correto e ele está ativo, mas continua com problemas para executar o aplicativo de exemplo, verifique sua política conforme descrito pela próxima etapa.

Também é possível tentar criar um objeto nova e um novo certificado seguindo as etapas de the section called “Criar um objeto”. Se criar um objeto nova, será preciso criar um novo nome para ela e baixar os novos arquivos de certificado em seu dispositivo.

Verifique a política anexada ao certificado

As políticas autorizam ações em AWS IoT. Se o certificado usado para se conectar ao AWS IoT não tiver uma política, ou se não tiver uma política que permita a conexão, a conexão será recusada mesmo se o certificado estiver ativo.

Para verificar as políticas que estão anexadas a um certificado

1. Localize o certificado conforme descrito no item anterior e abra sua página de detalhes.
2. No menu esquerdo da página de detalhes do certificado, selecione Políticas para visualizar as políticas que estão anexadas ao certificado.
3. Caso não haja políticas anexadas ao certificado, adicione uma selecionando o menu Ações e, depois, selecionando Anexar política.

   Selecione a política criada anteriormente em the section called “Crie AWS IoT recursos”.
4. Se houver uma política anexada, selecione o bloco da política para abrir sua página de detalhes.

   Na página de detalhes, examine o Documento da política para se certificar de que ele contém as mesmas informações criadas em the section called “Crie uma AWS IoT política”.

Verifique a linha de comando

Certifique-se de ter usado a linha de comando correta para o seu sistema. Os comandos usados em sistemas Linux e macOS são, em geral, diferentes dos usados em sistemas Windows.

Verifique o endereço do endpoint

Revise o comando inserido e verifique novamente se o endereço do endpoint em seu comando coincide com aquele do console do AWS IoT.

Verifique os nomes dos arquivos de certificado

Compare os nomes dos arquivos no comando digitado com os nomes dos arquivos de certificado no diretório `certs`.

Alguns sistemas podem exigir que nomes de arquivos estejam entre aspas para funcionar corretamente.

Verifique a instalação do SDK

Verifique de a instalação do SDK está completa e correta.

Em caso de dúvida, instale o SDK novamente no dispositivo. Na maioria dos casos, basta encontrar a seção do tutorial intitulada Instalar o SDK do AWS IoT dispositivo ***SDK language*** e seguir o procedimento novamente.

Se você estiver usando o AWS IoT Device SDK para JavaScript, lembre-se de instalar os aplicativos de amostra antes de tentar executá-los. A instalação do SDK não instala os aplicativos de exemplo de forma automática. Os aplicativos de exemplo devem ser instalados manualmente depois da instalação do SDK.

Exibir mensagens MQTT com o cliente AWS IoT MQTT

Esta seção descreve como usar o cliente de teste AWS IoT MQTT no AWS IoT console para observar as mensagens MQTT enviadas e recebidas pelo. AWS IoT O exemplo usado nesta seção está relacionado aos exemplos usados emIntrodução aos AWS IoT Core tutoriais; no entanto, você pode substituir o *topicName* usado nos exemplos por qualquer nome de tópico ou filtro de tópico usado pela sua solução de IoT.

Os dispositivos publicam mensagens MQTT que são identificadas por tópicos para comunicar seu estado e AWS IoT publicam mensagens MQTT para informar os dispositivos e aplicativos sobre mudanças e eventos. AWS IoTÉ possível usar o cliente MQTT para se inscrever nesses tópicos e observar as mensagens à medida que elas ocorrem. Você também pode usar o cliente de teste MQTT para publicar mensagens MQTT em dispositivos e serviços assinados em seu. Conta da AWS

Conteúdo

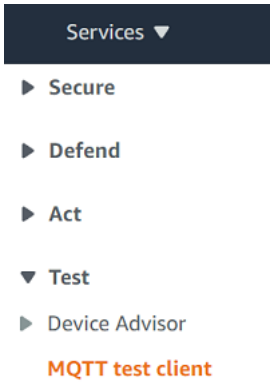
- Visualizar mensagens MQTT no cliente MQTT
- Publicar mensagens MQTT do cliente MQTT
- Testar assinaturas compartilhadas no cliente MQTT

Visualizar mensagens MQTT no cliente MQTT

O procedimento a seguir explica como se inscrever em um tópico específico do MQTT no qual seu dispositivo publica mensagens e visualizá-las no console de AWS IoT.

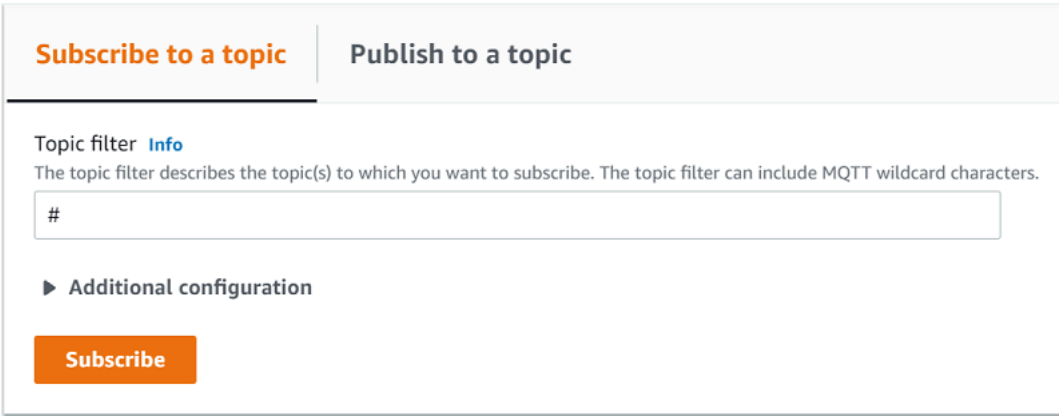
Para visualizar mensagens MQTT no cliente de teste MQTT

1. No console do AWS IoT, no menu esquerdo, selecione Testar e, depois, selecione Cliente de teste MQTT.

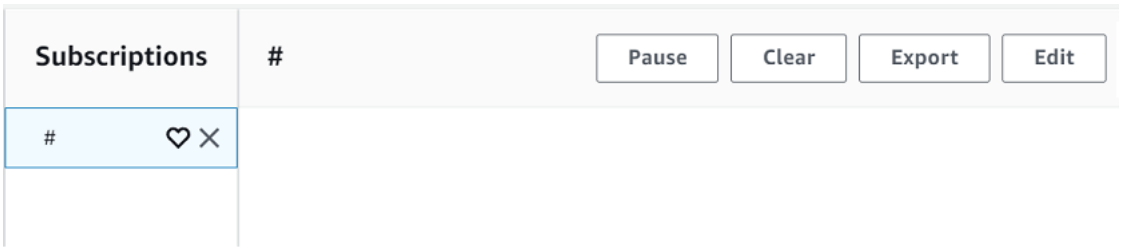


2. Na guia Inscrever-se em um tópico, insira *topicName* para assinar o tópico no qual seu dispositivo publica. Para começar a usar o aplicativo de exemplo, assine **#**, que assina todos os tópicos de mensagens.

   Continuando com o exemplo de introdução, na guia Assinar um tópico, no campo Filtro de tópicos, insira **#** e selecione Assinar.



   A página de log de mensagens do tópico, #, abrirá e **#** aparecerá na lista Assinaturas. Se o dispositivo que você configurou the section called “Configurar o dispositivo” estiver executando o programa de exemplo, você deverá ver as mensagens para as quais ele envia AWS IoT no registro de mensagens #. As entradas do registro de mensagens aparecerão abaixo da seção Publicar quando as mensagens com o tópico inscrito forem recebidas por AWS IoT.



3. Na página de log de mensagens #, também é possível publicar mensagens em um tópico, mas você precisará especificar o nome do tópico. Não é possível publicar no tópico #.

   As mensagens publicadas nos tópicos assinados aparecem no log de mensagens à medida que são recebidas, com a mensagem mais recente em primeiro lugar.

Solução de problemas com mensagens MQTT

Usar o filtro de tópicos curinga

Se suas mensagens não forem exibidas no log de mensagens conforme esperado, tente assinar um filtro de tópicos curinga, conforme descrito em Filtros de nomes de tópicos. O filtro de tópico curinga de vários níveis MQTT é o sinal de jogo da velha ou hashtag (#) e ele pode ser usado como filtro de tópicos no campo Tópico de assinatura.

Ao assinar o filtro de tópicos #, você assina todos os tópicos recebidos pelo agente de mensagens. Você pode restringir o filtro substituindo elementos do caminho do filtro de tópicos por um caractere curinga de vários níveis # ou pelo caractere curinga de nível único '+'.

Ao usar curingas em um filtro de tópicos

- O caractere curinga de vários níveis deve ser o último caractere no filtro de tópicos.
- O caminho do filtro de tópicos pode ter apenas um caractere curinga de nível único por nível de tópico.

Por exemplo:

Filtro de tópicos	Exibe mensagens com
#	Qualquer nome de tópico
`topic_1/#`	Um nome de tópico que inicia com `topic_1/`

Filtro de tópicos	Exibe mensagens com
`topic_1/level_2/#`	Um nome de tópico que inicia com `topic_1/level_2/`
`topic_1/+/level_3`	Um nome de tópico que inicia com `topic_1/`, termina com `/level_3` e tem um elemento de qualquer valor intermediário.

Para acessar mais informações sobre filtros de tópicos, consulte Filtros de nomes de tópicos.

Verificar se há erros no nome do tópico

Os nomes de tópicos MQTT e os filtros de tópicos diferenciam letras maiúsculas de minúsculas. Se, por exemplo, seu dispositivo estiver publicando mensagens em `Topic_1` (com T maiúsculo) em vez de `topic_1`, que você assinou, suas mensagens não serão exibidas no cliente de teste MQTT. Assinar o filtro de tópicos curinga, entretanto, mostraria que o dispositivo está publicando mensagens e você poderia ver que ele estava usando um nome de tópico diferente do esperado.

Publicar mensagens MQTT do cliente MQTT

Como publicar uma mensagem em um tópico MQTT

1. Na página do cliente de teste do MQTT, na guia Publicar em um tópico, no campo Nome do tópico, insira a *topicName* mensagem. Neste exemplo, use **my/topic**.

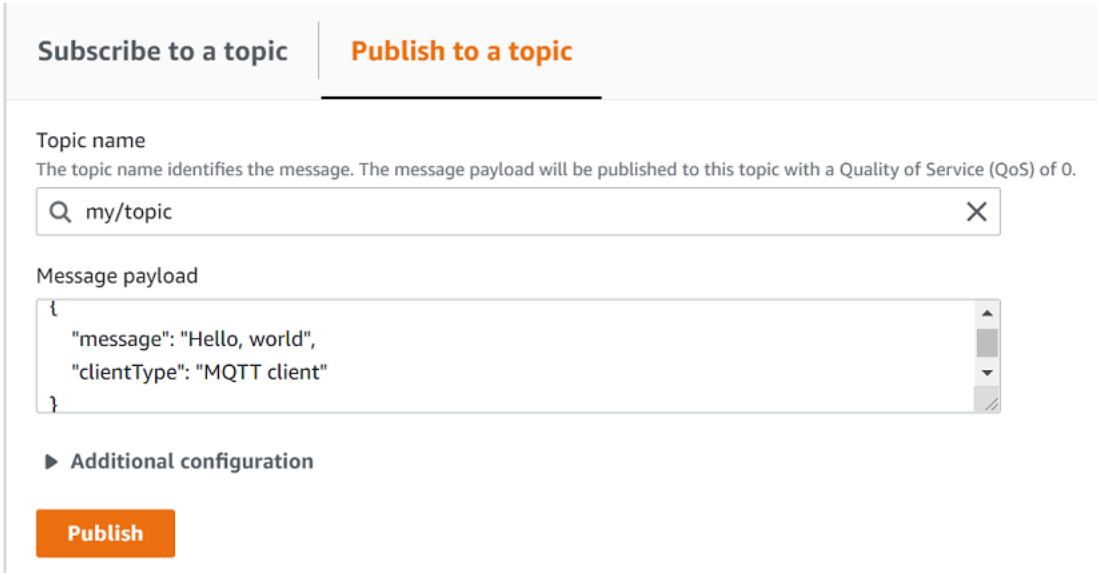
   > **Note**
   >
   > Não use informações de identificação pessoal em nomes de tópicos, seja no cliente de teste MQTT ou na implementação do seu sistema. Os nomes de tópicos pode surgir em comunicações e relatórios não criptografados.

2. Na janela de carga da mensagem, insira o JSON a seguir:

```
{
    "message": "Hello, world",
    "clientType": "MQTT test client"
}
```

3. Selecione Publicar para publicar sua mensagem no AWS IoT.

> **Note**
>
> Certifique-se de ser assinante do tópico my/topic antes de publicar a mensagem.



4. Na lista Assinaturas, selecione my/topic para ver a mensagem. Você deve poder ver a mensagem aparecer no cliente de teste MQTT abaixo da janela de carga da mensagem de publicação.

Você pode publicar mensagens MQTT *topicName* em outros tópicos alterando o campo Nome do tópico e escolhendo o botão Publicar.

> **Important**
>
> Quando você cria várias assinaturas com tópicos sobrepostos (por exemplo, sonda1/temperatura e sonda1/#), uma única mensagem publicada em um tópico correspondente às duas assinaturas pode ser entregue várias vezes, uma para cada assinatura sobreposta.

Testar assinaturas compartilhadas no cliente MQTT

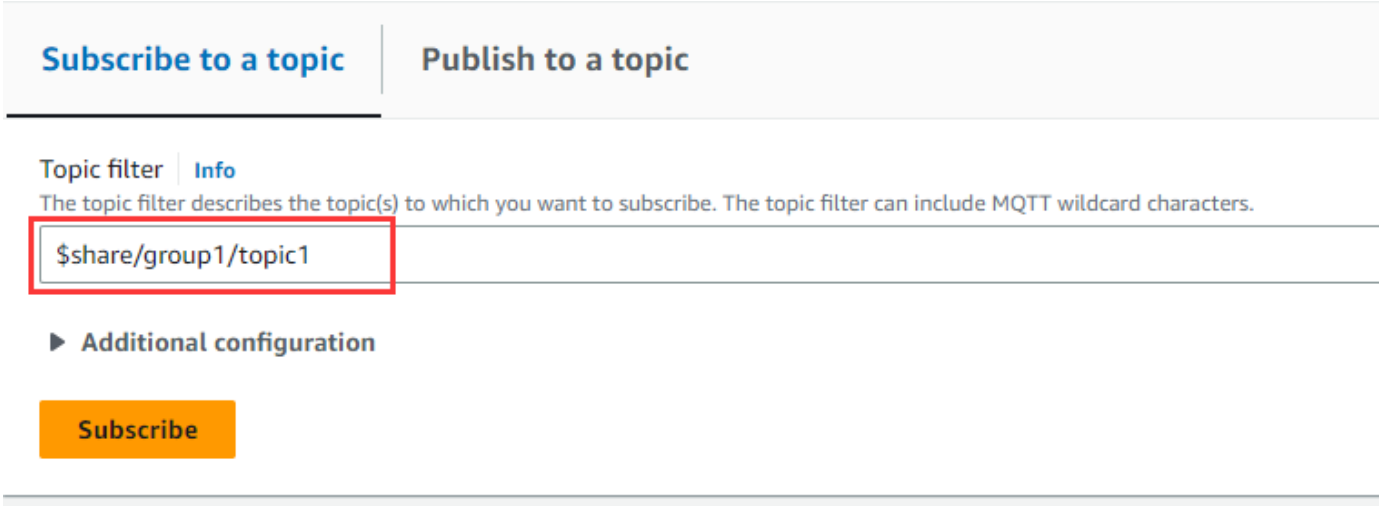
Esta seção descreve como usar o cliente AWS IoT MQTT no AWS IoT console para observar as mensagens MQTT enviadas e recebidas AWS IoT usando assinaturas compartilhadas. ???permitem que vários clientes compartilhem uma assinatura de um tópico com apenas um cliente recebendo mensagens publicadas nesse tópico usando uma distribuição aleatória. Para simular vários clientes MQTT (neste exemplo, dois clientes MQTT) compartilhando a mesma assinatura, você abre o cliente AWS IoT MQTT no AWS IoT console a partir de vários navegadores da web. O exemplo usado nesta seção não está relacionado aos exemplos contidos em Introdução aos AWS IoT Core tutoriais. Para acessar mais informações, consulte Assinaturas compartilhadas.

Para compartilhar uma assinatura de um tópico MQTT

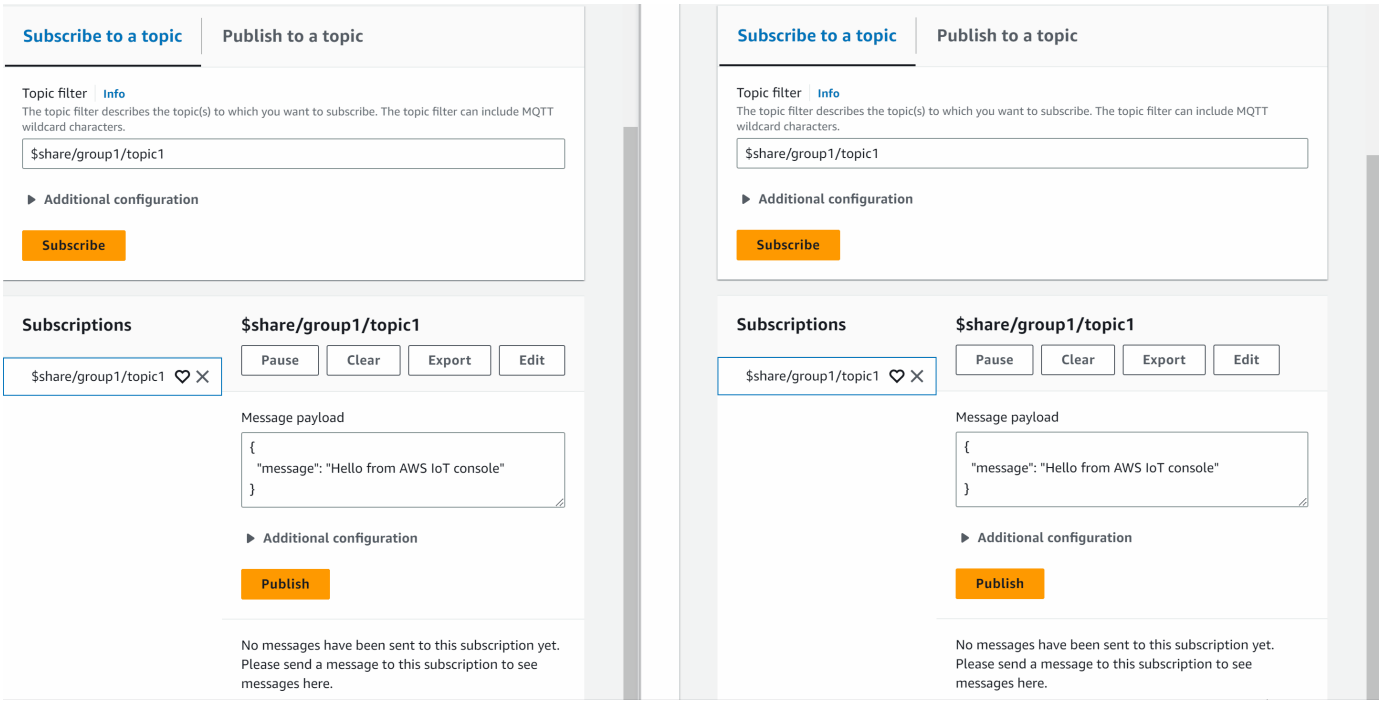
1. No console do AWS IoT, no painel de navegação, selecione Testar e, em seguida, selecione cliente de teste MQTT.
2. Na guia Inscrever-se em um tópico, insira *topicName* para assinar o tópico no qual seu dispositivo publica. Para usar assinaturas compartilhadas, assine um filtro de tópicos de uma assinatura compartilhada da seguinte maneira:

   ```
   $share/{ShareName}/{TopicFilter}
   ```

   Um exemplo de filtro de tópicos pode ser **$share/group1/topic1**, que assina o tópico de mensagens **topic1**.



3. Abra outro navegador da Web e repita as etapas 1 e 2. Assim, você estará simulando dois clientes MQTT diferentes que compartilham a mesma assinatura **`$share/group1/topic1`**.
4. Escolha um cliente MQTT, na guia Publicar em um tópico, no campo Nome do tópico, insira o *`topicName`* da sua mensagem. Neste exemplo, use **`topic1`**. Experimente publicar a mensagem algumas vezes. Na lista Assinaturas de ambos os clientes MQTT, você deve poder ver que os clientes recebem a mensagem usando uma distribuição randômica. No exemplo a seguir, publicamos a mensagem "Hello from AWS IoT console" três vezes. O cliente MQTT da esquerda recebeu a mensagem duas vezes e o cliente MQTT da direita recebeu a mensagem uma única vez.



AWS IoT tutoriais

Os AWS IoT tutoriais são divididos em dois caminhos de aprendizado para apoiar dois objetivos diferentes. Escolha o melhor caminho de aprendizado para sua meta.

- Você quer criar um proof-of-concept para testar ou demonstrar uma ideia de AWS IoT solução

  Para demonstrar tarefas e aplicativos comuns de IoT usando o AWS IoT Device Client em seus dispositivos, siga o caminho de the section called “Criando demonstrações com o AWS IoT Device Client” aprendizado. O AWS IoT Device Client fornece software de dispositivo com o qual você pode aplicar seus próprios recursos de nuvem para demonstrar uma end-to-end solução com desenvolvimento mínimo.

  Para obter informações sobre o AWS IoT Device Client, consulte o AWS IoT Device Client.
- Você quer aprender a criar software de produção para implantar sua solução

  Para criar seu próprio software de solução que atenda aos seus requisitos específicos usando um AWS IoT Device SDK, siga o plano de the section called “Construindo soluções com o AWS IoT dispositivo SDKs” aprendizado.

  Para obter informações sobre o AWS IoT dispositivo disponível SDKs, consulte???. Para obter informações sobre o AWS SDKs, consulte Ferramentas para desenvolver AWS.

AWS IoT opções de plano de aprendizado tutorial

- Criando demonstrações com o AWS IoT Device Client
- Construindo soluções com o AWS IoT dispositivo SDKs

Criando demonstrações com o AWS IoT Device Client

Os tutoriais neste plano de aprendizado orientam você pelas etapas para desenvolver um software de demonstração usando o AWS IoT Device Client. O AWS IoT Device Client fornece software que é executado em seu dispositivo de IoT para testar e demonstrar aspectos de uma solução de IoT baseada em. AWS IoT

O objetivo desses tutoriais é facilitar a exploração e a experimentação para que você tenha certeza de que a solução é AWS IoT compatível com sua solução antes de desenvolver o software do seu dispositivo.

O que você aprenderá nesses tutoriais:

- Como preparar um Raspberry Pi para uso como um dispositivo de IoT com AWS IoT
- Como demonstrar AWS IoT recursos usando o AWS IoT Device Client em seu dispositivo

Neste plano de aprendizado, você instalará o AWS IoT Device Client em seu próprio Raspberry Pi e criará os AWS IoT recursos na nuvem para demonstrar ideias de soluções de IoT. Embora os tutoriais desse plano de aprendizado demonstrem atributos usando um Raspberry Pi, eles explicam as metas e os procedimentos para ajudar você a adaptá-los a outros dispositivos.

Pré-requisitos para criar demonstrações com o Device Client AWS IoT

Esta seção descreve o que você precisará ter antes de iniciar os tutoriais neste plano de aprendizado.

Para fazer os tutoriais desse plano de aprendizado, você precisará:

- Um Conta da AWS

  Você pode usar o existente Conta da AWS, se tiver um, mas talvez seja necessário adicionar funções ou permissões adicionais para usar os AWS IoT recursos que esses tutoriais usam.

  Se você precisar criar um novo Conta da AWS, consultethe section called “Configurar Conta da AWS”.
- Um Raspberry Pi ou dispositivo IoT compatível

  Os tutoriais usam o Raspberry Pi porque, além de seus diferentes formatos, ele é amplamente usado e um dispositivo de demonstração relativamente barato. Os tutoriais foram testados no Raspberry Pi 3 Modelo B+, no Raspberry Pi 4 Modelo B e em uma instância do Amazon EC2 executando o Ubuntu Server 20.04 LTS (HVM). Para usar AWS CLI e executar os comandos, recomendamos que você use a versão mais recente do sistema operacional Raspberry Pi (Raspberry Pi OS (64 bits) ou OS Lite). Versões anteriores do SO podem funcionar, mas nós não as testamos.

> **Note**
>
> Os tutoriais explicam os objetivos de cada etapa para ajudar você a adaptá-los ao hardware de IoT que ainda não testamos. No entanto, eles não descrevem especificamente como adaptá-los a outros dispositivos.

- Familiaridade com o sistema operacional do dispositivo de IoT

  As etapas desses tutoriais pressupõem que você esteja familiarizado com o uso de comandos e operações básicas do Linux na interface de linha de comando suportada por um Raspberry Pi. Se você não estiver familiarizado com essas operações, talvez queira dedicar mais tempo para concluir os tutoriais.

  Para concluir esses tutoriais, você já deve entender como:

  - Executar com segurança as operações básicas do dispositivo, como montar e conectar componentes, conectar o dispositivo às fontes de alimentação necessárias e instalar e remover cartões de memória.
  - Carregar e baixar o software e os arquivos do sistema para o dispositivo. Se o dispositivo não usar um dispositivo de armazenamento removível, como um cartão microSD, você precisará saber como se conectar ao dispositivo e fazer o upload e o download do software e dos arquivos do sistema para o dispositivo.
  - Conecte seu dispositivo às redes nas quais você planeja usá-lo.
  - Conecte-se ao seu dispositivo a partir de outro computador usando um terminal SSH ou programa similar.
  - Use uma interface de linha de comando para criar, copiar, mover, renomear e definir as permissões de arquivos e diretórios no dispositivo.
  - Instale novos programas no dispositivo.
  - Transfira arquivos de e para o seu dispositivo usando ferramentas como FTP ou SCP.
- Um ambiente de desenvolvimento e teste para sua solução de IoT

  Os tutoriais descrevem o software e o hardware necessários; no entanto, presumem que você poderá realizar operações que talvez não estejam descritas explicitamente. Exemplos desse hardware e dessas operações incluem:

  - Um computador host local para baixar e armazenar arquivos

Para o Raspberry Pi, geralmente é um computador pessoal ou laptop que pode ler e gravar em cartões de memória microSD. O computador host local deve:

- Estar conectado à Internet.
- Ter a AWS CLI instalada e configurada.
- Tenha um navegador da Web compatível com o AWS console.

- Uma forma de conectar seu computador host local ao seu dispositivo para se comunicar com ele, inserir comandos e transferir arquivos

  No Raspberry Pi, isso geralmente é feito usando SSH e SCP do computador host local.

- Um monitor, mouse e teclado para se conectar ao dispositivo de IoT

  Eles podem ser úteis, mas não são necessários para concluir os tutoriais.

- Uma forma de seu computador host local e seus dispositivos de IoT se conectarem à Internet

  Pode ser uma conexão de rede com fio ou sem fio a um roteador ou gateway conectado à Internet. O host local também deve ser capaz de se conectar ao Raspberry Pi. Isso pode exigir que eles estejam na mesma rede local. Os tutoriais não mostram como configurar isso para seu dispositivo específico ou configuração de dispositivo, mas mostram como você pode testar essa conectividade.

- Acesso ao roteador da sua rede local para visualizar os dispositivos conectados

  Para concluir os tutoriais desse plano de aprendizado, você precisará encontrar o endereço IP do seu dispositivo de IoT.

  Em uma rede local, isso pode ser feito acessando a interface administrativa do roteador de rede ao qual seus dispositivos se conectam. Se você puder atribuir um endereço IP fixo ao seu dispositivo no roteador, poderá simplificar a reconexão após cada reinicialização do dispositivo.

  Se você tiver um teclado e um monitor conectados ao dispositivo, ifconfig poderá exibir o endereço IP do dispositivo.

  Se nada disso for uma opção, você precisará encontrar uma maneira de identificar o endereço IP do dispositivo após cada reinicialização.

Depois de ter todos os seus materiais, continue para the section called “Preparar-se para usar o IoT Device Client”.

Tutoriais neste plano de aprendizado

- Tutorial: Preparando seus dispositivos para o AWS IoT Device Client
- Tutorial: Instalando e configurando o AWS IoT Device Client
- Tutorial: Demonstre a comunicação de mensagens MQTT com o AWS IoT Device Client
- Tutorial: Demonstre ações remotas (trabalhos) com o AWS IoT Device Client
- Tutorial: Limpando depois de executar os tutoriais do AWS IoT Device Client

Tutorial: Preparando seus dispositivos para o AWS IoT Device Client

Este tutorial orienta você na inicialização do Raspberry Pi para prepará-lo para os tutoriais subsequentes neste percurso de aprendizado.

O objetivo desse tutorial é instalar a versão atual do sistema operacional do dispositivo e garantir que você possa se comunicar com o dispositivo no contexto do ambiente de desenvolvimento.

Pré-requisitos

Antes de começar este tutorial, verifique se você tem os itens listados em the section called “Pré-requisitos para criar demonstrações com o Device Client AWS IoT” disponíveis e prontos para uso.

A conclusão desse tutorial requer cerca de 90 minutos.

Neste tutorial, você vai:

- Instalar e atualizar o sistema operacional do dispositivo.
- Instalar e verificar qualquer software adicional necessário para executar os tutoriais.
- Testar a conectividade do dispositivo e instalar os certificados necessários.

Depois de concluir este tutorial, o próximo tutorial prepara seu dispositivo para as demonstrações que usam o AWS IoT Device Client.

Procedimentos deste tutorial

- Instalar e atualizar o sistema operacional do dispositivo
- Instalar e verificar o software necessário no seu dispositivo
- Testar o dispositivo e salvar o certificado de CA da Amazon

Instalar e atualizar o sistema operacional do dispositivo

Os procedimentos nesta seção descrevem como inicializar o cartão microSD que o Raspberry Pi usa como unidade do sistema. O cartão microSD do Raspberry Pi contém o software do sistema operacional (OS), bem como espaço para o armazenamento de arquivos da aplicação. Se você não estiver usando um Raspberry Pi, siga as instruções do dispositivo para instalar e atualizar o software do sistema operacional do dispositivo.

Depois de concluir esta seção, você poderá iniciar o dispositivo de IoT e conectar-se a ele por meio do programa de terminal no computador host local.

Equipamentos necessários:

- O ambiente local de desenvolvimento e testes
- Um Raspberry Pi, ou dispositivo de IoT, que pode se conectar à Internet
- Um cartão de memória microSD com capacidade de pelo menos 8 GB ou armazenamento suficiente para o sistema operacional e o software necessário.

> **Note**
>
> Ao selecionar um cartão microSD para esses exercícios, escolha um que seja tão grande quanto necessário, mas tão pequeno quanto possível.
> Um cartão SD pequeno será mais rápido de fazer backup e atualizar. No Raspberry Pi, você não precisará de mais do que um cartão microSD de 8 GB para esses tutoriais.
> Se você precisar de mais espaço para a aplicação específica, os arquivos de imagem menores salvos nesses tutoriais poderão redimensionar o sistema de arquivos em um cartão maior para usar todo o espaço compatível do cartão que você escolher.

Equipamento opcional:

- Um teclado USB conectado ao Raspberry Pi
- Um monitor HDMI e um cabo para conectar o monitor ao Raspberry Pi

Procedimentos desta seção:

- Carregue o sistema operacional do dispositivo no cartão microSD
- Inicie o dispositivo de IoT com o novo sistema operacional

- Conectar o computador host local ao dispositivo

Carregue o sistema operacional do dispositivo no cartão microSD

Esse procedimento usa o computador host local para carregar o sistema operacional do dispositivo em um cartão microSD.

> **Note**
>
> Se o dispositivo não usar uma mídia de armazenamento removível para o sistema operacional, instale o sistema operacional usando o procedimento para esse dispositivo e continue na the section called “Iniciar o dispositivo IoT”.

Para instalar o sistema operacional no Raspberry Pi

1. No computador host local, baixe e descompacte a imagem do sistema operacional Raspberry Pi que você deseja usar. As versões mais recentes estão disponíveis em https://www.raspberrypi.com/software/sistemas operacionais/

   Como escolher uma versão do sistema operacional Raspberry Pi

   Este tutorial usa a versão Raspberry Pi OS Lite porque é a menor versão compatível com esses tutoriais neste percurso de aprendizado. Essa versão do sistema operacional Raspberry Pi tem apenas uma interface de linha de comando e não tem uma interface gráfica de usuário. Uma versão do sistema operacional Raspberry Pi mais recente com uma interface gráfica de usuário também funcionará com esses tutoriais; no entanto, os procedimentos descritos neste percurso de aprendizado usam somente a interface da linha de comando para realizar operações no Raspberry Pi.
2. Insira o cartão microSD no computador host local.
3. Usando uma ferramenta de imagem de cartão SD, grave o arquivo de imagem do sistema operacional descompactado no cartão microSD.
4. Depois de gravar a imagem do sistema operacional Raspberry Pi no cartão microSD:

   a. Abra a partição BOOT no cartão microSD em uma janela da linha de comando ou janela do explorador de arquivos.

   b. Na partição BOOT do cartão microSD, no diretório raiz, crie um arquivo vazio chamado `ssh` sem extensão de arquivo e sem conteúdo. Isso faz com que o Raspberry Pi ative as comunicações SSH na primeira vez em que for iniciado.

5. Ejete o cartão microSD e remova-o com segurança do computador host local.

O cartão microSD está pronto para the section called “Iniciar o dispositivo IoT”.

Inicie o dispositivo de IoT com o novo sistema operacional

Este procedimento instala o cartão microSD e inicia o Raspberry Pi pela primeira vez usando o sistema operacional baixado.

Para iniciar o dispositivo de IoT com o novo sistema operacional

1. Com a alimentação desconectada do dispositivo, insira o cartão microSD da etapa anterior, the section called “Carregar o SO”, no Raspberry Pi.
2. Conecte o dispositivo a uma rede com fios.
3. Esses tutoriais interagirão com o Raspberry Pi por meio do computador host local usando um terminal SSH.

   Se você também quiser interagir diretamente com o dispositivo, você pode:

   a. Conectar um monitor HDMI a ele para assistir às mensagens do console do Raspberry Pi antes de conectar a janela do terminal do computador host local ao Raspberry Pi.
   b. Conectar um teclado USB a ele se quiser interagir diretamente com o Raspberry Pi.

4. Conectar a alimentação ao Raspberry Pi e esperar cerca de um minuto para que ele seja inicializado.

   Se você tiver um monitor conectado ao Raspberry Pi, poderá ver processo de inicialização nele.

5. Descubra o endereço IP do dispositivo:

   - Se você conectou um monitor HDMI ao Raspberry Pi, o endereço IP aparecerá nas mensagens exibidas no monitor
   - Se você tiver acesso ao roteador ao qual o Raspberry Pi está conectado, poderá ver o endereço na interface de administração do roteador.

Depois de ter o endereço IP do Raspberry Pi, você estará pronto para the section called “Conectar seu computador host”.

Conectar o computador host local ao dispositivo

Este procedimento usa o programa de terminal no computador host local para se conectar ao Raspberry Pi e alterar a senha padrão.

Para conectar o computador host local ao dispositivo

1. No computador host local, abra o programa do terminal SSH:

   - Windows: `PuTTY`
   - Linux/macOS: `Terminal`

   > **Note**
   >
   > O PuTTY não é instalado automaticamente no Windows. Se não estiver no computador, talvez seja necessário baixá-lo e instalá-lo.

2. Conecte o programa do terminal ao endereço IP do Raspberry Pi e faça login usando as credenciais padrão.

   ```
   username: pi
   password: raspberry
   ```

3. Depois de fazer login no Raspberry Pi, altere a senha do usuário `pi`.

   ```
   passwd
   ```

   Siga os prompts para alterar a senha.

   ```
   Changing password for pi.
   Current password: raspberry
   New password: YourNewPassword
   Retype new password: YourNewPassword
   passwd: password updated successfully
   ```

Depois de inserir o prompt da linha de comando do Raspberry Pi na janela do terminal e alterar a senha, você estará pronto para continuar em the section called “Instalar e verificar o software necessário”.

Instalar e verificar o software necessário no seu dispositivo

Os procedimentos nesta seção continuam com a seção anterior para atualizar o sistema operacional do Raspberry Pi e instalar o software no Raspberry Pi que será usado na próxima seção para criar e instalar o AWS IoT Device Client.

Depois de concluir esta seção, seu Raspberry Pi terá um sistema up-to-date operacional, o software exigido pelos tutoriais neste plano de aprendizado, e será configurado para sua localização.

Equipamentos necessários:

- O ambiente local de desenvolvimento e teste da seção anterior
- O Raspberry Pi usado na seção anterior
- O cartão de memória microSD da seção anterior

> **Note**
>
> O Raspberry Pi Model 3+ e o Raspberry Pi Model 4 podem executar todos os comandos descritos neste percurso de aprendizado. Se seu dispositivo de IoT não conseguir compilar o software ou executar o AWS Command Line Interface, talvez seja necessário instalar os compiladores necessários no computador host local para criar o software e depois transferi-lo para o dispositivo de IoT. Para obter mais informações sobre como instalar e compilar software para o dispositivo, consulte a documentação do software do dispositivo.

Procedimentos desta seção:

- Atualizar o software do sistema operacional
- Instale as aplicações e bibliotecas obrigatórias
- (Opcional) Salve a imagem do cartão microSD

Atualizar o software do sistema operacional

Esse procedimento atualiza o software do sistema operacional.

Para atualizar o software do sistema operacional no Raspberry Pi

Execute essas etapas na janela do terminal do computador host local.

1. Digite esses comandos para atualizar o software do sistema no Raspberry Pi.

```
sudo apt-get -y update
sudo apt-get -y upgrade
sudo apt-get -y autoremove
```

2. Atualize as configurações de localidade e fuso horário do Raspberry Pi (opcional).

   Digite esse comando para atualizar as configurações de localidade e fuso horário do dispositivo.

```
sudo raspi-config
```

   a. Para definir a localidade do dispositivo:

      i. Na tela Ferramenta de Configuração do Software Raspberry Pi (raspi-config), escolha a opção 5.

         5 Localisation Options Configure language and regional settings

         Use a tecla Tab para ir para <Select> e, em seguida, pressione a space bar.

      ii. No menu de opções de localização, escolha a opção L1.

         L1 Locale Configure language and regional settings

         Use a tecla Tab para ir para <Select> e, em seguida, pressione a space bar.

      iii. Na lista de opções de localidade, escolha as localidades que você deseja instalar no Raspberry Pi usando as teclas de seta para rolar e space bar para marcar as que deseja.

         Nos Estados Unidos, **en_US.UTF-8** é uma boa opção.

      iv. Depois de selecionar as localidades para o dispositivo, use a tecla Tab para escolher <OK> e pressione a space bar para exibir a página de confirmação de Configuração de localidades.

   b. Para definir o fuso horário do dispositivo:

      i. Na tela raspi-config, escolha a opção 5.

```
5 Localisation Options Configure language and regional settings
```

Use a tecla Tab para ir para <Select> e, em seguida, pressione a space bar.

ii. No menu de opções de localização, use a tecla de seta para escolher a opção L2:

```
L2 time zone Configure time zone
```

Use a tecla Tab para ir para <Select> e, em seguida, pressione a space bar.

iii. No menu Configuração de tzdata, escolha a área geográfica na lista.

Use a tecla Tab para ir para <OK> e, em seguida, pressione a space bar.

iv. Na lista de cidades, use as teclas de seta para escolher uma cidade no fuso horário.

Para definir o fuso horário, use a tecla Tab para ir para <OK> e, em seguida, pressione a space bar.

c. Quando terminar de atualizar as configurações, use a tecla Tab para acessar <Finish> e pressione a space bar para fechar a aplicação raspi-config.

3. Digite este comando para reiniciar o Raspberry Pi.

```
sudo shutdown -r 0
```

4. Aguarde a reinicialização do Raspberry Pi.
5. Depois que o Raspberry Pi for reiniciado, reconecte a janela do terminal no computador host local ao Raspberry Pi.

O software do sistema Raspberry Pi agora está configurado e você está pronto para continuar na the section called “Instalar aplicações e bibliotecas”.

Instale as aplicações e bibliotecas obrigatórias

Esse procedimento instala o software da aplicação e as bibliotecas que os tutoriais subsequentes usam.

Se você estiver usando um Raspberry Pi ou se puder compilar o software necessário no dispositivo de IoT, execute essas etapas na janela do terminal no computador host local. Se você precisar compilar software para o dispositivo de IoT no computador host local, analise a documentação do software do dispositivo de IoT para obter informações sobre como executar essas etapas no dispositivo.

Para instalar o software da aplicação e as bibliotecas no Raspberry Pi

1. Digite esse comando para instalar o software da aplicação e as bibliotecas.

```
sudo apt-get -y install build-essential libssl-dev cmake unzip git python3-pip
```

2. Insira esses comandos para confirmar que a versão correta do software foi instalada.

```
gcc --version
cmake --version
openssl version
git --version
```

3. Confirme se essas versões do software da aplicação estão instaladas:

   - `gcc` 9.3.0 ou posterior
   - `cmake` 3.10.x ou posterior
   - `OpenSSL`: 1.1.1 ou posterior
   - `git`: 2.20.1 ou posterior

Se o Raspberry Pi tiver versões aceitáveis do software da aplicação necessário, você está pronto para continuar na the section called “(Opcional) Salvar a imagem do microSD”.

(Opcional) Salve a imagem do cartão microSD

Ao longo dos tutoriais deste percurso de aprendizado, você encontrará esses procedimentos para salvar uma cópia da imagem do cartão microSD do Raspberry Pi em um arquivo no computador host local. Embora incentivadas, elas não são tarefas obrigatórias. Ao salvar a imagem do cartão microSD onde sugerido, você pode pular os procedimentos que precedem o ponto de salvamento neste percurso de aprendizado, o que pode economizar tempo se você precisar tentar algo novamente. A consequência de não salvar a imagem do cartão microSD periodicamente é que talvez você precise reiniciar os tutoriais do percurso de aprendizado desde o início se o cartão microSD estiver danificado ou se você acidentalmente definir uma aplicação ou as configurações incorretamente.

Neste ponto, o cartão microSD do Raspberry Pi tem um sistema operacional atualizado e o software básico da aplicação carregado. Você pode economizar o tempo necessário para concluir as etapas anteriores salvando o conteúdo do cartão microSD em um arquivo agora. Ter a imagem atual da imagem do cartão microSD do dispositivo permite que você comece a partir deste ponto para

continuar ou repetir um tutorial ou procedimento sem a necessidade de instalar e atualizar o software do zero.

Para salvar a imagem do cartão microSD em um arquivo

1. Digite este comando para desligar o Raspberry Pi.

   ```
   sudo shutdown -h 0
   ```

2. Depois que o Raspberry Pi for desligado completamente, desligue a energia.
3. Remova o cartão microSD do Raspberry Pi.
4. No computador host local:

   a. Insira o cartão microSD.

   b. Usando uma ferramenta de imagem de cartão SD, grave a imagem do cartão microSD em um arquivo.

   c. Depois que a imagem do cartão microSD for salva, ejete o cartão do computador host local.
5. Com a alimentação desconectada do Raspberry Pi, insira o cartão microSD no Raspberry Pi.
6. Ligue o Raspberry Pi.
7. Depois de esperar cerca de um minuto, no computador host local, reconecte a janela do terminal no computador host local que estava conectado ao Raspberry Pi. e, em seguida, faça login no Raspberry Pi.

Testar o dispositivo e salvar o certificado de CA da Amazon

Os procedimentos nesta seção continuam com a seção anterior para instalar o AWS Command Line Interface e o certificado da Autoridade de Certificação usado para autenticar suas conexões com AWS IoT Core.

Depois de concluir esta seção, você saberá que seu Raspberry Pi tem o software de sistema necessário para instalar o AWS IoT Device Client e que ele tem uma conexão ativa com a Internet.

Equipamentos necessários:

- O ambiente local de desenvolvimento e teste da seção anterior
- O Raspberry Pi usado na seção anterior
- O cartão de memória microSD da seção anterior

Procedimentos desta seção:

- Instale o AWS Command Line Interface
- Configure suas Conta da AWS credenciais
- Fazer download do certificado da CA raiz da Amazon
- (Opcional) Salve a imagem do cartão microSD

Instale o AWS Command Line Interface

Este procedimento instala o AWS CLI em seu Raspberry Pi.

Se você estiver usando um Raspberry Pi ou se puder compilar o software no dispositivo de IoT, execute essas etapas na janela do terminal no computador host local. Se você precisar compilar software para o dispositivo de IoT no computador host local, analise a documentação do software do dispositivo de IoT para obter informações sobre as bibliotecas que ele requer.

Para instalar o AWS CLI em seu Raspberry Pi

1. Use esses comandos para baixar e instalar a AWS CLI.

```
export PATH=$PATH:~/.local/bin # configures the path to include the directory with
 the AWS CLI
git clone https://github.com/aws/aws-cli.git # download the AWS CLI code from
 GitHub
cd aws-cli && git checkout v2 # go to the directory with the repo and checkout
 version 2
pip3 install -r requirements.txt # install the prerequisite software
```

2. Execute este comando para instalar AWS CLI o. Este comando pode levar até 15 minutos para ser concluído.

```
pip3 install . # install the AWS CLI
```

3. Execute esse comando para confirmar se a versão correta do AWS CLI foi instalada.

```
aws --version
```

   A versão do AWS CLI deve ser 2.2 ou posterior.

Se AWS CLI exibiu sua versão atual, você está pronto para continuarthe section called “Configurar as credenciais da conta”.

Configure suas Conta da AWS credenciais

Neste procedimento, você obterá Conta da AWS as credenciais e as adicionará para uso em seu Raspberry Pi.

Para adicionar suas Conta da AWS credenciais ao seu dispositivo

1. Obtenha um ID de chave de acesso e uma chave de acesso secreta Conta da AWS para autenticá-los AWS CLI em seu dispositivo.

   Se você é novo no AWS IAM, o https://aws.amazon.com/premiumsupport/knowledge-center/create-access-key/descreve o processo a ser executado no AWS console para criar credenciais do AWS IAM para usar em seu dispositivo.
2. Na janela do terminal no computador host local que está conectado ao Raspberry Pi e com as credenciais de ID da chave de acesso e Chave de acesso secreta do dispositivo:

   a. Execute o aplicativo AWS configure com este comando:

   ```
   aws configure
   ```

   b. Insira as credenciais e informações de configuração quando solicitado:

   ```
   AWS Access Key ID: your Access Key ID
   AWS Secret Access Key: your Secret Access Key
   Default region name: your Região da AWS code
   Default output format: json
   ```
3. Execute esse comando para testar o acesso do seu dispositivo ao seu Conta da AWS AWS IoT Core terminal.

   ```
   aws iot describe-endpoint --endpoint-type iot:Data-ATS
   ```

   Ele deve retornar seu endpoint AWS IoT de dados Conta da AWS específico, como este exemplo:

   ```
   {
       "endpointAddress": "a3EXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
   ```

```
}
```

Se você ver seu endpoint Conta da AWS de AWS IoT dados específico, seu Raspberry Pi tem a conectividade e as permissões para continuar. the section called “Fazer download do certificado da CA raiz da Amazon”

> **Important**
>
> Suas Conta da AWS credenciais agora estão armazenadas no cartão microSD do seu Raspberry Pi. Embora isso facilite as interações futuras para você e para o software que você criará nesses tutoriais, elas também serão salvas e duplicadas em qualquer imagem de cartão microSD que você criar após essa etapa, por padrão. AWS
> Para proteger a segurança de suas Conta da AWS credenciais, antes de salvar mais imagens do cartão microSD, considere apagar as credenciais `aws configure` executando novamente e inserindo caracteres aleatórios para o ID da chave de acesso e a chave de acesso secreta para evitar Conta da AWS que suas credenciais sejam comprometidas.
> Se você descobrir que salvou suas Conta da AWS credenciais inadvertidamente, poderá desativá-las no console do IAM. AWS

Fazer download do certificado da CA raiz da Amazon

Este procedimento baixa e salva uma cópia de um certificado da Autoridade de Certificação (CA) raiz da Amazon. O download desse certificado o salva para uso nos tutoriais subsequentes e também testa a conectividade do dispositivo com os serviços da AWS .

Para fazer download do certificado da CA raiz da Amazon

1. Execute o comando a seguir a fim de criar um diretório para o certificado.

   ```
   mkdir ~/certs
   ```

2. Execute este comando para fazer download do certificado da CA raiz da Amazon.

   ```
   curl -o ~/certs/AmazonRootCA1.pem https://www.amazontrust.com/repository/
   AmazonRootCA1.pem
   ```

3. Execute esses comandos para definir o acesso ao diretório do certificado e o arquivo.

```
chmod 745 ~
chmod 700 ~/certs
chmod 644 ~/certs/AmazonRootCA1.pem
```

4. Execute esse comando para ver o arquivo de certificado da CA no novo diretório.

   ```
   ls -l ~/certs
   ```

   Você deve ver uma entrada como essa. A data e a hora serão diferentes; no entanto, o tamanho do arquivo e todas as outras informações deverão ser iguais às mostradas aqui.

   ```
   -rw-r--r-- 1 pi pi 1188 Oct 28 13:02 AmazonRootCA1.pem
   ```

   Se o tamanho do arquivo não for 1188, verifique os parâmetros do comando curl. Você pode ter baixado um arquivo incorreto.

(Opcional) Salve a imagem do cartão microSD

Neste ponto, o cartão microSD do Raspberry Pi tem um sistema operacional atualizado e o software básico da aplicação carregado.

Para salvar a imagem do cartão microSD em um arquivo

1. Na janela do terminal do computador host local, limpe as credenciais da AWS .

   a. Execute o aplicativo AWS configure com este comando:

      ```
      aws configure
      ```

   b. Substitua as credenciais quando solicitado. Você pode deixar o Nome da região padrão e o Formato de saída padrão como estão pressionando Enter.

      ```
      AWS Access Key ID [****************YT2H]: XYXYXYXYX
      AWS Secret Access Key [****************9plH]: XYXYXYXYX
      Default region name [us-west-2]:
      Default output format [json]:
      ```

2. Digite este comando para desligar o Raspberry Pi.

```
sudo shutdown -h 0
```

3. Depois que o Raspberry Pi for desligado completamente, remova o conector de alimentação.
4. Remova o cartão microSD do dispositivo.
5. No computador host local:
   a. Insira o cartão microSD.
   b. Usando uma ferramenta de imagem de cartão SD, grave a imagem do cartão microSD em um arquivo.
   c. Depois que a imagem do cartão microSD for salva, ejete o cartão do computador host local.
6. Com a alimentação desconectada do Raspberry Pi, insira o cartão microSD no Raspberry Pi.
7. Ligue o dispositivo.
8. Após cerca de um minuto, no computador host local, reinicie a sessão da janela do terminal e faça login no dispositivo.

   Não insira novamente suas Conta da AWS credenciais ainda.

Depois de reiniciar e fazer login no Raspberry Pi, você estará pronto para continuar em the section called “Instalar e configurar o IoT Device Client”.

Tutorial: Instalando e configurando o AWS IoT Device Client

Este tutorial mostra a instalação e a configuração do AWS IoT Device Client e a criação dos AWS IoT recursos que você usará nesta e em outras demonstrações.

Para iniciar este tutorial:

- Prepare o computador host local e o Raspberry Pi com o tutorial anterior.

Este tutorial pode levar cerca de 90 minutos para ser concluído.

Quando você concluir este tópico:

- Seu dispositivo de IoT estará pronto para ser usado em outras demonstrações do AWS IoT Device Client.
- Você terá provisionado seu dispositivo de IoT em. AWS IoT Core

- Você terá baixado e instalado o AWS IoT Device Client em seu dispositivo.
- Você terá salvo uma imagem do cartão microSD do dispositivo que pode ser usada em tutoriais subsequentes.

Equipamentos necessários:

- O ambiente local de desenvolvimento e teste da seção anterior
- O Raspberry Pi usado na seção anterior
- O cartão de memória microSD do Raspberry Pi usado na seção anterior

Procedimentos deste tutorial

- Baixe e salve o AWS IoT Device Client
- Provisione seu Raspberry Pi em AWS IoT
- Configurar o cliente do AWS IoT dispositivo para testar a conectividade

Baixe e salve o AWS IoT Device Client

Os procedimentos nesta seção baixam o AWS IoT Device Client, compilam e instalam no seu Raspberry Pi. Depois de testar a instalação, você pode salvar a imagem do cartão microSD do Raspberry Pi para usar mais tarde quando quiser experimentar os tutoriais novamente.

Procedimentos desta seção:

- Baixe e crie o AWS IoT Device Client
- Crie os diretórios usados pelos tutoriais
- (Opcional) Salve a imagem do cartão microSD

Baixe e crie o AWS IoT Device Client

Este procedimento instala o AWS IoT Device Client no seu Raspberry Pi.

Execute esses comandos na janela do terminal do computador host local conectado ao Raspberry Pi.

Para instalar o AWS IoT Device Client em seu Raspberry Pi

1. Digite esses comandos para baixar e criar o AWS IoT Device Client no seu Raspberry Pi.

```
cd ~
git clone https://github.com/awslabs/aws-iot-device-client aws-iot-device-client
mkdir ~/aws-iot-device-client/build && cd ~/aws-iot-device-client/build
cmake ../
```

2. Execute esse comando para criar o AWS IoT Device Client. Este comando pode levar até 15 minutos para ser concluído.

   ```
   cmake --build . --target aws-iot-device-client
   ```

   As mensagens de aviso exibidas durante a compilação do AWS IoT Device Client podem ser ignoradas.

   Esses tutoriais foram testados com o AWS IoT Device Client integradogcc, versão (Raspbian 10.2.1-6+rpi1) 10.2.1 20210110 na versão de 30 de outubro de 2021 do Raspberry Pi OS (bullseye) on, versão (Raspbian 8.3.0-6+rpi1) 8.3.0 na versão de 7 de maio de 2021 do Raspberry Pi OS (gccbuster).
3. Depois que o AWS IoT Device Client terminar de criar, teste-o executando esse comando.

   ```
   ./aws-iot-device-client --help
   ```

Se você ver a ajuda da linha de comando para o AWS IoT Device Client, o AWS IoT Device Client foi criado com sucesso e está pronto para ser usado.

Crie os diretórios usados pelos tutoriais

Este procedimento cria os diretórios no Raspberry Pi que serão usados para armazenar os arquivos usados pelos tutoriais neste percurso de aprendizado.

Para criar os diretórios usados pelos tutoriais neste percurso de aprendizado:

1. Execute esses comandos para criar os diretórios necessários.

   ```
   mkdir ~/dc-configs
   mkdir ~/policies
   mkdir ~/messages
   mkdir ~/certs/testconn
   mkdir ~/certs/pubsub
   ```

```
mkdir ~/certs/jobs
```

2. Execute esses comandos para definir as permissões nos novos diretórios.

```
chmod 745 ~
chmod 700 ~/certs/testconn
chmod 700 ~/certs/pubsub
chmod 700 ~/certs/jobs
```

Depois de criar esses diretórios e definir permissões, continue em the section called “(Opcional) Salve a imagem do cartão microSD”.

(Opcional) Salve a imagem do cartão microSD

Neste ponto, o cartão microSD do seu Raspberry Pi tem um sistema operacional atualizado, o software básico do aplicativo e AWS IoT o Device Client.

Se você quiser voltar para tentar esses exercícios e tutoriais novamente, pule os procedimentos anteriores gravando a imagem do cartão microSD que você salvou com esse procedimento em um novo cartão microSD e continue com os tutoriais em the section called “Provisionar o Raspberry Pi”.

Para salvar a imagem do cartão microSD em um arquivo:

Na janela do terminal do computador host local conectado ao Raspberry Pi:

1. Confirme se suas Conta da AWS credenciais não foram armazenadas.

   a. Execute o aplicativo AWS configure com este comando:

   ```
   aws configure
   ```

   b. Se suas credenciais tiverem sido armazenadas (se forem exibidas no prompt), insira a string **XYXYXYXYX** quando solicitada, conforme mostrado aqui. Deixe o Nome da região padrão e o Formato de saída padrão em branco.

   ```
   AWS Access Key ID [****************YXYX]: XYXYXYXYX
   AWS Secret Access Key [****************YXYX]: XYXYXYXYX
   Default region name:
   Default output format:
   ```

2. Digite este comando para desligar o Raspberry Pi.

```
sudo shutdown -h 0
```

3. Depois que o Raspberry Pi for desligado completamente, remova o conector de alimentação.
4. Remova o cartão microSD do dispositivo.
5. No computador host local:

   a. Insira o cartão microSD.
   b. Usando uma ferramenta de imagem de cartão SD, grave a imagem do cartão microSD em um arquivo.
   c. Depois que a imagem do cartão microSD for salva, ejete o cartão do computador host local.

Você pode continuar com esse cartão microSD inserido em the section called “Provisionar o Raspberry Pi”.

Provisione seu Raspberry Pi em AWS IoT

Os procedimentos nesta seção começam com a imagem microSD salva que tem o AWS CLI e o AWS IoT Device Client instalado e criam os AWS IoT recursos e certificados de dispositivo que provisionam seu Raspberry Pi. AWS IoT

Instale o cartão microSD no Raspberry Pi

Este procedimento instala o cartão microSD com o software necessário carregado e configurado no Raspberry Pi e o configura para que Conta da AWS você possa continuar com os tutoriais neste caminho de aprendizado.

Use um cartão microSD de the section called “(Opcional) Salve a imagem do cartão microSD” que tenha o software necessário para os exercícios e tutoriais deste percurso de aprendizado.

Para instalar o cartão microSD no Raspberry Pi

1. Com a alimentação desconectada do Raspberry Pi, insira o cartão microSD no Raspberry Pi.
2. Ligue o Raspberry Pi.
3. Após cerca de um minuto, no computador host local, reinicie a sessão da janela do terminal e faça login no Raspberry Pi.
4. No computador host local, na janela do terminal e com as credenciais de ID da chave de acesso de Chave de acesso secreta do Raspberry Pi:

a. Execute o aplicativo AWS configure com este comando:

```
aws configure
```

b. Insira suas Conta da AWS credenciais e informações de configuração quando solicitado:

```
AWS Access Key ID [****************YXYX]: your Access Key ID
AWS Secret Access Key [****************YXYX]: your Secret Access Key
Default region name [us-west-2]: your Região da AWS code
Default output format [json]: json
```

Depois de restaurar suas Conta da AWS credenciais, você estará pronto para continuarthe section called “Provisione seu dispositivo em AWS IoT Core”.

Provisione seu dispositivo em AWS IoT Core

Os procedimentos nesta seção criam os AWS IoT recursos que provisionam seu Raspberry Pi em AWS IoT. Ao criar esses recursos, você deverá registrar várias informações. Essas informações são usadas pela configuração do AWS IoT Device Client no próximo procedimento.

Para que seu Raspberry Pi funcione AWS IoT, ele deve ser provisionado. O provisionamento é o processo de criar e configurar os AWS IoT recursos necessários para oferecer suporte ao seu Raspberry Pi como um dispositivo de IoT.

Com o Raspberry Pi ligado e reiniciado, conecte a janela do terminal no computador host local ao Raspberry Pi e conclua esses procedimentos.

Procedimentos desta seção:

- Crie e baixe arquivos de certificado
- Crie AWS IoT recursos

Crie e baixe arquivos de certificado

Este procedimento cria os arquivos de certificado de dispositivo para esta demonstração.

Para criar e baixar os arquivos de certificado de dispositivo para o Raspberry Pi

1. Na janela do terminal do computador host local, insira esses comandos para criar os arquivos de certificado de dispositivo para o dispositivo.

```
mkdir ~/certs/testconn
aws iot create-keys-and-certificate \
--set-as-active \
--certificate-pem-outfile "~/certs/testconn/device.pem.crt" \
--public-key-outfile "~/certs/testconn/public.pem.key" \
--private-key-outfile "~/certs/testconn/private.pem.key"
```

O comando retorna uma resposta como a seguinte. Registre o valor de *certificateArn* para uso posterior.

```
{
    "certificateArn": "arn:aws:iot:us-
west-2:57EXAMPLE833:cert/76e7e4edb3e52f52334be2f387a06145b2aa4c7fcd810f3aea2d92abc227d269",
    "certificateId":
 "76e7e4edb3e52f5233EXAMPLE7a06145b2aa4c7fcd810f3aea2d92abc227d269",
    "certificatePem": "-----BEGIN CERTIFICATE-----
\nMIIDWTCCAkGgAwIBAgI_SHORTENED_FOR_EXAMPLE_Lgn4jfgtS\n-----END CERTIFICATE-----
\n",
    "keyPair": {
        "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BA_SHORTENED_FOR_EXAMPLE_ImwIDAQAB\n-----END PUBLIC KEY-----
\n",
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIBAAKCAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
    }
}
```

2. Insira os comandos a seguir para definir as permissões no diretório de certificados e arquivos.

```
chmod 745 ~
chmod 700 ~/certs/testconn
chmod 644 ~/certs/testconn/*
chmod 600 ~/certs/testconn/private.pem.key
```

3. Execute este comando para examinar as permissões nos diretórios e arquivos do certificado.

```
ls -l ~/certs/testconn
```

A saída do comando deve ser a mesma vista aqui, com exceção das datas e horários do arquivo, que serão diferentes.

```
-rw-r--r-- 1 pi pi 1220 Oct 28 13:02 device.pem.crt
-rw------- 1 pi pi 1675 Oct 28 13:02 private.pem.key
-rw-r--r-- 1 pi pi  451 Oct 28 13:02 public.pem.key
```

Neste ponto, você tem os arquivos de certificado do dispositivo instalados no Raspberry Pi e pode continuar em the section called “Crie AWS IoT recursos”.

Crie AWS IoT recursos

Esse procedimento provisiona seu dispositivo AWS IoT criando os recursos necessários para acessar AWS IoT recursos e serviços.

Para provisionar seu dispositivo em AWS IoT

1. Na janela do terminal do computador host local, insira o comando a seguir para obter o endereço do endpoint de dados do dispositivo da sua Conta da AWS.

   ```
   aws iot describe-endpoint --endpoint-type IoT:Data-ATS
   ```

   O comando das etapas anteriores retorna uma resposta como a seguir. Registre o valor de *endpointAddress* para uso posterior.

   ```
   {
       "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. Digite este comando para criar um recurso para AWS IoT o seu Raspberry Pi.

   ```
   aws iot create-thing --thing-name "DevCliTestThing"
   ```

   Se seu recurso AWS IoT Thing foi criado, o comando retornará uma resposta como essa.

   ```
   {
       "thingName": "DevCliTestThing",
       "thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/DevCliTestThing",
       "thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
   }
   ```

3. Na janela do terminal:

a. Abra um editor de texto, como o `nano`.

b. Copie este documento de política JSON e cole-o no editor de texto aberto.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Subscribe",
                "iot:Receive",
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

> **Note**
>
> Este documento de política concede generosamente a cada recurso permissão para se conectar, receber, publicar e assinar. Normalmente, as políticas concedem permissão somente a recursos específicos para realizar ações específicas. No entanto, para o teste inicial de conectividade do dispositivo, essa política excessivamente geral e permissiva é usada para minimizar a chance de um problema de acesso durante esse teste. Nos tutoriais subsequentes, documentos de políticas com escopo mais restrito serão usados para demonstrar práticas recomendadas na elaboração de políticas.

c. Salve o arquivo no editor de texto como **`~/policies/dev_cli_test_thing_policy.json`**.

4. Execute esse comando para usar o documento de política das etapas anteriores para criar uma AWS IoT política.

```
aws iot create-policy \
--policy-name "DevCliTestThingPolicy" \
--policy-document "file://~/policies/dev_cli_test_thing_policy.json"
```

Se a política for criada, o comando retornará uma resposta como esta.

```
{
    "policyName": "DevCliTestThingPolicy",
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/DevCliTestThingPolicy",
    "policyDocument": "{\n    \"Version\": \"2012-10-17\",       \n    \"Statement
\": [\n        {\n            \"Effect\": \"Allow\",\n            \"Action\": [\n
                \"iot:Publish\",\n                \"iot:Subscribe\",\n
    \"iot:Receive\",\n                \"iot:Connect\"\n            ],\n
 \"Resource\": [\n                \"*\"\n            ]\n        }\n    ]\n}\n",
    "policyVersionId": "1"
}
```

5. Execute este comando para anexar a política ao certificado do dispositivo. Substitua *certificateArn* pelo valor de `certificateArn` salvo anteriormente.

```
aws iot attach-policy \
--policy-name "DevCliTestThingPolicy" \
--target "certificateArn"
```

Se houver êxito, o comando não retornará nada.

6. Execute esse comando para anexar o certificado do dispositivo ao recurso da AWS IoT coisa. Substitua *certificateArn* pelo valor de `certificateArn` salvo anteriormente.

```
aws iot attach-thing-principal \
--thing-name "DevCliTestThing" \
--principal "certificateArn"
```

Se houver êxito, o comando não retornará nada.

Depois de provisionar seu dispositivo com sucesso AWS IoT, você estará pronto para continuar. the section called “Configurar o Device Client e testar a conectividade”

Configurar o cliente do AWS IoT dispositivo para testar a conectividade

Os procedimentos nesta seção configuram o AWS IoT Device Client para publicar uma mensagem MQTT do seu Raspberry Pi.

Procedimentos desta seção:

- Crie o arquivo de configuração
- Abra o cliente de teste MQTT
- Execute AWS IoT o cliente do dispositivo

Crie o arquivo de configuração

Esse procedimento cria o arquivo de configuração para testar o AWS IoT Device Client.

Para criar o arquivo de configuração para testar o AWS IoT Device Client

- Na janela do terminal do computador host local conectado ao Raspberry Pi:

  a. Insira estes comandos para criar um diretório para os arquivos de configuração e definir a permissão no diretório:

  ```
  mkdir ~/dc-configs
  chmod 745 ~/dc-configs
  ```

  b. Abra um editor de texto, como o `nano`.

  c. Copie este documento JSON e cole-o no editor de texto aberto.

  ```
  {
    "endpoint": "a3qEXAMPLEaffp-ats.iot.us-west-2.amazonaws.com",
    "cert": "~/certs/testconn/device.pem.crt",
    "key": "~/certs/testconn/private.pem.key",
    "root-ca": "~/certs/AmazonRootCA1.pem",
    "thing-name": "DevCliTestThing",
    "logging": {
      "enable-sdk-logging": true,
      "level": "DEBUG",
      "type": "STDOUT",
      "file": ""
    },
    "jobs": {
  ```

```
    "enabled": false,
    "handler-directory": ""
  },
  "tunneling": {
    "enabled": false
  },
  "device-defender": {
    "enabled": false,
    "interval": 300
  },
  "fleet-provisioning": {
    "enabled": false,
    "template-name": "",
    "template-parameters": "",
    "csr-file": "",
    "device-key": ""
  },
  "samples": {
    "pub-sub": {
      "enabled": true,
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "",
      "subscribe-topic": "test/dc/subtopic",
      "subscribe-file": ""
    }
  },
  "config-shadow": {
    "enabled": false
  },
  "sample-shadow": {
    "enabled": false,
    "shadow-name": "",
    "shadow-input-file": "",
    "shadow-output-file": ""
  }
}
```

d. Substitua o *endpoint* valor pelo endpoint de dados do dispositivo Conta da AWS que você encontrou emthe section called “Provisione seu dispositivo em AWS IoT Core”.

e. Salve o arquivo no editor de texto como **~/dc-configs/dc-testconn-config.json**.

f. Execute este comando para definir permissões no novo arquivo de configuração.

```
chmod 644 ~/dc-configs/dc-testconn-config.json
```

Após salvar o arquivo, você estará pronto para continuar em the section called “Abra o cliente de teste MQTT”.

Abra o cliente de teste MQTT

Esse procedimento prepara o cliente de teste MQTT no AWS IoT console para assinar a mensagem MQTT que o AWS IoT Device Client publica quando é executado.

Para preparar o cliente de teste MQTT para assinar todas as mensagens MQTT

1. No computador host local, no console da AWS IoT, selecione o cliente de teste MQTT.
2. Na guia Assinar um tópico, em Filtro de tópicos, insira **#** (um único sinal de cerquilha) e selecione Assinar para assinar cada tópico do MQTT.
3. Abaixo do rótulo Assinaturas, confirme que pode ver **#** (um único sinal de jogo da velha).

Deixe a janela com o cliente de teste MQTT aberta enquanto continua na the section called “Execute AWS IoT o cliente do dispositivo”.

Execute AWS IoT o cliente do dispositivo

Esse procedimento executa o AWS IoT Device Client para que ele publique uma única mensagem MQTT que o cliente de teste MQTT receba e exiba.

Para enviar uma mensagem MQTT do AWS IoT Device Client

1. Certifique-se de que a janela do terminal conectada ao Raspberry Pi e a janela com o cliente de teste MQTT estejam visíveis enquanto você realiza esse procedimento.
2. Na janela do terminal, insira esses comandos para executar o AWS IoT Device Client usando o arquivo de configuração criado emthe section called “Crie o arquivo de configuração”.

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-testconn-config.json
   ```

   Na janela do terminal, o AWS IoT Device Client exibe mensagens informativas e quaisquer erros que ocorram durante a execução.

Caso nenhum erro seja exibido na janela do terminal, examine o cliente de teste do MQTT.

3. No cliente de teste do MQTT, na janela Assinaturas, consulte a mensagem Olá, mundo! enviada ao tópico de mensagem `test/dc/pubtopic`.
4. Se o AWS IoT Device Client não exibir erros e você ver Hello World! enviada para a `test/dc/pubtopic` mensagem no cliente de teste do MQTT, você demonstrou uma conexão bem-sucedida.
5. Na janela do terminal, digite **^C** (Ctrl-C) para parar o AWS IoT Device Client.

Depois de demonstrar que o AWS IoT Device Client está funcionando corretamente em seu Raspberry Pi e pode se comunicar com AWS IoT, você pode continuar com o. the section called "Comunicar-se com o cliente do dispositivo usando MQTT"

Tutorial: Demonstre a comunicação de mensagens MQTT com o AWS IoT Device Client

Este tutorial demonstra como o AWS IoT Device Client pode assinar e publicar mensagens MQTT, que são comumente usadas em soluções de IoT.

Para iniciar este tutorial:

- Configure seu computador host local e Raspberry Pi como na seção anterior.

  Se você salvou a imagem do cartão microSD depois de instalar o AWS IoT Device Client, você pode usar um cartão microSD com essa imagem com seu Raspberry Pi.
- Se você já executou essa demonstração antes, revise ??? para excluir todos os AWS IoT recursos que você criou em execuções anteriores para evitar erros de recursos duplicados.

Este tutorial leva cerca de 45 minutos para ser concluído.

Quando você concluir este tópico:

- Você terá demonstrado diferentes maneiras pelas quais seu dispositivo de IoT pode assinar mensagens MQTT AWS IoT e publicar mensagens MQTT em. AWS IoT

Equipamentos necessários:

- O ambiente local de desenvolvimento e teste da seção anterior

- O Raspberry Pi usado na seção anterior
- O cartão de memória microSD do Raspberry Pi usado na seção anterior

Procedimentos deste tutorial

- Preparar o Raspberry Pi para demonstrar a comunicação de mensagens MQTT
- Demonstre a publicação de mensagens com o AWS IoT Device Client
- Demonstre a assinatura de mensagens com o AWS IoT Device Client

Preparar o Raspberry Pi para demonstrar a comunicação de mensagens MQTT

Esse procedimento cria os recursos no AWS IoT e no Raspberry Pi para demonstrar a comunicação de mensagens MQTT usando o AWS IoT Device Client.

Procedimentos desta seção:

- Criar os arquivos de certificado para demonstrar a comunicação MQTT
- Provisione seu dispositivo para demonstrar a comunicação MQTT
- Configure o arquivo de configuração do AWS IoT Device Client e o cliente de teste MQTT para demonstrar a comunicação MQTT

Criar os arquivos de certificado para demonstrar a comunicação MQTT

Este procedimento cria os arquivos de certificado de dispositivo para esta demonstração.

Para criar e baixar os arquivos de certificado de dispositivo para o Raspberry Pi

1. Na janela do terminal do computador host local, insira o comando a seguir para criar os arquivos de certificado de dispositivo para o dispositivo.

```
mkdir ~/certs/pubsub
aws iot create-keys-and-certificate \
--set-as-active \
--certificate-pem-outfile "~/certs/pubsub/device.pem.crt" \
--public-key-outfile "~/certs/pubsub/public.pem.key" \
--private-key-outfile "~/certs/pubsub/private.pem.key"
```

O comando retorna uma resposta como a seguinte. Salve o valor de *certificateArn* para uso posterior.

```
{
"certificateArn": "arn:aws:iot:us-
west-2:57EXAMPLE833:cert/76e7e4edb3e52f52334be2f387a06145b2aa4c7fcd810f3aea2d92abc227d269",
"certificateId":
 "76e7e4edb3e52f5233EXAMPLE7a06145b2aa4c7fcd810f3aea2d92abc227d269",
"certificatePem": "-----BEGIN CERTIFICATE-----
\nMIIDWTCCAkGgAwIBAgI_SHORTENED_FOR_EXAMPLE_Lgn4jfgtS\n-----END CERTIFICATE-----
\n",
"keyPair": {
    "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BA_SHORTENED_FOR_EXAMPLE_ImwIDAQAB\n-----END PUBLIC KEY-----
\n",
    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIBAAKCAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
}
}
```

2. Insira os comandos a seguir para definir as permissões no diretório de certificados e arquivos.

```
chmod 700 ~/certs/pubsub
chmod 644 ~/certs/pubsub/*
chmod 600 ~/certs/pubsub/private.pem.key
```

3. Execute este comando para examinar as permissões nos diretórios e arquivos do certificado.

```
ls -l ~/certs/pubsub
```

A saída do comando deve ser a mesma vista aqui, com exceção das datas e horários do arquivo, que serão diferentes.

```
-rw-r--r-- 1 pi pi 1220 Oct 28 13:02 device.pem.crt
-rw------- 1 pi pi 1675 Oct 28 13:02 private.pem.key
-rw-r--r-- 1 pi pi  451 Oct 28 13:02 public.pem.key
```

4. Digite esses comandos para criar os diretórios para os arquivos de log.

```
mkdir ~/.aws-iot-device-client
mkdir ~/.aws-iot-device-client/log
```

```
chmod 745 ~/.aws-iot-device-client/log
echo " " > ~/.aws-iot-device-client/log/aws-iot-device-client.log
echo " " > ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
chmod 600 ~/.aws-iot-device-client/log/*
```

Provisione seu dispositivo para demonstrar a comunicação MQTT

Esta seção cria os AWS IoT recursos que provisionam seu Raspberry Pi em AWS IoT.

Para provisionar seu dispositivo no AWS IoT:

1. Na janela do terminal do computador host local, insira o comando a seguir para obter o endereço do endpoint de dados do dispositivo da sua Conta da AWS.

   ```
   aws iot describe-endpoint --endpoint-type IoT:Data-ATS
   ```

   O valor do endpoint não foi alterado desde quando este comando foi executado no tutorial anterior. Uma nova execução deste comando facilita localizar e colar o valor do endpoint de dados no arquivo de configuração usado neste tutorial.

   O comando das etapas anteriores retorna uma resposta como a seguir. Registre o valor de *endpointAddress* para uso posterior.

   ```
   {
   "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. Digite este comando para criar um novo recurso para AWS IoT o seu Raspberry Pi.

   ```
   aws iot create-thing --thing-name "PubSubTestThing"
   ```

   Como um recurso de AWS IoT coisas é uma representação virtual do seu dispositivo na nuvem, podemos criar vários recursos de coisas AWS IoT para serem usados para finalidades diferentes. Todos os recursos de objetos podem ser usados pelo mesmo dispositivo físico de IoT para representar aspectos diferentes do dispositivo.

   Estes tutoriais usarão apenas um recurso de objeto por vez para representar o Raspberry Pi. Dessa forma, nesses tutoriais, eles representam as diferentes demonstrações para que,

depois de criar os AWS IoT recursos para uma demonstração, você possa voltar e repetir a demonstração usando os recursos criados especificamente para cada uma.

Se seu recurso AWS IoT Thing foi criado, o comando retornará uma resposta como essa.

```
{
"thingName": "PubSubTestThing",
"thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/PubSubTestThing",
"thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
}
```

3. Na janela do terminal:

   a. Abra um editor de texto, como o `nano`.

   b. Copie este documento JSON e cole-o no editor de texto aberto.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:client/PubSubTestThing"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/pubtopic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
```

```
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topicfilter/test/dc/
subtopic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/subtopic"
            ]
        }
    ]
}
```

c. No editor, em cada `Resource` seção do documento de política, *us-west-2:57EXAMPLE833* substitua por seu Região da AWS, um caractere de dois pontos (:)) e seu número de 12 dígitos Conta da AWS .

d. Salve o arquivo no editor de texto como **~/policies/pubsub_test_thing_policy.json**.

4. Execute esse comando para usar o documento de política das etapas anteriores para criar uma AWS IoT política.

```
aws iot create-policy \
--policy-name "PubSubTestThingPolicy" \
--policy-document "file://~/policies/pubsub_test_thing_policy.json"
```

Se a política for criada, o comando retornará uma resposta como esta.

```
{
                                    "policyName": "PubSubTestThingPolicy",
                                    "policyArn": "arn:aws:iot:us-
west-2:57EXAMPLE833:policy/PubSubTestThingPolicy",
                                    "policyDocument": "{\n\"Version\":
 \"2012-10-17\",       \n\"Statement\": [\n{\n\"Effect\": \"Allow\",\n
\"Action\": [\n\"iot:Connect\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:client/PubSubTestThing\"\n]\n},\n{\n\"Effect\": \"Allow
\",\n\"Action\": [\n\"iot:Publish\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:topic/test/dc/pubtopic\"\n]\n},\n{\n\"Effect\": \"Allow
```

```
\",\n\"Action\": [\n\"iot:Subscribe\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:topicfilter/test/dc/subtopic\"\n]\n},\n{\n\"Effect\": \"Allow
\",\n\"Action\": [\n\"iot:Receive\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:topic/test/dc/*\"\n]\n}\n]\n}\n",
                                    "policyVersionId": "1"
                                    }
```

5. Execute este comando para anexar a política ao certificado do dispositivo. Substitua *certificateArn* pelo valor de certificateArn salvo anteriormente nesta seção.

```
aws iot attach-policy \
--policy-name "PubSubTestThingPolicy" \
--target "certificateArn"
```

   Se houver êxito, o comando não retornará nada.

6. Execute o comando a seguir para anexar o certificado de dispositivo ao recurso de objeto do AWS IoT . Substitua *certificateArn* pelo valor de certificateArn salvo anteriormente nesta seção.

```
aws iot attach-thing-principal \
--thing-name "PubSubTestThing" \
--principal "certificateArn"
```

   Se houver êxito, o comando não retornará nada.

Depois de provisionar seu dispositivo com sucesso AWS IoT, você estará pronto para continuarthe section called “Configurar o arquivo de configuração do Device Client e o cliente MQTT”.

Configure o arquivo de configuração do AWS IoT Device Client e o cliente de teste MQTT para demonstrar a comunicação MQTT

Esse procedimento cria um arquivo de configuração para testar o AWS IoT Device Client.

Para criar o arquivo de configuração para testar o AWS IoT Device Client

1. Na janela do terminal do computador host local conectado ao Raspberry Pi:

   a. Abra um editor de texto, como o nano.

   b. Copie este documento JSON e cole-o no editor de texto aberto.

```
{
  "endpoint": "a3qEXAMPLEaffp-ats.iot.us-west-2.amazonaws.com",
  "cert": "~/certs/pubsub/device.pem.crt",
  "key": "~/certs/pubsub/private.pem.key",
  "root-ca": "~/certs/AmazonRootCA1.pem",
  "thing-name": "PubSubTestThing",
  "logging": {
    "enable-sdk-logging": true,
    "level": "DEBUG",
    "type": "STDOUT",
    "file": ""
  },
  "jobs": {
    "enabled": false,
    "handler-directory": ""
  },
  "tunneling": {
    "enabled": false
  },
  "device-defender": {
    "enabled": false,
    "interval": 300
  },
  "fleet-provisioning": {
    "enabled": false,
    "template-name": "",
    "template-parameters": "",
    "csr-file": "",
    "device-key": ""
  },
  "samples": {
    "pub-sub": {
      "enabled": true,
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "",
      "subscribe-topic": "test/dc/subtopic",
      "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
    }
  },
  "config-shadow": {
    "enabled": false
  },
  "sample-shadow": {
```

```
    "enabled": false,
    "shadow-name": "",
    "shadow-input-file": "",
    "shadow-output-file": ""
  }
}
```

c. Substitua o *endpoint* valor pelo endpoint de dados do dispositivo Conta da AWS que você encontrou emthe section called “Provisione seu dispositivo em AWS IoT Core”.

d. Salve o arquivo no editor de texto como **~/dc-configs/dc-pubsub-config.json**.

e. Execute este comando para definir permissões no novo arquivo de configuração.

```
chmod 644 ~/dc-configs/dc-pubsub-config.json
```

2. Para preparar o cliente de teste MQTT para assinar todas as mensagens MQTT:

a. No computador host local, no console da AWS IoT, selecione o cliente de teste MQTT.

b. Na guia Assinar um tópico, em Filtro de tópicos, insira **#** (um único sinal de jogo da velha) e selecione Assinar.

c. Abaixo do rótulo Assinaturas, confirme que pode ver **#** (um único sinal de jogo da velha).

Deixe a janela com o cliente de teste MQTT aberta enquanto prossegue com este tutorial.

Depois de salvar o arquivo e configurar o cliente de teste MQTT, você estará pronto para avançar para the section called “Publicar mensagens com o IoT Device Client”.

Demonstre a publicação de mensagens com o AWS IoT Device Client

Os procedimentos nesta seção demonstram como o AWS IoT Device Client pode enviar mensagens MQTT padrão e personalizadas.

As seguintes declarações de política na política criada na etapa anterior para esses exercícios concedem ao Raspberry Pi permissão para realizar estas ações:

- **`iot:Connect`**

  Fornece o nome do clientePubSubTestThing, seu Raspberry Pi executando o AWS IoT Device Client, para se conectar.

```
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing"
      ]
    }
```

- **`iot:Publish`**

  Permite que o Raspberry Pi publique mensagens com um tópico MQTT de `test/dc/pubtopic`.

```
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic"
      ]
    }
```

  A ação `iot:Publish` concede permissão para publicar em tópicos MQTT listados na matriz de recursos. A declaração de política não controla o conteúdo dessas mensagens.

Publique a mensagem padrão usando o AWS IoT Device Client

Esse procedimento executa o AWS IoT Device Client para que ele publique uma única mensagem MQTT padrão que o cliente de teste MQTT recebe e exibe.

Para enviar a mensagem MQTT padrão do AWS IoT Device Client

1. Certifique-se de que a janela do terminal do computador host local conectado ao Raspberry Pi e a janela com o cliente de teste MQTT estejam visíveis enquanto você realiza esse procedimento.
2. Na janela do terminal, insira esses comandos para executar o AWS IoT Device Client usando o arquivo de configuração criado emthe section called “Crie o arquivo de configuração”.

```
cd ~/aws-iot-device-client/build
```

```
./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-config.json
```

Na janela do terminal, o AWS IoT Device Client exibe mensagens informativas e quaisquer erros que ocorram durante a execução.

Caso nenhum erro seja exibido na janela do terminal, examine o cliente de teste do MQTT.

3. No cliente de teste do MQTT, na janela Assinaturas, consulte a mensagem Olá, mundo! enviada ao tópico de mensagem `test/dc/pubtopic`.
4. Se o AWS IoT Device Client não exibir erros e você ver Hello World! enviada para a `test/dc/pubtopic` mensagem no cliente de teste do MQTT, você demonstrou uma conexão bem-sucedida.
5. Na janela do terminal, digite **^C** (Ctrl-C) para parar o AWS IoT Device Client.

Depois de demonstrar que o AWS IoT Device Client publicou a mensagem padrão do MQTT, você pode continuar com o. the section called “Publicar mensagem MQTT personalizada”

Publique uma mensagem personalizada usando o AWS IoT Device Client

Os procedimentos desta seção criam uma mensagem MQTT personalizada e, depois, executam o AWS IoT Device Client para que ele publique a mensagem MQTT personalizada uma vez para que o cliente de teste MQTT a receba e exiba.

Crie uma mensagem MQTT personalizada para o AWS IoT Device Client

Realize as seguintes etapas na janela do terminal do computador host local que está conectado ao Raspberry Pi.

Para criar uma mensagem personalizada para o AWS IoT Device Client publicar

1. Na janela do terminal, abra um editor de texto, como o `nano`.
2. No editor de texto, copie e cole o seguinte documento JSON. Essa será a carga útil da mensagem MQTT que o AWS IoT Device Client publica.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
```

```
    "velocity": 22,
    "bearing": 255
  }
}
```

3. Salve o conteúdo do editor de texto como **~/messages/sample-ws-message.json**.
4. Digite o comando a seguir para definir as permissões do arquivo de mensagem que você acabou de criar.

```
chmod 600 ~/messages/*
```

Para criar um arquivo de configuração para o AWS IoT Device Client usar para enviar a mensagem personalizada

1. Na janela do terminal, em um editor de texto comonano, abra o arquivo de configuração existente do AWS IoT Device Client:**~/dc-configs/dc-pubsub-config.json**.
2. Edite o objeto `samples` para que ele fique assim. Não são necessárias alterações em outras partes deste arquivo.

```
  "samples": {
    "pub-sub": {
      "enabled": true,
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "~/messages/sample-ws-message.json",
      "subscribe-topic": "test/dc/subtopic",
      "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
```

3. Salve o conteúdo do editor de texto como **~/dc-configs/dc-pubsub-custom-config.json**.
4. Execute este comando para definir permissões no novo arquivo de configuração.

```
chmod 644 ~/dc-configs/dc-pubsub-custom-config.json
```

Publique a mensagem MQTT personalizada usando o AWS IoT Device Client

Essa alteração afeta somente o conteúdo da carga da mensagem MQTT, então a política atual permanecerá funcionando. No entanto, se o tópico MQTT (conforme definido pelo valor de `publish-topic` em`~/dc-configs/dc-pubsub-custom-config.json`) fosse alterado,

a declaração de política `iot::Publish` também precisaria ser modificada para permitir que o Raspberry Pi publique no novo tópico do MQTT.

Para enviar a mensagem MQTT do AWS IoT Device Client

1. Certifique-se de que a janela do terminal e a janela com o cliente de teste MQTT estejam visíveis enquanto você realiza esse procedimento. Além disso, certifique-se de que o cliente de teste MQTT ainda seja assinante no filtro de tópicos #. Se não o for, assine novamente o filtro de tópicos #.
2. Na janela do terminal, digite os seguintes comandos para executar o AWS IoT Device Client usando o arquivo de configuração criado em the section called “Crie o arquivo de configuração”.

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-custom-config.json
   ```

   Na janela do terminal, o AWS IoT Device Client exibe mensagens informativas e quaisquer erros que ocorram durante a execução.

   Caso nenhum erro seja exibido na janela do terminal, examine o cliente de teste do MQTT.
3. No cliente de teste MQTT, na janela Assinaturas, examine a carga de mensagem personalizada enviada ao tópico de mensagem `test/dc/pubtopic`.
4. Se o AWS IoT Device Client não exibir erros e você ver a carga da mensagem personalizada que você publicou na `test/dc/pubtopic` mensagem no cliente de teste do MQTT, você publicou uma mensagem personalizada com sucesso.
5. Na janela do terminal, digite **^C** (Ctrl-C) para parar o AWS IoT Device Client.

Depois de demonstrar que o AWS IoT Device Client publicou uma carga de mensagem personalizada, você pode continuarthe section called “Assinar mensagens com o IoT Device Client”.

Demonstre a assinatura de mensagens com o AWS IoT Device Client

Nesta seção, você demonstrará dois tipos de assinaturas de mensagens:

- Assinatura de tópico único
- Assinatura de tópicos curinga

As seguintes declarações de política na política criada para esses exercícios concedem ao Raspberry Pi permissão para realizar estas ações:

- **`iot:Receive`**

  Permite que o AWS IoT Device Client receba tópicos do MQTT que correspondam aos nomeados no `Resource` objeto.

  ```
  {
    "Effect": "Allow",
    "Action": [
      "iot:Receive"
    ],
    "Resource": [
      "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/subtopic"
    ]
  }
  ```

- **`iot:Subscribe`**

  Permite que o AWS IoT Device Client assine filtros de tópicos do MQTT que correspondam aos nomeados no `Resource` objeto.

  ```
  {
    "Effect": "Allow",
    "Action": [
      "iot:Subscribe"
    ],
    "Resource": [
      "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic"
    ]
  }
  ```

Assinatura de um único tópico de mensagem MQTT

Este procedimento demonstra como o AWS IoT Device Client pode assinar e registrar mensagens MQTT.

Na janela do terminal do computador host local que está conectado ao Raspberry Pi, liste o conteúdo de **`~/dc-configs/dc-pubsub-custom-config.json`** ou abra o arquivo em um editor de texto para examinar seu conteúdo. Localize o objeto `samples`, que deve ser semelhante ao seguinte.

```
  "samples": {
    "pub-sub": {
      "enabled": true,
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "~/messages/sample-ws-message.json",
      "subscribe-topic": "test/dc/subtopic",
      "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
```

Observe que o valor de `subscribe-topic` é o tópico MQTT que o AWS IoT Device Client assinará quando for executado. O AWS IoT Device Client grava as cargas de mensagens que recebe dessa assinatura no arquivo nomeado no `subscribe-file` valor.

Para assinar um tópico de mensagem MQTT a partir do AWS IoT Device Client

1. Certifique-se de que a janela do terminal e a janela com o cliente de teste MQTT estejam visíveis enquanto você realiza esse procedimento. Além disso, certifique-se de que o cliente de teste MQTT ainda seja assinante no filtro de tópicos #. Se não o for, assine novamente o filtro de tópicos #.
2. Na janela do terminal, insira esses comandos para executar o AWS IoT Device Client usando o arquivo de configuração criado emthe section called “Crie o arquivo de configuração”.

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-custom-config.json
   ```

   Na janela do terminal, o AWS IoT Device Client exibe mensagens informativas e quaisquer erros que ocorram durante a execução.

   Caso nenhum erro seja exibido na janela do terminal, continue no console do AWS IoT .
3. No AWS IoT console, no cliente de teste do MQTT, escolha a guia Publicar em um tópico.
4. Em Nome do tópico, digite **`test/dc/subtopic`**
5. Em Carga da mensagem, examine o conteúdo da mensagem.
6. Selecione Publicar para publicar a mensagem MQTT.
7. Na janela do terminal, observe a entrada da mensagem recebida do AWS IoT Device Client que se parece com esta.

   ```
   2021-11-10T16:02:20.890Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
    subscribe topic, size: 45 bytes
   ```

8. Depois de ver a entrada da mensagem recebida que mostra que a mensagem foi recebida, digite **^C** (Ctrl-C) para interromper o AWS IoT Device Client.
9. Insira o seguinte comando para visualizar o final do arquivo de log de mensagens e ver a mensagem que foi publicada no cliente de teste MQTT.

```
tail ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
```

Ao visualizar a mensagem no arquivo de log, demonstrou-se que o AWS IoT Device Client recebeu a mensagem publicada a partir do cliente de teste MQTT.

Assinar diversos tópicos de mensagens MQTT usando caracteres curinga

Esses procedimentos demonstram como o AWS IoT Device Client pode assinar e registrar mensagens MQTT usando caracteres curinga. Para fazer isso, você deve:

1. Atualize o filtro de tópicos que o AWS IoT Device Client usa para assinar tópicos do MQTT.
2. Atualizar a política usada pelo dispositivo para permitir novas assinaturas.
3. Execute o AWS IoT Device Client e publique mensagens do console de teste do MQTT.

Para criar um arquivo de configuração para assinar diversos tópicos de mensagens MQTT com um filtro de tópicos MQTT curinga

1. Na janela do terminal do computador host local que está conectado ao Raspberry Pi, abra **~/dc-configs/dc-pubsub-custom-config.json** para edição e localize o objeto `samples`.
2. No editor de texto, localize o objeto `samples` e atualize o valor de `subscribe-topic` para que se assemelhe a isto.

```
"samples": {
  "pub-sub": {
    "enabled": true,
    "publish-topic": "test/dc/pubtopic",
    "publish-file": "~/messages/sample-ws-message.json",
    "subscribe-topic": "test/dc/#",
    "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
```

O novo valor de `subscribe-topic` é um filtro de tópicos MQTT com um caractere curinga MQTT no final. Isso descreve uma assinatura de todos os tópicos MQTT que começam com

`test/dc/`. O AWS IoT Device Client grava as cargas de mensagens que recebe dessa assinatura no arquivo nomeado em`subscribe-file`.

3. Salve o arquivo de configuração modificado como **`~/dc-configs/dc-pubsub-wild-config.json`** e saia do editor de texto.

Para modificar a política usada pelo Raspberry Pi para permitir a assinatura e recebimento de diversos tópicos de mensagens MQTT

1. Na janela do terminal do seu computador host local conectado ao Raspberry Pi, em seu editor de texto favorito, abra **`~/policies/pubsub_test_thing_policy.json`** para edição e, depois, localize as declarações de política `iot::Subscribe` e `iot::Receive` no arquivo.
2. Na declaração de política `iot::Subscribe`, atualize a string no objeto Resource para substituir `subtopic` por `*`, de modo que fique semelhante ao seguinte.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Subscribe"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/*"
  ]
}
```

> **Note**
>
> Os caracteres curinga do filtro de tópicos MQTT são + (sinal de adição) e # (sinal de jogo da velha). Uma solicitação de assinatura com # no final assina todos os tópicos que começam com a string que precede o caractere # (como, nesse caso, `test/dc/`). Entretanto, o valor do recurso na declaração de política autorizando essa assinatura deve usar * (asterisco) no lugar de # (sinal de jogo da velha) no ARN do filtro de tópicos. Isso ocorre porque o processador de políticas usa um caractere curinga diferente daquele usado pelo MQTT.
>
> Para acessar mais informações sobre como usar caracteres curinga para tópicos e filtros de tópicos em políticas, consulte Usando caracteres curinga no MQTT e nas políticas AWS IoT Core.

3. Na declaração de política `iot::Receive`, atualize a string no objeto Resource para substituir `subtopic` por `*`, de modo que fique semelhante ao seguinte.

```
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/*"
      ]
    }
```

4. Salve o documento de política atualizado como **`~/policies/pubsub_wild_test_thing_policy.json`** e saia do editor.

5. Insira o seguinte comando para atualizar a política deste tutorial para usar as novas definições de recursos.

```
aws iot create-policy-version \
--set-as-default \
--policy-name "PubSubTestThingPolicy" \
--policy-document "file://~/policies/pubsub_wild_test_thing_policy.json"
```

Em caso de êxito, ele retornará uma resposta como a seguinte. Observe que `policyVersionId` agora é `2`, o que indica que esta é a segunda versão desta política.

Se você obteve êxito em atualizar a política, avance para o próximo procedimento.

```
{
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/PubSubTestThingPolicy",
    "policyDocument": "{\n  \"Version\": \"2012-10-17\",       \n  \"Statement
\": [\n    {\n      \"Effect\": \"Allow\",\n      \"Action\": [\n
 \"iot:Connect\"\n      ],\n      \"Resource\": [\n        \"arn:aws:iot:us-
west-2:57EXAMPLE833:client/PubSubTestThing\"\n      ]\n    },\n    {\n
 \"Effect\": \"Allow\",\n      \"Action\": [\n        \"iot:Publish\"\n      ],
\n      \"Resource\": [\n        \"arn:aws:iot:us-west-2:57EXAMPLE833:topic/
test/dc/pubtopic\"\n      ]\n    },\n    {\n      \"Effect\": \"Allow\",\n
 \"Action\": [\n        \"iot:Subscribe\"\n      ],\n      \"Resource\": [\n
  \"arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/*\"\n      ]\n    },\n
   {\n      \"Effect\": \"Allow\",\n      \"Action\": [\n        \"iot:Receive\"\n
```

```
     ],\n      \"Resource\": [\n        \"arn:aws:iot:us-west-2:57EXAMPLE833:topic/
test/dc/*\"\n      ]\n    }\n  ]\n}\n",
    "policyVersionId": "2",
    "isDefaultVersion": true
}
```

Caso receba um erro informando que há versões da política demais para que uma nova seja salva, insira o seguinte comando para listar as versões atuais da política. Examine a lista retornada pelo comando a seguir para encontrar uma versão da política que possa ser excluída.

```
aws iot list-policy-versions --policy-name "PubSubTestThingPolicy"
```

Digite o comando a seguir para excluir uma versão que não é mais necessária. Não é possível excluir a versão padrão da política. A versão padrão da política é aquela com um valor de `isDefaultVersion` de `true`.

```
aws iot delete-policy-version \
--policy-name "PubSubTestThingPolicy" \
--policy-version-id policyId
```

Depois de excluir uma versão da política, tente a etapa novamente.

Com o arquivo de configuração e a política atualizados, você está pronto para demonstrar assinaturas curinga com o AWS IoT Device Client.

Para demonstrar como o AWS IoT Device Client assina e recebe vários tópicos de mensagens do MQTT

1. No cliente de teste MQTT, verifique as assinaturas. Se o cliente de teste MQTT for assinante do filtro de tópicos **#**, avance para a próxima etapa. Caso contrário, no cliente de teste do MQTT, na guia Assinar um tópico, em Filtro de tópicos, insira **#** (caractere de cerquilha) e escolha Assinar para assinar.
2. Na janela do terminal do computador host local conectado ao Raspberry Pi, execute os comandos a seguir para iniciar o AWS IoT Device Client.

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-wild-config.json
```

3. Enquanto observa a saída do AWS IoT Device Client na janela do terminal no computador host local, retorne ao cliente de teste MQTT. Na guia Publicar em um tópico, em Nome do tópico, insira **`test/dc/subtopic`** e selecione Publicar.
4. Na janela do terminal, confirme se a mensagem foi recebida procurando uma mensagem como:

```
2021-11-10T16:34:20.101Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
 subscribe topic, size: 76 bytes
```

5. Enquanto observa a saída do AWS IoT Device Client na janela do terminal do computador host local, retorne ao cliente de teste MQTT. Na guia Publicar em um tópico, em Nome do tópico, insira **`test/dc/subtopic2`** e selecione Publicar.
6. Na janela do terminal, confirme se a mensagem foi recebida procurando uma mensagem como:

```
2021-11-10T16:34:32.078Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
 subscribe topic, size: 77 bytes
```

7. Depois de ver as mensagens que confirmam que ambas foram recebidas, digite **`^C`** (Ctrl-C) para interromper o AWS IoT Device Client.
8. Insira o seguinte comando para visualizar o final do arquivo de log de mensagens e ver a mensagem que foi publicada no cliente de teste MQTT.

```
tail -n 20 ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
```

> **Note**
>
> O arquivo de log contém somente cargas de mensagens. Os tópicos de mensagens não são registrados no arquivo de log de mensagens recebidas.
> Você também pode ver a mensagem publicada pelo AWS IoT Device Client no registro recebido. Isso ocorre porque o filtro de tópicos curinga inclui esse tópico de mensagem e, por vezes, a solicitação de assinatura pode ser processada pelo agente de mensagens antes que a mensagem publicada seja enviada aos assinantes.

As entradas no arquivo de log apontam que as mensagens foram recebidas. É possível repetir esse procedimento usando outros nomes de tópicos. Todas as mensagens com um nome de tópico que comece com `test/dc/` devem ser recebidas e registradas em log. Mensagens com nomes de tópicos começando com outro texto são ignoradas.

Depois de demonstrar como o AWS IoT Device Client pode publicar e assinar mensagens MQTT, continue. Tutorial: Demonstre ações remotas (trabalhos) com o AWS IoT Device Client

Tutorial: Demonstre ações remotas (trabalhos) com o AWS IoT Device Client

Nestes tutoriais, vamos configurar e implantar trabalhos em um Raspberry Pi para demonstrar como você pode enviar operações remotas para dispositivos de IoT.

Para iniciar este tutorial:

- Configure seu computador host local e Raspberry Pi como na seção anterior.
- Se você não concluiu o tutorial na seção anterior, pode tentar este tutorial usando o Raspberry Pi com um cartão microSD que contém a imagem que você salvou depois de instalar AWS IoT o Device Client. (Opcional) Salve a imagem do cartão microSD
- Se você já executou essa demonstração antes, revise ??? para excluir todos os AWS IoT recursos que você criou em execuções anteriores para evitar erros de recursos duplicados.

Este tutorial leva cerca de 45 minutos para ser concluído.

Quando você concluir este tópico:

- Você terá demonstrado diferentes maneiras pelas quais seu dispositivo de IoT pode usar o AWS IoT Core para executar operações remotas gerenciadas por. AWS IoT

Equipamentos necessários:

- O ambiente local de desenvolvimento e teste já testado em uma seção anterior
- O Raspberry Pi testado em uma seção anterior
- O cartão de memória microSD do Raspberry Pi testado em uma seção anterior

Procedimentos deste tutorial

- Preparar o Raspberry Pi para execução de trabalhos
- Crie e execute o trabalho AWS IoT com o AWS IoT Device Client

Preparar o Raspberry Pi para execução de trabalhos

Os procedimentos nesta seção descrevem como preparar seu Raspberry Pi para executar trabalhos usando o AWS IoT Device Client.

> **Note**
>
> Estes procedimentos são específicos ao dispositivo. Caso queira realizar os procedimentos desta seção com mais de um dispositivo simultaneamente, cada dispositivo precisará de sua própria política e de um certificado e um nome de item exclusivos e específicos ao dispositivo. Para que cada dispositivo tenha seus recursos exclusivos, execute o procedimento uma vez para cada dispositivo enquanto modifica os elementos específicos do dispositivo, como descrito nos procedimentos.

Procedimentos deste tutorial

- Provisione seu Raspberry Pi para demonstrar trabalhos
- Configurar o AWS IoT Device Client para executar o agente de trabalhos

Provisione seu Raspberry Pi para demonstrar trabalhos

Os procedimentos nesta seção provisionam seu Raspberry Pi AWS IoT criando AWS IoT recursos e certificados de dispositivo para ele.

Tópicos

- Crie e baixe arquivos de certificado de dispositivo para demonstrar AWS IoT trabalhos
- Crie AWS IoT recursos para demonstrar AWS IoT empregos

Crie e baixe arquivos de certificado de dispositivo para demonstrar AWS IoT trabalhos

Este procedimento cria os arquivos de certificado de dispositivo para esta demonstração.

Se você estiver preparando mais de um dispositivo, esse procedimento deve ser executado em cada dispositivo.

Para criar e baixar os arquivos de certificado de dispositivo para o Raspberry Pi:

Na janela do terminal do computador host local conectado ao Raspberry Pi, digite os comandos a seguir.

1. Digite o comando a seguir para criar os arquivos de certificado de dispositivo.

```
aws iot create-keys-and-certificate \
--set-as-active \
--certificate-pem-outfile "~/certs/jobs/device.pem.crt" \
--public-key-outfile "~/certs/jobs/public.pem.key" \
--private-key-outfile "~/certs/jobs/private.pem.key"
```

   O comando retorna uma resposta como a seguinte. Salve o valor de *certificateArn* para uso posterior.

```
{
"certificateArn": "arn:aws:iot:us-
west-2:57EXAMPLE833:cert/76e7e4edb3e52f52334be2f387a06145b2aa4c7fcd810f3aea2d92abc227d269",
"certificateId":
 "76e7e4edb3e52f5233EXAMPLE7a06145b2aa4c7fcd810f3aea2d92abc227d269",
"certificatePem": "-----BEGIN CERTIFICATE-----
\nMIIDWTCCAkGgAwIBAgI_SHORTENED_FOR_EXAMPLE_Lgn4jfgtS\n-----END CERTIFICATE-----
\n",
"keyPair": {
    "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BA_SHORTENED_FOR_EXAMPLE_ImwIDAQAB\n-----END PUBLIC KEY-----
\n",
    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIBAAKCAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
}
}
```

2. Insira os comandos a seguir para definir as permissões no diretório de certificados e arquivos.

```
chmod 700 ~/certs/jobs
chmod 644 ~/certs/jobs/*
chmod 600 ~/certs/jobs/private.pem.key
```

3. Execute este comando para examinar as permissões nos diretórios e arquivos do certificado.

```
ls -l ~/certs/jobs
```

A saída do comando deve ser a mesma vista aqui, com exceção das datas e horários do arquivo, que serão diferentes.

```
-rw-r--r-- 1 pi pi 1220 Oct 28 13:02 device.pem.crt
-rw------- 1 pi pi 1675 Oct 28 13:02 private.pem.key
-rw-r--r-- 1 pi pi  451 Oct 28 13:02 public.pem.key
```

Após baixar os arquivos de certificado de dispositivo para o Raspberry Pi, você estará pronto para avançar para the section called “Provisionar o Raspberry Pi para trabalhos”.

Crie AWS IoT recursos para demonstrar AWS IoT empregos

Crie os AWS IoT recursos para esse dispositivo.

Se você estiver preparando mais de um dispositivo, esse procedimento deve ser executado para todos os dispositivos.

Para provisionar seu dispositivo no AWS IoT:

Na janela do terminal do computador host local conectado ao Raspberry Pi:

1. Digite o seguinte comando para obter o endereço do endpoint de dados do dispositivo da sua Conta da AWS.

   ```
   aws iot describe-endpoint --endpoint-type IoT:Data-ATS
   ```

   O valor do endpoint não foi alterado desde a última vez que este comando foi executado. Executar o comando mais uma vez facilita localizar e colar o valor do endpoint de dados no arquivo de configuração usado neste tutorial.

   O comando describe-endpoint retorna uma resposta como a seguinte. Registre o valor de *endpointAddress* para uso posterior.

   ```
   {
   "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. *uniqueThingName*Substitua por um nome exclusivo para seu dispositivo. Caso queira realizar este tutorial com vários dispositivos, dê um nome único a cada dispositivo. Como, por exemplo, **TestDevice01**, **TestDevice02**, e assim por diante.

   Digite este comando para criar um novo recurso para AWS IoT o seu Raspberry Pi.

   ```
   aws iot create-thing --thing-name "uniqueThingName"
   ```

   Como um recurso de AWS IoT coisas é uma representação virtual do seu dispositivo na nuvem, podemos criar vários recursos de coisas AWS IoT para serem usados para finalidades diferentes. Todos os recursos de objetos podem ser usados pelo mesmo dispositivo físico de IoT para representar aspectos diferentes do dispositivo.

   > **Note**
   >
   > Para proteger a política para vários dispositivos, você pode usar `${iot:Thing.ThingName}` em vez do nome do objeto estática, *uniqueThingName*.

   Estes tutoriais usarão apenas um recurso de objeto por vez por dispositivo. Dessa forma, nesses tutoriais, eles representam as diferentes demonstrações para que, depois de criar os AWS IoT recursos para uma demonstração, você possa voltar e repetir as demonstrações usando os recursos criados especificamente para cada uma.

   Se seu recurso AWS IoT Thing foi criado, o comando retornará uma resposta como essa. Registre o valor de *thingArn* para uso posterior quando for criar o trabalho a ser executado nesse dispositivo.

   ```
   {
   "thingName": "uniqueThingName",
   "thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/uniqueThingName",
   "thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
   }
   ```

3. Na janela do terminal:

   a. Abra um editor de texto, como o `nano`.

   b. Copie este documento JSON e cole-o no editor de texto aberto.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:client/uniqueThingName"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/pubtopic",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/events/job/
*",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/events/
jobExecution/*",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topicfilter/test/dc/
subtopic",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/events/
jobExecution/*",
                "arn:aws:iot:us-west-2:123456789012:topicfilter/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/subtopic",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:DescribeJobExecution",
                "iot:GetPendingJobExecutions",
                "iot:StartNextPendingJobExecution",
                "iot:UpdateJobExecution"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName"
            ]
        }
    ]
}
```

c. No editor, na `Resource` seção de cada declaração de política, *us-west-2:57EXAMPLE833* substitua por seu Região da AWS, um caractere de dois pontos (:)) e seu número de 12 dígitos Conta da AWS .

d. No editor, em cada declaração de política, *uniqueThingName* substitua pelo nome da coisa que você deu ao recurso.

e. Salve o arquivo no editor de texto como **~/policies/jobs_test_thing_policy.json**.

   Caso esteja executando esse procedimento para diversos dispositivos, salve o arquivo com esse nome de arquivo em cada dispositivo.

4. *uniqueThingName*Substitua pelo nome do dispositivo e, em seguida, execute esse comando para criar uma AWS IoT política personalizada para esse dispositivo.

```
aws iot create-policy \
--policy-name "JobTestPolicyFor uniqueThingName" \
--policy-document "file://~/policies/jobs_test_thing_policy.json"
```

Se a política for criada, o comando retornará uma resposta como esta.

```
{
    "policyName": "JobTestPolicyFor uniqueThingName",
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/
JobTestPolicyFor uniqueThingName",
    "policyDocument": "{\n\"Version\": \"2012-10-17\",\n\"Statement\": [\n{\n
\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Connect\"\n],\n\"Resource\":
 [\n\"arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing\"\n]\n},\n{\n
\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Publish\"\n],\n\"Resource\":
 [\n\"arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic\"\n]\n},\n{\n
\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Subscribe\"\n],\n\"Resource\": [\n
\"arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic\"\n]\n},\n{\n
\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Receive\"\n],\n\"Resource\": [\n
\"arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/*\"\n]\n}\n]\n}\n",
    "policyVersionId": "1"
}
```

5. *uniqueThingName*Substitua pelo nome do dispositivo e *certificateArn* pelo `certificateArn` valor que você salvou anteriormente nesta seção para esse dispositivo e, em seguida, execute esse comando para anexar a política ao certificado do dispositivo.

```
aws iot attach-policy \
--policy-name "JobTestPolicyFor uniqueThingName" \
--target "certificateArn"
```

Se houver êxito, o comando não retornará nada.

6. *uniqueThingName*Substitua pelo nome da coisa do dispositivo, *certificateArn* substitua pelo `certificateArn` valor que você salvou anteriormente nesta seção e, em seguida, execute esse comando para anexar o certificado do dispositivo ao recurso da AWS IoT coisa.

```
aws iot attach-thing-principal \
--thing-name "uniqueThingName" \
--principal "certificateArn"
```

Se houver êxito, o comando não retornará nada.

Após provisionar o Raspberry Pi com êxito, você estará pronto para repetir esta seção com outro Raspberry Pi em um teste ou, se todos os dispositivos tiverem sido provisionados, avançar para the section called “Configurar o Device Client para executar trabalhos”.

Configurar o AWS IoT Device Client para executar o agente de trabalhos

Este procedimento cria um arquivo de configuração para o AWS IoT Device Client executar o agente de trabalhos:.

Nota: se você estiver preparando mais de um dispositivo, esse procedimento deve ser executado em cada dispositivo.

Para criar o arquivo de configuração para testar o AWS IoT Device Client:

1. Na janela do terminal do computador host local conectado ao Raspberry Pi:

   a. Abra um editor de texto, como o `nano`.

   b. Copie este documento JSON e cole-o no editor de texto aberto.

```
{
  "endpoint": "a3qEXAMPLEaffp-ats.iot.us-west-2.amazonaws.com",
  "cert": "~/certs/jobs/device.pem.crt",
  "key": "~/certs/jobs/private.pem.key",
  "root-ca": "~/certs/AmazonRootCA1.pem",
  "thing-name": "uniqueThingName",
  "logging": {
    "enable-sdk-logging": true,
    "level": "DEBUG",
    "type": "STDOUT",
    "file": ""
  },
  "jobs": {
    "enabled": true,
    "handler-directory": ""
  },
  "tunneling": {
    "enabled": false
  },
  "device-defender": {
    "enabled": false,
    "interval": 300
  },
  "fleet-provisioning": {
```

```
    "enabled": false,
    "template-name": "",
    "template-parameters": "",
    "csr-file": "",
    "device-key": ""
  },
  "samples": {
    "pub-sub": {
      "enabled": false,
      "publish-topic": "",
      "publish-file": "",
      "subscribe-topic": "",
      "subscribe-file": ""
    }
  },
  "config-shadow": {
    "enabled": false
  },
  "sample-shadow": {
    "enabled": false,
    "shadow-name": "",
    "shadow-input-file": "",
    "shadow-output-file": ""
  }
}
```

c. Substitua o *endpoint* valor pelo valor do endpoint de dados do dispositivo Conta da AWS que você encontrou emthe section called “Provisione seu dispositivo em AWS IoT Core”.

d. *uniqueThingName*Substitua pelo nome da coisa que você usou para este dispositivo.

e. Salve o arquivo no editor de texto como **~/dc-configs/dc-jobs-config.json**.

2. Execute o comando a seguir para definir as permissões de arquivo do novo arquivo de configuração.

```
chmod 644 ~/dc-configs/dc-jobs-config.json
```

O cliente de teste MQTT não será utilizado neste teste. Embora o dispositivo troque mensagens MQTT relacionadas aos trabalhos AWS IoT, as mensagens de progresso do trabalho são trocadas somente com o dispositivo que está executando o trabalho. Como as mensagens de progresso do trabalho são trocadas somente com o dispositivo que está executando o trabalho, você não pode assiná-las de outro dispositivo, como o AWS IoT console.

Após salvar o arquivo de configuração, você estará pronto para avançar para the section called “Criar e executar um trabalho de IoT”.

Crie e execute o trabalho AWS IoT com o AWS IoT Device Client

Os procedimentos desta seção criam um documento de trabalho e um recurso de AWS IoT trabalho. Depois de criar o recurso de trabalho, AWS IoT envia o documento de trabalho para os destinos de trabalho especificados nos quais um agente de trabalhos aplica o documento de trabalho ao dispositivo ou cliente.

Procedimentos desta seção

- Criar e armazenar o documento de trabalho para o trabalho de IoT
- Execute um trabalho em AWS IoT um dispositivo de IoT

Criar e armazenar o documento de trabalho para o trabalho de IoT

Esse procedimento cria um documento de trabalho simples para incluir em um recurso de AWS IoT trabalho. O documento de trabalho mostra “Olá, mundo” no destino do trabalho.

Para criar e armazenar um documento de trabalho:

1. Selecione o bucket do Amazon S3 no qual você salvará seu documento de trabalho. Se você ainda não tiver um bucket do Amazon S3 existente para isso, será preciso criar um. Para obter informações sobre a criação de buckets do Amazon S3, consulte os tópicos de Introdução ao Amazon S3.
2. Crie e salve o documento de trabalho deste trabalho

   a. Abra um editor de texto no computador host local.

   b. Copie e cole o texto a seguir no editor.

   ```
   {
       "operation": "echo",
       "args": ["Hello world!"]
   }
   ```

   c. No computador host local, salve o conteúdo do editor em um arquivo nomeado **`hello-world-job.json`**.

d. Confirme se o arquivo foi salvo com êxito. Alguns editores de texto acrescentam automaticamente `.txt` ao final do nome de um arquivo ao salvar um arquivo de texto. Se o editor tiver acrescentado `.txt` ao nome do arquivo, faça a correção antes de continuar.

3. *path_to_file*Substitua o pelo caminho para**hello-world-job.json**, se não estiver em seu diretório atual, *s3_bucket_name* substitua pelo caminho do bucket do Amazon S3 até o bucket selecionado e, em seguida, execute esse comando para colocar seu documento de trabalho no bucket do Amazon S3.

```
aws s3api put-object \
--key hello-world-job.json \
--body path_to_file/hello-world-job.json --bucket s3_bucket_name
```

A URL do documento de trabalho que identifica o documento de trabalho que você armazenou no Amazon S3 é determinada pela substituição de *AWS_region* e na URL *s3_bucket_name* a seguir. Registre o URL resultante para usar posteriormente como *job_document_path*

```
https://s3_bucket_name.s3.AWS_Region.amazonaws.com/hello-world-job.json
```

> **Note**
>
> AWS a segurança impede que você possa abrir esse URL fora do seu Conta da AWS, por exemplo, usando um navegador. O URL é usado pelo mecanismo de AWS IoT trabalhos, que tem acesso ao arquivo, por padrão. Em um ambiente de produção, será preciso garantir que seus serviços do AWS IoT tenham permissão para acessar os documentos de trabalho armazenados no Amazon S3.

Após salvar o URL do documento de trabalho, avance para the section called “Executar trabalho para um único dispositivo”.

Execute um trabalho em AWS IoT um dispositivo de IoT

Os procedimentos nesta seção iniciam o AWS IoT Device Client em seu Raspberry Pi para executar o agente de trabalhos no dispositivo e aguardar a execução dos trabalhos. Ele também cria um recurso de trabalho em AWS IoT, que enviará o trabalho e será executado em seu dispositivo de IoT.

> **Note**
>
> Esse procedimento executa um trabalho em apenas um único dispositivo.

Para iniciar o atendente de trabalhos no Raspberry Pi:

1. Na janela do terminal do seu computador host local que está conectado ao seu Raspberry Pi, execute este comando para iniciar o AWS IoT Device Client.

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-jobs-config.json
```

2. Na janela do terminal, confirme se o AWS IoT Device Client exibe essas mensagens

```
2021-11-15T18:45:56.708Z [INFO]  {Main.cpp}: Jobs is enabled
                        .
                        .
                        .
2021-11-15T18:45:56.708Z [INFO]  {Main.cpp}: Client base has been notified that
 Jobs has started
2021-11-15T18:45:56.708Z [INFO]  {JobsFeature.cpp}: Running Jobs!
2021-11-15T18:45:56.708Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 startNextPendingJobExecution accepted and rejected
2021-11-15T18:45:56.708Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 nextJobChanged events
2021-11-15T18:45:56.708Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 updateJobExecutionStatusAccepted for jobId +
2021-11-15T18:45:56.738Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToUpdateJobExecutionAccepted with code {0}
2021-11-15T18:45:56.739Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 updateJobExecutionStatusRejected for jobId +
2021-11-15T18:45:56.753Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToNextJobChanged with code {0}
2021-11-15T18:45:56.760Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToStartNextJobRejected with code {0}
2021-11-15T18:45:56.776Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToStartNextJobAccepted with code {0}
2021-11-15T18:45:56.776Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToUpdateJobExecutionRejected with code {0}
2021-11-15T18:45:56.777Z [DEBUG] {JobsFeature.cpp}: Publishing
 startNextPendingJobExecutionRequest
```

```
2021-11-15T18:45:56.785Z [DEBUG] {JobsFeature.cpp}: Ack received for
 StartNextPendingJobPub with code {0}
2021-11-15T18:45:56.785Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
```

3. Na janela do terminal, depois de ver a seguinte mensagem, avance para o próximo procedimento e crie o recurso de trabalho. Observe que esta pode não ser a última entrada da lista.

```
2021-11-15T18:45:56.785Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
```

Para criar um recurso AWS IoT de trabalho

1. No computador host local:

   a. *job_document_url* Substitua pelo URL do documento de trabalho dethe section called "Criar e armazenar documento do trabalho".

   b. *thing_arn* Substitua pelo ARN do recurso que você criou para o seu dispositivo e execute esse comando.

   ```
   aws iot create-job \
   --job-id hello-world-job-1 \
   --document-source "job_document_url" \
   --targets "thing_arn" \
   --target-selection SNAPSHOT
   ```

   Em caso de êxito, o comando retornará um resultado semelhante a este.

   ```
   {
     "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-job-1",
     "jobId": "hello-world-job-1"
   }
   ```

2. Na janela do terminal, você deve ver uma saída do AWS IoT Device Client como esta.

```
2021-11-15T18:02:26.688Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Job ids differ
```

```
2021-11-15T18:10:24.890Z [INFO]  {JobsFeature.cpp}: Executing job: hello-world-
job-1
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Attempting to update job
 execution status!
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Not including stdout with the
 status details
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Not including stderr with the
 status details
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Assuming executable is in PATH
2021-11-15T18:10:24.890Z [INFO]  {JobsFeature.cpp}: About to execute: echo Hello
 world!
2021-11-15T18:10:24.890Z [DEBUG] {Retry.cpp}: Retryable function starting, it will
 retry until success
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Created EphermalPromise for
 ClientToken 3TEWba9Xj6 in the updateJobExecution promises map
2021-11-15T18:10:24.890Z [DEBUG] {JobEngine.cpp}: Child process now running
2021-11-15T18:10:24.890Z [DEBUG] {JobEngine.cpp}: Child process about to call
 execvp
2021-11-15T18:10:24.890Z [DEBUG] {JobEngine.cpp}: Parent process now running, child
 PID is 16737
2021-11-15T18:10:24.891Z [DEBUG] {16737}: Hello world!
2021-11-15T18:10:24.891Z [DEBUG] {JobEngine.cpp}: JobEngine finished waiting for
 child process, returning 0
2021-11-15T18:10:24.891Z [INFO]  {JobsFeature.cpp}: Job exited with status: 0
2021-11-15T18:10:24.891Z [INFO]  {JobsFeature.cpp}: Job executed successfully!
2021-11-15T18:10:24.891Z [DEBUG] {JobsFeature.cpp}: Attempting to update job
 execution status!
2021-11-15T18:10:24.891Z [DEBUG] {JobsFeature.cpp}: Not including stdout with the
 status details
2021-11-15T18:10:24.891Z [DEBUG] {JobsFeature.cpp}: Not including stderr with the
 status details
2021-11-15T18:10:24.892Z [DEBUG] {Retry.cpp}: Retryable function starting, it will
 retry until success
2021-11-15T18:10:24.892Z [DEBUG] {JobsFeature.cpp}: Created EphermalPromise for
 ClientToken GmQ0HTzWGg in the updateJobExecution promises map
2021-11-15T18:10:24.905Z [DEBUG] {JobsFeature.cpp}: Ack received for
 PublishUpdateJobExecutionStatus with code {0}
2021-11-15T18:10:24.905Z [DEBUG] {JobsFeature.cpp}: Removing ClientToken 3TEWba9Xj6
 from the updateJobExecution promises map
2021-11-15T18:10:24.905Z [DEBUG] {JobsFeature.cpp}: Success response after
 UpdateJobExecution for job hello-world-job-1
2021-11-15T18:10:24.917Z [DEBUG] {JobsFeature.cpp}: Ack received for
 PublishUpdateJobExecutionStatus with code {0}
```

```
2021-11-15T18:10:24.918Z [DEBUG] {JobsFeature.cpp}: Removing ClientToken GmQ0HTzWGg
 from the updateJobExecution promises map
2021-11-15T18:10:24.918Z [DEBUG] {JobsFeature.cpp}: Success response after
 UpdateJobExecution for job hello-world-job-1
2021-11-15T18:10:25.861Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
```

3. Enquanto o AWS IoT Device Client estiver em execução e aguardando um trabalho, você pode enviar outro trabalho alterando o `job-id` valor e reexecutando o create-job da Etapa 1.

Quando terminar de executar os trabalhos, na janela do terminal, digite ^C (Control-c) para interromper o AWS IoT Device Client.

Tutorial: Limpando depois de executar os tutoriais do AWS IoT Device Client

Os procedimentos deste tutorial orientam você na remoção dos arquivos e recursos que você criou ao concluir os tutoriais neste percurso de aprendizado.

Procedimentos deste tutorial

- Etapa 1: Limpar seus dispositivos após criar demonstrações com o AWS IoT Device Client
- Etapa 2: Limpar suas demonstrações Conta da AWS após a criação com o AWS IoT Device Client

Etapa 1: Limpar seus dispositivos após criar demonstrações com o AWS IoT Device Client

Este tutorial descreve duas opções de como limpar o cartão microSD depois de criar as demonstrações neste percurso de aprendizado. Escolha a opção que fornece o nível de segurança de que você precisa.

Observe que a limpeza do cartão microSD do dispositivo não remove nenhum AWS IoT recurso que você criou. Para limpar os AWS IoT recursos depois de limpar o cartão microSD do dispositivo, você deve revisar o tutorial em. the section called “Limpando depois de criar demonstrações com o AWS IoT Device Client”

Opção 1: como limpar regravando o cartão microSD

A maneira mais fácil e completa de limpar o cartão microSD depois de concluir os tutoriais neste percurso de aprendizado é substituir o cartão microSD por um arquivo de imagem salvo que você criou ao preparar o dispositivo pela primeira vez.

Esse procedimento usa o computador host local para gravar uma imagem salva do cartão microSD em um cartão microSD.

> **Note**
>
> Se o dispositivo não usa uma mídia de armazenamento removível para o sistema operacional, consulte o procedimento desse dispositivo.

Para gravar uma nova imagem no cartão microSD

1. No computador host local, localize a imagem salva do cartão microSD que você deseja gravar no cartão microSD.
2. Insira o cartão microSD no computador host local.
3. Usando uma ferramenta de imagem de cartão SD, grave o arquivo de imagem selecionado no cartão microSD.
4. Depois de gravar a imagem do sistema operacional Raspberry Pi no cartão microSD, ejete o cartão microSD e remova-o com segurança do computador host local.

O cartão microSD está pronto para ser usado.

Opção 2: como limpar excluindo diretórios de usuários

Para limpar o cartão microSD depois de concluir os tutoriais sem regravar a imagem do cartão microSD, você pode excluir os diretórios do usuário individualmente. Isso não é tão completo quanto regravar o cartão microSD por meio de uma imagem salva, pois não remove nenhum arquivo do sistema que possa ter sido instalado.

Se a remoção dos diretórios do usuário for suficientemente completa para suas necessidades, siga este procedimento.

Para excluir os diretórios de usuário desse percurso de aprendizado do dispositivo

1. Execute esses comandos para excluir os diretórios, subdiretórios e todos os arquivos que foram criados nesse percurso de aprendizado, na janela do terminal conectada ao dispositivo.

   > **Note**
   >
   > Depois de excluir esses diretórios e arquivos, você não poderá executar as demonstrações sem concluir os tutoriais novamente.

   ```
   rm -Rf ~/dc-configs
   rm -Rf ~/policies
   rm -Rf ~/messages
   rm -Rf ~/certs
   rm -Rf ~/.aws-iot-device-client
   ```

2. Execute esses comandos para excluir os diretórios e arquivos de origem da aplicação na janela do terminal conectada ao dispositivo.

   > **Note**
   >
   > Esses comandos não desinstalam nenhum programa. Eles removem apenas os arquivos de origem usados para criá-los e instalá-los. Depois de excluir esses arquivos, o AWS CLI e o AWS IoT Device Client podem não funcionar.

   ```
   rm -Rf ~/aws-cli
   rm -Rf ~/aws
   rm -Rf ~/aws-iot-device-client
   ```

Etapa 2: Limpar suas demonstrações Conta da AWS após a criação com o AWS IoT Device Client

Esses procedimentos ajudam você a identificar e remover os AWS recursos que você criou ao concluir os tutoriais neste plano de aprendizado.

Limpe AWS IoT os recursos

Esse procedimento ajuda você a identificar e remover os AWS IoT recursos que você criou ao concluir os tutoriais neste plano de aprendizado.

AWS IoT recursos criados neste plano de aprendizado

Tutorial	Recurso de objetos	Recurso de políticas
the section called “Instalar e configurar o IoT Device Client”	DevCliTestThing	DevCliTestThingPolicy
the section called “Comunicar-se com o cliente do dispositivo usando MQTT”	PubSubTestThing	PubSubTestThingPolicy
the section called “Executar trabalhos de IoT com o Device Client”	definido pelo usuário (pode haver mais de um)	definido pelo usuário (pode haver mais de um)

Para excluir os AWS IoT recursos, siga este procedimento para cada recurso que você criou

1. Substitua *thing_name* pelo nome do recurso do objeto que você deseja excluir e, em seguida, execute esse comando para listar os certificados anexados ao recurso do objeto por meio do computador host local.

   ```
   aws iot list-thing-principals --thing-name thing_name
   ```

   Esse comando retorna uma resposta como essa, que lista os certificados anexados a *thing_name*. Na maioria dos casos, haverá apenas um certificado na lista.

   ```
   {
       "principals": [
           "arn:aws:iot:us-
   west-2:57EXAMPLE833:cert/23853eea3cf0edc7f8a69c74abeafa27b2b52823cab5b3e156295e94b26ae8ac"
       ]
   }
   ```

2. Para cada certificado listado pelo comando anterior:

a. Substitua *certificate_ID* pelo ID de certificado do comando anterior. O ID do certificado são os caracteres alfanuméricos que seguem `cert/` no ARN retornado pelo comando anterior. Em seguida, execute esse comando para inativar o certificado.

```
aws iot update-certificate --new-status INACTIVE --certificate-
id certificate_ID
```

Se houver êxito, esse comando não retornará nada.

b. Substitua *certificate_ARN* pelo ARN do certificado da lista de certificados retornados anteriormente e, em seguida, execute esse comando para listar as políticas anexadas a esse certificado.

```
aws iot list-attached-policies --target certificate_ARN
```

Esse comando retorna uma resposta como essa, que lista as políticas anexadas ao certificado. Na maioria dos casos, haverá apenas uma política na lista.

```
{
    "policies": [
        {
            "policyName": "DevCliTestThingPolicy",
            "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/
DevCliTestThingPolicy"
        }
    ]
}
```

c. Para cada política anexada ao certificado:

i. Substitua *policy_name* pelo valor `policyName` do comando anterior, substitua *certificate_ARN* pelo ARN do certificado e execute esse comando para separar a política do certificado.

```
aws iot detach-policy --policy-name policy_name --target certificate_ARN
```

Se houver êxito, esse comando não retornará nada.

ii. Substitua *policy_name* pelo valor `policyName` e, em seguida, execute esse comando para ver se a política está anexada a mais certificados.

```
aws iot list-targets-for-policy --policy-name policy_name
```

Se o comando retornar uma lista vazia como essa, a política não será anexada a nenhum certificado, e você continuará listando as versões da política. Se ainda houver certificados anexados à política, continue com a etapa detach-thing-principal.

```
{
    "targets": []
}
```

iii. Substitua *policy_name* pelo valor `policyName` e, em seguida, execute esse comando para verificar as versões da política. Para excluir a política, ela deve ter somente uma versão.

```
aws iot list-policy-versions --policy-name policy_name
```

Se a política tiver apenas uma versão, como neste exemplo, você poderá pular para a etapa delete-policy e excluir a política agora.

```
{
    "policyVersions": [
        {
            "versionId": "1",
            "isDefaultVersion": true,
            "createDate": "2021-11-18T01:02:46.778000+00:00"
        }
    ]
}
```

Se a política tiver mais de uma versão, como neste exemplo, as versões da política com um valor `isDefaultVersion` de `false` devem ser excluídas antes que a política possa ser excluída.

```
{
    "policyVersions": [
        {
            "versionId": "2",
            "isDefaultVersion": true,
            "createDate": "2021-11-18T01:52:04.423000+00:00"
```

```
        },
        {
            "versionId": "1",
            "isDefaultVersion": false,
            "createDate": "2021-11-18T01:30:18.083000+00:00"
        }
    ]
}
```

Se você precisar excluir uma versão da política, substitua *policy_name* pelo valor `policyName`, substitua *version_ID* pelo valor `versionId` do comando anterior e execute esse comando para excluir uma versão da política.

```
aws iot delete-policy-version --policy-name policy_name --policy-version-
id version_ID
```

Se houver êxito, esse comando não retornará nada.

Depois de excluir uma versão da política, repita essa etapa até que a política tenha somente uma versão da política.

iv. Substitua *policy_name* pelo valor `policyName` e, em seguida, execute esse comando para excluir a política.

```
aws iot delete-policy --policy-name policy_name
```

d. Substitua *thing_name* pelo nome do objeto, substitua *certificate_ARN* pelo ARN do certificado e, em seguida, execute esse comando para separar o certificado do recurso do objeto.

```
aws iot detach-thing-principal --thing-name thing_name --
principal certificate_ARN
```

Se houver êxito, esse comando não retornará nada.

e. Substitua *certificate_ID* pelo ID de certificado do comando anterior. O ID do certificado são os caracteres alfanuméricos que seguem `cert/` no ARN retornado pelo comando anterior. Em seguida, execute esse comando para excluir o recurso do certificado.

```
aws iot delete-certificate --certificate-id certificate_ID
```

Se houver êxito, esse comando não retornará nada.

3. Substitua *thing_name* pelo nome do objeto e, em seguida, execute esse comando para excluir o objeto.

```
aws iot delete-thing --thing-name thing_name
```

Se houver êxito, esse comando não retornará nada.

Limpe AWS os recursos

Esse procedimento ajuda você a identificar e remover outros AWS recursos que você criou ao concluir os tutoriais neste plano de aprendizado.

Outros AWS recursos criados neste plano de aprendizado

Tutorial	Tipo de atributo	Nome ou ID do recurso
the section called “Executar trabalhos de IoT com o Device Client”	Objetos do Amazon S3	hello-world-job.json
the section called “Executar trabalhos de IoT com o Device Client”	AWS IoT recursos de trabalho	definido pelo usuário

Para excluir os AWS recursos criados nesse plano de aprendizado

1. Para excluir os trabalhos criados neste percurso de aprendizado

   a. Execute este comando para listar os trabalhos em seu Conta da AWS.

   ```
   aws iot list-jobs
   ```

   O comando retorna uma lista dos AWS IoT trabalhos em seu Conta da AWS e Região da AWS que se parece com isso.

   ```
   {
       "jobs": [
   ```

```
        {
            "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-
job-2",
            "jobId": "hello-world-job-2",
            "targetSelection": "SNAPSHOT",
            "status": "COMPLETED",
            "createdAt": "2021-11-16T23:40:36.825000+00:00",
            "lastUpdatedAt": "2021-11-16T23:40:41.375000+00:00",
            "completedAt": "2021-11-16T23:40:41.375000+00:00"
        },
        {
            "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-
job-1",
            "jobId": "hello-world-job-1",
            "targetSelection": "SNAPSHOT",
            "status": "COMPLETED",
            "createdAt": "2021-11-16T23:35:26.381000+00:00",
            "lastUpdatedAt": "2021-11-16T23:35:29.239000+00:00",
            "completedAt": "2021-11-16T23:35:29.239000+00:00"
        }
    ]
}
```

b. Para cada trabalho que você reconhece na lista como um trabalho que você criou neste plano de aprendizado, *`jobId`* substitua pelo `jobId` valor do trabalho a ser excluído e, em seguida, execute esse comando para excluir um AWS IoT trabalho.

```
aws iot delete-job --job-id jobId
```

Se o comando for bem-sucedido, ele não retornará nada.

2. Para excluir os documentos de trabalho que você armazenou em um bucket do Amazon S3 neste percurso de aprendizado.

a. Substitua *`bucket`* pelo nome do bucket que você usou e, em seguida, execute esse comando para listar os objetos no bucket do Amazon S3 que você usou.

```
aws s3api list-objects --bucket bucket
```

O comando retorna uma lista dos objetos do Amazon S3 no bucket que se parece com isso.

```
{
```

```
    "Contents": [
        {
            "Key": "hello-world-job.json",
            "LastModified": "2021-11-18T03:02:12+00:00",
            "ETag": "\"868c8bc3f56b5787964764d4b18ed5ef\"",
            "Size": 54,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "EXAMPLE",
                "ID":
 "e9e3d6ec1EXAMPLEf5bfb5e6bd0a2b6ed03884d1ed392a82ad011c144736a4ee"
            }
        },
        {
            "Key": "iot_job_firmware_update.json",
            "LastModified": "2021-04-13T21:57:07+00:00",
            "ETag": "\"7c68c591949391791ecf625253658c61\"",
            "Size": 66,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "EXAMPLE",
                "ID":
 "e9e3d6ec1EXAMPLEf5bfb5e6bd0a2b6ed03884d1ed392a82ad011c144736a4ee"
            }
        },
        {
            "Key": "order66.json",
            "LastModified": "2021-04-13T21:57:07+00:00",
            "ETag": "\"bca60d5380b88e1a70cc27d321caba72\"",
            "Size": 29,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "EXAMPLE",
                "ID":
 "e9e3d6ec1EXAMPLEf5bfb5e6bd0a2b6ed03884d1ed392a82ad011c144736a4ee"
            }
        }
    ]
}
```


b. Para cada objeto que você reconhece na lista como um objeto que você criou neste percurso de aprendizado, substitua *bucket* pelo valor nome do bucket e *key* pelo valor

chave do objeto a ser excluído e, em seguida, execute esse comando para excluir um objeto do Amazon S3.

```
aws s3api delete-object --bucket bucket --key key
```

Se o comando for bem-sucedido, ele não retornará nada.

Depois de excluir todos os AWS recursos e objetos que você criou ao concluir esse plano de aprendizado, você pode recomeçar e repetir os tutoriais.

Construindo soluções com o AWS IoT dispositivo SDKs

Os tutoriais desta seção ajudam você a percorrer as etapas para desenvolver uma solução de IoT que possa ser implantada em um ambiente de produção usando o. AWS IoT

Esses tutoriais podem levar mais tempo para serem concluídos do que os da seção sobre, the section called “Criando demonstrações com o AWS IoT Device Client” pois eles usam o AWS IoT dispositivo SDKs e explicam os conceitos que estão sendo aplicados com mais detalhes para ajudá-lo a criar soluções seguras e confiáveis.

Comece a criar soluções com o AWS IoT dispositivo SDKs

Esses tutoriais orientam você em diferentes AWS IoT cenários. Quando apropriado, os tutoriais usam o AWS IoT dispositivo. SDKs

Tópicos

- Tutorial: Conectando um dispositivo AWS IoT Core usando o AWS IoT Device SDK
- Criação de AWS IoT regras para rotear dados do dispositivo para outros serviços
- Reter o estado do dispositivo enquanto o dispositivo está off-line com as sombras do dispositivo
- Tutorial: Criando um autorizador personalizado para AWS IoT Core
- Tutorial: Monitorando a umidade do solo com o AWS IoT Raspberry Pi

Tutorial: Conectando um dispositivo AWS IoT Core usando o AWS IoT Device SDK

Este tutorial demonstra como conectar um dispositivo para AWS IoT Core que ele possa enviar e receber dados de AWS IoT e para. Depois de concluir este tutorial, seu dispositivo será configurado para se conectar AWS IoT Core e você entenderá como os dispositivos se comunicam com ele AWS IoT.

Tópicos

- Pré-requisitos
- Prepare seu dispositivo para AWS IoT
- Revisar o protocolo MQTT
- Revise o aplicativo de amostra do SDK do dispositivo pubsub.py
- Conecte seu dispositivo e comunique-se com AWS IoT Core
- Reveja os resultados
- Tutorial: Usar a AWS IoT Device SDK para C incorporado

Pré-requisitos

Antes de começar este tutorial, verifique se você tem o seguinte:

- Concluído Introdução aos AWS IoT Core tutoriais

  Na seção desse tutorial em que você deve the section called “Configurar o dispositivo”, selecione a opção the section called “Conectar um Raspberry Pi ou outro dispositivo” para seu dispositivo e use as opções da linguagem Python para configurá-lo.

  > **Note**
  >
  > Mantenha aberta a janela do terminal que você usa nesse tutorial porque você também vai usá-la.

- Um dispositivo que pode executar o AWS IoT Device SDK v2 para Python.

  Este tutorial mostra como conectar um dispositivo AWS IoT Core usando exemplos de código Python, que exigem um dispositivo relativamente poderoso. Se você estiver trabalhando com

dispositivos com recursos limitados, esses exemplos de código podem não funcionar com eles. Nesse caso, talvez você tenha mais êxito com o tutorial the section called “Como usar o AWS IoT Device SDK para C incorporado”.

- Recebeu as informações necessárias para se conectar ao dispositivo

  Para conectar seu dispositivo a AWS IoT, você deve ter informações sobre o nome da coisa, o nome do host e o número da porta.

  > **Note**
  >
  > Também é possível usar a autenticação personalizada para conectar dispositivos ao AWS IoT Core. Os dados de conexão transmitidos para a função do Lambda do autorizador dependem do protocolo usado.

  - Nome da coisa: O nome da AWS IoT coisa à qual você deseja se conectar. Você deve ter se registrado como seu dispositivo como uma AWS IoT coisa. Para obter mais informações, consulte Gerenciando dispositivos com AWS IoT.
  - Nome do host: o nome do host do endpoint de IoT específico da conta.
  - Número da porta: o número da porta à qual se conectar.

  Você pode usar o `configureEndpoint` método no SDK do AWS IoT Python para configurar o nome do host e o número da porta.

  ```
  myAWSIoTMQTTClient.configureEndpoint("random.iot.region.amazonaws.com", 8883)
  ```

Prepare seu dispositivo para AWS IoT

Em Introdução aos AWS IoT Core tutoriais, você preparou seu dispositivo e conta AWS para que eles pudessem se comunicar. Esta seção analisa os aspectos dessa preparação que se aplicam a qualquer conexão de dispositivo com AWS IoT Core o.

Para que um dispositivo se conecte a AWS IoT Core:

1. É necessário ter uma Conta da AWS.

   O procedimento em Configurar Conta da AWS descreve como criar um, Conta da AWS se você ainda não tiver um.

2. Nessa conta, você deve ter os seguintes AWS IoT recursos definidos para o dispositivo em sua Conta da AWS região.

   O procedimento em Crie AWS IoT recursos descreve como criar esses recursos para o dispositivo em sua Conta da AWS e Região.

   - Um certificado de dispositivo registrado AWS IoT e ativado para autenticar o dispositivo.

     O certificado geralmente é criado com e anexado a um AWS IoT objeto. Embora um objeto não seja necessário para a conexão de um dispositivo AWS IoT, ele disponibiliza AWS IoT recursos adicionais para o dispositivo.

   - Uma política anexada ao certificado do dispositivo que o autoriza a se conectar AWS IoT Core e realizar todas as ações desejadas.

3. Uma conexão com a Internet que pode acessar os endpoints Conta da AWS do seu dispositivo.

   Os endpoints do dispositivo estão descritos AWS IoT dados do dispositivo e endpoints de serviço e podem ser vistos na página de configurações do AWS IoT console.

4. Software de comunicação, como o SDKs fornecido pelo AWS IoT dispositivo. Este tutorial usa o AWS IoT Device SDK v2 para Python.

Revisar o protocolo MQTT

Antes de falarmos sobre o aplicativo de exemplo, é útil entender o protocolo MQTT. O protocolo MQTT oferece algumas vantagens sobre outros protocolos de comunicação de rede, como o HTTP, o que o torna uma escolha popular para dispositivos IoT. Esta seção analisa os principais aspectos do MQTT que se aplicam a este tutorial. Para obter informações sobre como o MQTT se compara ao HTTP, consulte Escolher um protocolo de aplicativo para a comunicação do dispositivo.

O MQTT usa um modelo de publish/subscribe comunicação

O protocolo MQTT usa um modelo publish/subscribe de comunicação com seu host. Esse modelo difere do request/response modelo usado pelo HTTP. Com o MQTT, os dispositivos estabelecem uma sessão com o host que é identificado por um ID de cliente exclusivo. Para enviar dados, os dispositivos publicam mensagens identificadas por tópicos para um agente de mensagens no host. Para receber mensagens do agente de mensagens, os dispositivos assinam tópicos enviando filtros de tópicos em solicitações de assinatura ao agente de mensagens.

O MQTT é compatível com sessões persistentes

O agente de mensagens recebe mensagens de dispositivos e publica mensagens em dispositivos que as assinaram. Com sessões persistentes —sessões que permanecem ativas mesmo quando o dispositivo iniciador está desconectado — os dispositivos podem recuperar mensagens que foram publicadas enquanto estavam desconectados. No lado do dispositivo, o MQTT é compatível com níveis de Qualidade de Serviço (QoS) que garantem que o host receba mensagens enviadas pelo dispositivo.

Revise o aplicativo de amostra do SDK do dispositivo pubsub.py

Esta seção analisa o aplicativo de amostra `pubsub.py` do AWS IoT Device SDK v2 para Python usado neste tutorial. Aqui, analisaremos como ele se conecta AWS IoT Core para publicar e assinar mensagens do MQTT. A próxima seção apresenta alguns exercícios para ajudar você a explorar como um dispositivo se conecta e se comunica com AWS IoT Core ele.

O aplicativo pubsub.py de amostra demonstra esses aspectos de uma conexão MQTT com: AWS IoT Core

- Protocolos de comunicação
- Sessões persistentes
- Qualidade do serviço
- Publicação de mensagens
- Assinatura de mensagens
- Desconexão e reconexão do dispositivo

Protocolos de comunicação

A amostra `pubsub.py` demonstra uma conexão MQTT usando os protocolos MQTT e MQTT via WSS. A biblioteca AWS common runtime (AWS CRT) fornece suporte ao protocolo de comunicação de baixo nível e está incluída no AWS IoT Device SDK v2 para Python.

MQTT

As chamadas de `pubsub.py` exemplo `mtls_from_path` (mostradas aqui) no `mqtt_connection_builder`para estabelecer uma conexão AWS IoT Core usando o protocolo MQTT. `mtls_from_path`usa certificados X.509 e TLS v1.2 para autenticar o dispositivo. A biblioteca AWS CRT lida com os detalhes de nível inferior dessa conexão.

```
mqtt_connection = mqtt_connection_builder.mtls_from_path(
```

```
    endpoint=args.endpoint,
    cert_filepath=args.cert,
    pri_key_filepath=args.key,
    ca_filepath=args.ca_file,
    client_bootstrap=client_bootstrap,
    on_connection_interrupted=on_connection_interrupted,
    on_connection_resumed=on_connection_resumed,
    client_id=args.client_id,
    clean_session=False,
    keep_alive_secs=6
)
```

`endpoint`

O Conta da AWS endpoint do seu dispositivo de IoT

No aplicativo de amostra, esse valor é passado pela linha de comando.

`cert_filepath`

O caminho para o arquivo de certificado do dispositivo

No aplicativo de amostra, esse valor é passado pela linha de comando.

`pri_key_filepath`

O caminho para o arquivo de chave privada do dispositivo que foi criado com seu arquivo de certificado

No aplicativo de amostra, esse valor é passado pela linha de comando.

`ca_filepath`

O caminho para o arquivo CA Raiz. Obrigatório somente se o servidor MQTT usar um certificado que ainda não esteja em seu armazenamento confiável.

No aplicativo de amostra, esse valor é passado pela linha de comando.

`client_bootstrap`

O objeto de runtime comum que trata as atividades de comunicação por soquete

No aplicativo de amostra, esse objeto é instanciado antes da chamada para `mqtt_connection_builder.mtls_from_path`.

`on_connection_interrupted`, `on_connection_resumed`

As funções de retorno de chamada para chamar quando a conexão do dispositivo for interrompida e retomada

`client_id`

O ID que identifica exclusivamente esse dispositivo no Região da AWS

No aplicativo de amostra, esse valor é passado pela linha de comando.

`clean_session`

Se deve iniciar uma nova sessão persistente ou, se houver uma, se reconectar a uma existente

`keep_alive_secs`

O valor keep alive, em segundos, a ser enviado na `CONNECT` solicitação. Um ping será enviado automaticamente nesse intervalo. Se o servidor não receber um ping após 1,5 vezes esse valor, ele assume que a conexão foi perdida.

MQTT via WSS

Os exemplos de chamada `pubsub.py` `websockets_with_default_aws_signing` (mostrados aqui) no `mqtt_connection_builder` para estabelecer conexão com o AWS IoT Core usando o protocolo MQTT via WSS. `websockets_with_default_aws_signing` cria uma conexão do MQTT via WSS usando o Signature V4 para autenticar o dispositivo.

```
mqtt_connection = mqtt_connection_builder.websockets_with_default_aws_signing(
    endpoint=args.endpoint,
    client_bootstrap=client_bootstrap,
    region=args.signing_region,
    credentials_provider=credentials_provider,
    websocket_proxy_options=proxy_options,
    ca_filepath=args.ca_file,
    on_connection_interrupted=on_connection_interrupted,
    on_connection_resumed=on_connection_resumed,
    client_id=args.client_id,
    clean_session=False,
    keep_alive_secs=6
)
```

`endpoint`

O Conta da AWS endpoint do seu dispositivo de IoT

No aplicativo de amostra, esse valor é passado pela linha de comando.

`client_bootstrap`

O objeto de runtime comum que trata as atividades de comunicação por soquete

No aplicativo de amostra, esse objeto é instanciado antes da chamada para `mqtt_connection_builder.websockets_with_default_aws_signing`.

`region`

A região de AWS assinatura usada pela autenticação Signature V4. Em `pubsub.py`, ele passa o parâmetro inserido na linha de comando.

No aplicativo de amostra, esse valor é passado pela linha de comando.

`credentials_provider`

As AWS credenciais fornecidas para uso na autenticação

No aplicativo de amostra, esse objeto é instanciado antes da chamada para `mqtt_connection_builder.websockets_with_default_aws_signing`.

`websocket_proxy_options`

Opções de proxy HTTP, se estiver usando um host proxy

No aplicativo de amostra, esse valor é inicializado antes da chamada para `mqtt_connection_builder.websockets_with_default_aws_signing`.

`ca_filepath`

O caminho para o arquivo CA Raiz. Obrigatório somente se o servidor MQTT usar um certificado que ainda não esteja em seu armazenamento confiável.

No aplicativo de amostra, esse valor é passado pela linha de comando.

`on_connection_interrupted`, `on_connection_resumed`

As funções de retorno de chamada para chamar quando a conexão do dispositivo for interrompida e retomada

`client_id`

O ID que identifica exclusivamente esse dispositivo no Região da AWS.

No aplicativo de amostra, esse valor é passado pela linha de comando.

`clean_session`

Se deve iniciar uma nova sessão persistente ou, se houver uma, se reconectar a uma existente

`keep_alive_secs`

O valor keep alive, em segundos, a ser enviado na `CONNECT` solicitação. Um ping será enviado automaticamente nesse intervalo. Se o servidor não receber um ping após 1,5 vezes esse valor, ele presume que a conexão foi perdida.

HTTPS

E quanto ao HTTPS? AWS IoT Core oferece suporte a dispositivos que publicam solicitações HTTPS. Do ponto de vista da programação, os dispositivos enviam solicitações HTTPS para AWS IoT Core o mesmo que qualquer outro aplicativo. Para ver um exemplo de um programa em Python que envia uma mensagem HTTP de um dispositivo, consulte o exemplo de código HTTPS usando a biblioteca do Python. `requests` Este exemplo envia uma mensagem para AWS IoT Core usar HTTPS de forma que a AWS IoT Core interprete como uma mensagem MQTT.

Embora AWS IoT Core ofereça suporte a solicitações HTTPS de dispositivos, certifique-se de revisar as informações para que você possa tomar uma decisão informada sobre qual protocolo usar para as comunicações do seu dispositivo. Escolher um protocolo de aplicativo para a comunicação do dispositivo

Sessões persistentes

No aplicativo de amostra, definir o parâmetro `clean_session` como `False` indica que a conexão deve ser persistente. Na prática, a conexão aberta por essa chamada se reconecta a uma sessão persistente existente, se houver. Caso contrário, ela cria e se conecta a uma nova sessão persistente.

Com uma sessão persistente, as mensagens enviadas ao dispositivo são armazenadas pelo agente de mensagens enquanto o dispositivo não está conectado. Quando um dispositivo se reconecta a uma sessão persistente, o agente de mensagens envia ao dispositivo todas as mensagens armazenadas nas quais ele se inscreveu.

Sem uma sessão persistente, o dispositivo não receberá mensagens enviadas enquanto o dispositivo não estiver conectado. A opção a ser usada depende do seu aplicativo e se as

mensagens que ocorrem enquanto um dispositivo não está conectado devem ser comunicadas. Para obter mais informações, consulte Sessões persistentes do MQTT.

Qualidade do serviço

Quando o dispositivo publica e assina mensagens, a Qualidade de Serviço (QoS) preferida pode ser definida. AWS IoT suporta os níveis de QoS 0 e 1 para operações de publicação e assinatura. Para obter mais informações sobre os níveis de QoS em AWS IoT, consulte. Opções de Qualidade de serviço (QoS) do MQTT

O tempo de execução do AWS CRT para Python define essas constantes para os níveis de QoS que ele suporta:

Níveis de qualidade de serviço do Python

Nível de QoS do MQTT	Valor simbólico do Python usado pelo SDK	Descrição
QoS nível 0	`mqtt.QoS.AT_MOST_ONCE`	Somente uma tentativa de enviar a mensagem será feita, seja ela recebida ou não. A mensagem pode não ser enviada, por exemplo, se o dispositivo não estiver conectado ou houver um erro na rede.
QoS nível 1	`mqtt.QoS.AT_LEAST_ONCE`	A mensagem é enviada repetidamente até que uma PUBACK confirmação seja recebida.

No aplicativo de amostra, as solicitações de publicação e assinatura são feitas com um nível de QoS de 1 (`mqtt.QoS.AT_LEAST_ONCE`).

- QoS na publicação

Quando um dispositivo publica uma mensagem com QoS nível 1, ele envia a mensagem repetidamente até receber uma PUBACK resposta do agente de mensagens. Se o dispositivo não estiver conectado, a mensagem será colocada na fila para ser enviada após a reconexão.

- QoS na assinatura

  Quando um dispositivo assina uma mensagem com QoS nível 1, o agente de mensagens salva as mensagens nas quais o dispositivo está inscrito até que elas possam ser enviadas ao dispositivo. O agente de mensagens reenvia as mensagens até receber uma resposta PUBACK do dispositivo.

Publicação de mensagens

Depois de estabelecer com sucesso uma conexão com AWS IoT Core, os dispositivos podem publicar mensagens. A amostra `pubsub.py` faz isso chamando a operação `publish` do `mqtt_connection` objeto.

```
mqtt_connection.publish(
    topic=args.topic,
    payload=message,
    qos=mqtt.QoS.AT_LEAST_ONCE
)
```

`topic`

O nome do tópico da mensagem que identifica a mensagem

No aplicativo de amostra, esse valor é passado pela linha de comando.

`payload`

A carga útil da mensagem formatada como uma string (por exemplo, um documento JSON)

No aplicativo de amostra, esse valor é passado pela linha de comando.

Um documento JSON é um formato de carga útil comum e reconhecido por outros AWS IoT serviços; no entanto, o formato de dados da carga útil da mensagem pode ser qualquer coisa com a qual os editores e assinantes concordem. Outros AWS IoT serviços, no entanto, reconhecem apenas JSON e CBOR, em alguns casos, para a maioria das operações.

`qos`

O nível de QoS para esta mensagem

Assinatura de mensagens

Para receber mensagens de AWS IoT outros serviços e dispositivos, os dispositivos assinam essas mensagens pelo nome do tópico. Os dispositivos podem assinar mensagens individuais especificando um nome de tópico e em um grupo de mensagens especificando um filtro de tópico, que pode incluir caracteres curinga. A amostra `pubsub.py` usa o código mostrado aqui para assinar mensagens e registrar as funções de retorno de chamada para processar a mensagem depois de recebida.

```
subscribe_future, packet_id = mqtt_connection.subscribe(
    topic=args.topic,
    qos=mqtt.QoS.AT_LEAST_ONCE,
    callback=on_message_received
)
subscribe_result = subscribe_future.result()
```

`topic`

Como assinar um tópico do. Pode ser um nome de tópico ou um filtro de tópico.

No aplicativo de amostra, esse valor é passado pela linha de comando.

`qos`

Se o agente de mensagens deve armazenar essas mensagens enquanto o dispositivo está desconectado.

Um valor de `mqtt.QoS.AT_LEAST_ONCE` (QoS nível 1) exige que uma sessão persistente seja especificada (`clean_session=False`) quando a conexão é criada.

`callback`

A função a ser chamada para processar a mensagem assinada.

A função `mqtt_connection.subscribe` retorna um future e um ID de pacote. Se a solicitação de assinatura foi iniciada com sucesso, o ID do pacote retornado será maior que 0. Para garantir que a assinatura tenha sido recebida e registrada pelo agente de mensagens, você deve aguardar o retorno do resultado da operação assíncrona, conforme mostrado no exemplo de código.

Função de retorno de chamada

O retorno de chamada na amostra `pubsub.py` processa as mensagens assinadas à medida que o dispositivo as recebe.

```
def on_message_received(topic, payload, **kwargs):
    print("Received message from topic '{}': {}".format(topic, payload))
    global received_count
    received_count += 1
    if received_count == args.count:
        received_all_event.set()
```

`topic`

O tópico da mensagem

Esse é o nome do tópico específico da mensagem recebida, mesmo que você tenha se inscrito em um filtro de tópicos.

`payload`

Carga útil da mensagem

O formato para isso é específico do aplicativo.

`kwargs`

Possíveis argumentos adicionais, conforme descrito em `mqtt.Connection.subscribe`.

Na amostra `pubsub.py`, `on_message_received` exibe somente o tópico e sua carga útil. Ele também conta as mensagens recebidas para finalizar o programa após o limite ser atingido.

Seu aplicativo avaliaria o tópico e a carga para determinar quais ações realizar.

Desconexão e reconexão do dispositivo

A amostra `pubsub.py` inclui funções de retorno de chamada que são chamadas quando o dispositivo é desconectado e quando a conexão é restabelecida. As ações que seu dispositivo executa nesses eventos são específicas do aplicativo.

Quando um dispositivo se conecta pela primeira vez, ele precisa se inscrever nos tópicos para receber. Se a sessão de um dispositivo estiver presente quando ele se reconectar, suas assinaturas serão restauradas e todas as mensagens armazenadas dessas assinaturas serão enviadas ao dispositivo após a reconexão.

Se a sessão de um dispositivo não existir mais quando ele se reconectar, ele deverá assinar novamente suas assinaturas. As sessões persistentes têm uma vida útil limitada e podem expirar quando o dispositivo é desconectado por muito tempo.

Conecte seu dispositivo e comunique-se com AWS IoT Core

Esta seção apresenta alguns exercícios para ajudá-lo a explorar diferentes aspectos da conexão do seu dispositivo a AWS IoT Core. Para esses exercícios, você usará o cliente de teste MQTT no AWS IoT console para ver o que seu dispositivo publica e publicar mensagens em seu dispositivo. Esses exercícios usam o exemplo `pubsub.py` do AWS IoT Device SDK v2 para Python e se baseiam na sua experiência com tutoriais do Tutoriais de conceitos básicos.

Nesta seção:

- Inscreva-se nos filtros de tópicos curinga
- Processar assinaturas de filtros de tópicos
- Publique mensagens do seu dispositivo

Para esses exercícios, você começará com o programa `pubsub.py` de amostra.

> **Note**
>
> Esses exercícios pressupõem que você concluiu os Tutoriais de conceitos básicos tutoriais e usou a janela do terminal do seu dispositivo a partir desse tutorial.

Inscreva-se nos filtros de tópicos curinga

Neste exercício, você modificará a linha de comando usada para chamar `pubsub.py` para assinar um filtro de tópico curinga e processará as mensagens recebidas com base no tópico da mensagem.

Procedimento de exercício

Para este exercício, imagine que seu dispositivo contenha um controle de temperatura e um controle de luz. Ele usa esses nomes de tópicos para identificar as mensagens sobre eles.

1. Antes de iniciar o exercício, tente executar esse comando nos Tutoriais de conceitos básicos tutoriais do seu dispositivo para garantir que tudo esteja pronto para o exercício.

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples
   ```

```
python3 pubsub.py --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/
certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
```

Você deve ver a mesma saída que viu no tutorial de introdução.

2. Para este exercício, altere esses parâmetros da linha de comando.

Ação	Parâmetro da linha de comando	Efeito
adicionar	--message ""	Configure `pubsub.py` para ouvir somente
adicionar	--count 2	Encerre o programa depois de receber duas mensagens
alteração	--topic `device/+/details`	Defina o filtro de tópicos para se inscrever em

Fazer essas alterações na linha de comando inicial resulta nessa linha de comando. Digite esse comando na janela do terminal do seu dispositivo.

```
python3 pubsub.py --message "" --count 2 --topic device/+/details --ca_file
 ~/certs/Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/
private.pem.key --endpoint your-iot-endpoint
```

O programa deve exibir algo semelhante ao seguinte:

```
Connecting to a3qexamplesffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-24d7cdcc-cc01-458c-8488-2d05849691e1'...
Connected!
Subscribing to topic 'device/+/details'...
Subscribed with QoS.AT_LEAST_ONCE
Waiting for all messages to be received...
```

Se você vir algo assim em seu terminal, seu dispositivo está pronto e ouvindo mensagens onde os nomes dos tópicos começam com `device` e terminam com `/detail`. Então, vamos testar isso.

3. Aqui estão algumas mensagens que seu dispositivo pode receber.

Nome do tópico	Carga útil da mensagem
`device/temp/details`	`{ "desiredTemp": 20, "currentTemp": 15 }`
`device/light/details`	`{ "desiredLight": 100, "currentLight": 50 }`

4. Usando o cliente de teste MQTT no AWS IoT console, envie as mensagens descritas na etapa anterior para o seu dispositivo.

   a. Abra o cliente de teste MQTT no AWS IoT console.

   b. Em Inscrever-se em um tópico, no campo assinatura do tópico, insira o filtro do tópico: **`device/+/details`** e, em seguida, selecione Assinar um tópico.

   c. Na coluna Assinaturas do cliente de teste MQTT, selecione dispositivo/+/detalhes.

   d. Para cada um dos tópicos na tabela anterior, faça o seguinte no cliente de teste do MQTT:

      1. Em Publicar, insira o valor da coluna Nome do tópico na tabela.
      2. No campo de carga útil da mensagem abaixo do nome do tópico, insira o valor da coluna Carga útil da mensagem na tabela.
      3. Observe a janela do terminal em que `pubsub.py` está sendo executada e, no cliente de teste do MQTT, escolha Publicar no tópico.

      Você deve ver que a mensagem foi recebida `pubsub.py` na janela do terminal.

Resultado do exercício

Com isso, `pubsub.py`, assinou as mensagens usando um filtro de tópico curinga, recebeu-as e as exibiu na janela do terminal. Observe como você se inscreveu em um único filtro de tópico e a função de retorno de chamada foi chamada para processar mensagens com dois tópicos distintos.

Processar assinaturas de filtros de tópicos

Com base no exercício anterior, modifique o aplicativo de `pubsub.py` amostra para avaliar os tópicos da mensagem e processar as mensagens inscritas com base no tópico.

Procedimento de exercício

Para avaliar o tópico da mensagem

1. Copie `pubsub.py` para `pubsub2.py`.
2. Abra `pubsub2.py` no seu editor de texto favorito ou IDE.
3. Em `pubsub2.py`, encontre a função `on_message_received`.
4. Em `on_message_received`, insira o código a seguir após a linha que começa com `print("Received message` e antes da linha que começa com `global received_count`.

```
    topic_parsed = False
    if "/" in topic:
        parsed_topic = topic.split("/")
        if len(parsed_topic) == 3:
            # this topic has the correct format
            if (parsed_topic[0] == 'device') and (parsed_topic[2] == 'details'):
                # this is a topic we care about, so check the 2nd element
                if (parsed_topic[1] == 'temp'):
                    print("Received temperature request: {}".format(payload))
                    topic_parsed = True
                if (parsed_topic[1] == 'light'):
                    print("Received light request: {}".format(payload))
                    topic_parsed = True
    if not topic_parsed:
        print("Unrecognized message topic.")
```

5. Salve suas alterações e execute o programa modificado usando essa linha de comando.

```
python3 pubsub2.py --message "" --count 2 --topic device/+/details --ca_file
 ~/certs/Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/
private.pem.key --endpoint your-iot-endpoint
```

6. No AWS IoT console, abra o cliente de teste MQTT.
7. Em Inscrever-se em um tópico, no campo assinatura do tópico, insira o filtro do tópico: **`device/+/details`** e, em seguida, selecione Assinar um tópico.
8. Na coluna Assinaturas do cliente de teste MQTT, selecione dispositivo/+/detalhes.
9. Para cada um dos tópicos nesta tabela, faça o seguinte no cliente de teste do MQTT:

Nome do tópico	Carga útil da mensagem
`device/temp/details`	`{ "desiredTemp": 20, "currentTemp": 15 }`
`device/light/details`	`{ "desiredLight": 100, "currentLight": 50 }`

1. Em Publicar, insira o valor da coluna Nome do tópico na tabela.
2. No campo de carga útil da mensagem abaixo do nome do tópico, insira o valor da coluna Carga útil da mensagem na tabela.
3. Observe a janela do terminal em que `pubsub.py` está sendo executada e, no cliente de teste do MQTT, escolha Publicar no tópico.

Você deve ver que a mensagem foi recebida `pubsub.py` na janela do terminal.

Você deverá ver algo semelhante a isso na janela do seu terminal.

```
Connecting to a3qexamplesffp-ats.iot.us-west-2.amazonaws.com with client ID 'test-
af794be0-7542-45a0-b0af-0b0ea7474517'...
Connected!
Subscribing to topic 'device/+/details'...
Subscribed with QoS.AT_LEAST_ONCE
Waiting for all messages to be received...
Received message from topic 'device/light/details': b'{ "desiredLight": 100,
 "currentLight": 50 }'
Received light request: b'{ "desiredLight": 100, "currentLight": 50 }'
Received message from topic 'device/temp/details': b'{ "desiredTemp": 20,
 "currentTemp": 15 }'
Received temperature request: b'{ "desiredTemp": 20, "currentTemp": 15 }'
2 message(s) received.
Disconnecting...
Disconnected!
```

Resultado do exercício

Neste exercício, você adicionou código para que o aplicativo de amostra reconheça e processe várias mensagens na função de retorno de chamada. Com isso, seu dispositivo pode receber mensagens e agir conforme elas.

Outra forma de seu dispositivo receber e processar várias mensagens é assinar mensagens diferentes separadamente e atribuir a cada assinatura sua própria função de retorno de chamada.

Publique mensagens do seu dispositivo

Você pode usar o aplicativo de amostra pubsub.py para publicar mensagens do seu dispositivo. Embora publique as mensagens como estão, elas não podem ser lidas como documentos JSON. Este exercício modifica o aplicativo de amostra para poder publicar documentos JSON na carga da mensagem que podem ser lidos por. AWS IoT Core

Procedimento de exercício

Neste exercício, a seguinte mensagem será enviada com o tópico `device/data`.

```
{
    "timestamp": 1601048303,
    "sensorId": 28,
    "sensorData": [
        {
        "sensorName": "Wind speed",
        "sensorValue": 34.2211224
        }
    ]
}
```

Para preparar seu cliente de teste MQTT para monitorar as mensagens deste exercício

1. Em Inscrever-se em um tópico, no campo assinatura do tópico, insira o filtro do tópico: **`device/data`** e, em seguida, selecione IAssinar um tópico.
2. Na coluna Assinaturas do cliente de teste MQTT, selecione dispositivo/dados.
3. Mantenha a janela do cliente de teste MQTT aberta para aguardar as mensagens do seu dispositivo.

Para enviar documentos JSON com o aplicativo de amostra pubsub.py

1. No seu dispositivo, copie `pubsub.py` para `pubsub3.py`.
2. Edite `pubsub3.py` para alterar a forma como ele formata as mensagens que publica.
   a. Abra `pubsub3.py` em um editor de texto.
   b. Localize esta linha de código:

      ```
      message = "{} [{}]".format(message_string, publish_count)
      ```

   c. Altere para:

      ```
      message = "{}".format(message_string)
      ```

   d. Localize esta linha de código:

      ```
      message_json = json.dumps(message)
      ```

   e. Altere para:

      ```
      message = "{}".json.dumps(json.loads(message))
      ```

   f. Salve as alterações.
3. No seu dispositivo, execute esse comando para enviar a mensagem duas vezes.

   ```
   python3 pubsub3.py  --ca_file ~/certs/Amazon-root-CA-1.pem  --cert ~/certs/
   device.pem.crt  --key ~/certs/private.pem.key  --topic device/data  --count 2 --
   message '{"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
    speed","sensorValue":34.2211224}]}'  --endpoint your-iot-endpoint
   ```

4. No cliente de teste MQTT, verifique se ele interpretou e formatou o documento JSON na carga útil da mensagem, da seguinte forma:

Por padrão, `pubsub3.py` também assina as mensagens que envia. Você deve ver que ele recebeu as mensagens na saída do aplicativo. A janela do terminal deve parecer com algo semelhante ao seguinte.

```
Connecting to a3qEXAMPLEsffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-5cff18ae-1e92-4c38-a9d4-7b9771afc52f'...
Connected!
Subscribing to topic 'device/data'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 2 message(s)
Publishing message to topic 'device/data':
 {"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}
Received message from topic 'device/data':
 b'{"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}'
Publishing message to topic 'device/data':
 {"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}
Received message from topic 'device/data':
 b'{"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}'
2 message(s) received.
Disconnecting...
Disconnected!
```

Resultado do exercício

Com isso, seu dispositivo pode gerar mensagens para enviar AWS IoT Core para testar a conectividade básica e fornecer mensagens do dispositivo AWS IoT Core para processamento. Por exemplo, você pode usar esse aplicativo para enviar dados de teste do seu dispositivo para testar ações de AWS IoT regras.

Reveja os resultados

Os exemplos deste tutorial proporcionaram a você uma experiência prática com os conceitos básicos de como os dispositivos podem se comunicar AWS IoT Core— uma parte fundamental de sua solução. AWS IoT Quando seus dispositivos conseguem se comunicar AWS IoT Core, eles podem passar mensagens para AWS serviços e outros dispositivos nos quais possam atuar. Da mesma forma, AWS serviços e outros dispositivos podem processar informações que resultam em mensagens enviadas de volta aos seus dispositivos.

Quando estiver pronto para explorar AWS IoT Core mais, experimente estes tutoriais:

- the section called “Envio de uma notificação do Amazon SNS”
- the section called “Armazenamento de dados do dispositivo em uma tabela do DynamoDB”
- the section called “Formatando uma notificação usando uma função AWS Lambda ”

Tutorial: Usar a AWS IoT Device SDK para C incorporado

Esta seção descreve como executar o AWS IoT Device SDK para C incorporado.

Procedimentos desta seção

- Etapa 1: instalar a AWS IoT Device SDK para C incorporado
- Etapa 2: configurar uma aplicação de amostra
- Etapa 3: criar e executar a aplicação de exemplo

Etapa 1: instalar a AWS IoT Device SDK para C incorporado

Em geral, o AWS IoT Device SDK para C incorporado destina-se a dispositivos com restrição de recursos que exigem um runtime de linguagem C otimizado. É possível usar o SDK em qualquer sistema operacional e hospedá-lo em qualquer tipo de processador (p. ex., MCUs e MPUs). Se você tiver mais memória e recursos de processamento disponíveis, recomendamos usar um dos SDKs de dispositivos e dispositivos móveis do AWS IoT de ordem superior (por exemplo, C++, Java, JavaScript e Python).

Em geral, o AWS IoT Device SDK para C incorporado destina-se a sistemas que usam MCUs ou MPUs low-end que executam sistemas operacionais incorporados. Para o exemplo de programação nesta seção, presumimos que o dispositivo usa Linux.

Example

1. Faça o download de AWS IoT Device SDK para C incorporado para o dispositivo no GitHub.

   ```
   git clone https://github.com/aws/aws-iot-device-sdk-embedded-c.git --recurse-
   submodules
   ```

   Isso cria um diretório chamado `aws-iot-device-sdk-embedded-c` no diretório atual.

2. Navegue até esse diretório e confira a versão mais recente. Consulte github.com/aws/aws-iot-device-sdk-embedded-C/tags para ver a tag de lançamento mais recente.

```
cd aws-iot-device-sdk-embedded-c
git checkout latest-release-tag
```

3. Instale o OpenSSL versão 1.1.0 ou posterior. As bibliotecas de desenvolvimento do OpenSSL geralmente são chamadas de “libssl-dev” ou “openssl-devel” quando instaladas por meio de um gerenciador de pacotes.

```
sudo apt-get install libssl-dev
```

Etapa 2: configurar uma aplicação de amostra

O AWS IoT Device SDK para C incorporado inclui aplicativos de exemplo para você experimentar. Para simplificar, este tutorial usa a aplicação `mqtt_demo_mutual_auth`, que ilustra como se conectar ao agente de mensagens do AWS IoT Core e assinar e publicar em tópicos MQTT.

1. Copie o certificado e a chave privada que você criou em Introdução aos AWS IoT Core tutoriais no diretório `build/bin/certificates`.

> **Note**
>
> Os certificados CA raiz e de dispositivo estão sujeitos a expiração ou revogação. Se esses certificados expirarem ou forem revogados, você deverá copiar um novo certificado da CA ou uma chave privada e um certificado do dispositivo no dispositivo.

2. É necessário configurar o exemplo com sua chave privada, certificado, certificado CA raiz e endpoint pessoal do AWS IoT Core. Navegue até o diretório `aws-iot-device-sdk-embedded-c/demos/mqtt/mqtt_demo_mutual_auth`.

   Se você tiver a AWS CLI instalada, será possível usar o comando para localizar o URL do endpoint da sua conta.

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

Se você não tiver a AWS CLI instalada, abra o console do AWS IoT. No painel de navegação, escolha Gerenciar e Objetos. Escolha o objeto de IoT para o dispositivo e escolha Interagir. Seu endpoint é exibido na seção HTTPS da página de detalhes do objeto.

3. Abra o arquivo `demo_config.h` e atualize os valores para o seguinte:

   AWS_IOT_ENDPOINT

   Seu endpoint pessoal.

   CLIENT_CERT_PATH

   O caminho do arquivo de certificado, por exemplo `certificates/device.pem.crt`".

   CLIENT_PRIVATE_KEY_PATH

   O nome do arquivo de chave privada, por exemplo `certificates/private.pem.key`.

   Por exemplo:

   ```
   // Get from demo_config.h
   // ================================================
   #define AWS_IOT_ENDPOINT                "my-endpoint-ats.iot.us-
   east-1.amazonaws.com"
   #define AWS_MQTT_PORT                   8883
   #define CLIENT_IDENTIFIER               "testclient"
   #define ROOT_CA_CERT_PATH               "certificates/AmazonRootCA1.crt"
   #define CLIENT_CERT_PATH                "certificates/my-device-cert.pem.crt"
   #define CLIENT_PRIVATE_KEY_PATH         "certificates/my-device-private-key.pem.key"
   // ================================================
   ```

4. Verifique se você tem a CMake instalada no dispositivo usando este comando.

   ```
   cmake --version
   ```

   Se vir as informações de versão para o compilador, você poderá continuar para a próxima seção.

   Se receber um erro ou não vir nenhuma informação, você deverá instalar o pacote cmake usando este comando.

```
sudo apt-get install cmake
```

Execute o comando cmake --version novamente e confirme se a CMake foi instalada e que você está pronto para continuar.

5. Verifique se você tem as ferramentas de desenvolvimento instaladas em seu dispositivo usando este comando.

```
gcc --version
```

Se vir as informações de versão para o compilador, você poderá continuar para a próxima seção.

Se receber um erro ou não vir nenhuma informação do compilador, você deverá instalar o pacote `build-essential` usando este comando.

```
sudo apt-get install build-essential
```

Execute o comando gcc --version novamente e confirme se as ferramentas de compilação foram instaladas e que você está pronto para continuar.

Etapa 3: criar e executar a aplicação de exemplo

Este procedimento explica como gerar a aplicação `mqtt_demo_mutual_auth` em seu dispositivo e conectá-la ao console de AWS IoT usando o AWS IoT Device SDK para C incorporado.

Como executar os aplicativos de exemplo do AWS IoT Device SDK para C incorporado

1. Navegue até `aws-iot-device-sdk-embedded-c` e crie um diretório de compilação.

```
mkdir build && cd build
```

2. Insira o comando CMake a seguir para gerar os arquivos de Makefiles necessários para compilação.

```
cmake ..
```

3. Digite o comando a seguir para compilar o arquivo de aplicação executável.

```
make
```

4. Execute o aplicativo `mqtt_demo_mutual_auth` com este comando.

```
cd bin
./mqtt_demo_mutual_auth
```

Você deve ver uma saída semelhante a:

```
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:584] Establishing a TLS session to a2zk5tjv9x07ct-ats.iot.us-west-2.amazonaws.com:8883.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1264] Creating an MQTT connection to a2zk5tjv9x07ct-ats.iot.us-west-2.amazonaws.com.
[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=2.
[INFO] [MQTT] [core_mqtt_serializer.c:970] CONNACK session present bit not set.
[INFO] [MQTT] [core_mqtt_serializer.c:912] Connection accepted.
[INFO] [MQTT] [core_mqtt.c:1526] Received MQTT CONNACK successfully from broker.
[INFO] [MQTT] [core_mqtt.c:1792] MQTT connection established with the broker.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1033] MQTT connection successfully established with broker.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1296] A clean MQTT connection is established. Cleaning up all the stored outgoing publishes.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1314] Subscribing to the MQTT topic testclient/example/topic.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1097] SUBSCRIBE sent for topic testclient/example/topic to broker.


[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=3.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:921] Subscribed to the topic testclient/example/topic. with maximum QoS 1.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1358] Sending Publish to the MQTT topic testclient/example/topic.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1195] PUBLISH sent for topic testclient/example/topic to broker with packet ID 2.


[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=2.
[INFO] [MQTT] [core_mqtt.c:1126] Ack packet deserialized with result: MQTTSuccess.
[INFO] [MQTT] [core_mqtt.c:1139] State record updated. New state=MQTTPublishDone.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:946] PUBACK received for packet id 2.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:672] Cleaned up outgoing publish packet with packet id 2.


[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=40.
[INFO] [MQTT] [core_mqtt.c:1015] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
```

O dispositivo está conectado à AWS IoT agora usando o AWS IoT Device SDK para C incorporado.

Também é possível usar o console da AWS IoT para visualizar as mensagens MQTT que a aplicação de exemplo está publicando. Para obter informações sobre como usar o cliente MQTT no console do AWS IoT, consulte the section called “Exibir mensagens MQTT com o cliente AWS IoT MQTT”.

Criação de AWS IoT regras para rotear dados do dispositivo para outros serviços

Esses tutoriais mostram como criar e testar AWS IoT regras usando algumas das ações de regras mais comuns.

AWS IoT as regras enviam dados de seus dispositivos para outros AWS serviços. Eles recebem mensagens MQTT específicas, formatam os dados nas cargas de mensagens e enviam o resultado para outros AWS serviços.

Recomendamos que você os experimente na ordem em que são mostrados aqui, mesmo que seu objetivo seja criar uma regra que use uma função do Lambda ou algo mais complexo. Os tutoriais são apresentados do básico ao complexo. Eles apresentam novos conceitos de forma incremental para ajudá-lo a aprender os conceitos que você pode usar para criar ações de regras que não têm um tutorial específico.

> **Note**
>
> AWS IoT as regras ajudam você a enviar os dados dos seus dispositivos de IoT para outros AWS serviços. Para fazer isso com sucesso, no entanto, você precisa de um conhecimento prático dos outros serviços para os quais deseja enviar dados. Embora esses tutoriais forneçam as informações necessárias para concluir as tarefas, talvez seja útil descobrir mais sobre os serviços para os quais você deseja enviar dados antes de usá-los em sua solução. Uma explicação detalhada dos outros AWS serviços está fora do escopo desses tutoriais.

Visão geral do cenário do tutorial

O cenário desses tutoriais é o de um dispositivo sensor climático que publica periodicamente os dados. Existem muitos desses dispositivos sensores nesse sistema imaginário. No entanto, os tutoriais desta seção se concentram em um único dispositivo e mostram como você pode acomodar vários sensores.

Os tutoriais desta seção mostram como usar AWS IoT regras para realizar as seguintes tarefas com esse sistema imaginário de dispositivos de sensores climáticos.

- Tutorial: como republicar uma mensagem MQTT

Este tutorial mostra como republicar uma mensagem MQTT recebida dos sensores meteorológicos como uma mensagem que contém somente a ID do sensor e o valor da temperatura. Ele usa somente serviços AWS IoT Core e demonstra uma consulta SQL simples e como usar o cliente MQTT para testar sua regra.

- Tutorial:r Como enviar uma notificação do Amazon SNS

  Este tutorial mostra como enviar uma mensagem SNS quando um valor de um dispositivo sensor meteorológico excede um valor específico. Ele se baseia nos conceitos apresentados no tutorial anterior e acrescenta como trabalhar com outro AWS serviço, o Amazon Simple Notification Service (Amazon SNS).

  Se você for novo no Amazon SNS, revise os exercícios de Conceitos básicos antes de começar este tutorial.

- Tutorial: Armazenamento de dados do dispositivo em uma tabela do DynamoDB

  Este tutorial mostra como armazenar os dados dos dispositivos de sensores climáticos em uma tabela de banco de dados. Ele usa a declaração de consulta de regras e os modelos de substituição para formatar os dados da mensagem para o serviço de destino, o Amazon DynamoDB.

  Se você é iniciante no DynamoDB, revise seus exercícios de introdução antes de começar este tutorial.

- Tutorial: Formatando uma notificação usando uma função AWS Lambda

  Este tutorial mostra como chamar uma função do Lambda para reformatar os dados do dispositivo e enviá-los como uma mensagem de texto. Ele adiciona um script Python e funções do AWS SDK em uma AWS Lambdafunção para formatar com os dados da carga útil da mensagem dos dispositivos de sensores meteorológicos e enviar uma mensagem de texto.

  Se você é novato no Lambda, revise seus exercícios de introdução antes de começar este tutorial.

AWS IoT visão geral das regras

Todos esses tutoriais criam AWS IoT regras.

Para que uma AWS IoT regra envie os dados de um dispositivo para outro AWS serviço, ela usa:

- Uma declaração de consulta de regra que consiste em:

  - Uma cláusula SQL SELECT que seleciona e formata os dados da carga útil da mensagem
  - Um filtro de tópico (o objeto FROM na instrução de consulta de regra) que identifica as mensagens a serem usadas
  - Uma declaração condicional opcional (uma cláusula SQL WHERE) que especifica condições específicas sobre as quais agir
- Pelo menos uma ação de regra

Os dispositivos publicam mensagens em tópicos MQTT. O filtro de tópicos na instrução SQL SELECT identifica os tópicos do MQTT aos quais aplicar a regra. Os campos especificados na instrução SQL SELECT formatam os dados da carga útil da mensagem MQTT recebida para uso pelas ações da regra. Para obter uma lista completa de ações de regra, consulte Ações de regra da AWS IoT.

Tutoriais nesta seção

- Tutorial: como republicar uma mensagem MQTT
- Tutorial:r Como enviar uma notificação do Amazon SNS
- Tutorial: Armazenamento de dados do dispositivo em uma tabela do DynamoDB
- Tutorial: Formatando uma notificação usando uma função AWS Lambda

Tutorial: como republicar uma mensagem MQTT

Este tutorial demonstra como criar uma AWS IoT regra que publica uma mensagem MQTT quando uma mensagem MQTT especificada é recebida. A carga útil da mensagem recebida pode ser modificada pela regra antes de ser publicada. Isso possibilita a criação de mensagens personalizadas para aplicativos específicos sem a necessidade de alterar o dispositivo ou o firmware. Você também pode usar o aspecto de filtragem de uma regra para publicar mensagens somente quando uma condição específica for atendida.

As mensagens republicadas por uma regra agem como mensagens enviadas por qualquer outro AWS IoT dispositivo ou cliente. Os dispositivos podem assinar as mensagens republicadas da mesma forma que assinam qualquer outro tópico de mensagem do MQTT.

O que você aprenderá neste tutorial:

- Como usar consultas e funções SQL simples em uma instrução de consulta de regra
- Como usar o cliente MQTT para testar uma regra AWS IoT

Este tutorial leva cerca de 30 minutos para ser concluído.

Neste tutorial, você vai:

- Analise os tópicos e AWS IoT regras do MQTT
- Etapa 1: criar uma AWS IoT regra para republicar uma mensagem MQTT
- Etapa 2: Testar a nova regra
- Etapa 3: revisar os resultados e as próximas etapas

Antes de começar este tutorial, verifique se você tem o seguinte:

- Configurar Conta da AWS

  Você precisará do seu AWS IoT console Conta da AWS e do console para concluir este tutorial.
- Revisado Exibir mensagens MQTT com o cliente AWS IoT MQTT

  Certifique-se de que você pode usar o cliente MQTT para fazer a assinatura e publicar em um tópico. Você usará o cliente MQTT para testar a nova regra neste procedimento.

Analise os tópicos e AWS IoT regras do MQTT

Antes de falar sobre AWS IoT regras, é útil entender o protocolo MQTT. Em soluções de IoT, o protocolo MQTT oferece algumas vantagens em relação a outros protocolos de comunicação de rede, como HTTP, o que o torna uma escolha popular para uso por dispositivos de IoT. Esta seção analisa os principais aspectos do MQTT conforme eles se aplicam a este tutorial. Para obter informações sobre como o MQTT se compara ao HTTP, consulte Escolher um protocolo de aplicativo para a comunicação do dispositivo.

Protocolo MQTT

O protocolo MQTT usa um modelo publish/subscribe de comunicação com seu host. Para enviar dados, os dispositivos publicam mensagens identificadas por tópicos para o agente de AWS IoT mensagens. Para receber mensagens do agente de mensagens, os dispositivos assinam os tópicos que receberão enviando filtros de tópicos nas solicitações de assinatura para o agente de mensagens. O mecanismo de regras AWS IoT recebe mensagens MQTT do agente de mensagens.

AWS IoT regras

AWS IoT as regras consistem em uma instrução de consulta de regras e em uma ou mais ações de regra. Quando o mecanismo de regras AWS IoT recebe uma mensagem MQTT, esses elementos agem na mensagem da seguinte maneira.

- Declaração de consulta de regra

  A instrução de consulta da regra descreve os tópicos do MQTT a serem usados, interpreta os dados da carga útil da mensagem e formata os dados conforme descrito por uma instrução SQL semelhante às instruções usadas por bancos de dados SQL populares. O resultado da instrução de consulta são os dados enviados para as ações da regra.

- Ação da regra

  Cada ação de regra em uma regra atua nos dados resultantes da declaração de consulta da regra. AWS IoT suporta muitas ações de regras. Neste tutorial, no entanto, você se concentrará na ação da regra Nova publicação, que publica o resultado da instrução de consulta como uma mensagem MQTT com um tópico específico.

Etapa 1: criar uma AWS IoT regra para republicar uma mensagem MQTT

A AWS IoT regra que você criará neste tutorial se inscreve nos tópicos do `device/`*`device_id`*`/data` MQTT, onde *`device_id`* está o ID do dispositivo que enviou a mensagem. Esses tópicos são descritos por um filtro de tópicos como`device/+/data`, em que + é um caractere curinga que corresponde a qualquer sequência de caracteres entre os dois caracteres de barra.

Quando a regra recebe uma mensagem de um tópico correspondente, ela republica os valores `device_id` e `temperature` e como uma nova mensagem MQTT com o tópico `device/data/temp`.

Por exemplo, a carga útil de uma mensagem MQTT com o tópico `device/22/data` tem a seguinte aparência:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
```

```
}
```

A regra pega o valor `temperature` da carga útil da mensagem e o `device_id` do tópico e os republica como uma mensagem MQTT com o tópico `device/data/temp` tópico e uma carga útil de mensagem com a seguinte aparência:

```
{
  "device_id": "22",
  "temperature": 28
}
```

Com essa regra, dispositivos que precisam apenas do ID do dispositivo e dos dados de temperatura se inscrevem no tópico `device/data/temp` para receber somente essas informações.

Para criar uma regra que republica uma mensagem MQTT

1. Abra o hub de regras do AWS IoT console.
2. Em Regras, escolha Criar e comece a criar sua nova regra.
3. Na parte superior da opção Criar uma regra:
   a. Em Nome, insira o nome da regra. Para este tutorial, dê o nome **`republish_temp`**.

      Lembre-se de que o nome de uma regra deve ser exclusivo em sua conta e região e não pode ter espaços. Usamos um caractere sublinhado nesse nome para separar as duas palavras no nome da regra.
   b. Em Descrição, descreva a regra.

      Uma descrição significativa ajuda você a lembrar ao que essa regra se refere e por que você a criou. A descrição pode ser tão longa quanto necessário, portanto, seja o mais detalhista possível.
4. Na declaração de consulta de regra de Criar uma regra:
   a. Em Uso da versão SQL, selecione**`2016-03-23`**.
   b. Na caixa de edição Instrução de consulta de regra, insira a instrução:

      ```
      SELECT topic(2) as device_id, temperature FROM 'device/+/data'
      ```

      Esta declaração:

- Recebe mensagens MQTT com um tópico que corresponda ao filtro de tópicos `device/+/data`.
- Seleciona o segundo elemento da cadeia de caracteres do tópico e o atribui ao campo `device_id`.
- Seleciona o campo de valor `temperature` da carga útil da mensagem e o atribui ao campo `temperature`.

5. Em Definir uma ou mais ações:

   a. Para abrir a lista de ações de regra para essa regra, escolha Adicionar ação.

   b. Em Selecionar uma ação, escolha Republicar uma mensagem em um AWS IoT tópico.

   c. Na parte inferior da lista de ações, escolha Configurar ação para abrir a página de configuração da ação selecionada.

6. Em Configurar ação:

   a. Em Tópico, insira **`device/data/temp`**. Esse é o tópico MQTT da mensagem que essa regra publicará.

   b. Em Qualidade de Serviço, escolha 0 - A mensagem é entregue zero ou mais vezes.

   c. Em Escolha ou crie uma função para conceder AWS IoT acesso para realizar essa ação:

      i. Selecione Criar perfil. A caixa de diálogo Criar um novo perfil é aberta.

      ii. Insira um nome que descreva o novo perfil. Neste tutorial, use **`republish_role`**.

      Quando você cria um novo perfil, as políticas corretas para executar a ação da regra são criadas e anexadas ao novo perfil. Se você alterar o tópico dessa ação de regra ou usar esse perfil em outra ação de regra, deverá atualizar a política desse perfil para autorizar o novo tópico ou ação. Para atualizar uma função existente, escolha Atualizar função nesta seção.

      iii. Escolha Criar função para criar a função e fechar a caixa de diálogo.

   d. Escolha Adicionar ação para adicionar a ação à regra e retornar à página Criar uma regra.

7. A ação Republicar uma mensagem em um AWS IoT tópico agora está listada em Definir uma ou mais ações.

   No bloco da nova ação, abaixo de Republicar uma mensagem em um AWS IoT tópico, você pode ver o tópico no qual sua ação de republicação será publicada.

Essa é a única ação de regra que você adicionará a essa regra.

8. Em Criar uma regra, desça até a parte inferior e escolha Criar regra para criar a regra e concluir esta etapa.

Etapa 2: Testar a nova regra

Para testar sua nova regra, você usará o cliente MQTT para publicar e assinar as mensagens MQTT usadas por essa regra.

Abra o cliente MQTT no AWS IoT console em uma nova janela. Isso permitirá que você edite a regra sem perder a configuração do seu cliente MQTT. O cliente MQTT não retém nenhuma assinatura ou logs de mensagens se você deixar que ele vá para outra página no console.

Você pode usar o cliente MQTT para testar a regra

1. No cliente MQTT no AWS IoT console, assine os tópicos de entrada, neste caso, `device/+/data`.

   a. No Cliente MQTT, em Assinaturas, selecione Assine um tópico.

   b. Em Tópico de assinatura, insira o tópico do filtro de tópico de entrada, **`device/+/data`**.

   c. Deixe os demais campos com as configurações padrão.

   d. Escolha Assinar um tópico.

   Na coluna Assinaturas, em Publicar em um tópico , **`device/+/data`** é exibido.

2. Assine o tópico que sua regra publicará: `device/data/temp`.

   a. Em Assinaturas, escolha Assinar um tópico novamente e, em Tópico de assinatura, insira o tópico da mensagem republicada, **`device/data/temp`**.

   b. Deixe os demais campos com as configurações padrão.

   c. Escolha Assinar um tópico.

   Na coluna Assinaturas, em dispositivo/+/dado, **`device/data/temp`** é exibido.

3. Publique uma mensagem no tópico de entrada com um ID de dispositivo específico, **`device/22/data`**. Você não pode publicar nos tópicos MQTT que contenham caracteres curinga.

   a. No cliente MQTT, em Assinaturas, selecione Publicar em um tópico.

b. No campo Publicar, insira o nome do tópico de entrada, **`device/22/data`**.

c. Copie os dados de amostra mostrados aqui e, na caixa de edição abaixo do nome do tópico, cole os dados de amostra.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

d. Para enviar sua mensagem MQTT, escolha Publicar no tópico.

4. Revise as mensagens enviadas.

a. No cliente MQTT, em Assinaturas, há um ponto verde ao lado dos dois tópicos nos quais você se inscreveu anteriormente.

Os pontos verdes indicam que uma ou mais mensagens novas foram recebidas desde a última vez que você as viu.

b. Em Assinaturas, escolha dispositivo/+/dado para verificar se a carga útil da mensagem corresponde ao que você acabou de publicar e tem a seguinte aparência:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

c. Em Assinaturas, escolha verificar se device/data/tempa carga útil da mensagem republicada tem a seguinte aparência:

```
{
  "device_id": "22",
```

```
  "temperature": 28
}
```

Observe que o valor `device_id` valor é uma string citada e o valor `temperature` é numérico. Isso ocorre porque a `topic()`função extraiu a string do nome do tópico da mensagem de entrada, enquanto o valor `temperature` usa o valor numérico da carga útil da mensagem de entrada.

Se você quiser transformar o valor `device_id` em um valor numérico, substitua `topic(2)` na instrução de consulta de regra por:

```
cast(topic(2) AS DECIMAL)
```

Observe que converter o valor `topic(2)` em um valor numérico só funcionará se essa parte do tópico contiver somente caracteres numéricos.

5. Se você perceber que a mensagem correta foi publicada no device/data/temptópico, sua regra funcionou. Veja o que mais você pode aprender sobre a ação da regra de republicação na próxima seção.

   Se você não ver que a mensagem correta foi publicada no dispositivo/+/data ou nos device/data/temptópicos, consulte as dicas de solução de problemas.

Solução de problemas da regra de republicação de mensagens

Aqui estão algumas objetos para verificar caso você não esteja vendo os resultados esperados.

- Você recebeu um banner de erro

  Se um erro apareceu quando você publicou a mensagem de entrada, corrija esse erro primeiro. As etapas a seguir podem ajudá-lo a corrigir esse erro.

- Você não vê a mensagem de entrada no cliente MQTT

  Toda vez que você publica sua mensagem de entrada no tópico `device/22/data`, essa mensagem deve aparecer no cliente MQTT, se tiver assinado o filtro de tópicos `device/+/data` conforme descrito no procedimento.

  Pontos importantes

  - Verifique o filtro de tópicos em que você fez a assinatura

Se você fez a assinatura no tópico da mensagem de entrada conforme descrito no procedimento, deverá ver uma cópia da mensagem de entrada toda vez que publicá-la.

Se você não visualizar a mensagem, verifique o nome do tópico em que você fez a assinatura e compare-o com o tópico no qual você publicou. Os nomes dos tópicos diferenciam maiúsculas de minúsculas e o tópico no qual você fez a assinatura deve ser idêntico ao tópico no qual você publicou a carga útil da mensagem.

- Verifique a função de publicação de mensagens

  No cliente MQTT, em Assinaturas, escolha dispositivo/+/dados, verifique o tópico da mensagem de publicação e escolha Publicar no tópico. Você deve ver a carga útil da mensagem na caixa de edição abaixo do tópico aparecer na lista de mensagens.

- Você não vê sua mensagem republicada no cliente MQTT

  Para que sua regra funcione, ela deve ter a política correta que a autorize a receber e republicar uma mensagem e deve receber a mensagem.

  Pontos importantes

  - Verifique o Região da AWS do seu cliente MQTT e a regra que você criou

    O console no qual você está executando o cliente MQTT deve estar na mesma região AWS da regra que você criou.

  - Verifique o tópico da mensagem de entrada na instrução de consulta da regra

    Para que a regra funcione, ela deve receber uma mensagem com o nome do tópico que corresponda ao filtro do tópico na cláusula FROM da instrução de consulta da regra.

    Verifique a ortografia do filtro de tópico na declaração de consulta de regra com a do tópico no cliente MQTT. Os nomes dos tópicos diferenciam maiúsculas de minúsculas e o tópico da mensagem deve corresponder ao filtro de tópico na instrução de consulta da regra.

  - Verifique o conteúdo da carga útil da mensagem de entrada

    Para que a regra funcione, ela deve encontrar o campo de dados na carga da mensagem declarada na instrução SELECT.

    Verifique a ortografia do campo `temperature` na instrução de consulta da regra com a da carga útil da mensagem no cliente MQTT. Os nomes dos campos diferenciam maiúsculas de

minúsculas e o campo `temperature` na instrução de consulta da regra deve ser idêntico ao campo `temperature` na carga útil da mensagem.

Verifique se o documento JSON na carga útil da mensagem está formatado corretamente. Se o JSON tiver algum erro, como uma vírgula ausente, a regra não poderá lê-lo.

- Verifique o tópico da mensagem republicada na ação da regra

  O tópico no qual a ação da regra de republicação publica a nova mensagem deve corresponder ao tópico no qual você fez a assinatura no cliente MQTT.

  Abra a regra que você criou no console e verifique o tópico no qual a ação da regra republicará a mensagem.

- Verifique a função que está sendo usada pela regra

  A ação da regra deve ter permissão para receber o tópico original e publicar o novo tópico.

  As políticas que autorizam a regra a receber dados de mensagens e republicá-los são específicas para os tópicos usados. Se você alterar o tópico usado para republicar os dados da mensagem, deverá atualizar a função da ação de regra para atualizar sua política conforme o tópico atual.

  Se você suspeitar que esse é o problema, edite a ação da regra de republicação e crie uma nova função. As novas funções criadas pela ação da regra recebem as autorizações necessárias para realizar essas ações.

Etapa 3: revisar os resultados e as próximas etapas

Neste tutorial

- Você usou uma consulta SQL simples e algumas funções em uma instrução de consulta de regra para produzir uma nova mensagem MQTT.
- Você criou uma regra que republicou essa nova mensagem.
- Você usou o cliente MQTT para testar sua AWS IoT regra.

Próximas etapas

Depois de republicar algumas mensagens com essa regra, experimente usá-la para ver como a alteração de alguns aspectos do tutorial afeta a mensagem republicada. Aqui estão algumas ideias para você começar.

- Altere o *device_id* tópico da mensagem de entrada e observe o efeito na carga útil da mensagem republicada.
- Altere os campos selecionados na instrução de consulta de regras e observe o efeito na carga útil da mensagem republicada.
- Experimente o próximo tutorial desta série e aprenda como Tutorial:r Como enviar uma notificação do Amazon SNS.

A ação de regra Republicar usada neste tutorial também pode ajudá-lo a depurar instruções de consulta de regras. Por exemplo, você pode adicionar essa ação a uma regra para ver como sua instrução de consulta de regra está formatando os dados usados por suas ações de regra.

Tutorial:r Como enviar uma notificação do Amazon SNS

Este tutorial demonstra como criar uma AWS IoT regra que envia dados de mensagens MQTT para um tópico do Amazon SNS para que possam ser enviados como uma mensagem de texto SMS.

Neste tutorial, você cria uma regra que envia dados de mensagens de um sensor meteorológico para todos os assinantes de um tópico do Amazon SNS, sempre que a temperatura exceder o valor definido na regra. A regra detecta quando a temperatura relatada excede o valor definido pela regra e, em seguida, cria uma nova carga útil de mensagem que inclui somente a ID do dispositivo, a temperatura relatada e o limite de temperatura que foi excedido. A regra envia a nova carga útil da mensagem como um documento JSON para um tópico do SNS, que notifica todos os assinantes do tópico do SNS.

O que você aprenderá neste tutorial:

- Como criar e testar uma notificação do Amazon SNS
- Como chamar uma notificação do Amazon SNS a partir de uma regra AWS IoT
- Como usar consultas e funções SQL simples em uma instrução de consulta de regra
- Como usar o cliente MQTT para testar uma regra AWS IoT

Este tutorial leva cerca de 30 minutos para ser concluído.

Neste tutorial, você vai:

- Etapa 1: criar um tópico do Amazon SNS que envia uma mensagem de texto SMS
- Etapa 2: criar uma AWS IoT regra para enviar a mensagem de texto
- Etapa 3: testar a AWS IoT regra e a notificação do Amazon SNS
- Etapa 4: revisar os resultados e as próximas etapas

Antes de começar este tutorial, verifique se você tem o seguinte:

- Configurar Conta da AWS

  Você precisará do seu AWS IoT console Conta da AWS e do console para concluir este tutorial.
- Revisado Exibir mensagens MQTT com o cliente AWS IoT MQTT

  Certifique-se de que você pode usar o cliente MQTT para fazer a assinatura e publicar em um tópico. Você usará o cliente MQTT para testar a nova regra neste procedimento.
- Revisado o Amazon Simple Notification Service

  Se você nunca usou o Amazon SNS, consulte a opção Configuração do acesso para o Amazon SNS. Se você já concluiu outros AWS IoT tutoriais, você já Conta da AWS deve estar configurado corretamente.

Etapa 1: criar um tópico do Amazon SNS que envia uma mensagem de texto SMS

Esse procedimento explica como criar o tópico do Amazon SNS para o qual seu sensor meteorológico pode enviar dados de mensagens. O tópico do Amazon SNS então notificará todos os assinantes por meio de uma mensagem de texto SMS sobre o limite de temperatura que foi excedido.

Para criar um tópico do Amazon SNS que envia uma mensagem de texto SMS

1. Crie um tópico do Amazon SNS.

   a. Faça login no console do Amazon SNS.
   b. No painel de navegação à esquerda, selecione Tópicos.
   c. Na página Tópicos, escolha Criar tópico.
   d. Em Detalhes, escolha o tipo Padrão. Por padrão, o console cria um tópico FIFO.
   e. Em Nome, insira o nome do tópico do SNS. Para este tutorial, insira **`high_temp_notice`**.

f. Role até o final do formulário e escolha Criar tópico.

O console abrirá a página Detalhes do tópico.

2. Crie uma assinatura do Amazon SNS.

> **Note**
>
> O número de telefone que você usa nesta assinatura pode incorrer em cobranças de mensagens de texto a partir das mensagens que você enviará neste tutorial.

a. Na página de detalhes do tópico high_temp_notice, selecione Criar assinatura.

b. Em Criar assinatura, na seção Detalhes, na lista Protocolo, escolha SMS.

c. No Endpoint, insira o número de um telefone que pode receber mensagens de texto. Certifique-se de inseri-lo de forma que comece com um +, inclua o código do país e da área e não inclua nenhum outro caractere de pontuação.

d. Selecione Criar assinatura.

3. Teste a notificação do Amazon SNS.

a. No console Amazon SNS , no painel de navegação esquerdo, selecione Tópicos.

b. Para abrir a página de detalhes do tópico, em Tópicos, na lista de tópicos, escolha high_temp_notice.

c. Para abrir a página Publicar mensagem no tópico, na página de detalhes high_temp_notice, selecione Publicar mensagem.

d. Em Publicar mensagem no tópico, na seção Corpo da mensagem, em Corpo da mensagem a ser enviada ao endpoint, insira uma mensagem curta.

e. Navegue para baixo até o final da página e selecione Publicar mensagem.

f. No telefone com o número usado anteriormente ao criar a assinatura, confirme se a mensagem foi recebida.

Se você não recebeu a mensagem de teste, verifique novamente o número de telefone e as configurações do seu telefone.

Certifique-se de poder publicar mensagens de teste no console do Amazon SNS antes de continuar o tutorial.

Etapa 2: criar uma AWS IoT regra para enviar a mensagem de texto

A AWS IoT regra que você criará neste tutorial se inscreve nos tópicos do `device/`*`device_id`*`/data` MQTT, onde *`device_id`* está o ID do dispositivo que enviou a mensagem. Esses tópicos são descritos em um filtro de tópicos como `device/+/data`, em que `+` é um caractere curinga que corresponde a qualquer sequência de caracteres entre os dois caracteres de barra. Essa regra também testa o valor do campo `temperature` na carga útil da mensagem.

Quando a regra recebe uma mensagem de um tópico correspondente, ela pega o valor *`device_id`* do nome do tópico, o valor `temperature` da carga útil da mensagem, adiciona um valor constante ao limite que está testando e envia esses valores como um documento JSON para um tópico de notificação do Amazon SNS.

Por exemplo, uma mensagem MQTT do dispositivo de sensor meteorológico número 32 usa o tópico `device/32/data` e tem uma carga de mensagem semelhante a esta:

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

A declaração de consulta de regra da regra pega o valor `temperature` da carga útil da mensagem, *`device_id`* do nome do tópico e adiciona o valor `max_temperature` constante para enviar uma carga útil de mensagem semelhante a esta para o tópico do Amazon SNS:

```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30
}
```

Para criar uma AWS IoT regra para detectar um valor de temperatura acima do limite e criar os dados a serem enviados ao tópico do Amazon SNS

1. Abra o hub de regras do AWS IoT console.

2. Se essa for sua primeira regra, escolha Criar ou Criar uma regra.
3. Em Criar uma regra:

   a. Em Nome, insira **`temp_limit_notify`**.

      Lembre-se de que o nome de uma regra deve ser exclusivo em sua região Conta da AWS e não pode ter espaços. Usamos um caractere de sublinhado nesse nome para separar as palavras no nome da regra.

   b. Em Descrição, descreva a regra.

      Uma descrição significativa facilita lembrar o que essa regra faz e por que você a criou. A descrição pode ser tão longa quanto necessário, portanto, seja o mais detalhista possível.
4. Na declaração de consulta de regra de Criar uma regra:

   a. Em Uso da versão SQL, selecione 23-03-2016.

   b. Na caixa de edição Instrução de consulta de regra, insira a instrução:

      ```
      SELECT topic(2) as device_id,
          temperature as reported_temperature,
          30 as max_temperature
        FROM 'device/+/data'
        WHERE temperature > 30
      ```

      Esta declaração:

      - Recebe mensagens MQTT com um tópico que corresponda ao filtro de tópicos `device/+/data` e que tenha um valor `temperature` maior que 30.
      - Seleciona o segundo elemento da cadeia de caracteres do tópico e o atribui ao campo `device_id`.
      - Seleciona o campo `temperature` de valor da carga útil da mensagem e o atribui ao campo `reported_temperature`.
      - Cria um valor constante `30` para representar o valor limite e o atribui ao campo `max_temperature`.
5. Para abrir a lista de ações de regra para essa regra, em Definir uma ou mais ações, escolha Adicionar ação.
6. Em Selecionar uma ação, selecione Enviar uma mensagem como uma notificação por push do SNS.

7. Para abrir a página de configuração da ação selecionada, na parte inferior da lista de ações, escolha Configurar ação.
8. Em Configurar ação:

   a. Em Destino do SNS, escolha Selecionar, encontre seu tópico do SNS chamado high_temp_notice e escolha Selecionar.

   b. Em Formato de mensagem, selecione RAW.

   c. Em Escolher ou criar uma função para conceder AWS IoT acesso para realizar essa ação, escolha Criar função.

   d. Em Criar uma nova função, em Nome, insira um nome exclusivo para a nova função. Para este tutorial, use **`sns_rule_role`**.

   e. Selecione Criar perfil.

   Se estiver repetindo este tutorial ou reutilizando uma função existente, escolha Atualizar função antes de continuar. Isso atualiza o documento de política da função para funcionar com o destino do SNS.
9. Escolha Adicionar ação e retorne à página Criar uma regra.

   No quadro da nova ação, abaixo de Enviar uma mensagem como notificação push do SNS, você pode ver o tópico do SNS que sua regra vai chamar.

   Essa é a única ação de regra que você adicionará a essa regra.
10. Para criar a regra e concluir essa etapa, em Criar uma regra, desloque o cursor para baixo até a parte inferior e escolha Criar regra.

Etapa 3: testar a AWS IoT regra e a notificação do Amazon SNS

Para testar sua nova regra, você usará o cliente MQTT para publicar e assinar as mensagens MQTT usadas por essa regra.

Abra o cliente MQTT no AWS IoT console em uma nova janela. Isso permitirá que você edite a regra sem perder a configuração do seu cliente MQTT. Se você deixar o cliente MQTT para ir para outra página no console, ele não reterá nenhuma assinatura ou logs de mensagens.

Você pode usar o cliente MQTT para testar a regra

1. No cliente MQTT no AWS IoT console, assine os tópicos de entrada, neste caso, `device/+/data`.

   a. No Cliente MQTT, em Assinaturas, selecione Assine um tópico.

   b. Em Tópico de assinatura, insira o tópico do filtro de tópico de entrada, **`device/+/data`**.

   c. Deixe os demais campos com as configurações padrão.

   d. Escolha Assinar um tópico.

      Na coluna Assinaturas, em Publicar em um tópico, **`device/+/data`** é exibido.

2. Publique uma mensagem no tópico de entrada com um ID de dispositivo específico, **`device/32/data`**. Você não pode publicar nos tópicos MQTT que contenham caracteres curinga.

   a. No cliente MQTT, em Assinaturas, selecione Publicar em um tópico.

   b. No campo Publicar, insira o nome do tópico de entrada, **`device/32/data`**.

   c. Copie os dados de amostra mostrados aqui e, na caixa de edição abaixo do nome do tópico, cole os dados de amostra.

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

   d. Escolha Publicar em um tópico para publicar a mensagem MQTT.

3. Confirme se a mensagem de texto foi enviada.

   a. No cliente MQTT, em Assinaturas, há um ponto verde ao lado do tópico assinado anteriormente.

      O ponto verde indica que uma ou mais mensagens novas foram recebidas desde a última vez que você as visualizou.

b. Em Assinaturas, escolha dispositivo/+/dado para verificar se a carga útil da mensagem corresponde ao que você acabou de publicar e tem a seguinte aparência:

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

c. Verifique o telefone utilizado para assinar o tópico do SNS e confirme se o conteúdo da carga útil da mensagem tem a seguinte aparência:

```
{"device_id":"32","reported_temperature":38,"max_temperature":30}
```

Observe que o valor `device_id` valor é uma string citada e o valor `temperature` é numérico. Isso ocorre porque a `topic()`função extraiu a string do nome do tópico da mensagem de entrada, enquanto o valor `temperature` usa o valor numérico da carga útil da mensagem de entrada.

Se você quiser transformar o valor `device_id` em um valor numérico, substitua `topic(2)` na instrução de consulta de regra por:

```
cast(topic(2) AS DECIMAL)
```

Observe que converter o valor `topic(2)` em um valor numérico `DECIMAL` só funcionará se essa parte do tópico contiver somente caracteres numéricos.

4. Tente enviar uma mensagem MQTT na qual a temperatura não exceda o limite.

a. No cliente MQTT, em Assinaturas, selecione Publicar em um tópico.

b. No campo Publicar, insira o nome do tópico de entrada, **`device/33/data`**.

c. Copie os dados de amostra mostrados aqui e, na caixa de edição abaixo do nome do tópico, cole os dados de amostra.

```
{
```

```
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

d. Para enviar sua mensagem MQTT, escolha Publicar no tópico.

Você deve visualizar a mensagem que enviou na assinatura **`device/+/data`**. No entanto, como o valor da temperatura está abaixo da temperatura máxima na consulta da regra, você não deve receber uma mensagem de texto.

Se não encontrar o comportamento correto, confira as dicas de solução de problemas.

Solução de problemas da regra de mensagens do SNS

Aqui estão algumas objetos para verificar caso você não esteja vendo os resultados esperados.

- Você recebeu um banner de erro

  Se um erro apareceu quando você publicou a mensagem de entrada, corrija esse erro primeiro. As etapas a seguir podem ajudá-lo a corrigir esse erro.
- Você não vê a mensagem de entrada no cliente MQTT

  Toda vez que você publica sua mensagem de entrada no tópico `device/22/data`, essa mensagem deve aparecer no cliente MQTT, se tiver assinado o filtro de tópicos `device/+/data` conforme descrito no procedimento.

  Pontos importantes

  - Verifique o filtro de tópicos em que você fez a assinatura

    Se você fez a assinatura no tópico da mensagem de entrada conforme descrito no procedimento, deverá ver uma cópia da mensagem de entrada toda vez que publicá-la.

    Se você não visualizar a mensagem, verifique o nome do tópico em que você fez a assinatura e compare-o com o tópico no qual você publicou. Os nomes dos tópicos diferenciam maiúsculas

de minúsculas e o tópico no qual você fez a assinatura deve ser idêntico ao tópico no qual você publicou a carga útil da mensagem.

- Verifique a função de publicação de mensagens

  No cliente MQTT, em Assinaturas, escolha dispositivo/+/dados, verifique o tópico da mensagem de publicação e escolha Publicar no tópico. Você deve ver a carga útil da mensagem na caixa de edição abaixo do tópico aparecer na lista de mensagens.

- Se você não receber uma mensagem SMS

  Para que sua regra funcione, ela deve ter a política correta que a autorize a receber uma mensagem e enviar uma notificação do SNS, e ela deve receber a mensagem.

  Pontos importantes

  - Verifique o Região da AWS do seu cliente MQTT e a regra que você criou

    O console no qual você está executando o cliente MQTT deve estar na mesma região AWS da regra que você criou.

  - Verifique se o valor da temperatura na carga útil da mensagem excede o limite de teste

    Se o valor da temperatura for menor ou igual a 30, conforme definido na instrução de consulta da regra, a regra não executará nenhuma das ações.

  - Verifique o tópico da mensagem de entrada na instrução de consulta da regra

    Para que a regra funcione, ela deve receber uma mensagem com o nome do tópico que corresponda ao filtro do tópico na cláusula FROM da instrução de consulta da regra.

    Verifique a ortografia do filtro de tópico na declaração de consulta de regra com a do tópico no cliente MQTT. Os nomes dos tópicos diferenciam maiúsculas de minúsculas e o tópico da mensagem deve corresponder ao filtro de tópico na instrução de consulta da regra.

  - Verifique o conteúdo da carga útil da mensagem de entrada

    Para que a regra funcione, ela deve encontrar o campo de dados na carga da mensagem declarada na instrução SELECT.

    Verifique a ortografia do campo `temperature` na instrução de consulta da regra com a da carga útil da mensagem no cliente MQTT. Os nomes dos campos diferenciam maiúsculas de minúsculas e o campo `temperature` na instrução de consulta da regra deve ser idêntico ao campo `temperature` na carga útil da mensagem.

Verifique se o documento JSON na carga útil da mensagem está formatado corretamente. Se o JSON tiver algum erro, como uma vírgula ausente, a regra não poderá lê-lo.

- Verifique o tópico da mensagem republicada na ação da regra

  O tópico no qual a ação da regra de republicação publica a nova mensagem deve corresponder ao tópico no qual você fez a assinatura no cliente MQTT.

  Abra a regra que você criou no console e verifique o tópico no qual a ação da regra republicará a mensagem.

- Verifique a função que está sendo usada pela regra

  A ação da regra deve ter permissão para receber o tópico original e publicar o novo tópico.

  As políticas que autorizam a regra a receber dados de mensagens e republicá-los são específicas para os tópicos usados. Se você alterar o tópico usado para republicar os dados da mensagem, deverá atualizar a função da ação de regra para atualizar sua política conforme o tópico atual.

  Se você suspeitar que esse é o problema, edite a ação da regra de republicação e crie uma nova função. As novas funções criadas pela ação da regra recebem as autorizações necessárias para realizar essas ações.

Etapa 4: revisar os resultados e as próximas etapas

Neste tutorial:

- Você criou e testou um tópico e uma assinatura de notificação do Amazon SNS.
- Você usou uma consulta SQL simples e funções em uma consulta de regra para criar uma nova mensagem para sua notificação.
- Você criou uma AWS IoT regra para enviar uma notificação do Amazon SNS usando sua carga de mensagem personalizada.
- Você usou o cliente MQTT para testar sua AWS IoT regra.

Próximas etapas

Depois de enviar algumas mensagens de texto com essa regra, experimente usá-la para ver como a alteração de alguns aspectos do tutorial afeta a mensagem e quando ela é enviada. Aqui estão algumas ideias para você começar.

- Altere o *device_id* tópico da mensagem de entrada e observe o efeito no conteúdo da mensagem de texto.
- Altere os campos selecionados na consulta da regra e observe o efeito no conteúdo da mensagem de texto.
- Altere o teste na consulta da regra para testar uma temperatura mínima em vez de uma temperatura máxima. Lembre-se de mudar o nome de `max_temperature`!
- Adicione uma ação de regra de republicação para enviar uma mensagem MQTT quando uma notificação do SNS for enviada.
- Experimente o próximo tutorial desta série e aprenda como Tutorial: Armazenamento de dados do dispositivo em uma tabela do DynamoDB.

Tutorial: Armazenamento de dados do dispositivo em uma tabela do DynamoDB

Este tutorial demonstra como criar uma AWS IoT regra que envia dados de mensagens para uma tabela do DynamoDB.

Neste tutorial, você cria uma regra que envia dados das mensagens de um dispositivo de sensor climático imaginário para uma tabela do DynamoDB. A regra formata os dados de vários sensores meteorológicos para que possam ser adicionados a uma única tabela de banco de dados.

O que você aprenderá neste tutorial

- Como criar uma tabela do DynamoDB
- Como enviar dados de mensagens para uma tabela do DynamoDB a partir de uma regra AWS IoT
- Como usar modelos de substituição em uma regra AWS IoT
- Como usar consultas e funções SQL simples em uma instrução de consulta de regra
- Como usar o cliente MQTT para testar uma regra AWS IoT

Este tutorial leva cerca de 30 minutos para ser concluído.

Neste tutorial, você vai:

- Etapa 1: Criar a tabela no DynamoDB para este tutorial

- Etapa 2: criar uma AWS IoT regra para enviar dados para a tabela do DynamoDB
- Etapa 3: testar a AWS IoT regra e a tabela do DynamoDB
- Etapa 4: revisar os resultados e as próximas etapas

Antes de começar este tutorial, verifique se você tem o seguinte:

- Configurar Conta da AWS

  Você precisará do seu AWS IoT console Conta da AWS e do console para concluir este tutorial.
- Revisado Exibir mensagens MQTT com o cliente AWS IoT MQTT

  Certifique-se de que você pode usar o cliente MQTT para fazer a assinatura e publicar em um tópico. Você usará o cliente MQTT para testar a nova regra neste procedimento.
- Analisada a visão geral do Amazon DynamoDB

  Se você nunca usou o DynamoDB, consulte Getting Started with DynamoDB para se familiarizar com os conceitos e operações básicos do DynamoDB.

Etapa 1: Criar a tabela no DynamoDB para este tutorial

Neste tutorial, você criará uma tabela do DynamoDB com esses atributos para registrar os dados dos dispositivos imaginários de sensores climáticos:

- `sample_time` é uma chave primária e descreve a hora em que a amostra foi registrada.
- `device_id` é uma chave de classificação e descreve o dispositivo que forneceu a amostra
- `device_data` são os dados recebidos do dispositivo e formatados pela instrução de consulta de regra

Para criar uma tabela do DynamoDB para este tutorial

1. Abra o console DynamoDB, e selecione Criar tabela.
2. Em Criar tabela:
   a. Em Nome da tabela, insira o nome da tabela: **`wx_data`**.
   b. Em Chave de partição insira **`sample_time`**, e na lista de opções ao lado do campo, escolha**`Number`**.

c. Em Chave de classificação, insira **`device_id`**, e, na lista de opções ao lado do campo, escolha**`Number`**.

d. Na parte inferior da página, selecione Criar.

Você definirá `device_data` posteriormente, quando configurar a ação da regra do DynamoDB.

Etapa 2: criar uma AWS IoT regra para enviar dados para a tabela do DynamoDB

Nesta etapa, você usará a consulta de regras para formatar os dados dos dispositivos imaginários de sensores climáticos para gravar na tabela do banco de dados.

Um exemplo de carga útil de mensagem recebida de um dispositivo de sensor climático tem o seguinte aspecto:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

Para a entrada do banco de dados, você usará a instrução de consulta de regra para nivelar a estrutura da carga útil da mensagem para ficar dessa forma:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind_velocity": 22,
  "wind_bearing": 255
}
```

Nessa regra, você também usará alguns Modelos de substituição. Os modelos de substituição são expressões que permitem inserir valores dinâmicos de funções e dados de mensagens.

Para criar a AWS IoT regra para enviar dados para a tabela do DynamoDB

1. Abra o hub Regras do AWS IoT console. Ou você pode abrir a AWS IoT página inicial em Console de gerenciamento da AWS e navegar até Roteamento de mensagens> Regras.
2. Para começar a criar sua nova regra em Regras, escolha Criar regra.
3. Em Propriedades da regra:

   a. Em Nome do perfil, insira **`wx_data_ddb`**.

      Lembre-se de que o nome de uma regra deve ser exclusivo em sua região Conta da AWS e não pode ter espaços. Usamos um caractere sublinhado nesse nome para separar as duas palavras no nome da regra.

   b. Em Descrição da regra, descreva a regra.

      Uma descrição significativa facilita lembrar o que essa regra faz e por que você a criou. A descrição pode ser tão longa quanto necessário, portanto, seja o mais detalhista possível.
4. Escolha Próximo para continuar.
5. Em instrução SQL:

   a. Na versão SQL, selecione**`2016-03-23`**.

   b. Na caixa de edição da instrução SQL, insira a instrução:

      ```
      SELECT temperature, humidity, barometer,
        wind.velocity as wind_velocity,
        wind.bearing as wind_bearing,
      FROM 'device/+/data'
      ```

      Esta declaração:

      - Recebe mensagens MQTT com um tópico que corresponda ao filtro de tópicos `device/+/data`.
      - Formata os elementos do atributo `wind` como atributos individuais.
      - Transmite o `temperature`, `humidity` e atributos inalterados `barometer`.
6. Escolha Próximo para continuar.
7. Em Ações de regra:

   a. Para abrir a lista de ações de regra para essa regra, na Ação 1, escolha **`DynamoDB`**.

> **Note**
>
> Certifique-se de escolher o DynamoDB e não o DBv2 Dynamo como ação de regra.

b. Em Nome da tabela, escolha o nome da tabela do DynamoDB que você criou na etapa anterior: **`wx_data`**.

   Os campos Tipo de chave de partição e Tipo de chave de classificação são preenchidos com os valores da tabela do DynamoDB.

c. Em Chave de partição, insira **`sample_time`**.

d. Em Valor da chave de partição, insira **`${timestamp()}`**.

   Esse é o primeiro dos Modelos de substituição que você usará nesta regra. Em vez de usar um valor da carga da mensagem, ele usará o valor retornado da função de datação. Para saber mais, consulte Datação na AWS IoT Core Guia do desenvolvedor.

e. Em Chave de classificação, insira **`device_id`**.

f. Em Valor da chave de classificação, insira **`${cast(topic(2) AS DECIMAL)}`**.

   Esse é o segundo dos Modelos de substituição que você vai usar nesta regra. Ele insere o valor do segundo elemento no nome do tópico, que é o ID do dispositivo, depois de convertê-lo em um valor DECIMAL para corresponder ao formato numérico da chave. Para saber mais sobre tópicos, consulte Tópicos no Guia do desenvolvedor da AWS IoT Core . Ou, para saber mais sobre a transmissão, consulte Orientação no AWS IoT Core Guia do desenvolvedor.

g. Na coluna Gravar os dados da mensagem nesta coluna, insira **`device_data`**.

   Isso criará a coluna `device_data` na tabela do DynamoDB.

h. Deixe o campo Operação em branco.

i. Em Perfil do IAM, selecione Criar novo perfil.

j. Na caixa de diálogo Criar perfil, em Nome do perfil, insira wx_ddb_role. Essa nova função conterá automaticamente uma política com o prefixo "aws-iot-rule" que permitirá que a **`wx_data_ddb`** regra envie dados para a tabela do **`wx_data`** DynamoDB que você criou.

k. Em Perfil do IAM, escolha **`wx_ddb_role`**.

l. Na parte inferior da página, selecione a opção Próximo.

8. Na parte inferior da página Revisar e criar, escolha a opção Criar para criar a regra.

Etapa 3: testar a AWS IoT regra e a tabela do DynamoDB

Para testar a nova regra, você usará o cliente MQTT para publicar e assinar as mensagens MQTT usadas neste teste.

Abra o cliente MQTT no AWS IoT console em uma nova janela. Isso permitirá que você edite a regra sem perder a configuração do seu cliente MQTT. O cliente MQTT não retém nenhuma assinatura ou logs de mensagens se você deixar que ele vá para outra página no console. Você também desejará que uma janela de console separada seja aberta para o hub do DynamoDB Tables no console para visualizar AWS IoT as novas entradas que sua regra envia.

Você pode usar o cliente MQTT para testar a regra

1. No cliente MQTT no AWS IoT console, assine o tópico de entrada,`device/+/data`.
   a. No cliente MQTT, escolha Assinar um tópico.
   b. Em Filtro de tópicos, insira o tópico do filtro de tópico de entrada, **`device/+/data`**.
   c. Escolha Assinar.
2. Agora, publique uma mensagem no tópico de entrada com um ID de dispositivo específico, **`device/22/data`**. Você não pode publicar nos tópicos MQTT que contenham caracteres curinga.
   a. No cliente MQTT, escolha Publicar em um tópico.
   b. Em Nome do tópico, insira um nome para o tópico, **`device/22/data`**.
   c. Em Carga útil da mensagem, insira os seguintes dados de exemplo.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

   d. Para publicar a mensagem MQTT, escolha Publicar.

e. Agora, no cliente MQTT, escolha Assinar um tópico. Na coluna Assinar, escolha a assinatura **`device/+/data`**. Confirme se os dados de amostra da etapa anterior aparecem.

3. Verifique se consegue visualizar a linha na tabela do DynamoDB que sua regra criou.

   a. No hub do DynamoDB Tables no console, escolha wx_data e, em AWS IoT seguida, escolha a guia Itens.

      Se você já estiver na guia Itens, talvez precise atualizar a exibição selecionando o ícone de atualização no canto superior direito do cabeçalho da tabela.

   b. Observe que os valores de sample_time na tabela são links. Abra um. Se acabou de enviar a primeira mensagem, ela será a única na lista.

      Esse link exibe todos os dados nessa linha da tabela.

   c. Expanda a entrada device_data para ver os dados que resultaram da consulta da regra.

   d. Explore as diferentes representações dos dados disponíveis nessa exibição. Você também pode editar os dados nessa exibição.

   e. Depois de concluir a revisão dessa linha de dados, para salvar as alterações feitas, escolha a opção Salvar ou, para sair sem salvar nenhuma alteração, escolha Cancelar.

Se não encontrar o comportamento correto, confira as dicas de solução de problemas.

Solução de problemas da regra do DynamoDB

Aqui estão algumas objetos para verificar caso você não esteja vendo os resultados esperados.

- Você recebeu um banner de erro

  Se um erro apareceu quando você publicou a mensagem de entrada, corrija esse erro primeiro. As etapas a seguir podem ajudá-lo a corrigir esse erro.

- Você não vê a mensagem de entrada no cliente MQTT

  Toda vez que você publica sua mensagem de entrada no tópico `device/22/data`, essa mensagem deve aparecer no cliente MQTT, se tiver assinado o filtro de tópicos `device/+/data` conforme descrito no procedimento.

  Pontos importantes

  - Verifique o filtro de tópicos em que você fez a assinatura

Se você fez a assinatura no tópico da mensagem de entrada conforme descrito no procedimento, deverá ver uma cópia da mensagem de entrada toda vez que publicá-la.

Se você não visualizar a mensagem, verifique o nome do tópico em que você fez a assinatura e compare-o com o tópico no qual você publicou. Os nomes dos tópicos diferenciam maiúsculas de minúsculas e o tópico no qual você fez a assinatura deve ser idêntico ao tópico no qual você publicou a carga útil da mensagem.

- Verifique a função de publicação de mensagens

  No cliente MQTT, em Assinaturas, escolha dispositivo/+/dados, verifique o tópico da mensagem de publicação e escolha Publicar no tópico. Você deve ver a carga útil da mensagem na caixa de edição abaixo do tópico aparecer na lista de mensagens.

- Você não vê os dados na tabela do DynamoDB

  A primeiro objeto a fazer é atualizar a exibição selecionando o ícone de atualização no canto superior direito do cabeçalho da tabela. Se os dados que você está procurando não forem exibidos, verifique o seguinte.

  Pontos importantes

  - Verifique o Região da AWS do seu cliente MQTT e a regra que você criou

    O console no qual você está executando o cliente MQTT deve estar na mesma região AWS da regra que você criou.

  - Verifique o tópico da mensagem de entrada na instrução de consulta da regra

    Para que a regra funcione, ela deve receber uma mensagem com o nome do tópico que corresponda ao filtro do tópico na cláusula FROM da instrução de consulta da regra.

    Verifique a ortografia do filtro de tópico na declaração de consulta de regra com a do tópico no cliente MQTT. Os nomes dos tópicos diferenciam maiúsculas de minúsculas e o tópico da mensagem deve corresponder ao filtro de tópico na instrução de consulta da regra.

  - Verifique o conteúdo da carga útil da mensagem de entrada

    Para que a regra funcione, ela deve encontrar o campo de dados na carga da mensagem declarada na instrução SELECT.

    Verifique a ortografia do campo `temperature` na instrução de consulta da regra com a da carga útil da mensagem no cliente MQTT. Os nomes dos campos diferenciam maiúsculas de

minúsculas e o campo `temperature` na instrução de consulta da regra deve ser idêntico ao campo `temperature` na carga útil da mensagem.

Verifique se o documento JSON na carga útil da mensagem está formatado corretamente. Se o JSON tiver algum erro, como uma vírgula ausente, a regra não poderá lê-lo.

- Verifique os nomes da chave e do campo usados na ação da regra

  Os nomes de campo usados na regra de tópico devem corresponder aos encontrados na carga útil da mensagem JSON da mensagem publicada.

  Abra a regra criada no console e verifique os nomes dos campos na configuração da ação da regra com aqueles usados no cliente MQTT.

- Verifique a função que está sendo usada pela regra

  A ação da regra deve ter permissão para receber o tópico original e publicar o novo tópico.

  As políticas que autorizam a regra a receber dados de mensagens e atualizar a tabela do DynamoDB são específicas para os tópicos utilizados. Se você alterar o tópico ou o nome da tabela do DynamoDB usado pela regra, deverá atualizar a função da ação da regra para que a política esteja de acordo.

  Se você suspeitar que esse é o problema, edite a ação da regra e crie um novo perfil. As novas funções criadas pela ação da regra recebem as autorizações necessárias para realizar essas ações.

Etapa 4: revisar os resultados e as próximas etapas

Depois de enviar algumas mensagens para a tabela do DynamoDB com essa regra, experimente usá-la para ver como a alteração de alguns aspectos do tutorial afeta os dados gravados na tabela. Aqui estão algumas ideias para você começar.

- Altere o tópico da mensagem de entrada e observe o efeito nos dados. *`device_id`* Você pode usar isso para simular o recebimento de dados de vários sensores meteorológicos.
- Altere os campos selecionados na instrução de consulta da regra e observe o efeito nos dados. Você pode usar isso para filtrar os dados armazenados na tabela.
- Adicione uma ação de regra de republicação para enviar uma mensagem MQTT para cada linha adicionada à tabela. Você pode usar isso para depuração.

Depois de concluir este tutorial, confira the section called “Formatando uma notificação usando uma função AWS Lambda ”.

Tutorial: Formatando uma notificação usando uma função AWS Lambda

Este tutorial demonstra como enviar dados da mensagem MQTT para uma AWS Lambda ação para formatação e envio para outro serviço. AWS Neste tutorial, a AWS Lambda ação usa o AWS SDK para enviar a mensagem formatada para o tópico do Amazon SNS que você criou no tutorial sobre como fazer isso. the section called “Envio de uma notificação do Amazon SNS”

No tutorial sobre como fazer issothe section called “Envio de uma notificação do Amazon SNS”, o documento JSON resultante da instrução de consulta da regra foi enviado como o corpo da mensagem de texto. O resultado foi uma mensagem de texto parecida com o seguinte exemplo:

```
{"device_id":"32","reported_temperature":38,"max_temperature":30}
```

Neste tutorial, você usará uma ação de AWS Lambda regra para chamar uma AWS Lambda função que formata os dados da instrução de consulta de regra em um formato mais amigável, como este exemplo:

```
Device 32 reports a temperature of 38, which exceeds the limit of 30.
```

A AWS Lambda função que você criará neste tutorial formata a string da mensagem usando os dados da declaração de consulta de regra e chama a função de publicação do SNS do AWS SDK para criar a notificação.

O que você aprenderá neste tutorial

- Como criar e testar uma AWS Lambda função
- Como usar o AWS SDK em uma AWS Lambda função para publicar uma notificação do Amazon SNS
- Como usar consultas e funções SQL simples em uma instrução de consulta de regra
- Como usar o cliente MQTT para testar uma regra AWS IoT

Este tutorial leva cerca de 45 minutos para ser concluído.

Neste tutorial, você vai:

- Etapa 1: criar uma AWS Lambda função que envie uma mensagem de texto

- Etapa 2: criar uma AWS IoT regra com uma ação de AWS Lambda regra
- Etapa 3: testar a AWS IoT regra e a ação da AWS Lambda regra
- Etapa 4: revisar os resultados e as próximas etapas

Antes de começar este tutorial, verifique se você tem o seguinte:

- Configurar Conta da AWS

  Você precisará do seu AWS IoT console Conta da AWS e do console para concluir este tutorial.
- Revisado Exibir mensagens MQTT com o cliente AWS IoT MQTT

  Certifique-se de que você pode usar o cliente MQTT para fazer a assinatura e publicar em um tópico. Você usará o cliente MQTT para testar a nova regra neste procedimento.
- Concluídos os outros tutoriais de regras desta seção

  Este tutorial requer o tópico de notificação do SNS que você criou no tutorial sobre como fazer isso the section called “Envio de uma notificação do Amazon SNS”. Também pressupõe que você tenha concluído os outros tutoriais relacionados às regras nesta seção.
- Revisou a visão geral do AWS Lambda

  Se você nunca usou AWS Lambda antes, revise AWS Lambdae Comece a usar o Lambda para aprender seus termos e conceitos.

Etapa 1: criar uma AWS Lambda função que envie uma mensagem de texto

A AWS Lambda função neste tutorial recebe o resultado da declaração de consulta da regra, insere os elementos em uma string de texto e envia a string resultante para o Amazon SNS como mensagem em uma notificação.

Ao contrário do tutorial sobre comothe section called “Envio de uma notificação do Amazon SNS”, que usou uma ação de AWS IoT regra para enviar a notificação, este tutorial envia a notificação da função Lambda usando uma função do AWS SDK. No entanto, o tópico real de notificação do Amazon SNS usado neste tutorial é o mesmo que você usou no tutorial sobre como fazer isso the section called “Envio de uma notificação do Amazon SNS”.

Para criar uma AWS Lambda função que envia uma mensagem de texto

1. Crie uma nova AWS Lambda função.

a. No AWS Lambda console, selecione Criar função.

b. Em Criar função, selecione Usar um esquema.

   Pesquise e selecione o esquema **`hello-world-python`** e, em seguida, escolha Configurar.

c. Em Informações básicas:

   i. Em Nome do perfil, insira o nome da função, **`format-high-temp-notification`**.

   ii. Em Função de execução, escolha Criar uma nova função a partir de modelos de AWS política.

   iii. No campo Nome do perfil, digite o nome da nova função, **`format-high-temp-notification-role`**.

   iv. Em Modelos de política - opcional, pesquise e selecione a política de publicação do Amazon SNS.

   v. Escolha a opção Criar função.

2. Modifique o código do esquema para formatar e enviar uma notificação do Amazon SNS.

   a. Depois de criar sua função, você deverá ver a página de format-high-temp-notificationdetalhes. Caso contrário, abra-a na página Função do Lambda

   b. Na página de format-high-temp-notificationdetalhes, escolha a guia Configuração e vá até o painel Código da função.

   c. Na janela Código da função, no painel Ambiente, escolha o arquivo Python, `lambda_function.py`.

   d. Na janela Código da função, exclua todo o código original do programa do esquema e substitua-o por esse código.

```
import boto3
#
#   expects event parameter to contain:
#   {
#       "device_id": "32",
#       "reported_temperature": 38,
#       "max_temperature": 30,
#       "notify_topic_arn": "arn:aws:sns:us-
east-1:57EXAMPLE833:high_temp_notice"
#   }
#
```

```
#   sends a plain text string to be used in a text message
#
#      "Device {0} reports a temperature of {1}, which exceeds the limit of
 {2}."
#
#   where:
#       {0} is the device_id value
#       {1} is the reported_temperature value
#       {2} is the max_temperature value
#
def lambda_handler(event, context):

    # Create an SNS client to send notification
    sns = boto3.client('sns')

    # Format text message from data
    message_text = "Device {0} reports a temperature of {1}, which exceeds the
 limit of {2}.".format(
            str(event['device_id']),
            str(event['reported_temperature']),
            str(event['max_temperature'])
        )

    # Publish the formatted message
    response = sns.publish(
            TopicArn = event['notify_topic_arn'],
            Message = message_text
        )

    return response
```

e. Escolha Implantar.

3. Em uma nova janela, consulte o nome do recurso da Amazon (ARN) do tópico do Amazon SNS no tutorial sobre como fazer isso the section called “Envio de uma notificação do Amazon SNS”.

   a. Em uma nova janela, abra o console página Tópicos no console do Amazon SNS.

   b. Na página Tópicos, encontre o tópico de notificação high_temp_notice na lista de tópicos do Amazon SNS.

   c. Encontre o ARN do tópico de notificação high_temp_notice para usar na próxima etapa.

4. Crie um caso de teste para sua função do Lambda.

a. Na página Lambda Functions do console, na página de format-high-temp-notificationdetalhes, escolha Selecionar um evento de teste no canto superior direito da página (mesmo que pareça desativado) e escolha Configurar eventos de teste.

b. Em Selecionar um evento de teste, escolha Configurar evento de teste.

c. Em Nome do evento, insira **`SampleRuleOutput`**.

d. No editor JSON abaixo do Nome do evento, cole esse exemplo de documento JSON. Esse é um exemplo do que sua AWS IoT regra enviará para a função Lambda.

```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30,
  "notify_topic_arn": "arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice"
}
```

e. Consulte a janela que tem o ARN do tópico de notificação high_temp_notice e copie o valor do ARN.

f. Substitua o valor `notify_topic_arn` no editor JSON pelo ARN do seu tópico de notificação.

   Mantenha essa janela aberta para que você possa usar esse valor de ARN novamente ao criar a regra AWS IoT .

g. Escolha Criar.

5. Teste a função com dados de amostra.

   a. Na página de format-high-temp-notificationdetalhes, no canto superior direito da página, confirme que SampleRuleOutputaparece ao lado do botão Testar. Caso contrário, selecione-o na lista de eventos de teste disponíveis.

   b. Para enviar a mensagem de saída da regra de amostra para a função, escolha Testar.

Se a função e a notificação funcionarem, você receberá uma mensagem de texto no telefone de assinatura da notificação.

Se você não recebeu uma mensagem de texto no telefone, verifique o resultado da operação. No painel Código da função, na guia Resultado da execução, revise a resposta para encontrar quaisquer erros ocorridos. Não prossiga para a próxima etapa até que sua função possa enviar a notificação para seu telefone.

Etapa 2: criar uma AWS IoT regra com uma ação de AWS Lambda regra

Nesta etapa, você usará a instrução de consulta de regras para formatar os dados do dispositivo de sensor climático imaginário para enviar para uma função do Lambda, que formatará e enviará uma mensagem de texto.

Um exemplo de carga útil de mensagem recebida dos dispositivos meteorológicos tem a seguinte aparência:

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

Nessa regra, você usará a instrução de consulta de regras para criar uma carga útil de mensagem para a função do Lambda que se parece com esta:

```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30,
  "notify_topic_arn": "arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice"
}
```

Ela contém todas as informações que a função do Lambda precisa para formatar e enviar a mensagem de texto correta.

Para criar a AWS IoT regra para chamar uma função Lambda

1. Abra o hub de regras do AWS IoT console.
2. Para começar a criar sua nova regra em Regras, escolha a opção Criar.
3. Na parte superior da opção Criar uma regra:
    a. Em Nome, insira o nome da regra, **`wx_friendly_text`**.

Lembre-se de que o nome de uma regra deve ser exclusivo em sua região Conta da AWS e não pode ter espaços. Usamos um caractere sublinhado nesse nome para separar as duas palavras no nome da regra.

b. Em Descrição, descreva a regra.

Uma descrição significativa facilita lembrar o que essa regra faz e por que você a criou. A descrição pode ser tão longa quanto necessário, portanto, seja o mais detalhista possível.

4. Na declaração de consulta de regra de Criar uma regra:

a. Em Uso da versão SQL, selecione**2016-03-23**.

b. Na caixa de edição Instrução de consulta de regra, insira a instrução:

```
SELECT
  cast(topic(2) AS DECIMAL) as device_id,
  temperature as reported_temperature,
  30 as max_temperature,
  'arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice' as notify_topic_arn
FROM 'device/+/data' WHERE temperature > 30
```

Esta declaração:

- Recebe mensagens MQTT com um tópico que corresponda ao filtro de tópicos `device/+/data` e que tenha um valor `temperature` maior que 30.
- Seleciona o segundo elemento da cadeia de caracteres do tópico, converte-o em um número decimal e o atribui ao campo `device_id`.
- Seleciona o valor do campo `temperature` na carga da mensagem e o atribui ao campo `reported_temperature`.
- Cria um valor constante, `30`, para representar o valor limite e o atribui ao campo `max_temperature`.
- Cria um valor constante para o campo `notify_topic_arn`.

c. Consulte a janela que tem o ARN do tópico de notificação high_temp_notice e copie o valor do ARN.

d. Substitua o valor do ARN (*arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice*) no editor de instruções de consulta de regras pelo ARN do seu tópico de notificação.

5. Em Definir uma ou mais ações:

a. Para abrir a lista de ações de regra para essa regra, escolha Adicionar ação.

b. Em Selecionar uma ação, escolha Enviar uma mensagem para uma função do Lambda.

c. Para abrir a página de configuração da ação selecionada, na parte inferior da lista de ações, escolha Configurar ação.

6. Em Configurar ação:

a. Em Nome do perfil, escolha Selecionar.

b. Selecione format-high-temp-notification.

c. Na parte inferior da opção Configurar ação, escolha Adicionar ação.

d. Para criar a regra, na parte inferior da opção Criar uma regra, escolha Criar regra.

Etapa 3: testar a AWS IoT regra e a ação da AWS Lambda regra

Para testar sua nova regra, você usará o cliente MQTT para publicar e assinar as mensagens MQTT usadas por essa regra.

Abra o cliente MQTT no AWS IoT console em uma nova janela. Agora você pode editar a regra sem perder a configuração do cliente MQTT. Se você deixar o cliente MQTT para ir para outra página no console, perderá as assinaturas ou registros de mensagens.

Você pode usar o cliente MQTT para testar a regra

1. No cliente MQTT no AWS IoT console, assine os tópicos de entrada, neste caso, `device/+/data`.

a. No Cliente MQTT, em Assinaturas, selecione Assine um tópico.

b. Em Tópico de assinatura, insira o tópico do filtro de tópico de entrada, **`device/+/data`**.

c. Deixe os demais campos com as configurações padrão.

d. Escolha Assinar um tópico.

Na coluna Assinaturas, em Publicar em um tópico, **`device/+/data`** é exibido.

2. Publique uma mensagem no tópico de entrada com um ID de dispositivo específico, **`device/32/data`**. Você não pode publicar nos tópicos MQTT que contenham caracteres curinga.

a. No cliente MQTT, em Assinaturas, selecione Publicar em um tópico.

b. No campo Publicar, insira o nome do tópico de entrada, **`device/32/data`**.

c. Copie os dados de amostra mostrados aqui e, na caixa de edição abaixo do nome do tópico, cole os dados de amostra.

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

d. Escolha Publicar em um tópico para publicar sua mensagem MQTT.

3. Confirme se a mensagem de texto foi enviada.

a. No cliente MQTT, em Assinaturas, há um ponto verde ao lado do tópico assinado anteriormente.

O ponto verde indica que uma ou mais mensagens novas foram recebidas desde a última vez que você as visualizou.

b. Em Assinaturas, escolha dispositivo/+/dado para verificar se a carga útil da mensagem corresponde ao que você acabou de publicar e tem a seguinte aparência:

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

c. Verifique o telefone utilizado para assinar o tópico do SNS e confirme se o conteúdo da carga útil da mensagem tem a seguinte aparência:

```
Device 32 reports a temperature of 38, which exceeds the limit of 30.
```

Se você alterar o elemento ID do tópico no tópico da mensagem, lembre-se de que converter o valor `topic(2)` em um valor numérico só funcionará se esse elemento no tópico da mensagem contiver somente caracteres numéricos.

4. Tente enviar uma mensagem MQTT na qual a temperatura não exceda o limite.

   a. No cliente MQTT, em Assinaturas, selecione Publicar em um tópico.

   b. No campo Publicar, insira o nome do tópico de entrada, **`device/33/data`**.

   c. Copie os dados de amostra mostrados aqui e, na caixa de edição abaixo do nome do tópico, cole os dados de amostra.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

   d. Para enviar sua mensagem MQTT, escolha Publicar no tópico.

   Você deve ver a mensagem que enviou na assinatura **`device/+/data`**; no entanto, como o valor da temperatura está abaixo da temperatura máxima na consulta da regra, você não deve receber uma mensagem de texto.

   Se não encontrar o comportamento correto, confira as dicas de solução de problemas.

Solução de problemas com sua AWS Lambda regra e notificação

Aqui estão algumas objetos para verificar caso você não esteja vendo os resultados esperados.

- Você recebeu um banner de erro

  Se um erro apareceu quando você publicou a mensagem de entrada, corrija esse erro primeiro. As etapas a seguir podem ajudá-lo a corrigir esse erro.

- Você não vê a mensagem de entrada no cliente MQTT

Toda vez que você publica sua mensagem de entrada no tópico `device/32/data`, essa mensagem deve aparecer no cliente MQTT, se tiver assinado o filtro de tópicos `device/+/data` conforme descrito no procedimento.

Pontos importantes

- Verifique o filtro de tópicos em que você fez a assinatura

  Se você fez a assinatura no tópico da mensagem de entrada conforme descrito no procedimento, deverá ver uma cópia da mensagem de entrada toda vez que publicá-la.

  Se você não visualizar a mensagem, verifique o nome do tópico em que você fez a assinatura e compare-o com o tópico no qual você publicou. Os nomes dos tópicos diferenciam maiúsculas de minúsculas e o tópico no qual você fez a assinatura deve ser idêntico ao tópico no qual você publicou a carga útil da mensagem.

- Verifique a função de publicação de mensagens

  No cliente MQTT, em Assinaturas, escolha dispositivo/+/dados, verifique o tópico da mensagem de publicação e escolha Publicar no tópico. Você deve ver a carga útil da mensagem na caixa de edição abaixo do tópico aparecer na lista de mensagens.

- Se você não receber uma mensagem SMS

  Para que sua regra funcione, ela deve ter a política correta que a autorize a receber uma mensagem e enviar uma notificação do SNS, e ela deve receber a mensagem.

  Pontos importantes

  - Verifique o Região da AWS do seu cliente MQTT e a regra que você criou

    O console no qual você está executando o cliente MQTT deve estar na mesma região AWS da regra que você criou.

  - Verifique se o valor da temperatura na carga útil da mensagem excede o limite de teste

    Se o valor da temperatura for menor ou igual a 30, conforme definido na instrução de consulta da regra, a regra não executará nenhuma das ações.

  - Verifique o tópico da mensagem de entrada na instrução de consulta da regra

    Para que a regra funcione, ela deve receber uma mensagem com o nome do tópico que corresponda ao filtro do tópico na cláusula FROM da instrução de consulta da regra.

Verifique a ortografia do filtro de tópico na declaração de consulta de regra com a do tópico no cliente MQTT. Os nomes dos tópicos diferenciam maiúsculas de minúsculas e o tópico da mensagem deve corresponder ao filtro de tópico na instrução de consulta da regra.

- Verifique o conteúdo da carga útil da mensagem de entrada

  Para que a regra funcione, ela deve encontrar o campo de dados na carga da mensagem declarada na instrução SELECT.

  Verifique a ortografia do campo `temperature` na instrução de consulta da regra com a da carga útil da mensagem no cliente MQTT. Os nomes dos campos diferenciam maiúsculas de minúsculas e o campo `temperature` na instrução de consulta da regra deve ser idêntico ao campo `temperature` na carga útil da mensagem.

  Verifique se o documento JSON na carga útil da mensagem está formatado corretamente. Se o JSON tiver algum erro, como uma vírgula ausente, a regra não poderá lê-lo.

- Verifique a notificação do Amazon SNS

  EmEtapa 1: criar um tópico do Amazon SNS que envia uma mensagem de texto SMS, consulte a etapa 3 que descreve como testar a notificação do Amazon SNS e testar a notificação para garantir que a notificação funcione.

- Verifique a função do Lambda

  Em Etapa 1: criar uma AWS Lambda função que envie uma mensagem de texto, consulte a etapa 5 que descreve como testar a função do Lambda usando dados de teste e testar a função do Lambda.

- Verifique a função que está sendo usada pela regra

  A ação da regra deve ter permissão para receber o tópico original e publicar o novo tópico.

  As políticas que autorizam a regra a receber dados de mensagens e republicá-los são específicas para os tópicos usados. Se você alterar o tópico usado para republicar os dados da mensagem, deverá atualizar a função da ação de regra para atualizar sua política conforme o tópico atual.

  Se você suspeitar que esse é o problema, edite a ação da regra de republicação e crie uma nova função. As novas funções criadas pela ação da regra recebem as autorizações necessárias para realizar essas ações.

Etapa 4: revisar os resultados e as próximas etapas

Neste tutorial:

- Você criou uma AWS IoT regra para chamar uma função Lambda que enviou uma notificação do Amazon SNS usando sua carga de mensagem personalizada.
- Você usou uma consulta SQL simples e funções em uma instrução de consulta de regra para criar uma nova carga de mensagem para a função do Lambda.
- Você usou o cliente MQTT para testar sua AWS IoT regra.

Próximas etapas

Depois de enviar algumas mensagens de texto com essa regra, experimente usá-la para ver como a alteração de alguns aspectos do tutorial afeta a mensagem e quando ela é enviada. Aqui estão algumas ideias para você começar.

- Altere o *`device_id`* tópico da mensagem de entrada e observe o efeito no conteúdo da mensagem de texto.
- Altere os campos selecionados na instrução de consulta de regras, atualize a função do Lambda para usá-los em uma nova mensagem e observe o efeito no conteúdo da mensagem de texto.
- Altere o teste na consulta da regra para testar uma temperatura mínima em vez de uma temperatura máxima. Atualize a função do Lambda para formatar uma nova mensagem e lembre-se de alterar o nome de `max_temperature`.
- Para saber mais sobre como encontrar erros que possam ocorrer durante o desenvolvimento e o uso de AWS IoT regras, consulteMonitoramento AWS IoT.

Reter o estado do dispositivo enquanto o dispositivo está off-line com as sombras do dispositivo

Esses tutoriais mostram como usar o serviço AWS IoT Device Shadow para armazenar e atualizar as informações de estado de um dispositivo. O documento Shadow, que é um documento JSON, mostra a alteração no estado do dispositivo com base nas mensagens publicadas por um dispositivo, aplicativo local ou serviço. Neste tutorial, o documento Shadow mostra a mudança na cor de uma lâmpada. Esses tutoriais também mostram como a sombra armazena essas informações mesmo quando o dispositivo está desconectado da Internet e passa as informações de estado mais recentes para o dispositivo quando ele volta a ficar on-line e solicita essas informações.

Recomendamos que você veja esses tutoriais na ordem em que são mostrados aqui, começando com os recursos AWS IoT que você precisa criar e a configuração de hardware necessária, o que também ajuda você a aprender os conceitos de forma incremental. Esses tutoriais mostram como configurar e conectar um dispositivo Raspberry Pi para uso com. AWS IoT Se você não tiver o hardware necessário, poderá seguir esses tutoriais adaptando-os a um dispositivo de sua escolha ou criando um dispositivo virtual com o Amazon EC2.

Visão geral do cenário do tutorial

O cenário desses tutoriais é um aplicativo ou serviço local que altera a cor de uma lâmpada e publica seus dados em tópicos de sombra reservados. Esses tutoriais são semelhantes à funcionalidade Sombra do Dispositivo descrita no tutorial interativo de introdução e são implementados em um dispositivo Raspberry Pi. Os tutoriais nesta seção concentram-se em uma única sombra clássica e mostram como você pode acomodar sombras nomeadas ou vários dispositivos.

Os tutoriais a seguir ajudarão você a aprender como usar o serviço AWS IoT Device Shadow.

- Tutorial: preparando seu Raspberry Pi para executar o aplicativo de sombra

  Este tutorial mostra como configurar um dispositivo Raspberry Pi para conexão com o. AWS IoT Você também criará um documento de AWS IoT política e um recurso, baixará os certificados e anexará a política a esse recurso. Este tutorial leva cerca de 30 minutos para ser concluído.

- Tutorial: instalando o Device SDK e executando o aplicativo de amostra para Sombras do Dispositivo

  Este tutorial mostra como instalar as ferramentas e o software necessários e o AWS IoT Device SDK para Python e, em seguida, executar o aplicativo shadow de amostra. Este tutorial se baseia nos conceitos apresentados em Conectar um Raspberry Pi ou outro dispositivo e leva 20 minutos para ser concluído.

- Tutorial: interagindo com a Sombra do Dispositivo usando o aplicativo de amostra e o cliente de teste MQTT

  Este tutorial mostra como você usa o aplicativo e o AWS IoT console de `shadow.py` amostra para observar a interação entre AWS IoT Device Shadows e as mudanças de estado da lâmpada. O tutorial também mostra como enviar mensagens MQTT para os tópicos reservados da Sombra do Dispositivo. A conclusão deste tutorial pode levar cerca de 45 minutos.

AWS IoT Visão geral do Device Shadow

Um Device Shadow é uma representação virtual persistente de um dispositivo gerenciado por um recurso criado por você no AWS IoT registro. O documento Shadow é um documento JSON ou de JavaScript notação usado para armazenar e recuperar as informações do estado atual de um dispositivo. Você pode usar a sombra para obter e definir o estado de um dispositivo por meio de tópicos do MQTT ou HTTP REST APIs, independentemente de o dispositivo estar conectado à Internet.

Um documento de sombra contém uma `state` propriedade que descreve esses aspectos do estado do dispositivo.

- `desired`: Os aplicativos especificam os estados desejados das propriedades do dispositivo atualizando o objeto `desired`.
- `reported`: Os dispositivos relatam seu estado atual no objeto `reported`.
- `delta`: AWS IoT relata diferenças entre o estado desejado e o estado relatado no `delta` objeto.

Veja um exemplo de documento do estado de sombra.

```
{
  "state": {
    "desired": {
      "color": "green"
      },
    "reported": {
      "color": "blue"
      },
    "delta": {
      "color": "green"
      }
   }
}
```

Para atualizar o documento Shadow de um dispositivo, você pode usar os tópicos reservados do MQTT, o Device Shadow REST APIs que suporta o`GET`,`UPDATE`, e `DELETE` as operações com HTTP e a AWS IoT CLI.

No exemplo anterior, digamos que você queira alterar a cor `desired` para `yellow`. Para fazer isso, envie uma solicitação para a API UpdateThingShadow ou publique uma mensagem no tópico Atualizar, `$aws/things/THING_NAME/shadow/update`.

```
{
  "state": {
    "desired": {
      "color": yellow
    }
  }
}
```

As atualizações afetam apenas os campos especificados na solicitação. Depois de atualizar com sucesso o Device Shadow, AWS IoT publica o novo `desired` estado no `delta` tópico,`$aws/things/THING_NAME/shadow/delta`. O documento de sombra, nesse caso, tem a seguinte aparência:

```
{
  "state": {
    "desired": {
      "color": yellow
    },
    "reported": {
      "color": green
    },
    "delta": {
      "color": yellow
      }
  }
}
```

O novo estado é então reportado ao AWS IoT Device Shadow usando o `Update` tópico `$aws/things/THING_NAME/shadow/update` com a seguinte mensagem JSON:

```
{
  "state": {
    "reported": {
      "color": yellow
    }
  }
}
```

Se você quiser obter as informações do estado atual, envie uma solicitação para a API GetThingShadow ou publique uma mensagem MQTT no tópico Obter, `$aws/things/THING_NAME/shadow/get`.

Para obter mais informações sobre como usar o serviço de sombra do dispositivo, consulte AWS IoT Serviço Device Shadow.

Para obter mais informações sobre o uso de sombra do dispositivo em dispositivos, aplicativos e serviços, consulte Usar sombras em dispositivos e Usar sombras em aplicativos e serviços.

Para obter informações sobre como interagir com AWS IoT sombras, consulte. Interagir com sombras

Para obter informações sobre os tópicos reservados do MQTT e o HTTP REST APIs, consulte Tópicos MQTT da Sombra do Dispositivo e. API REST da Sombra do Dispositivo

Tutorial: preparando seu Raspberry Pi para executar o aplicativo de sombra

Este tutorial demonstra como instalar e configurar um dispositivo Raspberry Pi e criar os AWS IoT recursos necessários para um dispositivo se conectar e trocar mensagens MQTT.

> **Note**
>
> Se estiver planejando the section called “Criar um dispositivo virtual com o Amazon EC2”, ignore esta página e avance para the section called “Configurar o dispositivo”. Você criará estes recursos quando criar suo objeto virtual. Se você quiser usar um dispositivo diferente em vez do Raspberry Pi, tente seguir esses tutoriais adaptando-os a um dispositivo de sua escolha.

Neste tutorial, você aprenderá:

- Configure um dispositivo Raspberry Pi e configure-o para uso com AWS IoT.
- Crie um documento AWS IoT de política que autorize seu dispositivo a interagir com os AWS IoT serviços.
- Crie um recurso de coisa nos certificados AWS IoT do dispositivo X.509 e, em seguida, anexe o documento de política.

  A objeto é a representação virtual de seu dispositivo no registro do AWS IoT . O certificado autentica seu dispositivo no AWS IoT Core, e o documento de política autoriza seu dispositivo a interagir com. AWS IoT

Como executar este tutorial

Para executar o aplicativo de exemplo das Sombras do Dispositivo `shadow.py`, você precisará de um dispositivo Raspberry Pi que se conecte ao AWS IoT. Recomendamos que você siga este tutorial na ordem apresentada aqui, começando com a configuração do Raspberry Pi e seus acessórios e, em seguida, criando uma política e anexando a política a um recurso criado por você. Em seguida, você pode seguir este tutorial usando a interface gráfica do usuário (GUI) suportada pelo Raspberry Pi para abrir o AWS IoT console no navegador da web do dispositivo, o que também facilita o download dos certificados diretamente no Raspberry Pi para conexão. AWS IoT

Antes de começar este tutorial, verifique se você tem o seguinte:

- Um Conta da AWS. Se você não possuir uma, conclua as etapas descritas em Configurar Conta da AWS antes de continuar. Você precisará do seu AWS IoT console Conta da AWS e do console para concluir este tutorial.
- O Raspberry Pi e seus acessórios necessários. Você precisará de:
  - Um Modelo B do Raspberry Pi 3 ou modelo mais recente. Este tutorial pode funcionar em versões anteriores do Raspberry Pi, mas não o testamos.
  - Raspberry Pi OS (32 bits) ou uma versão posterior. Recomendamos usar a versão mais recente do SO do Raspberry Pi. Versões anteriores do SO podem funcionar, mas nós não as testamos.
  - Uma conexão Ethernet ou Wi-Fi.
  - Teclado, mouse, monitor, cabos e fontes de alimentação.

Este tutorial leva cerca de 30 minutos para ser concluído.

Etapa 1: instalar e configurar o dispositivo Raspberry Pi

Nesta seção, configuraremos um dispositivo Raspberry Pi para uso com AWS IoT.

> **Important**
>
> Adaptar estas instruções a outros dispositivos e sistemas operacionais pode ser um desafio. Você precisará conhecer seu dispositivo o bastante para interpretar estas instruções e aplicá-las ao seu dispositivo. Se encontrar dificuldades, você pode tentar uma das outras opções de dispositivo como alternativa, como Criar um dispositivo virtual com o Amazon EC2 ou Use seu PC Windows ou Linux ou Mac como um AWS IoT dispositivo.

Você precisará configurar seu Raspberry Pi para que ele possa iniciar o sistema operacional (SO), conectar-se à Internet e permitir que você interaja com ele em uma interface de linha de comando. Você também pode usar a interface gráfica do usuário (GUI) compatível com o Raspberry Pi para abrir o AWS IoT console e executar o restante deste tutorial.

Para configurar o Raspberry Pi

1. Insira o cartão SD no slot para cartão MicroSD do Raspberry Pi. Alguns cartões SD vêm pré-carregados com um gerenciador de instalação que exibe um menu para instalar o sistema operacional após inicializar a placa. Você também pode usar o Raspberry Pi Imager para instalar o sistema operacional em sua placa.
2. Conecte uma TV ou monitor HDMI ao cabo HDMI que se conecta à porta HDMI do Raspberry Pi.
3. Conecte o teclado e o mouse às portas USB do Raspberry Pi e, em seguida, conecte o adaptador de alimentação para inicializar a placa.

Após a inicialização do Raspberry Pi, se o cartão SD vier pré-carregado com o gerenciador de instalação, aparecerá um menu para instalar o sistema operacional. Se tiver problemas para instalar o sistema operacional, você pode tentar as etapas a seguir. Para obter mais informações sobre como configurar o Raspberry Pi, consulte Configurando o Raspberry Pi.

Se você estiver com problemas para configurar o Raspberry Pi:

- Verifique se você inseriu o cartão SD antes de inicializar a placa. Se você conectar o cartão SD após inicializar a placa, o menu de instalação pode não aparecer.
- Verifique se a TV ou o monitor estão ligados e se a entrada correta está selecionada.
- Verifique se você está usando um software compatível com Raspberry Pi.

Depois de instalar e configurar o sistema operacional Raspberry Pi, abra o navegador da Web do Raspberry Pi e navegue até o AWS IoT Core console para continuar as demais etapas deste tutorial.

Se você conseguir abrir o AWS IoT Core console, seu Raspberry Pi está pronto e você pode continuarthe section called “Provisionando seu dispositivo em AWS IoT ”.

Se você estiver com problemas ou precisar de ajuda adicional, consulte Como obter ajuda para seu Raspberry Pi.

Tutorial: Provisionando seu dispositivo em AWS IoT

Esta seção cria os AWS IoT Core recursos que seu tutorial usará.

Etapas para provisionar seu dispositivo em AWS IoT

- Etapa 1: Criar uma AWS IoT política para o Device Shadow
- Etapa 2: criar um recurso de objeto e vincular a política ao objeto
- Etapa 3: revisar os resultados e as próximas etapas

Etapa 1: Criar uma AWS IoT política para o Device Shadow

Certificados X.509 com os quais autenticam seu dispositivo. AWS IoT Core AWS IoT as políticas são anexadas ao certificado que permite ao dispositivo realizar AWS IoT operações, como assinar ou publicar tópicos reservados do MQTT usados pelo serviço Device Shadow. Seu dispositivo apresenta seu certificado quando se conecta e envia mensagens para AWS IoT Core o.

Siga as etapas para criar uma política que permita que o dispositivo execute as operações do AWS IoT que são necessárias para executar o programa de exemplo. Recomendamos que você crie uma política que conceda apenas as permissões necessárias para executar a tarefa. Você cria a AWS IoT política primeiro e depois a anexa ao certificado do dispositivo que você criará posteriormente.

Para criar uma AWS IoT política

1. No menu à esquerda, escolha Seguro e escolha Políticas. Se sua conta tiver políticas existentes, escolha Criar, caso contrário, na página Você ainda não tem uma política, escolha Criar uma política.
2. Na página Criar uma política:

   a. No campo Nome, insira um nome para a política (por exemplo, **`My_Device_Shadow_policy`**). Não use informações de identificação pessoal nos nomes de política.

   b. No documento de política, você descreve ações de conexão, assinatura, recebimento e publicação que dão permissão ao dispositivo para publicar e assinar os tópicos reservados do MQTT.

      Copie o exemplo de política a seguir e cole-o em seu documento de política. `thingname`Substitua pelo nome da coisa que você criará (por exemplo,`My_light_bulb`), `region` pela AWS IoT região em que você está usando os serviços e `account` pelo

seu Conta da AWS número. Para obter mais informações sobre AWS IoT políticas, consulteAWS IoT Core políticas.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/get",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/update"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/get/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/get/rejected",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/update/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/update/rejected",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/
things/thingname/shadow/update/delta"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
```

```
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingname/shadow/get/accepted",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingname/shadow/get/rejected",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingname/shadow/update/accepted",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingname/shadow/update/rejected",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingname/shadow/update/delta"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Connect",
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/test-*"
        }
    ]
}
```

Etapa 2: criar um recurso de objeto e vincular a política ao objeto

Os dispositivos conectados AWS IoT podem ser representados por recursos de coisas no AWS IoT registro. Um recurso de objeto representa um dispositivo específico ou uma entidade lógica, como a lâmpada deste tutorial.

Para aprender a criar algo em AWS IoT, siga as etapas descritas emCriar um objeto. Aqui estão algumas objetos importantes a serem observadas ao seguir as etapas desse tutorial:

1. Escolha Criar uma único objeto e, no campo Nome, insira um nome para o objeto que seja igual ao `thingname` (por exemplo, `My_light_bulb`) que você especificou quando criou a política anteriormente.

   Não é possível alterar um nome de objeto após sua criação. Se você deu a ele um nome diferente de `thingname`, crie um objeto nova com o nome como `thingname` e exclua o objeto antiga.

> **Note**
>
> Não use informações de identificação pessoal nos nomes de objetos. O nome do objeto pode surgir em comunicações e relatórios não criptografados.

2. Recomendamos que você baixe cada um dos arquivos de certificado na página Certificado criado! em um local onde você possa encontrá-los facilmente. Você precisará instalar esses arquivos para executar o aplicativo de amostra.

   Recomendamos que você baixe os arquivos em um subdiretório `certs` em seu diretório home no Raspberry Pi e nomeie cada um deles com um nome mais simples, conforme sugerido na tabela a seguir.

   Nomes de arquivos de certificado

Arquivo	Caminho do arquivo
Certificado CA raiz	`~/certs/Amazon-root-CA-1.pem`
Certificado de dispositivo	`~/certs/device.pem.crt`
Chave privada	`~/certs/private.pem.key`

3. Depois de ativar o certificado para habilitar conexões AWS IoT, escolha Anexar uma política e certifique-se de anexar a política que você criou anteriormente (por exemplo,**`My_Device_Shadow_policy`**) à coisa.

   Depois de criar uma coisa, você pode ver seu recurso exibido na lista de itens no AWS IoT console.

Etapa 3: revisar os resultados e as próximas etapas

Neste tutorial, você aprendeu a:

- Instalar e configurar o dispositivo Raspberry Pi.
- Crie um documento AWS IoT de política que autorize seu dispositivo a interagir com os AWS IoT serviços.
- Crie um recurso de objeto e um certificado de dispositivo X.509 associado e anexe o documento de política a ele.

Próximas etapas

Agora você pode instalar o SDK do AWS IoT dispositivo para Python, executar `shadow.py` o aplicativo de amostra e usar Device Shadows para controlar o estado. Para obter mais informações

sobre como executar este tutorial, consulte Tutorial: instalando o Device SDK e executando o aplicativo de amostra para Sombras do Dispositivo.

Tutorial: instalando o Device SDK e executando o aplicativo de amostra para Sombras do Dispositivo

Esta seção mostra como você pode instalar o software necessário e o AWS IoT Device SDK para Python e executar `shadow.py` o aplicativo de amostra para editar o documento Shadow e controlar o estado do shadow.

Neste tutorial, você aprenderá:

- Use o software instalado e o AWS IoT Device SDK for Python para executar o aplicativo de amostra.
- Saiba como inserir um valor usando o aplicativo de exemplo publica o valor desejado no console AWS IoT .
- Revise o aplicativo de amostra `shadow.py` e como ele usa o protocolo MQTT para atualizar o estado da sombra.

Antes de executar este tutorial:

Você deve ter configurado seu Conta da AWS dispositivo Raspberry Pi e criado uma AWS IoT política que dê ao dispositivo permissões para publicar e assinar os tópicos reservados do MQTT do serviço Device Shadow. Para obter mais informações, consulte Tutorial: preparando seu Raspberry Pi para executar o aplicativo de sombra.

Você também deve ter instalado o Git, o Python e o Device AWS IoT SDK for Python. Este tutorial se baseia nos conceitos apresentados no tutorial Conectar um Raspberry Pi ou outro dispositivo. Se você ainda não experimentou esse tutorial, recomendamos que siga as etapas descritas nesse tutorial para instalar os arquivos de certificado e o SDK do dispositivo e, em seguida, volte a este tutorial para executar o aplicativo de amostra `shadow.py`.

Neste tutorial, você vai:

- Etapa 1: execute o aplicativo de exemplo shadow.py
- Etapa 2: revise o aplicativo de amostra do SDK do dispositivo shadow.py
- Etapa 3: solucionar problemas com o aplicativo de amostra shadow.py
- Etapa 4: revisar os resultados e as próximas etapas

Este tutorial leva cerca de 20 minutos para ser concluído.

Etapa 1: execute o aplicativo de exemplo shadow.py

Antes de executar o aplicativo de amostra `shadow.py`, você precisará das informações a seguir, além dos nomes e do local dos arquivos de certificado instalados.

Valores de parâmetros de aplicação

Parâmetro	Onde encontrar o valor
your-iot-thing-name	Nome da AWS IoT coisa que você criou anteriormente emthe section called “Etapa 2: criar um recurso de objeto e vincular a política ao objeto”. Para encontrar esse valor, no AWS IoT console, escolha Gerenciar e, em seguida, escolha Objetos.
your-iot-endpoint	O *your-iot-endpoint* valor tem um formato de:*endpoint_id* `-ats.iot.` *region*`.amazonaws.com` , por exemplo,`a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com` . Para localizar este valor: 1. No AWS IoT console, escolha Gerenciar e, em seguida, escolha Objetos. 2. Selecione o objeto de IoT criada para seu dispositivo, My_light_bulb, que você usou anteriormente, e escolha Interagir. Na página de detalhes do objeto, seu endpoint é exibido na seção HTTPS.

Instalar e executar o aplicativo de exemplo

1. Navegue até o diretório do aplicativo de exemplo.

```
cd ~/aws-iot-device-sdk-python-v2/samples/service-clients
```

2. Na janela da linha de comando, substitua *your-iot-endpoint* e *your-iot-thing-name* conforme indicado e execute esse comando.

```
python3 shadow.py --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/certs/
device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint --
thing_name your-iot-thing-name
```

3. Observe que o aplicativo de exemplo:

   1. Conecta-se ao serviço de AWS IoT da sua conta.
   2. Assina eventos `Delta` e respostas `Update` e `Get`.
   3. Solicita que você insira o valor desejado no terminal.
   4. Exibe uma saída semelhante à seguinte:

```
Connecting to a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-0c8ae2ff-cc87-49d2-a82a-ae7ba1d0ca5a'...
Connected!
Subscribing to Delta events...
Subscribing to Update responses...
Subscribing to Get responses...
Requesting current shadow state...
Launching thread to read user input...
Finished getting initial shadow state.
Shadow contains reported value 'off'.
Enter desired value:
```

> **Note**
>
> Se você estiver com problemas para executar o aplicativo de amostra `shadow.py`, examine the section called “Etapa 3: solucionar problemas com o aplicativo de amostra `shadow.py`”. Para obter informações adicionais que possam ajudá-lo a corrigir o problema, adicione o parâmetro `--verbosity debug` à linha de comando para que o aplicativo de amostra exiba mensagens detalhadas sobre o que está fazendo.

Insira valores e observe as atualizações no documento de sombra

Você pode inserir valores no terminal para especificar o valor `desired`, o que também atualiza o valor `reported`. Digamos que você insira a cor `yellow` no terminal. O valor `reported` também é atualizado para a cor `yellow`. O seguinte mostra as mensagens exibidas no terminal:

```
Enter desired value:
yellow
Changed local shadow value to 'yellow'.
Updating reported shadow value to 'yellow'...
Update request published.
Finished updating reported shadow value to 'yellow'.
```

Quando você publica essa solicitação de atualização, AWS IoT cria uma sombra clássica padrão para o recurso thing. Você pode observar a solicitação de atualização que você publicou nos `desired` valores `reported` e no AWS IoT console examinando o documento Shadow do recurso que você criou (por exemplo,`My_light_bulb`). Para ver a atualização no documento de sombra:

1. No AWS IoT console, escolha Gerenciar e, em seguida, escolha Coisas.
2. Na lista de itens exibidos, selecione o que você criou, escolha Sombras e, em seguida, escolha Sombra clássica.

O documento de sombra deve ser semelhante ao seguinte, mostrando os valores `reported` e `desired` definidos para a cor `yellow`. Você vê esses valores na seção Estado da sombra do documento.

```
{
"desired": {
  "welcome": "aws-iot",
  "color": "yellow"
},
"reported": {
  "welcome": "aws-iot",
  "color": "yellow"
}
}
```

Você também vê uma seção de Metadados que contém as informações de data e hora e o número da versão da solicitação.

Você pode usar a versão do documento de estado para garantir que está atualizando a versão mais recente de um documento de sombra do dispositivo. Se você enviar outra solicitação de atualização, o número da versão será incrementado em 1. Ao fornecer uma versão com uma solicitação de atualização, o serviço rejeitará a solicitação com um código de resposta de conflito HTTP 409 se a versão atual do documento de estado não corresponder à versão fornecida.

```
{
"metadata": {
  "desired": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620156893
    }
  },
  "reported": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620156893
    }
  }
},
"version": 10
}
```

Para saber mais sobre o documento de sombra e observar as alterações nas informações de estado, vá para o próximo tutorial Tutorial: interagindo com a Sombra do Dispositivo usando o aplicativo de amostra e o cliente de teste MQTT conforme descrito na seção Etapa 4: revisar os resultados e as próximas etapas deste tutorial. Opcionalmente, você também pode aprender sobre o código de amostra `shadow.py` e como ele usa o protocolo MQTT na seção a seguir.

Etapa 2: revise o aplicativo de amostra do SDK do dispositivo shadow.py

Esta seção analisa o aplicativo de amostra `shadow.py` do AWS IoT Device SDK v2 para Python usado neste tutorial. Aqui, analisaremos como ele se conecta AWS IoT Core usando o protocolo MQTT e MQTT over WSS. A biblioteca AWS common runtime (AWS-CRT) fornece suporte ao protocolo de comunicação de baixo nível e está incluída no AWS IoT Device SDK v2 para Python.

Embora este tutorial use MQTT e MQTT sobre WSS, AWS IoT oferece suporte a dispositivos que publicam solicitações HTTPS. Para ver um exemplo de um programa em Python que envia uma mensagem HTTP de um dispositivo, consulte o exemplo de código HTTPS usando a biblioteca do Python `requests`.

Para obter informações sobre como você pode tomar uma decisão informada sobre qual protocolo usar para as comunicações do seu dispositivo, consulte Escolher um protocolo de aplicativo para a comunicação do dispositivo.

MQTT

Os exemplos de chamada `shadow.py` `mtls_from_path` (mostrados aqui) no `mqtt_connection_builder` para estabelecer conexão com o AWS IoT Core usando o protocolo MQTT. O `mtls_from_path` usa certificados X.509 e TLS v1.2 para autenticar o dispositivo. A biblioteca AWS-CRT manipula os detalhes de nível inferior dessa conexão.

```
mqtt_connection = mqtt_connection_builder.mtls_from_path(
  endpoint=args.endpoint,
  cert_filepath=args.cert,
  pri_key_filepath=args.key,
  ca_filepath=args.ca_file,
  client_bootstrap=client_bootstrap,
  on_connection_interrupted=on_connection_interrupted,
  on_connection_resumed=on_connection_resumed,
  client_id=args.client_id,
  clean_session=False,
  keep_alive_secs=6
)
```

- `endpoint`é o AWS IoT endpoint que você transmitiu pela linha de comando e `client_id` é o ID que identifica exclusivamente esse dispositivo no. Região da AWS
- `cert_filepath`, `pri_key_filepath`, e `ca_filepath` são os caminhos para os arquivos de certificado e chave privada do dispositivo e para o arquivo CA raiz.
- `client_bootstrap` é o objeto de runtime comum que manipula as atividades de comunicação por soquete e é instanciado antes da chamada para `mqtt_connection_builder.mtls_from_path`.
- `on_connection_interrupted` e `on_connection_resumed` são funções de retorno de chamada para ligar quando a conexão do dispositivo é interrompida e retomada.

- `clean_session` é iniciar uma sessão nova e persistente ou, se houver uma, reconectar-se a uma sessão existente. `keep_alive_secs` é o valor keep alive, em segundos, para enviar a solicitação `CONNECT`. Um ping será enviado automaticamente nesse intervalo. O servidor assume que a conexão será perdida se não receber um ping após 1,5 vezes esse valor.

A amostra `shadow.py` também chama `websockets_with_default_aws_signing` no `mqtt_connection_builder` para estabelecer uma conexão com AWS IoT Core usando o protocolo MQTT sobre WSS. MQTT sobre WSS também usa os mesmos parâmetros que MQTT e usa estes parâmetros adicionais:

- `region`é a região de AWS assinatura usada pela autenticação Signature V4 e `credentials_provider` são AWS as credenciais fornecidas para uso na autenticação. A região é passada pela linha de comando e o objeto `credentials_provider` é instanciado logo antes da chamada para `mqtt_connection_builder.websockets_with_default_aws_signing`.
- `websocket_proxy_options` são as opções de proxy HTTP, se estiver usando um host proxy. No aplicativo de amostra `shadow.py`, esse valor é instanciado logo antes da chamada para `mqtt_connection_builder.websockets_with_default_aws_signing`.

Inscreva-se em tópicos e eventos de sombra

A amostra `shadow.py` tenta estabelecer uma conexão e espera ser totalmente conectada. Se não estiver conectado, os comandos serão colocados na fila. Uma vez conectado, o exemplo assina eventos delta e atualiza e obtém mensagens, e publica mensagens com um nível de Qualidade de Serviço (QoS) de 1 (`mqtt.QoS.AT_LEAST_ONCE`).

Quando um dispositivo assina uma mensagem com QoS nível 1, o agente de mensagens salva as mensagens nas quais o dispositivo está inscrito até que elas possam ser enviadas ao dispositivo. O agente de mensagens reenvia as mensagens até receber uma resposta `PUBACK` do dispositivo.

Para obter mais informações sobre o protocolo MQTT, consulte Revisar o protocolo MQTT e MQTT.

Para obter mais informações sobre como MQTT, MQTT sobre WSS, sessões persistentes e níveis de QoS são usados neste tutorial, consulte. Revise o aplicativo de amostra do SDK do dispositivo pubsub.py

Etapa 3: solucionar problemas com o aplicativo de amostra `shadow.py`

Ao executar o aplicativo de amostra `shadow.py`, você deve ver algumas mensagens exibidas no terminal e uma solicitação para inserir um valor `desired`. Se o programa gerar um erro, para depurar o erro, você pode começar verificando se executou o comando correto para o seu sistema.

Em alguns casos, a mensagem de erro pode indicar problemas de conexão e ser semelhante a: `Host name was invalid for dns resolution` ou `Connection was closed unexpectedly`. Nesses casos, aqui estão algumas objetos que você pode verificar:

- Verifique o endereço do endpoint no comando

  Revise o argumento `endpoint` no comando inserido para executar o aplicativo de amostra (por exemplo, `a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com`) e verifique esse valor no AWS IoT console.

  Para verificar se você usou o valor correto:

  1. No AWS IoT console, escolha Gerenciar e, em seguida, escolha Objetos.
  2. Escolha o que você criou para seu aplicativo de amostra (por exemplo, My_light_bulb) e escolha Interagir.

  Na página de detalhes do objeto, seu endpoint é exibido na seção HTTPS. Você também deve ver uma mensagem que diz: `This thing already appears to be connected.`

- Verifique a ativação do certificado

  Os certificados autenticam seu dispositivo com AWS IoT Core.

  Para verificar se o seu certificado está ativo:

  1. No AWS IoT console, escolha Gerenciar e, em seguida, escolha Objetos.
  2. Escolha o que você criou para seu aplicativo de exemplo (por exemplo, My_light_bulb) e escolha Segurança.
  3. Selecione o certificado e, na página de detalhes do certificado, escolha Selecionar o certificado e, na página de detalhes do certificado, escolha Ações.

  Se na lista suspensa Ativar não estiver disponível e você só puder escolher Desativar, seu certificado estará ativo. Caso contrário, escolha Ativar e execute novamente o programa de amostra.

Se o programa ainda não for executado, verifique os nomes dos arquivos do certificado na pasta `certs`.

- Verifique a política anexada ao recurso do objeto

  Embora os certificados autentiquem seu dispositivo, AWS IoT as políticas permitem que o dispositivo realize AWS IoT operações, como assinar ou publicar tópicos reservados do MQTT.

  Para verificar se a política correta está anexada:

  1. Encontre o certificado conforme descrito anteriormente e escolha Políticas.
  2. Escolha a política exibida e verifique se ela descreve as ações `connect`, `subscribe`, `receive` e `publish` que dão permissão ao dispositivo para publicar e assinar os tópicos reservados do MQTT.

     Para obter um exemplo de política, consulte Etapa 1: Criar uma AWS IoT política para o Device Shadow.

  Se você receber mensagens de erro que indicam problemas na conexão AWS IoT, isso pode ser devido às permissões que você está usando para a política. Se for esse o caso, recomendamos que você comece com uma política que forneça acesso total aos AWS IoT recursos e, em seguida, execute novamente o programa de amostra. Você pode editar a política atual ou escolher a política atual, escolher Desanexar e, em seguida, criar outra política que forneça acesso total e a anexe ao seu recurso. Posteriormente, você poderá restringir a política apenas às ações e políticas necessárias para executar o programa.

  ```
  {
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:*"
            ],
            "Resource": "*"
        }
    ]
  }
  ```

- Verifique a instalação do SDK do seu dispositivo

Se o programa ainda não for executado, você poderá reinstalar o Device SDK para garantir que a instalação do SDK esteja completa e correta.

Etapa 4: revisar os resultados e as próximas etapas

Neste tutorial, você aprendeu a:

- Instale o software, as ferramentas e o AWS IoT Device SDK para Python necessários.
- Entenda como o aplicativo de amostra, a `shadow.py`, usa o protocolo MQTT para recuperar e atualizar o estado atual da sombra.
- Execute o aplicativo de amostra para Device Shadows e observe a atualização do documento Shadow no AWS IoT console. Você também aprendeu a solucionar quaisquer problemas e corrigir erros ao executar o programa.

Próximas etapas

Agora você pode executar o aplicativo de amostra `shadow.py` e usar as Sombras do Dispositivo para controlar o estado. Você pode observar as atualizações no documento Shadow no console AWS IoT e observar os eventos delta aos quais o aplicativo de amostra responde. Usando o cliente de teste MQTT, é possível assinar os tópicos de sombra reservados e observar as mensagens recebidas pelos tópicos ao executar o programa de amostra. Para obter mais informações sobre como executar este tutorial, consulte Tutorial: interagindo com a Sombra do Dispositivo usando o aplicativo de amostra e o cliente de teste MQTT.

Tutorial: interagindo com a Sombra do Dispositivo usando o aplicativo de amostra e o cliente de teste MQTT

Para interagir com o aplicativo de amostra `shadow.py`, insira um valor no terminal para o valor `desired`. Por exemplo, você pode especificar cores que se assemelhem aos semáforos e AWS IoT respondam à solicitação e atualizem os valores relatados.

Neste tutorial, você aprenderá:

- Usar o aplicativo de amostra `shadow.py` para especificar os estados desejados e atualizar o estado atual da sombra.
- Editar o documento Shadow para observar os eventos delta e como o aplicativo de amostra `shadow.py` responde a eles.

- Utilizar o cliente de teste MQTT para assinar tópicos de sombra e observar atualizações ao executar o programa de amostra.

Antes de executar este tutorial, você precisa:

Configure seu Conta da AWS, configure seu dispositivo Raspberry Pi e crie uma AWS IoT coisa e uma política. Você também deve ter instalado o software necessário, o Device SDK, os arquivos de certificado e executado o programa de amostra no terminal. Para obter mais informações, consulte os tutoriais anteriores Tutorial: preparando seu Raspberry Pi para executar o aplicativo de sombra e Etapa 1: execute o aplicativo de exemplo shadow.py. Você deve concluir esses tutoriais, caso ainda não o tenha feito.

Neste tutorial, você vai:

- Etapa 1: atualizar os valores desejados e relatados usando o aplicativo de amostra shadow.py
- Etapa 2: exibir mensagens do aplicativo de amostra shadow.py no cliente de teste MQTT
- Etapa 3: solucionar problemas com as interações da Sombra do Dispositivo
- Etapa 4: revisar os resultados e as próximas etapas

Este tutorial leva cerca de 45 minutos para ser concluído.

Etapa 1: atualizar os valores desejados e relatados usando o aplicativo de amostra **`shadow.py`**

No tutorial anteriorEtapa 1: execute o aplicativo de exemplo shadow.py, você aprendeu a observar uma mensagem publicada no documento Shadow no AWS IoT console ao inserir o valor desejado, conforme descrito na seçãoTutorial: instalando o Device SDK e executando o aplicativo de amostra para Sombras do Dispositivo.

No exemplo anterior, definimos a cor desejada como `yellow`. Depois de inserir cada valor, o terminal solicita que você insira outro valor `desired`. Se você inserir novamente o mesmo valor (`yellow`), o aplicativo reconhecerá isso e solicitará que você insira um novo valor `desired`.

```
Enter desired value:
yellow
Local value is already 'yellow'.
Enter desired value:
```

Agora, digamos que você insira a corgreen. AWS IoT responde à solicitação e atualiza o `reported` valor paragreen. É assim que a atualização acontece quando o estado `desired` é diferente do estado `reported`, causando um delta.

Como o aplicativo de amostra **`shadow.py`** simula as interações da Sombra do Dispositivo:

1. Insira um valor `desired` (digamos `yellow`) no terminal para publicar o estado desejado.
2. Como o estado `desired` é diferente do estado `reported` (digamos, a cor `green`), ocorre um delta e o aplicativo que está inscrito no delta recebe essa mensagem.
3. O aplicativo responde à mensagem e atualiza seu estado para o valor `desired`, `yellow`.
4. O aplicativo então publica uma mensagem de atualização com o novo valor relatado do estado do dispositivo, `yellow`.

A seguir, são mostradas as mensagens exibidas no terminal que mostram como a solicitação de atualização é publicada.

```
Enter desired value:
green
Changed local shadow value to 'green'.
Updating reported shadow value to 'green'...
Update request published.
Finished updating reported shadow value to 'green'.
```

No AWS IoT console, o documento Shadow reflete o valor atualizado `green` para os `desired` campos `reported` e, e o número da versão é incrementado em 1. Por exemplo, se o número da versão anterior foi exibido como 10, o número da versão atual será exibido como 11.

> **Note**
>
> A exclusão de uma sombra não redefine o número da versão para 0. Você verá que a versão sombra é incrementada em 1 quando você publica uma solicitação de atualização ou cria outra sombra com o mesmo nome.

Edite o documento Shadow para observar eventos delta

O aplicativo de amostra `shadow.py` também está inscrito em eventos `delta` e responde quando há uma alteração no valor `desired`. Por exemplo, você pode alterar o valor `desired` para a cor

`red`. Para fazer isso, no AWS IoT console, edite o documento Shadow clicando em Editar e defina o `desired` valor como `red` no JSON, mantendo o `reported` valor como`green`. Antes de salvar as alterações, mantenha o terminal do Raspberry Pi aberto, pois você verá as mensagens exibidas no terminal quando a alteração ocorrer.

```
{
"desired": {
  "welcome": "aws-iot",
  "color": "red"
},
"reported": {
  "welcome": "aws-iot",
  "color": "green"
}
}
```

Depois de salvar o novo valor, o aplicativo de amostra `shadow.py` responde a essa alteração e exibe mensagens no terminal indicando o delta. Em seguida, você deverá ver as seguintes mensagens aparecerem abaixo da solicitação para inserir o valor `desired`.

```
Enter desired value:
Received shadow delta event.
Delta reports that desired value is 'red'. Changing local value...
Changed local shadow value to 'red'.
Updating reported shadow value to 'red'...
Finished updating reported shadow value to 'red'.
Enter desired value:
Update request published.
Finished updating reported shadow value to 'red'.
```

Etapa 2: exibir mensagens do aplicativo de amostra **`shadow.py`** no cliente de teste MQTT

Você pode usar o cliente de teste do MQTT no AWS IoT console para monitorar as mensagens do MQTT que são passadas no seu Conta da AWS. Ao assinar tópicos MQTT reservados usados pelo serviço Sombra do Dispositivo, você pode observar as mensagens recebidas pelos tópicos ao executar o aplicativo de amostra.

Se você ainda não usou o cliente de teste MQTT, pode revisar Exibir mensagens MQTT com o cliente AWS IoT MQTT. Isso ajudará você a aprender a usar o cliente de teste MQTT no console do AWS IoT para visualizar as mensagens MQTT à medida que elas passam pelo agente de mensagens.

1. Abra o cliente de teste MQTT

   Abra o cliente de teste do MQTT no console do AWS IoT em uma nova janela para que você possa observar as mensagens recebidas pelos tópicos do MQTT sem perder a configuração do seu cliente de teste do MQTT. O cliente de teste MQTT não retém assinaturas ou logs de mensagens se você deixá-lo para ir para outra página no console. Nesta seção do tutorial, você pode abrir o documento Shadow do seu AWS IoT produto e o cliente de teste MQTT em janelas separadas para observar mais facilmente a interação com o Device Shadows.

2. Assine os tópicos Shadow reservados do MQTT

   Você pode usar o cliente de teste MQTT para inserir os nomes dos tópicos reservados do MQTT da Sombra do Dispositivo e se inscrever neles para receber atualizações ao executar o aplicativo de amostra `shadow.py`. Para se inscrever nos tópicos:

   a. No cliente de teste MQTT no AWS IoT console, escolha Subscribe to a topic (Se inscreva em um tópico).

   b. Na seção Filtro de tópicos, digite: $aws/things/ /shadow/update/ # ***thingname***. Aqui, `thingname` é o nome do recurso de objeto que você criou anteriormente (por exemplo, `My_light_bulb`).

   c. Mantenha os valores padrão para as configurações adicionais e escolha Inscrever-se.

   Ao usar o caractere curinga # na assinatura do tópico, você pode se inscrever em vários tópicos do MQTT ao mesmo tempo e observar todas as mensagens trocadas entre o dispositivo e sua sombra em uma única janela. Para obter mais informações sobre os caracteres curinga e seu uso, consulteTópicos do MQTT.

3. Execute um programa de amostra **`shadow.py`** e observe as mensagens

   Na janela de linha de comando do Raspberry Pi, se você desconectou o programa, execute o aplicativo de exemplo novamente e observe as mensagens no cliente de teste MQTT no console AWS IoT .

   a. Execute o comando a seguir para reiniciar o programa de amostra. *`your-iot-endpoint`*Substitua *`your-iot-thing-name`* e pelos nomes da AWS IoT coisa que você criou anteriormente (por exemplo,`My_light_bulb`) e pelo endpoint para interagir com o dispositivo.

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples/service-clients
   ```

```
python3 shadow.py --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/certs/
device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint --
thing_name your-iot-thing-name
```

Em seguida, o aplicativo de amostra `shadow.py` é executado e recupera o estado de sombra atual. Se você excluiu a sombra ou limpou os estados atuais, o programa define o valor atual como `off` e, em seguida, solicita que você insira um valor `desired`.

```
Connecting to a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-0c8ae2ff-cc87-49d2-a82a-ae7ba1d0ca5a'...
Connected!
Subscribing to Delta events...
Subscribing to Update responses...
Subscribing to Get responses...
Requesting current shadow state...
Launching thread to read user input...
Finished getting initial shadow state.
Shadow document lacks 'color' property. Setting defaults...
Changed local shadow value to 'off'.
Updating reported shadow value to 'off'...
Update request published.
Finished updating reported shadow value to 'off'...
Enter desired value:
```

Por outro lado, se o programa estava em execução e você o reiniciou, você verá o valor de cor mais recente relatado no terminal. No cliente de teste do MQTT, você verá uma atualização nos tópicos $aws/things/ /shadow/get e ***thingname*** $aws/things//. ***thingname*** shadow/get/accepted

Suponha que a última cor relatada tenha sido `green`. A seguir, mostra o conteúdo do arquivo ***thingname***$aws/things//JSON. shadow/get/accepted

```
{
"state": {
  "desired": {
    "welcome": "aws-iot",
    "color": "green"
  },
  "reported": {
    "welcome": "aws-iot",
    "color": "green"
```

```
  }
},
"metadata": {
  "desired": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620161643
    }
  },
  "reported": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620161643
    }
  }
},
"version": 10,
"timestamp": 1620173908
}
```

b. Insira um valor `desired` no terminal, como `yellow`. O aplicativo de amostra `shadow.py` responde e exibe as seguintes mensagens no terminal que mostram a alteração no valor `reported` para `yellow`.

```
Enter desired value:
yellow
Changed local shadow value to 'yellow'.
Updating reported shadow value to 'yellow'...
Update request published.
Finished updating reported shadow value to 'yellow'.
```

No cliente de teste do MQTT no console da AWS IoT , em Assinaturas, você vê que os seguintes tópicos receberam uma mensagem:

- $aws/things/ ***thingname*** /shadow/update: mostra que os valores e os valores mudam para a cor. `desired` `updated` `yellow`

- $aws/things/***thingname***/shadow/update/accepted: mostra os valores atuais dos `reported` estados `desired` e e seus metadados e informações de versão.
- $aws/things/***thingname***/shadow/update/documents: mostra os valores anteriores e atuais dos `reported` estados e e seus metadados `desired` e informações de versão.

Como o documento $aws/things/***thingname***/shadow/update/documentstambém contém informações contidas nos outros dois tópicos, podemos revisá-lo para ver as informações do estado. O estado anterior mostra o valor relatado definido como `green`, seus metadados e informações de versão e o estado atual que mostra o valor relatado atualizado para `yellow`.

```
{
"previous": {
  "state": {
    "desired": {
      "welcome": "aws-iot",
      "color": "green"
    },
    "reported": {
      "welcome": "aws-iot",
      "color": "green"
    }
  },
  "metadata": {
    "desired": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297898
      }
    },
    "reported": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297898
      }
    }
  },
```

```
  "version": 10
},
"current": {
  "state": {
    "desired": {
      "welcome": "aws-iot",
      "color": "yellow"
    },
    "reported": {
      "welcome": "aws-iot",
      "color": "yellow"
    }
  },
  "metadata": {
    "desired": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297904
      }
    },
    "reported": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297904
      }
    }
  },
  "version": 11
},
"timestamp": 1617297904
}
```

c. Agora, se você inserir outro valor `desired`, verá mais alterações nos valores `reported` e nas atualizações de mensagens recebidas por esses tópicos. O número da versão também aumenta em 1. Por exemplo, se você inserir o valor `green`, o estado anterior reportará o valor `yellow` e o estado atual relatará o valor `green`.

4. Edite o documento de sombra para observar eventos delta

   Para observar as alterações no tópico delta, edite o documento de sombra no console AWS IoT . Por exemplo, você pode alterar o valor `desired` para a cor `red`. Para fazer isso, no AWS IoT console, escolha Editar e defina o `desired` valor como vermelho no JSON, mantendo o `reported` valor definido como. `green` Antes de salvar a alteração, mantenha o terminal aberto, pois você verá a mensagem delta relatada no terminal.

```
{
"desired": {
  "welcome": "aws-iot",
  "color": "red"
},
"reported": {
  "welcome": "aws-iot",
  "color": "green"
}
}
```

   O aplicativo de amostra `shadow.py` responde a essa alteração e exibe mensagens no terminal indicando o delta. No cliente de teste MQTT, os tópicos `update` receberão uma mensagem mostrando as alterações nos valores `desired` e `reported`.

   Você também vê que o tópico $aws/things//***thingname***recebeu uma mensagem. shadow/update/delta Para ver a mensagem, escolha esse tópico, que está listado em Assinaturas.

```
{
"version": 13,
"timestamp": 1617318480,
"state": {
  "color": "red"
},
"metadata": {
  "color": {
    "timestamp": 1617318480
  }
}
}
```

Etapa 3: solucionar problemas com as interações da Sombra do Dispositivo

Ao executar o aplicativo de amostra de sombra, você pode encontrar problemas ao observar as interações com o serviço Sombra do Dispositivo.

Se o programa for executado com êxito e solicitar que você insira um valor `desired`, você poderá observar as interações da Sombra do Dispositivo usando o documento de sombra e o cliente de teste MQTT, conforme descrito anteriormente. No entanto, se você não conseguir ver as interações, aqui estão algumas objetos que você pode verificar:

- Verifique o nome da coisa e sua sombra no AWS IoT console

  Se você não vê as mensagens no documento de sombra, revise o comando e verifique se ele corresponde ao nome do objeto no AWS IoT console. Você também pode verificar se tem uma sombra clássica escolhendo seu recurso e, em seguida, escolhendo Sombras. Este tutorial se concentra, principalmente, nas interações com a sombra clássica.

  Você também pode confirmar se o dispositivo usado está conectado à Internet. No AWS IoT console, escolha o que você criou anteriormente e, em seguida, escolha Interagir. Na página de detalhes do objeto, você deve ver uma mensagem aqui que diz: `This thing already appears to be connected.`
- Verifique os tópicos reservados do MQTT nos quais você se inscreveu

  Se você não vir as mensagens aparecendo no cliente de teste MQTT, verifique se os tópicos que você assinou estão formatados corretamente. Os tópicos do MQTT Device Shadow têm um formato $aws/things/ ***`thingname`*** /shadow/ e podem ter`update`, ou `delete` segui-lo`get`, dependendo das ações que você deseja realizar na sombra. Este tutorial usa o tópico $aws/things/ ***`thingname`*** /shadow/ #. Portanto, certifique-se de inseri-lo corretamente ao assinar o tópico na seção Filtro de tópicos do cliente de teste.

  Ao inserir o nome do tópico, certifique-se de que *`thingname`* seja igual ao nome da AWS IoT coisa que você criou anteriormente. Você também pode se inscrever em tópicos adicionais do MQTT para ver se uma atualização foi realizada com sucesso. Por exemplo, você pode se inscrever no tópico $aws/things/***`thingname`***/para receber uma mensagem sempre que uma solicitação de atualização falhar, shadow/update/rejected para que você possa depurar problemas de conexão. Para obter mais informações sobre os tópicos reservados, consulte the section called “Tópicos de sombra” e Tópicos MQTT da Sombra do Dispositivo.

Etapa 4: revisar os resultados e as próximas etapas

Neste tutorial, você aprendeu a:

- Usar o aplicativo de amostra `shadow.py` para especificar os estados desejados e atualizar o estado atual da sombra.
- Editar o documento Shadow para observar os eventos delta e como o aplicativo de amostra `shadow.py` responde a eles.
- Utilizar o cliente de teste MQTT para assinar tópicos de sombra e observar atualizações ao executar o programa de amostra.

Próximas etapas

Você pode assinar tópicos adicionais reservados do MQTT para observar atualizações no aplicativo da sombra. Por exemplo, se você se inscrever apenas no tópico $aws/things/***thingname***/shadow/update/accepted, verá somente as informações do estado atual quando uma atualização for realizada com sucesso.

Você também pode assinar tópicos de sombra adicionais para depurar problemas ou aprender mais sobre as interações da Sombra do Dispositivo e também depurar quaisquer problemas com as interações da Sombra do Dispositivo. Para obter mais informações, consulte the section called "Tópicos de sombra" e Tópicos MQTT da Sombra do Dispositivo.

Você também pode optar por estender seu aplicativo usando sombras nomeadas ou usando hardware adicional conectado ao Raspberry Pi LEDs e observar alterações em seu estado usando mensagens enviadas do terminal.

Para obter mais informações sobre o serviço Sombra do Dispositivo e como usar o serviço em dispositivos, aplicativos e serviços, consulte AWS IoT Serviço Device Shadow, Usar sombras em dispositivos e Usar sombras em aplicativos e serviços.

Tutorial: Criando um autorizador personalizado para AWS IoT Core

Este tutorial demonstra as etapas para criar, validar e usar a autenticação personalizada usando a AWS CLI. Opcionalmente, usando este tutorial, você pode usar o Postman para enviar dados ao AWS IoT Core usando a API de publicação HTTP.

Este tutorial mostra como criar um exemplo de função do Lambda que implementa a lógica de autorização e autenticação e um autorizador personalizado usando a chamada create-authorizer

com a assinatura de token ativada. O autorizador é então validado usando o. Etest-invoke-authorizer, finalmente, você pode enviar dados AWS IoT Core usando a API HTTP Publish para um tópico de teste do MQTT. A solicitação de amostra especificará o autorizador a ser invocado usando o `x-amz-customauthorizer-name` cabeçalho e transmitirá os cabeçalhos token-key-name e `x-amz-customauthorizer-signature` na solicitação.

O que você aprenderá neste tutorial:

- Como criar uma função do Lambda para ser um manipulador de autorizador personalizado
- Como criar um autorizador personalizado usando o AWS CLI com a assinatura de token ativada
- Como testar seu autorizador personalizado usando o comando test-invoke-authorizer
- Como publicar um tópico do MQTT usando o Postman e validar a solicitação com seu autorizador personalizado

Este tutorial leva cerca de 60 minutos para ser concluído.

Neste tutorial, você vai:

- Etapa 1: criar uma função do Lambda para seu autorizador personalizado
- Etapa 2: criar um par de chaves pública e privada do autorizador privado
- Etapa 3: Criar um recurso de autorizador personalizado e sua autorização
- Etapa 4: testar o autorizador ligando test-invoke-authorizer
- Etapa 5: testar a publicação da mensagem MQTT usando o Postman
- Etapa 6: visualizar mensagens no cliente de teste MQTT
- Etapa 7: revisar os resultados e as próximas etapas
- Etapa 8: Limpeza

Antes de começar este tutorial, verifique se você tem o seguinte:

- Configurar Conta da AWS

  Você precisará do seu AWS IoT console Conta da AWS e do console para concluir este tutorial.

  A conta que você usa para este tutorial funciona melhor quando inclui pelo menos as seguintes políticas gerenciadas da AWS :

  - `IAMFullAccess`

- `AWSIoTFullAccess`
- `AWSLambda_FullAccess`

> **Important**
>
> As políticas do IAM usadas neste tutorial são mais permissivas do que você deveria seguir em uma implementação de produção. Em um ambiente de produção, certifique-se de que as políticas de sua conta e recursos concedam somente as permissões necessárias.
> Ao criar políticas do IAM para produção, determine qual acesso os usuários e perfis precisam e, em seguida, crie políticas que permitam que eles executem apenas essas tarefas.
> Para obter mais informações, consulte Práticas recomendadas de segurança no IAM.

- Instalou o AWS CLI

  Para obter informações sobre como instalar o AWS CLI, consulte Instalando a AWS CLI. Este tutorial requer uma AWS CLI versão `aws-cli/2.1.3 Python/3.7.4 Darwin/18.7.0 exe/x86_64` ou posterior.

- Ferramentas do OpenSSL

  Os exemplos neste tutorial usam o LibreSSL 2.6.5. Você também pode usar as ferramentas OpenSSL v1.1.1i para este tutorial.

- Revisou a visão geral do AWS Lambda

  Se você nunca usou AWS Lambda antes, revise AWS Lambdae Comece a usar o Lambda para aprender seus termos e conceitos.

- Consultar como criar solicitações no Postman

  Para obter mais informações, consulte Solicitações de criação.

- Remover autorizadores personalizados do tutorial anterior

  Você só Conta da AWS pode ter um número limitado de autorizadores personalizados configurados ao mesmo tempo. Para ver informações sobre como remover um autorizador personalizado, consulte the section called “Etapa 8: Limpeza”.

Etapa 1: criar uma função do Lambda para seu autorizador personalizado

A autenticação personalizada AWS IoT Core usa recursos do autorizador que você cria para autenticar e autorizar clientes. A função que você criará nesta seção autentica e autoriza os clientes à medida que eles se conectam AWS IoT Core e AWS IoT acessam os recursos.

A função do Lambda faz o seguinte:

- Se uma solicitação vier do test-invoke-authorizer, ela retornará uma política do IAM com uma ação `Deny`.
- Se uma solicitação vier do Postman usando HTTP e o parâmetro `actionToken` tiver um valor de `allow`, ela retornará uma política do IAM com uma ação `Allow`. Caso contrário, ele retornará uma política do IAM com uma ação `Deny`.

Para criar uma função do Lambda para seu autorizador personalizado

1. No console do Lambda, abra Funções.
2. Escolha a opção Criar função.
3. Confirme se Criar do zero está selecionado.
4. Em Basic information:
   a. Em Nome do perfil, insira **`custom-auth-function`**.
   b. Em Runtime, confirme Node.js 18.x
5. Escolha a opção Criar função.

   O Lambda cria uma função Node.js e uma função de execução que concede à função permissão para fazer upload de logs. A função Lambda assume a função de execução quando você invoca sua função e usa a função de execução para criar credenciais para o AWS SDK e ler dados de fontes de eventos.
6. Para ver o código e a configuração da função no AWS Cloud9editor, escolha custom-auth-functionna janela do designer e escolha index.js no painel de navegação do editor.

   Para linguagens de script, como o Node.js, o Lambda inclui uma função básica que retorna uma resposta de sucesso. Você pode usar o editor do AWS Cloud9 para editar sua função, desde que seu código-fonte não exceda 3 MB.
7. Substitua o código index.js no editor pelo código a seguir:

```
// A simple Lambda function for an authorizer. It demonstrates
// How to parse a CLI and Http password to generate a response.

export const handler = async (event, context, callback) => {

    //Http parameter to initiate allow/deny request
    const HTTP_PARAM_NAME='actionToken';
    const ALLOW_ACTION = 'Allow';
    const DENY_ACTION = 'Deny';

    //Event data passed to Lambda function
    var event_str = JSON.stringify(event);
    console.log('Complete event :'+ event_str);

    //Read protocolData from the event json passed to Lambda function
    var protocolData = event.protocolData;
    console.log('protocolData value---> ' + protocolData);

    //Get the dynamic account ID from function's ARN to be used
    // as full resource for IAM policy
    var ACCOUNT_ID = context.invokedFunctionArn.split(":")[4];
    console.log("ACCOUNT_ID---"+ACCOUNT_ID);

    //Get the dynamic region from function's ARN to be used
    // as full resource for IAM policy
    var REGION = context.invokedFunctionArn.split(":")[3];
    console.log("REGION---"+REGION);

    //protocolData data will be undefined if testing is done via CLI.
    // This will help to test the set up.
    if (protocolData === undefined) {

        //If CLI testing, pass deny action as this is for testing purpose only.
        console.log('Using the test-invoke-authorizer cli for testing only');
        callback(null, generateAuthResponse(DENY_ACTION,ACCOUNT_ID,REGION));

    } else{

        //Http Testing from Postman
        //Get the query string from the request
        var queryString = event.protocolData.http.queryString;
        console.log('queryString values -- ' + queryString);
        /*         global URLSearchParams       */
```

```
        const params = new URLSearchParams(queryString);
        var action = params.get(HTTP_PARAM_NAME);

        if(action!=null && action.toLowerCase() === 'allow'){

            callback(null, generateAuthResponse(ALLOW_ACTION,ACCOUNT_ID,REGION));

        }else{

            callback(null, generateAuthResponse(DENY_ACTION,ACCOUNT_ID,REGION));

        }

    }

};

// Helper function to generate the authorization IAM response.
var generateAuthResponse = function(effect,ACCOUNT_ID,REGION) {

    var full_resource = "arn:aws:iot:"+ REGION + ":" + ACCOUNT_ID + ":*";
    console.log("full_resource---"+full_resource);

    var authResponse = {};
    authResponse.isAuthenticated = true;
    authResponse.principalId = 'principalId';

    var policyDocument = {};
    policyDocument.Version = '2012-10-17';
    policyDocument.Statement = [];
    var statement = {};
    statement.Action = 'iot:*';
    statement.Effect = effect;
    statement.Resource = full_resource;
    policyDocument.Statement[0] = statement;
    authResponse.policyDocuments = [policyDocument];
    authResponse.disconnectAfterInSeconds = 3600;
    authResponse.refreshAfterInSeconds = 600;

    console.log('custom auth policy function called from http');
    console.log('authResponse --> ' + JSON.stringify(authResponse));
    console.log(authResponse.policyDocuments[0]);

    return authResponse;
```

```
}
```

8. Escolha Implantar.
9. Depois que Alterações implantadas aparecer acima do editor:
   a. Role até a seção Visão geral da função acima do editor.
   b. Copie o ARN da função e salve-o para usar posteriormente neste tutorial.
10. Teste a função do .
    a. Selecione a guia Testar.
    b. Usando as configurações de teste padrão, escolha Invocar.
    c. Se o teste for concluído com êxito, nos Resultados da execução, abra a visualização Detalhes. Você deve ver o documento de política que a função retornou.

       Se o teste falhou ou você não vê um documento de política, analise o código para encontrar e corrigir os erros.

Etapa 2: criar um par de chaves pública e privada do autorizador privado

Seu autorizador personalizado requer uma chave pública e privada para autenticá-lo. Os comandos nesta seção usam ferramentas OpenSSL para criar esse par de chaves.

Parar criar o par de chaves pública e privada do autorizador privado

1. Crie o arquivo da chave privada.

```
openssl genrsa -out private-key.pem 4096
```

2. Verifique o arquivo da chave privada criado.

```
openssl rsa -check -in private-key.pem -noout
```

   Se o comando não exibir nenhum erro, o arquivo de chave privada é válido.

3. Crie o arquivo da chave pública.

```
openssl rsa -in private-key.pem -pubout -out public-key.pem
```

4. Verifique o arquivo da chave pública.

```
openssl pkey -inform PEM -pubin -in public-key.pem -noout
```

Se o comando não exibir nenhum erro, o arquivo de chave pública é válido.

Etapa 3: Criar um recurso de autorizador personalizado e sua autorização

O autorizador AWS IoT personalizado é o recurso que une todos os elementos criados nas etapas anteriores. Nesta seção, você criará um recurso de autorizador personalizado e concederá a ele permissão para executar a função do Lambda criada anteriormente. Você pode criar um recurso de autorização personalizado usando o AWS IoT console AWS CLI, o ou a AWS API.

Para este tutorial, você só precisa criar um autorizador personalizado. Esta seção descreve como criar usando o AWS IoT console e o AWS CLI, para que você possa usar o método mais conveniente para você. Não há diferença entre os recursos de autorizador personalizado criados por nenhum dos métodos.

Criar um recurso de autorizador personalizado

Escolha uma destas opções para criar seu recurso de autorizador personalizado

- Crie um autorizador personalizado usando o console AWS IoT
- Criar um autorizador personalizado usando a AWS CLI

Para criar um autorizador personalizado (console)

1. Abra a página do autorizador personalizado do AWS IoT console e escolha Criar autorizador.
2. Em Criar autorizador:
    a. Em Nome do autorizador, insira **`my-new-authorizer`**.
    b. Em Status do autorizador, marque Ativo.
    c. Em Função do autorizador, escolha a função do Lambda criada anteriormente.
    d. Em Validação de token - opcional:
        i. Ative a Validação do token.
        ii. Em Nome da chave do token, insira **`tokenKeyName`**.
        iii. Escolha Adicionar chave.

iv. Em Nome da chave, insira **`FirstKey`**.

v. Em Chave pública, insira o conteúdo do arquivo `public-key.pem`. Certifique-se de incluir as linhas do arquivo com `-----BEGIN PUBLIC KEY-----` e `-----END PUBLIC KEY-----` e não adicionar ou remover nenhum avanço de linha, retornos de carro ou outros caracteres do conteúdo do arquivo. A string inserida deve ser semelhante a este exemplo.

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAvEBzOk4vhN+3LgslvEWt
sLCqNmt5Damas3bmiTRvq2gjRJ6KXGTGQChqArAJwL1a9dkS9+maaXC3vc6xzx9z
QPu/vQOe5tyzz1MsKdmtFGxMqQ3qjEXAMPLEOmqyUKPP5mff58k6ePSfXAnzBH0q
lg2HioefrpU5OSAnpuRAjYKofKjbc2Vrn6N2G7hV+IfTBvCElf0csalS/Rk4phD5
oa4Y0GHISRnevypg5C8n9Rrz91PWGqP6M/q5DNJJXjMyleG92hQgu1N696bn5Dw8
FhedszFa6b2x6xrItZFzewNQkPMLMFhNrQIIyvshtT/F1LVCS5+v8AQ8UGGDfZmv
QeqAMAF7WgagDMXcfgKSVU8yid2sIm56qsCLMvD2Sq8Lgzpey9N5ON1o1Cvldwvc
KrJJtgwW6hVqRGuShnownLpgG86M6neZ5sRMbVNZO8OzcobLngJ0Ibw9KkcUdklW
gvZ6HEJqBY2XE70iEXAMPLETPHzhqvK6Ei1HGxpHsXx6BNft582J1VpgYjXha8oa
/NN7l7Zbj/euAb41IVtmX8JrD9z613d1iM5L8HluJlUzn62Q+VeNV2tdA7MfPfMC
8btGYladFAnitThaz6+F0VSBJPu7pZQoLnqyEp5zLMtF+kFl2yOBmGAP0RBivRd9
JWBUCG0bqcLQPeQyjbXSOfUCAwEAAQ==
-----END PUBLIC KEY-----
```

3. Selecione Criar autorizador.
4. Se o recurso do autorizador personalizado tiver sido criado, você verá a lista de autorizadores personalizados, onde deve aparecer o seu novo autorizador personalizado, e você poderá continuar na próxima seção para testá-lo.

   Se você vir um erro, analise-o, tente criar seu autorizador personalizado novamente e verifique as entradas. Observe que cada recurso de autorizador personalizado deve ter um nome exclusivo.

Para criar um autorizador personalizado (AWS CLI)

1. Substitua seus valores por `authorizer-function-arn` e `token-signing-public-keys` e, em seguida, execute o seguinte comando:

```
aws iot create-authorizer \
--authorizer-name "my-new-authorizer" \
--token-key-name "tokenKeyName" \
```

```
--status ACTIVE \
--no-signing-disabled \
--authorizer-function-arn "arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-function" \
--token-signing-public-keys FirstKey="-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAvEBzOk4vhN+3LgslvEWt
sLCqNmt5Damas3bmiTRvq2gjRJ6KXGTGQChqArAJwL1a9dkS9+maaXC3vc6xzx9z
QPu/vQOe5tyzz1MsKdmtFGxMqQ3qjEXAMPLEOmqyUKPP5mff58k6ePSfXAnzBH0q
lg2HioefrpU5OSAnpuRAjYKofKjbc2Vrn6N2G7hV+IfTBvCElf0csalS/Rk4phD5
oa4Y0GHISRnevypg5C8n9Rrz91PWGqP6M/q5DNJJXjMyleG92hQgu1N696bn5Dw8
FhedszFa6b2x6xrItZFzewNQkPMLMFhNrQIIyvshtT/F1LVCS5+v8AQ8UGGDfZmv
QeqAMAF7WgagDMXcfgKSVU8yid2sIm56qsCLMvD2Sq8Lgzpey9N5ON1o1Cvldwvc
KrJJtgwW6hVqRGuShnownLpgG86M6neZ5sRMbVNZO8OzcobLngJ0Ibw9KkcUdklW
gvZ6HEJqBY2XE70iEXAMPLETPHzhqvK6Ei1HGxpHsXx6BNft582J1VpgYjXha8oa
/NN7l7Zbj/euAb41IVtmX8JrD9z613d1iM5L8HluJlUzn62Q+VeNV2tdA7MfPfMC
8btGYladFAnitThaz6+F0VSBJPu7pZQoLnqyEp5zLMtF+kFl2yOBmGAP0RBivRd9
JWBUCG0bqcLQPeQyjbXSOfUCAwEAAQ==
-----END PUBLIC KEY-----"
```

Em que:

- O valor `authorizer-function-arn` é o nome do recurso da Amazon (ARN) da função do Lambda que você criou para o seu autorizador personalizado.
- O valor `token-signing-public-keys` inclui o nome da chave, **`FirstKey`** e o conteúdo do arquivo `public-key.pem`. Certifique-se de incluir as linhas do arquivo com `-----BEGIN PUBLIC KEY-----` e `-----END PUBLIC KEY-----` e não adicionar ou remover nenhum avanço de linha, retornos de carro ou outros caracteres do conteúdo do arquivo.

  Nota: tome cuidado ao inserir a chave pública, pois qualquer alteração em seu valor a tornará inutilizável.

2. Se o autorizador personalizado for criado, o comando retornará o nome e o ARN do novo recurso, como o seguinte.

```
{
    "authorizerName": "my-new-authorizer",
    "authorizerArn": "arn:aws:iot:Region:57EXAMPLE833:authorizer/my-new-authorizer"
}
```

Salve o valor `authorizerArn` para uso na próxima etapa.

Lembre-se de que cada recurso de autorizador personalizado deve ter um nome exclusivo.

Autorizar o recurso de autorizador personalizado

Nesta seção, você concederá permissão ao recurso de autorizador personalizado que acabou de criar para executar a função do Lambda. Para conceder a permissão, você pode usar o comando da CLI add-permission.

Conceda permissão para sua função Lambda usando o AWS CLI

1. Depois de inserir os valores, digite o seguinte comando. Observe que o valor `statement-id` deve ser exclusivo. Substitua *Id-1234* por outro valor se você já tiver executado esse tutorial antes ou se receber um erro `ResourceConflictException`.

```
aws lambda add-permission  \
--function-name "custom-auth-function" \
--principal "iot.amazonaws.com" \
--action "lambda:InvokeFunction" \
--statement-id "Id-1234" \
--source-arn authorizerArn
```

2. Se o comando for executado com êxito, ele retornará uma declaração de permissão, como neste exemplo. Você pode continuar na próxima seção para testar o autorizador personalizado.

```
{
    "Statement": "{\"Sid\":\"Id-1234\",\"Effect\":\"Allow\",\"Principal
\":{\"Service\":\"iot.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction
\",\"Resource\":\"arn:aws:lambda:Region:57EXAMPLE833:function:custom-
auth-function\",\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":
\"arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-function\"}}}"
}
```

Se o comando não for executado com êxito, ele retornará um erro, como neste exemplo. Será necessário verificar e corrigir o erro antes de continuar.

```
An error occurred (AccessDeniedException) when calling the AddPermission operation:
 User: arn:aws:iam::57EXAMPLE833:user/EXAMPLE-1 is not authorized to perform:
 lambda:AddPer
```

```
mission on resource: arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-
function
```

Etapa 4: testar o autorizador ligando test-invoke-authorizer

Com todos os recursos definidos, nesta seção, você ligará test-invoke-authorizer da linha de comando para testar o passe de autorização.

Observe que, ao invocar o autorizador a partir da linha de comando, `protocolData` não está definido; portanto, o autorizador sempre retornará um documento DENY. No entanto, esse teste confirma que seu autorizador personalizado e a função do Lambda estão configurados corretamente, mesmo que não teste totalmente a função do Lambda.

Para testar seu autorizador personalizado e sua função Lambda usando o AWS CLI

1. No diretório que contém o arquivo `private-key.pem` criado na etapa anterior, execute o seguinte comando.

   ```
   echo -n "tokenKeyValue" | openssl dgst -sha256 -sign private-key.pem | openssl
    base64 -A
   ```

   Esse comando cria uma string de assinatura para ser usada na próxima etapa. A string de assinatura é semelhante a esta:

   ```
   dBwykzlb+fo+JmSGdwoGr8dyC2qB/IyLefJJr+rbCvmu9Jl4KHAA9DG+V
   +MMWu09YSA86+64Y3Gt4tOykpZqn9mn
   VB1wyxp+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvsluv7i2IMjEg
   +CPY0zrWt1jr9BikgGPDxWkjaeeh
   bQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm90OyZ94R4bdJsHNig1JePgnuOBvMGCEFE09jGjj
   szEHfgAUAQIWXiVGQj16BU1xKpTGSiTAwheLKUjITOEXAMPLECK3aHKYKY
   +d1vTvdthKtYHBq8MjhzJ0kggbt29V
   QJCb8RilN/P5+vcVniSXWPplyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtcspACi6ca
   +tsDuX
   f3LzCwQQF/YSUy02u5XkWn
   +sto6KCkpNlkD0wU8gl3+kOzxrthnQ8gEajd5Iylx230iqcXo3osjPha7JDyWM5o+K
   EWckTe91I1mokDr5sJ4JXixvnJTVSx1li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h
   +flFeloZlAWQFH
   xRlXsPqiVKS1ZIUClaZWprh/orDJplpiWfBgBIOgokJIDGP9gwhXIIk7zWrGmWpMK9o=
   ```

Copie essa string de assinatura para usá-la na próxima etapa. Tome cuidado para não incluir caracteres extras nem omitir nenhum.

2. Nesse comando, substitua o valor `token-signature` pela string de assinatura da etapa anterior e execute esse comando para testar seu autorizador.

```
aws iot test-invoke-authorizer \
--authorizer-name my-new-authorizer \
--token tokenKeyValue \
--token-signature dBwykzlb+fo+JmSGdwoGr8dyC2qB/IyLefJJr
+rbCvmu9Jl4KHAA9DG+V+MMWu09YSA86+64Y3Gt4tOykpZqn9mnVB1wyxp
+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvsluv7i2IMjEg
+CPY0zrWt1jr9BikgGPDxWkjaeehbQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm90OyZ94R4bdJsh
+d1vTvdthKtYHBq8MjhzJ0kggbt29VQJCb8RilN/
P5+vcVniSXWPplyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtcspACi6ca
+tsDuXf3LzCwQQF/YSUy02u5XkWn
+sto6KCkpNlkD0wU8gl3+kOzxrthnQ8gEajd5Iylx230iqcXo3osjPha7JDyWM5o
+KEWckTe91I1mokDr5sJ4JXixvnJTVSx1li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h
+flFeloZlAWQFHxRlXsPqiVKS1ZIUClaZWprh/orDJplpiWfBgBIOgokJIDGP9gwhXIIk7zWrGmWpMK9o=
```

Se o comando for executado com êxito, ele retornará as informações geradas por sua função de autorizador personalizado, como neste exemplo.

```
{
    "isAuthenticated": true,
    "principalId": "principalId",
    "policyDocuments": [
        "{\"Version\":\"2012-10-17\",       \"Statement\":[{\"Action\":\"iot:*\",
\"Effect\":\"Deny\",\"Resource\":\"arn:aws:iot:Region:57EXAMPLE833:*\"}]}"
    ],
    "refreshAfterInSeconds": 600,
    "disconnectAfterInSeconds": 3600
}
```

Se o comando retornar um erro, analise-o e verifique novamente os comandos usados nesta seção.

Etapa 5: testar a publicação da mensagem MQTT usando o Postman

1. Para obter o endpoint de dados do dispositivo a partir da linha de comando, chame describe-endpoint conforme mostrado aqui

   ```
   aws iot describe-endpoint --output text --endpoint-type iot:Data-ATS
   ```

   Salve esse endereço para uso como *device_data_endpoint_address* em uma etapa posterior.

2. Abra uma nova janela do Postman e crie uma nova solicitação HTTP POST.

   a. No seu computador, abra o aplicativo Postman.

   b. No Postman, no menu Arquivo, escolha Novo….

   c. Na caixa de diálogo Novo, selecione Solicitação.

   d. Em Salvar solicitação,

      i. Em Nome da solicitação, insira **Custom authorizer test request**.

      ii. Em Selecionar uma coleção ou pasta para salvar: escolha ou crie uma coleção na qual salvar essa solicitação.

      iii. Escolha Salvar em ***collection_name***.

3. Crie a solicitação do POST para testar seu autorizador personalizado.

   a. No seletor do método de solicitação ao lado do campo URL, escolha POST.

   b. No campo URL, crie o URL para sua solicitação usando o seguinte URL com o comando *device_data_endpoint_address* from the describe-endpoint em uma etapa anterior.

      ```
      https://device_data_endpoint_address:443/topics/test/cust-auth/topic?
      qos=0&actionToken=allow
      ```

      Observe que esse URL inclui o parâmetro de consulta `actionToken=allow` que fará com que a função do Lambda retorne um documento de política que permita acesso à AWS IoT. Depois de inserir o URL, os parâmetros de consulta também serão exibidos na guia Parâmetros do Postman.

   c. Na guia Autenticação, no campo Tipo, escolha Sem autenticação.

   d. Na guia Cabeçalhos:

i. Se houver uma chave de host marcada, desmarque-a.

ii. Na parte inferior da lista de cabeçalhos, adicione estes novos cabeçalhos e confirme se estão marcados. Substitua o **`Host`** valor pelo seu *`device_data_endpoint_address`* e o **`x-amz-customauthorizer-signature`** valor pela string de assinatura que você usou com o test-invoke-authorize comando na seção anterior.

Chave	Valor
`x-amz-customauthorizer-name`	**`my-new-authorizer`**
`Host`	*`device_data_endpoint_address`*
`tokenKeyName`	**`tokenKeyValue`**

Chave	Valor
x-amz-customauthorizer-signature	*dBwykzlb+fo+JmSGdwoGr8dyC2qB/IyLefJJr+rbCvmu9Jl4KHAA9DG+V+MMWu09YSA86+64Y3Gt4tOykpZqn9mnVB1wyxp+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvsluv7i2IMjEg+CPY0zrWt1jr9BikgGPDxWkjaeehbQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm90OyZ94R4bdJsHNig1JePgnuOBvMGCEFE09jGjjszEHfgAUAQIWXiVGQj16BU1xKpTGSiTAwheLKUjITOEXAMPLECK3aHKYKY+d1vTvdthKtYHBq8MjhzJ0kggbt29VQJCb8RilN/P5+vcVniSXWPplyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtcspACi6ca+tsDuXf3LzCwQQF/YSUy02u5XkWn+sto6KCkpNlkD0wU8gl3+kOzxrthnQ8gEajd5Iylx230iqcXo3osjPha7JDyWM5o+KEWckTe91I1mokDr5sJ4JXixvnJTVSx1li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h+flFeloZlAWQFHxRlXsPqiVKS1ZIUClaZWprh/orDJplpiWfBgBIOgokJIDGP9gwhXIIk7zWrGmWpMK9o=*

e. Na guia Corpo:

i. Na caixa de opção de formato de dados, escolha Bruto.

ii. Na lista de tipos de dados, escolha JavaScript.

iii. No campo de texto, insira esta carga de mensagem JSON para sua mensagem de teste:

```
{
```

```
    "data_mode": "test",
    "vibration": 200,
    "temperature": 40
}
```

4. Escolha Enviar para enviar a solicitação.

   Se a solicitação for concluída com êxito, ela retornará:

   ```
   {
       "message": "OK",
       "traceId": "ff35c33f-409a-ea90-b06f-fbEXAMPLE25c"
   }
   ```

   A resposta bem-sucedida indica que seu autorizador personalizado permitiu a conexão AWS IoT e que a mensagem de teste foi entregue ao broker in AWS IoT Core.

   Se retornar um erro, revise a mensagem de erro*device_data_endpoint_address*, a string de assinatura e os outros valores do cabeçalho.

Mantenha essa solicitação no Postman para uso na próxima seção.

Etapa 6: visualizar mensagens no cliente de teste MQTT

Na etapa anterior, você enviou mensagens simuladas do dispositivo AWS IoT usando o Postman. A resposta bem-sucedida indicou que seu autorizador personalizado permitiu a conexão à AWS IoT e que a mensagem de teste foi entregue ao agente no AWS IoT Core. Nesta seção, você usará o cliente de teste MQTT no AWS IoT console para ver o conteúdo da mensagem da mesma forma que outros dispositivos e serviços.

Para ver as mensagens de teste autorizadas pelo seu autorizador personalizado

1. No AWS IoT console, abra o cliente de teste MQTT.
2. Na guia Assinar um tópico, em Filtro de tópicos, insira **`test/cust-auth/topic`**, que é o tópico da mensagem usado no exemplo do Postman da seção anterior.
3. Escolha Assinar.

   Mantenha essa janela visível para a próxima etapa.
4. No Postman, na solicitação criada na seção anterior, escolha Enviar.

Analise a resposta para verificar se ela foi concluída com êxito. Caso contrário, solucione o erro conforme descrito na seção anterior.

5. No cliente de teste do MQTT, deve ser possível ver uma nova entrada que mostra o tópico da mensagem e, se expandida, a carga da mensagem da solicitação enviada pelo Postman.

   Se você não vir suas mensagens no cliente de teste do MQTT, aqui estão alguns itens que você deve verificar:

   - Certifique-se de que sua solicitação do Postman tenha sido retornada com êxito. Se AWS IoT rejeitar a conexão e retornar um erro, a mensagem na solicitação não será passada para o agente de mensagens.
   - Verifique se o Conta da AWS e Região da AWS usado para abrir o AWS IoT console é o mesmo que você está usando no URL do Postman.
   - Garanta que você esteja usando o endpoint adequado para o autorizador personalizado. O endpoint de IoT padrão pode não aceitar o uso de autorizadores personalizados com funções do Lambda. Em vez disso, você pode usar configurações de domínio para definir um novo endpoint e depois especificá-lo para o autorizador personalizado.
   - Verifique se você inseriu o tópico corretamente no cliente de teste do MQTT. O filtro do tópico diferencia letras maiúsculas de minúsculas. Em caso de dúvida, você também pode se inscrever no **#** tópico, que assina todas as mensagens MQTT que passam pelo agente de mensagens Conta da AWS e Região da AWS usadas para abrir o AWS IoT console.

Etapa 7: revisar os resultados e as próximas etapas

Neste tutorial:

- Você criou uma função do Lambda para ser um manipulador de autorizador personalizado
- Você criou um autorizador personalizado com a assinatura de token ativada
- Você testou seu autorizador personalizado usando o comando test-invoke-authorizer
- Você publicou um tópico do MQTT usando o Postman e validou a solicitação com seu autorizador personalizado
- Você usou o cliente de teste MQTT para visualizar as mensagens enviadas do seu teste do Postman

Próximas etapas

Depois de enviar algumas mensagens do Postman para verificar se o autorizador personalizado está funcionando, experimente analisar como a alteração de diferentes aspectos deste tutorial afeta os resultados. Aqui estão alguns exemplos para você começar.

- Altere a string da assinatura para que não seja mais válido ver como as tentativas de conexão não autorizadas são processadas. Você deve receber uma resposta de erro, como essa, e a mensagem não deve aparecer no cliente de teste do MQTT.

```
{
    "message": "Forbidden",
    "traceId": "15969756-a4a4-917c-b47a-5433e25b1356"
}
```

- Para saber mais sobre como encontrar erros que possam ocorrer durante o desenvolvimento e o uso de AWS IoT regras, consulteMonitoramento AWS IoT.

Etapa 8: Limpeza

Se quiser repetir este tutorial, talvez seja necessário remover alguns dos autorizadores personalizados. Você Conta da AWS pode ter apenas um número limitado de autorizadores personalizados configurados ao mesmo tempo e você pode obter um `LimitExceededException` ao tentar adicionar um novo sem remover um autorizador personalizado existente.

Para remover um autorizador personalizado (console)

1. Abra a página do autorizador personalizado do AWS IoT console e, na lista de autorizadores personalizados, encontre o autorizador personalizado a ser removido.
2. Abra a página de detalhes do autorizador personalizado e, no menu Ações, escolha Editar.
3. Desmarque a opção Ativar autorizador e, em seguida, selecione Atualizar.

   Não é possível excluir um autorizador personalizado enquanto ele estiver ativo.
4. Na página de detalhes do autorizador personalizado, abra o menu Ações e selecione Excluir.

Para remover um autorizador personalizado (AWS CLI)

1. Liste os autorizadores personalizados que você instalou e localize o nome do autorizador personalizado que deseja excluir.

```
aws iot list-authorizers
```

2. Defina o autorizador personalizado como `inactive` executando esse comando após substituir *`Custom_Auth_Name`* pelo `authorizerName` do autorizador personalizado a ser excluído.

```
aws iot update-authorizer --status INACTIVE --authorizer-name Custom_Auth_Name
```

3. Exclua o autorizador personalizado executando esse comando após substituir *`Custom_Auth_Name`* pelo `authorizerName` do autorizador personalizado a ser excluído.

```
aws iot delete-authorizer --authorizer-name Custom_Auth_Name
```

Tutorial: Monitorando a umidade do solo com o AWS IoT Raspberry Pi

Este tutorial mostra como usar um Raspberry Pi, um sensor de umidade, e AWS IoT monitorar o nível de umidade do solo de uma planta doméstica ou jardim. O Raspberry Pi executa um código que lê o nível de umidade e a temperatura do sensor e, em seguida, envia os dados para AWS IoT. Você cria uma regra AWS IoT que envia um e-mail para um endereço inscrito em um tópico do Amazon SNS quando o nível de umidade cai abaixo de um limite.

> **Note**
>
> Este tutorial pode não estar atualizado. Algumas referências podem ter sido substituídas desde que este tópico foi publicado originalmente.

Sumário

- Pré-requisitos
- Conf AWS IoT iguração
  - Etapa 1: criar a AWS IoT política
  - Etapa 2: criar a AWS IoT coisa, o certificado e a chave privada
  - Etapa 3: criar um tópico e uma assinatura do Amazon SNS
  - Etapa 4: criar uma AWS IoT regra para enviar um e-mail
- Configuração do Raspberry Pi e do sensor de umidade

Pré-requisitos

Para concluir este tutorial, é necessário:

- Um Conta da AWS.
- Um usuário do IAM com permissões de administrador.
- Um computador de desenvolvimento que execute Windows, macOS, Linux ou Unix para acessar o console do AWS IoT.
- Um Raspberry Pi 3B ou 4B que execute a versão mais recente do Raspberry Pi OS. Para acessar instruções de instalação, consulte Install an operating system no site do Raspberry Pi.
- Um monitor, um teclado, um mouse e uma rede Wi-Fi ou conexão Ethernet para o Raspberry Pi.
- Um sensor de umidade compatível com o Raspberry Pi. O sensor usado neste tutorial é um Sensor capacitivo de umidade Adafruit STEMMA I2C com um JST de 4 pinos com conector de cabo de soquete fêmea.

Conf AWS IoT iguração

Para concluir este tutorial, é necessário criar os recursos a seguir. Para conectar um dispositivo AWS IoT, você cria uma coisa de IoT, um certificado de dispositivo e uma AWS IoT política.

- Qualquer AWS IoT coisa.

  Umo objeto representa um dispositivo físico (nesse caso, o Rasberry Pi) e contém metadados estáticos sobre o dispositivo.
- Um certificado de dispositivo.

  Todos os dispositivos devem ter um certificado de dispositivo para se conectar e autenticar com o AWS IoT.
- Uma AWS IoT política.

  Cada certificado de dispositivo tem uma ou mais AWS IoT políticas associadas a ele. Essas políticas determinam quais AWS IoT recursos o dispositivo pode acessar.
- Um certificado CA AWS IoT raiz.

  Os dispositivos e outros clientes usam um certificado CA AWS IoT raiz para autenticar o AWS IoT servidor com o qual estão se comunicando. Para obter mais informações, consulte Autenticação do servidor.

- Uma AWS IoT regra.

  Uma regra contém uma consulta e uma ou mais ações de regra. A consulta extrai dados de mensagens do dispositivo para determinar se os dados da mensagem devem ser processados. A ação da regra especifica o que fazer se os dados corresponderem à consulta.

- Uma assinatura e um tópico do Amazon SNS.

  A regra recebe dados de umidade do Raspberry Pi. Se o valor estiver abaixo de um limite, ele enviará uma mensagem ao tópico do Amazon SNS. O Amazon SNS envia essa mensagem para todos os endereços de e-mail assinantes do tópico.

Etapa 1: criar a AWS IoT política

Crie uma AWS IoT política que permita que seu Raspberry Pi se conecte e envie mensagens para o. AWS IoT

1. No console do AWS IoT, se um botão Começar for exibido, selecione-o. Caso contrário, no painel de navegação, expanda a opção Proteger e selecione a opção Políticas.
2. Se a caixa de diálogo Você ainda não tem políticas, selecione Criar uma política. Caso contrário, escolha a opção Criar.
3. Insira um nome para a AWS IoT política (por exemplo,**`MoistureSensorPolicy`**).
4. Na seção Adicionar instruções, substitua a política existente pelo JSON a seguir. *`account`*Substitua *`region`* e pelo seu Região da AWS Conta da AWS número e.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:Connect",
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/RaspberryPi"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Publish",
            "Resource": [
```

```
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/update",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/delete",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/get"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Receive",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/update/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/delete/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/get/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/update/rejected",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
RaspberryPi/shadow/delete/rejected"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
RaspberryPi/shadow/update/accepted",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
RaspberryPi/shadow/delete/accepted",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
RaspberryPi/shadow/get/accepted",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
RaspberryPi/shadow/update/rejected",
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
RaspberryPi/shadow/delete/rejected"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:GetThingShadow",
```

```
                "iot:UpdateThingShadow",
                "iot:DeleteThingShadow"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:thing/RaspberryPi"
        }
    ]
}
```

5. Escolha Criar.

Etapa 2: criar a AWS IoT coisa, o certificado e a chave privada

Crie uma coisa no AWS IoT registro para representar seu Raspberry Pi.

1. No console do AWS IoT, no painel de navegação, selecione Gerenciar e Objetos.
2. Se uma caixa de diálogo Você ainda não tem objetos for exibida, selecione a opção Registrar um objeto. Caso contrário, escolha Criar.
3. Na página Criando AWS IoT coisas, escolha Criar uma única coisa.
4. Na página Adicionar o dispositivo ao registro do dispositivo, insira um nome para o objeto de IoT (por exemplo, **`RaspberryPi`**) e selecione Próximo. Você não pode alterar o nome de um objeto depois de criá-lo. Para alterar o nome de um objeto, é necessário criar um objeto, fornecer o novo nome e, depois, excluir o objeto antiga.
5. Na página Adicionar um certificado ao objeto, escolha Criar certificado.
6. Escolha os links Download para fazer download do certificado, da chave privada e do certificado CA raiz.

   > **Important**
   >
   > Esta é a única vez que você pode fazer download do certificado e da chave privada.

7. Para ativar o certificado, selecione a opção Ativar. O certificado deve estar ativo para que um dispositivo se conecte ao AWS IoT.
8. Selecione a opção Anexar uma política.
9. Em Adicionar uma política para sua coisa, escolha e MoistureSensorPolicy, em seguida, escolha Registrar coisa.

Etapa 3: criar um tópico e uma assinatura do Amazon SNS

Criar um tópico e uma assinatura do Amazon SNS.

1. No console SNS do AWS, no painel de navegação, selecione Tópicos e, em seguida, selecione Criar tópico.
2. Escolha o tipo como Padrão e insira um nome para o tópico (por exemplo, **`MoistureSensorTopic`**).
3. Insira um nome de exibição para o tópico (por exemplo, **`Moisture Sensor Topic`**). Esse é o nome exibido para o tópico no console do Amazon SNS .
4. Escolha Criar tópico.
5. Na página de detalhes do tópico do Amazon SNS, selecione Criar assinatura.
6. Em Protocolo, escolha Email.
7. Para Endpoint, insira seu endereço de e-mail.
8. Selecione Criar assinatura.
9. Abra o cliente de e-mail e procure uma mensagem com o assunto **`MoistureSensorTopic`**. Abra o e-mail e clique no link Confirmar assinatura.

> **Important**
>
> Você não receberá nenhum alerta por e-mail deste tópico do Amazon SNS até confirmar a assinatura.

Você deve receber uma mensagem de e-mail com o texto digitado.

Etapa 4: criar uma AWS IoT regra para enviar um e-mail

Uma AWS IoT regra define uma consulta e uma ou mais ações a serem tomadas quando uma mensagem é recebida de um dispositivo. O mecanismo de AWS IoT regras escuta as mensagens enviadas pelos dispositivos e usa os dados nas mensagens para determinar se alguma ação deve ser tomada. Para obter mais informações, consulte Regras para AWS IoT.

Neste tutorial, o Raspberry Pi publica mensagens no `aws/things/RaspberryPi/shadow/update`. Este é um tópico MQTT interno usado por dispositivos e pelo serviço Thing Shadow. O Raspberry Pi publica mensagens que têm o seguinte formato:

```
{
    "reported": {
        "moisture" : moisture-reading,
        "temp" : temperature-reading
    }
}
```

Você cria uma consulta que extrai os dados de umidade e temperatura da mensagem recebida. Você também cria uma ação do Amazon SNS que usa os dados e os envia aos assinantes do tópico do Amazon SNS se a leitura de umidade estiver abaixo de um valor limite.

Criar uma regra do Amazon SNS

1. No AWS IoT console, escolha Encaminhamento de mensagens e, em seguida, escolha Regras. Se uma caixa de diálogo Você ainda não tem regras, selecione Criar uma regra. Caso contrário, selecione Criar regra.
2. Na página Propriedades da regra, insira um nome de regra como **`MoistureSensorRule`**, e forneça uma breve descrição da regra, como**`Sends an alert when soil moisture level readings are too low`**.
3. Escolha Próximo e configure sua instrução SQL. Escolha a versão SQL como 2016-03-23 e insira a seguinte AWS IoT instrução de consulta SQL:

   ```
   SELECT * FROM '$aws/things/RaspberryPi/shadow/update/accepted' WHERE
    state.reported.moisture < 400
   ```

   Essa instrução aciona a ação da regra quando a leitura de `moisture` é menor que `400`.

   > **Note**
   >
   > Talvez seja necessário usar um valor diferente. Depois de ter o código em execução no Raspberry Pi, você poderá ver os valores obtidos do sensor tocando no sensor, colocando-o na água ou colocando-o em um vaso.

4. Escolha a opção Próximo e anexe ações de regra. Para a Ação 1, escolha Serviço de Notificação Simples. A descrição dessa ação de regra é Enviar uma mensagem como uma notificação push do SNS.

5. Para o tópico SNS, escolha o tópico que você criou em Etapa 3: criar um tópico e uma assinatura do Amazon SNS MoistureSensorTopic, e deixe o formato da mensagem como RAW. Em Perfil do IAM, selecione Criar uma nova função. Insira um nome para a função, por exemplo, **`LowMoistureTopicRole`**, e escolha Criar função.
6. Escolha Próximo para revisar e, em seguida, escolha Criar para criar a regra.

Configuração do Raspberry Pi e do sensor de umidade

Insira o cartão microSD no Raspberry Pi, conecte o monitor, o teclado, o mouse e, se você não estiver usando a rede Wi-Fi, o cabo Ethernet. Não conecte o cabo de alimentação ainda.

Conecte o cabo jumper JST ao sensor de umidade. O outro lado do jumper tem quatro cabos:

- Verde: I2C SCL
- Branco: I2C SDA
- Vermelho: alimentação (3,5 V)
- Preto: terra

Mantenha o Raspberry Pi com o conector Ethernet à direita. Nessa orientação, há duas linhas de pinos GPIO na parte superior. Conecte os cabos do sensor de umidade à linha inferior de pinos na ordem a seguir. A partir do pino mais à esquerda, conecte o vermelho (alimentação), o branco (SDA) e o verde (SCL). Ignore um pino e conecte o fio preto (terra). Para obter mais informações, consulte Cabeamento de computador Python.

Conecte o cabo de alimentação ao Raspberry Pi e conecte a outra extremidade a uma tomada para ligá-lo.

Configurar o Raspberry Pi

1. Em Boas-vindas ao Raspberry Pi, selecione Próximo.
2. Escolha o país, o idioma, o fuso horário e o layout do teclado. Escolha Próximo.
3. Insira uma senha para o Raspberry Pi e selecione Próximo.
4. Escolha a rede Wi-Fi e selecione Próximo. Se você não estiver usando uma rede Wi-Fi, selecione Ignorar.
5. Escolha Próximo para verificar se há atualizações de software. Quando as atualizações forem concluídas, selecione Reiniciar para reiniciar o Raspberry Pi.

Depois que o Raspberry Pi for iniciado, habilite a interface I2C.

1. No canto superior esquerdo da área de trabalho do Raspbian, clique no ícone do Raspberry, selecione Preferências e Configuração do Raspberry Pi.
2. Na guia Interfaces para I2C, selecione Habilitar.
3. Escolha OK.

As bibliotecas do sensor de umidade Adafruit STEMMA foram criadas para. CircuitPython Para executá-las em um Raspberry Pi, é necessário instalar a versão mais recente do Python 3.

1. Execute os seguintes comandos em um prompt de comando para atualizar o software do Raspberry Pi:

   ```
   sudo apt-get update
   ```

   ```
   sudo apt-get upgrade
   ```
2. Execute o seguinte comando para atualizar a instalação do Python 3:

   ```
   sudo pip3 install --upgrade setuptools
   ```
3. Execute o seguinte comando para instalar as bibliotecas GPIO do Raspberry Pi:

   ```
   pip3 install RPI.GPIO
   ```
4. Execute o seguinte comando para instalar as bibliotecas Adafruit Blinka:

   ```
   pip3 install adafruit-blinka
   ```

   Para obter mais informações, consulte Instalando CircuitPython bibliotecas no Raspberry Pi.
5. Execute o seguinte comando para instalar as bibliotecas Adafruit Seesaw:

   ```
   sudo pip3 install adafruit-circuitpython-seesaw
   ```
6. Execute o comando a seguir para instalar o AWS IoT Device SDK para Python:

   ```
   pip3 install AWSIoTPythonSDK
   ```

Agora o Raspberry Pi tem todas as bibliotecas necessárias. Crie um arquivo chamado **`moistureSensor.py`** e copie o seguinte código Python no arquivo:

```
from adafruit_seesaw.seesaw import Seesaw
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTShadowClient
from board import SCL, SDA

import logging
import time
import json
import argparse
import busio

# Shadow JSON schema:
#
# {
#   "state": {
#       "desired":{
#           "moisture":<INT VALUE>,
#           "temp":<INT VALUE>
#       }
#   }
# }

# Function called when a shadow is updated
def customShadowCallback_Update(payload, responseStatus, token):

    # Display status and data from update request
    if responseStatus == "timeout":
        print("Update request " + token + " time out!")

    if responseStatus == "accepted":
        payloadDict = json.loads(payload)
        print("~~~~~~~~~~~~~~~~~~~~~~")
        print("Update request with token: " + token + " accepted!")
        print("moisture: " + str(payloadDict["state"]["reported"]["moisture"]))
        print("temperature: " + str(payloadDict["state"]["reported"]["temp"]))
        print("~~~~~~~~~~~~~~~~~~~~~~\n\n")

    if responseStatus == "rejected":
        print("Update request " + token + " rejected!")

# Function called when a shadow is deleted
def customShadowCallback_Delete(payload, responseStatus, token):

     # Display status and data from delete request
    if responseStatus == "timeout":
```

```
        print("Delete request " + token + " time out!")

    if responseStatus == "accepted":
        print("~~~~~~~~~~~~~~~~~~~~~")
        print("Delete request with token: " + token + " accepted!")
        print("~~~~~~~~~~~~~~~~~~~~~\n\n")

    if responseStatus == "rejected":
        print("Delete request " + token + " rejected!")


# Read in command-line parameters
def parseArgs():

    parser = argparse.ArgumentParser()
    parser.add_argument("-e", "--endpoint", action="store", required=True, dest="host",
 help="Your device data endpoint")
    parser.add_argument("-r", "--rootCA", action="store", required=True,
 dest="rootCAPath", help="Root CA file path")
    parser.add_argument("-c", "--cert", action="store", dest="certificatePath",
 help="Certificate file path")
    parser.add_argument("-k", "--key", action="store", dest="privateKeyPath",
 help="Private key file path")
    parser.add_argument("-p", "--port", action="store", dest="port", type=int,
 help="Port number override")
    parser.add_argument("-n", "--thingName", action="store", dest="thingName",
 default="Bot", help="Targeted thing name")
    parser.add_argument("-id", "--clientId", action="store", dest="clientId",
 default="basicShadowUpdater", help="Targeted client id")

    args = parser.parse_args()
    return args


# Configure logging
# AWSIoTMQTTShadowClient writes data to the log
def configureLogging():

    logger = logging.getLogger("AWSIoTPythonSDK.core")
    logger.setLevel(logging.DEBUG)
    streamHandler = logging.StreamHandler()
    formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s -
 %(message)s')
    streamHandler.setFormatter(formatter)
```

```
    logger.addHandler(streamHandler)


# Parse command line arguments
args = parseArgs()

if not args.certificatePath or not args.privateKeyPath:
    parser.error("Missing credentials for authentication.")
    exit(2)

# If no --port argument is passed, default to 8883
if not args.port:
    args.port = 8883


# Init AWSIoTMQTTShadowClient
myAWSIoTMQTTShadowClient = None
myAWSIoTMQTTShadowClient = AWSIoTMQTTShadowClient(args.clientId)
myAWSIoTMQTTShadowClient.configureEndpoint(args.host, args.port)
myAWSIoTMQTTShadowClient.configureCredentials(args.rootCAPath, args.privateKeyPath,
 args.certificatePath)

# AWSIoTMQTTShadowClient connection configuration
myAWSIoTMQTTShadowClient.configureAutoReconnectBackoffTime(1, 32, 20)
myAWSIoTMQTTShadowClient.configureConnectDisconnectTimeout(10) # 10 sec
myAWSIoTMQTTShadowClient.configureMQTTOperationTimeout(5) # 5 sec

# Initialize Raspberry Pi's I2C interface
i2c_bus = busio.I2C(SCL, SDA)

# Intialize SeeSaw, Adafruit's Circuit Python library
ss = Seesaw(i2c_bus, addr=0x36)

# Connect to AWS IoT
myAWSIoTMQTTShadowClient.connect()

# Create a device shadow handler, use this to update and delete shadow document
deviceShadowHandler =
 myAWSIoTMQTTShadowClient.createShadowHandlerWithName(args.thingName, True)

# Delete current shadow JSON doc
deviceShadowHandler.shadowDelete(customShadowCallback_Delete, 5)

# Read data from moisture sensor and update shadow
```

```
while True:

    # read moisture level through capacitive touch pad
    moistureLevel = ss.moisture_read()

    # read temperature from the temperature sensor
    temp = ss.get_temp()

    # Display moisture and temp readings
    print("Moisture Level: {}".format(moistureLevel))
    print("Temperature: {}".format(temp))

    # Create message payload
    payload = {"state":{"reported":{"moisture":str(moistureLevel),"temp":str(temp)}}}

    # Update shadow
    deviceShadowHandler.shadowUpdate(json.dumps(payload), customShadowCallback_Update,
 5)
    time.sleep(1)
```

Salve o arquivo em um local onde você possa encontrá-lo. Execute `moistureSensor.py` na linha de comando com os seguintes parâmetros:

endpoint

Seu AWS IoT endpoint personalizado. Para obter mais informações, consulte API REST da Sombra do Dispositivo.

rootCA

O caminho completo para seu certificado CA AWS IoT raiz.

cert

O caminho completo para o certificado AWS IoT do seu dispositivo.

key

O caminho completo para a chave privada AWS IoT do certificado do seu dispositivo.

thingName

O nome do objeto (neste caso, `RaspberryPi`).

clientId

O ID do cliente MQTT. Use `RaspberryPi`.

A linha de comando deve ser semelhante a esta:

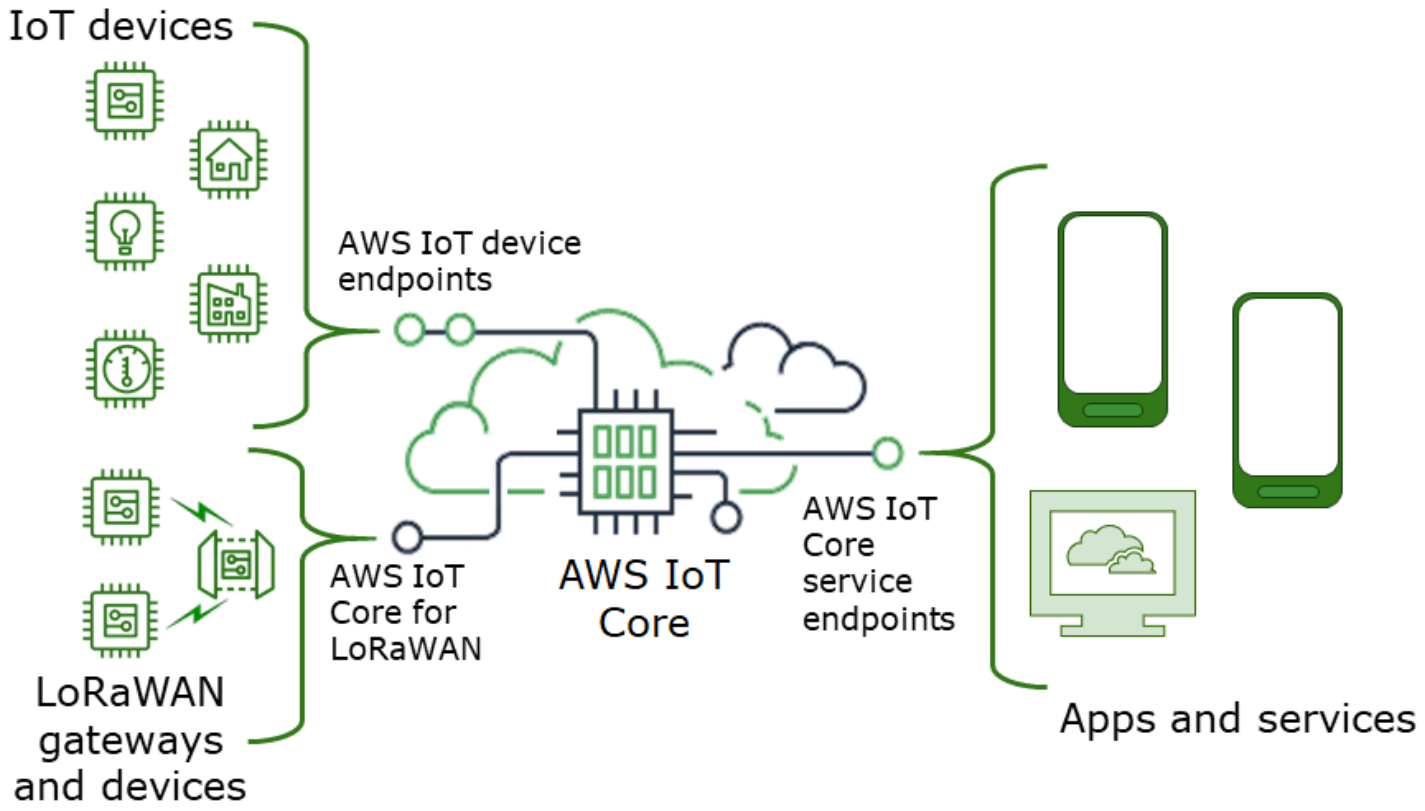
```
python3 moistureSensor.py --endpoint your-endpoint --rootCA ~/certs/
AmazonRootCA1.pem --cert ~/certs/raspberrypi-certificate.pem.crt --key
~/certs/raspberrypi-private.pem.key --thingName RaspberryPi --clientId
RaspberryPi
```

Tente tocar no sensor, colocá-lo em um vaso ou em um copo de água para ver como o sensor responde a vários níveis de umidade. Se for necessário, você poderá alterar o valor limite no `MoistureSensorRule`. Quando a leitura do sensor de umidade fica abaixo do valor especificado na instrução de consulta SQL da sua regra, AWS IoT publica uma mensagem no tópico do Amazon SNS. Você deve receber uma mensagem de e-mail que contém os dados de umidade e temperatura.

Depois de verificar o recebimento de mensagens de e-mail do Amazon SNS, pressione CTRL +C para interromper o programa Python. É improvável que o programa Python envie mensagens suficientes para gerar cobranças, mas é uma prática recomendada interromper o programa ao concluir.

Conecte-se a AWS IoT Core

AWS IoT Core suporta conexões com dispositivos de IoT, gateways sem fio, serviços e aplicativos. Os dispositivos se conectam para que AWS IoT Core possam enviar e receber dados de AWS IoT serviços e outros dispositivos. Aplicativos e outros serviços também se conectam AWS IoT Core para controlar e gerenciar os dispositivos de IoT e processar os dados da sua solução de IoT. Esta seção descreve como escolher a melhor maneira de se conectar e se comunicar com AWS IoT Core cada aspecto da sua solução de IoT.



Há várias maneiras de interagir com AWS IoT. Aplicativos e serviços podem usar o AWS IoT Core - endpoints do ambiente de gerenciamento e os dispositivos podem se conectar AWS IoT Core usando AWS IoT endpoints do dispositivo ou AWS IoT Core para regiões e endpoints de LoRa WAN.

AWS IoT Core - endpoints do ambiente de gerenciamento

Os AWS IoT Core terminais do plano de controle fornecem acesso às funções que controlam e gerenciam sua AWS IoT solução.

- Endpoints

Os endpoints do ambiente de gerenciamento do AWS IoT Core e do ambiente de gerenciamento do AWS IoT Core Device Advisor são específicos da região e estão listados em Endpoints e cotas do AWS IoT Core. Os formatos dos endpoints são os seguintes.

Propósito do endpoint	Formato do endpoint	Atende
AWS IoT Core - ambiente de gerenciamento	AWS IoT Pontos finais do plano de controle	AWS IoT API do plano de controle
AWS IoT Core Device Advisor - plano de controle	`api.iotdeviceadvisor.` *`aws-region`* `.amazonaws.com`	AWS IoT Core API do plano de controle do Device Advisor

- IPv4 endpoints — os IPv4 endpoints oferecem suporte somente ao IPv4 tráfego e estão disponíveis para todas as regiões.

  IPv4 os endpoints usam a seguinte convenção de nomenclatura:

  ```
  iot.aws-region.amazonaws.com
  ```

  Por exemplo, o nome do IPv4 endpoint para a região us-east-1 é. `iot.us-east-1.amazonaws.com`

- Endpoints de pilha dupla (IPv4 e IPv6) — Os endpoints de pilha dupla oferecem suporte tanto ao tráfego quanto ao tráfego. IPv4 IPv6 Quando uma solicitação é feita para um endpoint de pilha dupla, o URL do endpoint é resolvido para um IPv4 endereço IPv6 ou, dependendo do protocolo usado pela rede e pelo cliente.

  Endpoints de pilha dupla usam a seguinte convenção de nomenclatura:

  ```
  iot.aws-region.api.aws
  ```

  Por exemplo, o nome do endpoint de pilha dupla para a região us-east-1 é `iot.us-east-1.api.aws`.

- SDKs e ferramentas

Eles AWS SDKsfornecem suporte específico de um idioma para AWS IoT Core APIs o e APIs de outros AWS serviços. O AWS Mobile SDKs fornece aos desenvolvedores de aplicativos suporte específico da plataforma para a AWS IoT Core API e outros AWS serviços em dispositivos móveis.

O AWS CLIfornece acesso por linha de comando às funções fornecidas pelos endpoints de AWS IoT serviço. AWS Tools for PowerShell fornece ferramentas para gerenciar AWS serviços e recursos no ambiente PowerShell de script.

- Autenticação

  Os endpoints do serviço usam usuários e AWS credenciais do IAM para autenticar usuários.

- Saiba mais

  Para obter mais informações e links para referências do SDK, consulte the section called "Conecte-se aos endpoints AWS IoT Core de serviço".

AWS IoT endpoints do dispositivo

Os endpoints do AWS IoT dispositivo oferecem suporte à comunicação entre seus dispositivos de IoT e. AWS IoT

- Endpoints

  O suporte AWS IoT Core e as AWS IoT Device Management funções dos endpoints do dispositivo. Eles são específicos para você Conta da AWS e você pode ver o que são usando o describe-endpoint comando.

Propósito do endpoint	Formato do endpoint	Atende
AWS IoT Core - plano de dados	Consulte ???.	AWS IoT API do plano de dados
AWS IoT Device Managemen t - dados de trabalhos	Consulte ???.	AWS IoT API do plano de dados de empregos
AWS IoT Device Advisor - plano de dados	Consulte ???.	Não aplicável

Propósito do endpoint	Formato do endpoint	Atende
AWS IoT Device Management - Encapsulamento seguro	`api.tunneling.iot.` *`aws-region`* `.amazonaws.com`	AWS IoT API de tunelamento seguro

Se você estiver usando endpoints de pilha dupla (IPv4 e IPv6) para operações de plano de dados, use o tipo de endpoint. `iot:Data-ATS` `iot:Jobs` IPv4 só pode ser usado para. Para obter mais informações sobre esses endpoints e as funções compatíveis com eles, consulte the section called "AWS IoT dados do dispositivo e endpoints de serviço".

- SDKs

  O AWS IoT dispositivo SDKs fornece suporte específico de idioma para os protocolos Message Queueing Telemetry Transport (MQTT) e WebSocket Secure (WSS), que os dispositivos usam para se comunicar. AWS IoTAWS Mobilidade SDKstambém fornecem suporte para comunicações de dispositivos MQTT e outros AWS serviços em dispositivos móveis. AWS IoT APIs APIs

- Autenticação

  Os endpoints do dispositivo usam certificados X.509 ou usuários AWS do IAM com credenciais para autenticar usuários.

- Saiba mais

  Para obter mais informações e links para referências do SDK, consulte the section called "AWS IoT Dispositivo SDKs".

AWS IoT Core para dispositivos e gateways de LoRa WAN

AWS IoT Core for LoRa WAN conecta gateways e dispositivos sem fio a. AWS IoT Core

- Endpoints

  AWS IoT Core for LoRa WAN gerencia as conexões de gateway para endpoints específicos da conta e da região AWS IoT Core . Os gateways podem se conectar ao endpoint do Servidor de Configuração e Atualização (CUPS) da sua conta, fornecido AWS IoT Core pela LoRa WAN.

Propósito do endpoint	Formato do endpoint	Atende
Servidor de configuração e atualização (CUPS)	*account-specific-prefix* .cups.lorawan. *aws-region* .amazonaws.com:443	Comunicação de gateway com o servidor de configuração e atualização fornecido pelo AWS IoT Core for LoRa WAN
LoRaServidor de rede WAN (LNS)	*account-specific-prefix* .gateway.lorawan. *aws-region* .amazonaws.com:443	Comunicação de gateway com o servidor de rede LoRa WAN fornecido pela AWS IoT Core for LoRa WAN

- SDKs

  A API AWS IoT sem fio na qual AWS IoT Core a LoRa WAN é construída é suportada pelo AWS SDK. Para obter mais informações, consulte Kits AWS SDKs de ferramentas.

- Autenticação

  AWS IoT Core para comunicações de dispositivos LoRa WAN, use certificados X.509 para proteger as comunicações com. AWS IoT

- Saiba mais

  Para obter mais informações sobre como configurar e conectar dispositivos sem fio, consulte AWS IoT Core Regiões e endpoints de LoRa WAN.

Conecte-se aos endpoints AWS IoT Core de serviço

Você pode acessar os recursos do plano de controle AWS IoT Core- usando o AWS CLI AWS SDK do seu idioma preferido ou chamando diretamente a API REST. Recomendamos usar o AWS CLI ou um AWS SDK para interagir, AWS IoT Core pois eles incorporam as melhores práticas para AWS serviços de chamadas. Chamar o REST APIs diretamente é uma opção, mas você deve fornecer as credenciais de segurança necessárias para permitir o acesso à API.

> **Note**
>
> Os dispositivos de IoT devem usar AWS IoT Dispositivo SDKs. O dispositivo SDKs é otimizado para uso em dispositivos, suporta a comunicação MQTT com AWS IoT e suporta os dispositivos AWS IoT APIs mais usados pelos dispositivos. Para obter mais informações sobre o dispositivo SDKs e os recursos que ele fornece, consulteAWS IoT Dispositivo SDKs. Os dispositivos móveis devem usar AWS Mobilidade SDKs. O Mobile SDKs fornece suporte para AWS IoT APIs comunicações de dispositivos MQTT e outros AWS serviços em dispositivos móveis. APIs Para obter mais informações sobre o celular SDKs e os recursos que eles fornecem, consulteAWS Mobilidade SDKs.

Você pode usar AWS Amplify ferramentas e recursos em aplicativos móveis e da Web para se conectar com mais facilidade AWS IoT Core. Para obter mais informações sobre como se conectar AWS IoT Core usando o Amplify, consulte PubSuba documentação do Amplify.

As seções a seguir descrevem as ferramentas SDKs que você pode usar para desenvolver e interagir com AWS IoT outros AWS serviços. Para ver a lista completa de AWS ferramentas e kits de desenvolvimento que estão disponíveis para criar e gerenciar aplicativos AWS, consulte Ferramentas para criar AWS.

AWS CLI para AWS IoT Core

O AWS CLI fornece acesso à linha de comando a. AWS APIs

- Instalação

  Para obter informações sobre como instalar o AWS CLI, consulte Instalando AWS CLI o.
- Autenticação

  O AWS CLI usa as credenciais do seu. Conta da AWS
- Referência

  Para obter informações sobre os AWS CLI comandos desses AWS IoT Core serviços, consulte:
  - AWS CLI Referência de comandos para IoT
  - AWS CLI Referência de comando para dados de IoT
  - AWS CLI Referência de comando para dados de trabalhos de IoT
  - AWS CLI Referência de comando para tunelamento seguro de IoT

Para obter ferramentas para gerenciar AWS serviços e recursos no ambiente PowerShell de script, consulte AWS Ferramentas para PowerShell.

AWS SDKs

Com AWS SDKs, seus aplicativos e dispositivos compatíveis podem fazer chamadas AWS IoT APIs e ligar para outros AWS serviços. APIs Esta seção fornece links para AWS SDKs e para a documentação de referência APIs da API para os AWS IoT Core serviços.

O AWS SDKs apoio a esses AWS IoT Core APIs

- AWS IoT
- AWS IoT Plano de dados
- AWS IoT Plano de dados de trabalhos
- AWS IoT Tunelamento seguro
- AWS IoT Sem fio

C++

Para instalar o AWS SDK para C++ e usá-lo para se conectar à AWS IoT:

1. Siga as instruções em Introdução ao uso do AWS SDK para C++

   Essas instruções descrevem como:

   - Instalar e compilar o SDK dos arquivos de origem
   - Fornecer credenciais para usar o SDK com sua Conta da AWS
   - Inicializar e desligar o SDK em seu aplicativo ou serviço
   - Crie um CMake projeto para criar seu aplicativo ou serviço
2. Criar e executar um aplicativo de exemplo. Para exemplos de aplicativos que usam o SDK da AWS para C++ Exemplos de códigos do AWS SDK para C++.

Documentação dos AWS IoT Core serviços que o AWS SDK para C++ suporta

- AWS: :IoTClient” documentação de referência
- Documentação de referência do Aws: :Io TData Plane: TData PlaneClient :Io
- Documentação de referência do Aws: :Io TJobsDataPlane: TJobs DataPlaneClient :Io

- Documentação de referência do Aws: :Io TSecure Tunneling: :Io TSecure TunnelingClient

Go

Para instalar o AWS SDK para Go e usá-lo para se conectar à AWS IoT:

1. Siga as instruções em Introdução ao AWS SDK para Go

   Essas instruções descrevem como:

   - Instale o AWS SDK para Go
   - Obter chaves de acesso para que o SDK acesse sua Conta da AWS
   - Importar pacotes para o código-fonte de nossos aplicativos ou serviços
2. Criar e executar um aplicativo de exemplo. Para ver exemplos de aplicativos que usam o AWS SDK para Go, consulte Exemplos de códigos do AWS SDK para Go.

Documentação dos AWS IoT Core serviços que o AWS SDK para Go suporta

- Documentação de referência de IoT
- Documentação de referência TData do Io Plane
- Documentação de TJobs DataPlane referência do Io
- Documentação de referência do Io TSecure Tunneling

Java

Para instalar o AWS SDK para Java e usá-lo para se conectar à AWS IoT:

1. Siga as instruções em Introdução ao AWS SDK for Java 2.x

   Essas instruções descrevem como:

   - Inscreva-se AWS e crie um usuário do IAM
   - Fazer download do SDK
   - Configurar AWS credenciais e região
   - Usar o SDK com o Apache Maven
   - Usar o SDK com o Gradle

2. Criar e executar um aplicativo de exemplo usando um dos Códigos de exemplo do AWS SDK for Java 2.x.
3. Analisar a Documentação de referência da API SDK

Documentação dos AWS IoT Core serviços que o AWS SDK para Java suporta

- IotClient documentação de referência
- IotDataPlaneClient documentação de referência
- IotJobsDataPlaneClient documentação de referência
- Documentação de TSecure TunnelingClient referência do Io

JavaScript

Para instalar o AWS SDK para JavaScript e usá-lo para se conectar a AWS IoT:

1. Siga as instruções em Configurar o AWS SDK para JavaScript. Essas instruções se aplicam ao uso do AWS SDK para JavaScript no navegador e com o Node.JS. Siga as instruções que se aplicam à sua instalação.

   Essas instruções descrevem como:

   - Verificar os pré-requisitos
   - Instale o SDK para JavaScript
   - Carregue o SDK para JavaScript
2. Criar e executar um aplicativo de exemplo para começar a usar o SDK, conforme descrito na opção de introdução do seu ambiente.
   - Comece a usar o AWS SDK para JavaScript no navegador ou
   - Comece a usar o AWS SDK para JavaScript em Node.js

Documentação dos AWS IoT Core serviços que o AWS SDK para JavaScript suporta

- `AWS.Iot reference documentation`
- `AWS.IotData reference documentation`
- `AWS.IotJobsDataPlane reference documentation`
- `AWS.IotSecureTunneling reference documentation`

.NET

Para instalar o AWS SDK para .NET e usá-lo para se conectar à AWS IoT:

1. Siga as instruções em Configurando seu AWS SDK para .NET ambiente
2. Siga as instruções em Configurando seu AWS SDK para .NET projeto

   Essas instruções descrevem como:

   - Iniciar um novo projeto
   - Obter e configurar AWS credenciais
   - Instalar AWS pacotes do SDK
3. Crie e execute um dos programas de amostra em Trabalhando com AWS serviços no AWS SDK for .NET
4. Analisar a Documentação de referência da API SDK

Documentação dos AWS IoT Core serviços que o AWS SDK para .NET suporta

- Documentação de referência de Amazon.IoT.Model
- Amazon. IotData.Documentação de referência do modelo
- Documentação de referência do Amazon.Io .Model TJobs DataPlane
- Documentação de referência do Amazon.Io Tunneling.Model TSecure

PHP

Para instalar o AWS SDK para PHP e usá-lo para se conectar à AWS IoT:

1. Siga as instruções em Introdução à AWS SDK para PHP versão 3

   Essas instruções descrevem como:

   - Verificar os pré-requisitos
   - Instalar o SDK
   - Aplicar o SDK a um script PHP
2. Criar e executar um aplicativo de exemplo usando um dos Códigos de exemplo do AWS SDK para PHP Versão 3

Documentação dos AWS IoT Core serviços que o AWS SDK para PHP suporta

- Documentação de TClient referência do Io
- Documentação de TData PlaneClient referência do Io
- Documentação de TJobs DataPlaneClient referência do Io
- Documentação de TSecure TunnelingClient referência do Io

Python

Para instalar o AWS SDK para Python (Boto3) e usá-lo para se conectar à AWS IoT:

1. Siga as instruções em Início rápido do AWS SDK para Python (Boto3)

   Essas instruções descrevem como:

   - Instalar o SDK
   - Configurar o SDK
   - Usar o SDK no seu código
2. Criar e executar um programa de exemplo que usa o AWS SDK para Python (Boto3)

   Esse programa exibe as opções de registro atualmente configuradas da conta. Depois de instalar o SDK e configurá-lo para sua conta, você deverá conseguir executar esse programa.

   ```
   import boto3
   import json

   # initialize client
   iot = boto3.client('iot')

   # get current logging levels, format them as JSON, and write them to stdout
   response = iot.get_v2_logging_options()
   print(json.dumps(response, indent=4))
   ```

   Para obter mais informações sobre a função usada nesse exemplo, consulte the section called “Configurar o AWS IoT registro”.

Documentação dos AWS IoT Core serviços que o AWS SDK para Python (Boto3) suporta

- Documentação de referência de IoT

- Documentação de referência TData do Io Plane
- Documentação de TJobs DataPlane referência do Io
- Documentação de referência do Io TSecure Tunneling

Ruby

Para instalar o AWS SDK para Ruby e usá-lo para se conectar à AWS IoT:

- Siga as instruções em Introdução ao AWS SDK para Ruby

  Essas instruções descrevem como:

  - Instalar o SDK
  - Configurar o SDK
- Criar e executar o tutorial Olá, mundo

Documentação dos AWS IoT Core serviços aos quais o AWS SDK for Ruby oferece suporte

- Documentação de referência de Aws::IoT::Client
- Documentação de referência do Aws: :Io TData Plane: :Client
- Documentação de referência do Aws: :Io TJobsDataPlane: :Client
- Aws: :Io TSecure Tunneling: :Documentação de referência do cliente

AWS Mobilidade SDKs

O AWS Mobile SDKs fornece aos desenvolvedores de aplicativos móveis suporte específico da plataforma para os serviços AWS IoT Core , a comunicação APIs de dispositivos de IoT usando o MQTT e outros serviços. APIs AWS

Android

AWS Mobile SDK para Android

AWS Mobile SDK para Android Ele contém uma biblioteca, exemplos e documentação para os desenvolvedores criarem aplicativos móveis conectados usando AWS. Esse SDK também inclui suporte para comunicações de dispositivos MQTT e chamadas APIs de serviços. AWS IoT Core Para saber mais, consulte:

- AWS SDK móvel para Android ativado GitHub
- AWS SDK móvel para Android Readme
- AWS Exemplos de SDK móvel para Android
- AWS Referência da API do SDK for Android
- AWSIoTClient Documentação de referência de classe

iOS

AWS Mobile SDK for iOS

O AWS Mobile SDK for iOS é um kit de desenvolvimento de software de código aberto, distribuído sob uma licença Apache Open Source. O SDK para iOS fornece uma biblioteca, exemplos de código e documentação para ajudar os desenvolvedores a criar aplicativos móveis conectados usando o. AWS Esse SDK também inclui suporte para comunicações de dispositivos MQTT e chamadas APIs de serviços. AWS IoT Core Para saber mais, consulte:

- AWS Mobile SDK for iOS em GitHub
- AWS SDK para iOS Readme
- AWS Amostras do SDK para iOS
- AWS IoT Documentos de referência de classe no AWS SDK para iOS

REST APIs dos AWS IoT Core serviços

O REST APIs dos AWS IoT Core serviços pode ser chamado diretamente usando solicitações HTTP.

- URL do endpoint

  Os endpoints de serviço que expõem o REST APIs dos AWS IoT Core serviços variam de acordo com a região e estão listados em AWS IoT Core Endpoints e Cotas. Você deve usar o endpoint da região que tem os AWS IoT recursos que você deseja acessar, porque AWS IoT os recursos são específicos da região.
- Autenticação

  O REST APIs dos AWS IoT Core serviços usa credenciais AWS do IAM para autenticação. Para obter mais informações, consulte Assinatura de solicitações da AWS API na Referência AWS geral.

- Referência de API

  Para obter informações sobre as funções específicas fornecidas pelo REST APIs dos AWS IoT Core serviços, consulte:

  - Referência de API para IoT.
  - Referência de API para dados de IoT.
  - Referência de API para dados de trabalhos de IoT.
  - Referência de API para encapsulamento seguro de IoT.

Conecte dispositivos a AWS IoT

Os dispositivos se conectam AWS IoT e outros serviços por meio de AWS IoT Core. Por meio AWS IoT Core disso, os dispositivos enviam e recebem mensagens usando endpoints de dispositivos específicos da sua conta. Os the section called “AWS IoT Dispositivo SDKs” permitem comunicações de dispositivos usando os protocolos MQTT e WSS. Para obter mais informações sobre protocolos que os dispositivos podem usar, consulte the section called “Protocolos de comunicação do dispositivo”.

O agente de mensagens

AWS IoT gerencia a comunicação do dispositivo por meio de um agente de mensagens. Dispositivos e clientes publicam mensagens no agente de mensagens e também assinam mensagens que o agente de mensagens publica. As mensagens são identificadas por um tópico definido pelo aplicativo. Quando o agente de mensagens recebe uma mensagem publicada por um dispositivo ou cliente, ele republica essa mensagem para os dispositivos e clientes que assinaram o tópico da mensagem. O agente de mensagens também encaminha mensagens para o mecanismo de AWS IoT regras, que pode agir sobre o conteúdo da mensagem.

AWS IoT segurança de mensagens

Conexões de dispositivos a serem AWS IoT usadas the section called “Certificados do cliente X.509” e AWS assinatura V4 para autenticação. As comunicações do dispositivo são protegidas pelo TLS versão 1.3 e AWS IoT exigem que os dispositivos enviem a extensão de Indicação de Nome do Servidor (SNI) quando se conectam. Para obter mais informações, consulte Segurança de transporte em AWS IoT.

AWS IoT dados do dispositivo e endpoints de serviço

> **Important**
>
> Você pode armazenar em cache ou armazenar os endpoints em seu dispositivo. Assim, você não precisará consultar a API `DescribeEndpoint` toda vez que um novo dispositivo for conectado. Os endpoints não mudarão depois de AWS IoT Core criá-los para sua conta.

Cada conta tem vários endpoints de dispositivo que são exclusivos da conta e são compatíveis com funções específicas de IoT. Os endpoints de dados do AWS IoT dispositivo oferecem suporte a um publish/subscribe protocolo projetado para as necessidades de comunicação dos dispositivos de IoT; no entanto, outros clientes, como aplicativos e serviços, também podem usar essa interface se o aplicativo exigir os recursos especializados que esses endpoints fornecem. Os endpoints de serviço do AWS IoT dispositivo oferecem suporte ao acesso centrado no dispositivo aos serviços de segurança e gerenciamento.

Para conhecer o endpoint de dados do dispositivo da sua conta, você pode encontrá-lo na página Configurações do seu AWS IoT Core console.

Para conhecer o endpoint do dispositivo da sua conta para uma finalidade específica, incluindo o endpoint de dados do dispositivo, use o comando describe-endpoint da CLI mostrado aqui ou a API REST `DescribeEndpoint` e insira o valor do parâmetro *`endpointType`* na tabela a seguir.

```
aws iot describe-endpoint --endpoint-type endpointType
```

Esse comando retorna um *`iot-endpoint`* no seguinte formato:*`account-specific-prefix`*`.iot.`*`aws-region`*`.amazonaws.com`.

Cada cliente tem um endpoint `iot:Data-ATS` e `iot:Data`. Cada endpoint usa um certificado X.509 para autenticar o cliente. É altamente recomendável que os clientes usem o tipo de endpoint `iot:Data-ATS` mais recente a fim de evitar problemas relacionados ao próximo cancelamento de confiança difundido pelas autoridades de certificação do Symantec. Fornecemos o `iot:Data` endpoint para dispositivos recuperarem dados de endpoints antigos que usam VeriSign certificados para compatibilidade com versões anteriores. Para obter mais informações, consulte Autenticação do servidor.

AWS IoT endpoints para dispositivos

Propósito do endpoint	Valor do ***endpointType***	Description
AWS IoT Core - Operações do plano de dados	`iot:Data-ATS`	Usado para enviar e receber dados dos componentes do agente de mensagens, da sombra do dispositivo e do mecanismo de regras da AWS IoT. `iot:Data-ATS` retorna um endpoint de dados assinados pela ATS.
AWS IoT Core - Operações do plano de dados (legado)	`iot:Data`	`iot:Data`retorna um endpoint de dados VeriSign assinado fornecido para compatibilidade com versões anteriores. O MQTT 5 não é compatível com endpoints Symantec (`iot:Data`).
AWS IoT Core acesso credencial	`iot:CredentialProvider`	Usado para trocar um certifica do X.509 integrado do dispositivo por credencia is temporárias a fim de se conectar diretamente a outros serviços da AWS . Para obter mais informações sobre como se conectar a outros AWS serviços, consulte Autorizaç ão de chamadas diretas para AWS serviços.
AWS IoT Device Managemen t- operações de dados de trabalhos	`iot:Jobs`	Usado para permitir que dispositivos interajam com o

Propósito do endpoint	Valor do *endpointType*	Description
		serviço AWS IoT Jobs usando o Jobs Device HTTPS APIs. `iot:Jobs` IPv4 só pode ser usado para. Se você estiver usando endpoints de pilha dupla (IPv4 e IPv6), use o tipo de endpoint. `iot:Data-ATS`
AWS IoT Operações do Device Advisor	`iot:DeviceAdvisor`	Um tipo de endpoint de teste usado para testar dispositivos com o Device Advisor. Para obter mais informações, consulte ???.
AWS IoT Core data beta (pré-visualização)	`iot:Data-Beta`	Um tipo de endpoint reservado para versões beta. Para obter mais informações sobre seu uso atual, consulte ???.

Você também pode usar seu próprio nome de domínio totalmente qualificado (FQDN), como*example.com*, e o certificado de servidor associado para conectar dispositivos usando. AWS IoT the section called “Configurações do domínio”

AWS IoT Dispositivo SDKs

O AWS IoT dispositivo SDKs ajuda você a conectar seus dispositivos de IoT AWS IoT Core e eles são compatíveis com os protocolos MQTT e MQTT sobre WSS.

O AWS IoT dispositivo é SDKs diferente do AWS SDKs porque SDKs suporta as AWS IoT necessidades de comunicação especializadas dos dispositivos de IoT, mas não oferece suporte a todos os serviços suportados pelo. AWS SDKs Os AWS IoT dispositivos SDKs são compatíveis com os AWS SDKs que oferecem suporte a todos os AWS serviços; no entanto, eles usam métodos de autenticação diferentes e se conectam a terminais diferentes, o que pode tornar o uso AWS SDKs impraticável em um dispositivo de IoT.

Dispositivos móveis

Eles the section called “AWS Mobilidade SDKs” suportam as comunicações do dispositivo MQTT APIs, alguns dos AWS IoT serviços e outros AWS serviços. APIs Se estiver desenvolvendo em um dispositivo móvel compatível, analise o SDK dele para ver se é a melhor opção para desenvolver sua solução de IoT.

C++

AWS IoT SDK de dispositivos C++

O AWS IoT C++ Device SDK permite que os desenvolvedores criem aplicativos conectados usando AWS e APIs dos AWS IoT Core serviços. Esse SDK foi especificamente projetado para dispositivos que não têm restrições de recursos e exigem recursos avançados, como enfileiramento de mensagens, suporte a vários threads e os mais recentes recursos de linguagem. Para mais informações, consulte:

- AWS IoT SDK do dispositivo C++ v2 ativado GitHub
- AWS IoT SDK do dispositivo C++ v2 Readme
- AWS IoT Amostras do Device SDK C++ v2
- AWS IoT Documentação da API C++ v2 do Device SDK

Python

AWS IoT SDK de dispositivo para Python

O AWS IoT Device SDK for Python possibilita que os desenvolvedores escrevam scripts em Python para usar seus dispositivos para acessar a plataforma por meio do MQTT ou AWS IoT do MQTT pelo protocolo Secure (WSS). WebSocket Ao conectar seus dispositivos aos AWS IoT Core serviços, os usuários podem trabalhar com segurança com o agente APIs de mensagens, as regras e o serviço Device Shadow que AWS IoT Core fornece e com outros AWS serviços AWS Lambda, como Amazon Kinesis, Amazon S3 e muito mais.

- AWS IoT SDK de dispositivo para Python v2 ativado GitHub
- AWS IoT SDK de dispositivo para Python v2 Readme
- AWS IoT Exemplos de SDK de dispositivo para Python v2
- AWS IoT Documentação da API Device SDK para Python v2

JavaScript

AWS IoT SDK do dispositivo para JavaScript

O AWS IoT Device SDK for JavaScript possibilita que os desenvolvedores criem JavaScript aplicativos que os APIs acessem AWS IoT Core usando MQTT ou MQTT por meio do protocolo. WebSocket Ele pode ser usado em ambientes Node.js e aplicações de navegador. Para saber mais, consulte:

- AWS IoT SDK do dispositivo para JavaScript v2 ativado GitHub
- AWS IoT SDK do dispositivo para Readme JavaScript v2
- AWS IoT SDK de dispositivo para amostras JavaScript v2
- AWS IoT Documentação da API Device SDK for JavaScript v2

Java

AWS IoT SDK de dispositivo para Java

O AWS IoT Device SDK for Java possibilita que os desenvolvedores Java acessem o do por meio APIs do MQTT ou AWS IoT Core do MQTT pelo protocolo. WebSocket O SDK é compatível com o serviço de sombra do dispositivo. Você pode acessar as sombras usando métodos HTTP, inclusive GET, UPDATE e DELETE. O SDK também oferece suporte a um modelo simplificado de acesso a sombras, o que permite que os desenvolvedores troquem dados com as sombras usando os métodos getter e setter, sem necessidade de serializar ou desserializar nenhum documento JSON. Para saber mais, consulte:

- AWS IoT SDK de dispositivo para Java v2 ativado GitHub
- AWS IoT SDK de dispositivo para Java v2 Readme
- AWS IoT Exemplos de SDK de dispositivo para Java v2
- AWS IoT Documentação da API Device SDK for Java v2

Embedded C

AWS IoT SDK de dispositivo para C incorporado

> **Important**
>
> Esse SDK é destinado ao uso por desenvolvedores de software incorporados experientes.

O AWS IoT Device SDK para C incorporado (C-SDK) é uma coleção de arquivos de origem C sob a licença de código aberto do MIT que podem ser usados em aplicativos incorporados para conectar com segurança dispositivos de IoT ao IoT Core. AWS Inclui as bibliotecas MQTT, JSON Parser e AWS IoT Device Shadow, entre outras. Ele é distribuído na forma de código-fonte e deve ser incorporado ao firmware do cliente juntamente com o código do aplicativo, outras bibliotecas e, opcionalmente, um RTOS (Sistema Operacional em Tempo Real).

Geralmente, AWS IoT Device SDK para C incorporado é direcionado a dispositivos com recursos limitados que exigem um tempo de execução otimizado da linguagem C. Você pode usar o SDK em qualquer sistema operacional e hospedá-lo em qualquer tipo de processador (por exemplo, MCUs e MPUs). Se seu dispositivo tiver memória e recursos de processamento suficientes disponíveis, recomendamos que você use um dos outros AWS IoT dispositivos e dispositivos móveis SDKs, como o SDK do AWS IoT dispositivo para C++, JavaScript Java ou Python.

Para saber mais, consulte:

- AWS IoT SDK de dispositivo para C incorporado em GitHub
- AWS IoT SDK de dispositivo para Leiame C incorporado
- AWS IoT SDK de dispositivo para amostras C incorporadas

Protocolos de comunicação do dispositivo

AWS IoT Core suporta dispositivos e clientes que usam os protocolos MQTT e MQTT over WebSocket Secure (WSS) para publicar e assinar mensagens, e dispositivos e clientes que usam o protocolo HTTPS para publicar mensagens. Todos os protocolos suportam IPv4 IPv6 e. Esta seção descreve as diferentes opções de conexão para dispositivos e clientes.

Versões do protocolo TLS

AWS IoT Core usa TLS versão 1.2 e TLS versão 1.3 para criptografar todas as comunicações. Você pode configurar versões adicionais da política de TLS para seu endpoint definindo as configurações

de TLS nas configurações de domínio. Ao conectar dispositivos a clientes AWS IoT Core, os clientes podem enviar a extensão Server Name Indication (SNI), que é necessária para recursos como registro de várias contas, endpoints configuráveis, domínios personalizados e endpoints da VPC. Para obter mais informações, consulte Segurança de transporte no AWS IoT.

Os AWS IoT Dispositivo SDKs oferecem suporte a MQTT e MQTT via WSS e aos requisitos de segurança das conexões do cliente. Recomendamos usar os AWS IoT Dispositivo SDKs para conectar clientes à AWS IoT.

Protocolos, mapeamentos de porta e autenticação

A forma como um dispositivo ou cliente se conecta ao agente de mensagens é configurável usando um tipo de autenticação. Por padrão, ou quando nenhuma extensão SNI é enviada, o método de autenticação é baseado no protocolo de aplicativo, na porta e na extensão TLS de negociação de protocolo da camada de aplicativo (ALPN) que os dispositivos usam. A tabela a seguir lista a autenticação esperada com base na porta, porta e ALPN.

Protocolos, autenticação e mapeamentos de porta

Protocolo	Operações compatíveis	Autenticação	Porta	Nome do protocolo ALPN
Acabou o MQTT WebSocket	Publicar/assinar	Signature versão 4	443	N/D
Acabou o MQTT WebSocket	Publicar/assinar	Autenticação personalizada	443	N/D
MQTT	Publicar/assinar	Certificado do cliente X.509	443†	`x-amzn-mqtt-ca`
MQTT	Publicar/assinar	Certificado do cliente X.509	8883	N/D
MQTT	Publicar/assinar	Autenticação personalizada	443†	`mqtt`
HTTPS	Somente publicação	Signature versão 4	443	N/D

Protocolo	Operações compatíveis	Autenticação	Porta	Nome do protocolo ALPN
HTTPS	Somente publicação	Certificado do cliente X.509	443†	`x-amzn-http-ca`
HTTPS	Somente publicação	Certificado do cliente X.509	8443	N/D
HTTPS	Somente publicação	Autenticação personalizada	443	N/D

Application Layer Protocol Negotiation (ALPN)

† Ao usar configurações de endpoint padrão, os clientes que se conectam na porta 443 com autenticação de certificado de cliente X.509 devem implementar a extensão TLS de negociação de protocolo de camada de aplicativo (ALPN) e usar o nome do protocolo ALPN listado no ALPN enviado pelo cliente como parte da mensagem ProtocolNameList . `ClientHello`
Na porta 443, o endpoint IoT:Data-ATS suporta ALPN x-amzn-http-ca HTTP, mas o endpoint IoT:Jobs não.
Na porta 8443 HTTPS e na porta 443 MQTT com ALPN x-amzn-mqtt-ca, a autenticação personalizada não pode ser usada.

Os clientes se conectam aos endpoints Conta da AWS de seus dispositivos. Consulte the section called “AWS IoT dados do dispositivo e endpoints de serviço” para ver informações sobre como encontrar os endpoints do dispositivo da sua conta.

Note

AWS SDKs não exigem o URL inteiro. Eles exigem apenas o nome do host do endpoint, como a `pubsub.py`amostra do AWS IoT Device SDK for Python on. GitHub Transmitir o URL inteiro conforme fornecido na tabela a seguir pode gerar um erro, como nome de host inválido.

Conectando-se a AWS IoT Core

Protocolo	Endpoint ou URL
MQTT	*iot-endpoint*
MQTT via WSS	wss://*iot-endpoint* /mqtt
HTTPS	https://*iot-endpoint* /topics

Escolher um protocolo de aplicativo para a comunicação do dispositivo

Para a maioria das comunicações de dispositivos de IoT por meio dos endpoints do dispositivo, convém usar os protocolos Secure MQTT ou MQTT over WebSocket Secure (WSS); no entanto, os endpoints do dispositivo também oferecem suporte a HTTPS.

A tabela a seguir compara como AWS IoT Core usa os dois protocolos de alto nível (MQTT e HTTPS) para comunicação com dispositivos.

AWS IoT protocolos de dispositivos (MQTT e HTTPS) side-by-side

Recurso	MQTT	HTTPS
Suporte para publicação/assinatura	Publicação e assinatura	Somente publicação
Compatibilidade com o SDK	AWS O dispositivo SDKs suporta os protocolos MQTT e WSS	Não há suporte para SDK, mas você pode usar métodos específicos de linguagem para fazer solicitações HTTPS
Suporte à qualidade de serviço	Níveis 0 e 1 de QoS do MQTT	A QoS é compatível enviando um parâmetro de string de consulta ?qos=qos em que o valor pode ser 0 ou 1. Você pode adicionar essa string de consulta para publicar uma mensagem com o valor de QoS desejado.

Recurso	MQTT	HTTPS
Pode receber mensagens perdidas enquanto o dispositivo estava off-line	Sim	Não
Suporte a campo de `clientId`	Sim	Não
Detecção de desconexão do dispositivo	Sim	Não
Comunicações seguras	Sim. Consulte ???	Sim. Consulte ???
Definições de tópico	Aplicativo definido	Aplicativo definido
Formato dos dados de mensagem	Aplicativo definido	Aplicativo definido
Sobrecarga do protocolo	Menor	Mais alto
Consumo de energia	Menor	Mais alto

Escolher um tipo de autenticação para a comunicação do seu dispositivo

Você pode configurar o tipo de autenticação para seu endpoint de IoT usando endpoints configuráveis. Como alternativa, use a configuração padrão e determine como seus dispositivos são autenticados com a combinação de protocolo de aplicativo, porta e extensão ALPN TLS. O tipo de autenticação escolhido determina como seus dispositivos serão autenticados ao se conectarem AWS IoT Core. Existem cinco tipos de autenticação:

certificado X.509

Autentique dispositivos usando certificados de cliente X.509, que são AWS IoT Core validados para autenticar o dispositivo. Esse tipo de autenticação funciona com os protocolos Secure MQTT (MQTT over TLS) e HTTPS.

Certificado X.509 com autorizador personalizado

Autentique dispositivos usando certificados de cliente X.509 e execute ações adicionais de autenticação usando um autorizador personalizado, que receberá informações do certificado de cliente X.509. Esse tipo de autenticação funciona com os protocolos Secure MQTT (MQTT over TLS) e HTTPS. Esse tipo de autenticação só é possível usando endpoints configuráveis com autenticação personalizada X.509. Não há opção ALPN.

AWS Signature versão 4 (SigV4)

Autentique dispositivos usando o Cognito ou seu serviço de backend, oferecendo suporte à federação social e corporativa. Esse tipo de autenticação funciona com os protocolos MQTT over WebSocket Secure (WSS) e HTTPS.

Autorizador personalizado

Autentique dispositivos configurando uma função do Lambda para processar as informações de autenticação personalizadas enviadas para o AWS IoT Core. Esse tipo de autenticação funciona com os protocolos Secure MQTT (MQTT over TLS), HTTPS e MQTT over WebSocket Secure (WSS).

Padrão

Autentique dispositivos com base na extensão de negociação de protocolo da camada de and/or aplicativo de porta (ALPN) que os dispositivos usam. Não há suporte para algumas opções adicionais de autenticação. Para obter mais informações, consulte ???.

A tabela abaixo mostra todas as combinações compatíveis de tipos de autenticação e protocolos de aplicativos.

Combinações com suporte de tipos de autenticação e protocolos de aplicativo

Tipo de autenticação	Secure MQTT (MQTT over TLS)	MQTT via WebSocket Secure (WSS)	HTTPS	Padrão
certificado X.509	✓		✓	
Certificado X.509 com autorizador personalizado	✓		✓	
AWS Signature versão 4 (SigV4)		✓	✓	

Tipo de autenticação	Secure MQTT (MQTT over TLS)	MQTT via WebSocket Secure (WSS)	HTTPS	Padrão
Autorizador personalizado	✓	✓	✓	
Padrão				✓

Limites de duração da conexão

Não é garantido que as conexões HTTPS durem mais do que o tempo necessário para receber e responder às solicitações.

A duração da conexão MQTT depende do atributo de autenticação usado. A tabela a seguir lista a duração máxima da conexão em condições ideais para cada atributo.

Duração da conexão MQTT por atributo de autenticação

Recurso	Duração máxima *
Certificado do cliente X.509	1 a 2 semanas
Autenticação personalizada	1 a 2 semanas
Signature versão 4	Até 24 horas

* Não garantido

Com certificados X.509 e autenticação personalizada, a duração da conexão não tem um limite rígido, mas pode ser de apenas alguns minutos. Interrupções de conexão podem ocorrer por vários motivos. A lista a seguir contém alguns dos motivos mais comuns.

- Interrupções na disponibilidade de Wi-Fi
- Interrupções na conexão do provedor de serviços de Internet (ISP)
- Patches de serviço
- Implantações de serviços
- Autoescalabilidade do serviço

- Host de serviço indisponível
- Problemas e atualizações do balanceador de carga
- Erros no lado do cliente,

Seus dispositivos devem implementar estratégias para detectar desconexões e reconectar-se. Para obter mais informações sobre eventos de desconexão e orientações sobre como lidar com eles, consulte ??? em ???.

MQTT

O MQTT (Message Queuing Telemetry Transport) é um protocolo de mensagens leve e amplamente adotado, projetado para dispositivos restritos. O suporte a AWS IoT Core para MQTT é baseado na especificação MQTT v3.1.1 e na especificação MQTT v5.0, com algumas diferenças, conforme documentado em the section called “AWS IoT diferenças das especificações do MQTT”. Como a versão mais recente do padrão, o MQTT 5 apresenta vários recursos principais que tornam um sistema baseado em MQTT mais robusto, incluindo novos aprimoramentos de escalabilidade, relatório de erros aprimorado com respostas de código de motivo, temporizadores de expiração de mensagens e sessões e cabeçalhos de mensagens de usuário personalizados. Para obter mais informações sobre os recursos compatíveis com o MQTT 5, consulte Recursos compatíveis com o MQTT 5. AWS IoT Core AWS IoT Core também suporta comunicação entre versões MQTT (MQTT 3 e MQTT 5). Um publicador do MQTT 3 pode enviar uma mensagem do MQTT 3 para um assinante do MQTT 5 que receberá uma mensagem de publicação do MQTT 5 e vice-versa.

AWS IoT Core suporta conexões de dispositivos que usam o protocolo MQTT e o protocolo MQTT over WSS e que são identificadas por um ID de cliente. O AWS IoT Dispositivo SDKs dá suporte a ambos os protocolos e são as formas recomendadas de conectar dispositivos a AWS IoT Core. O AWS IoT dispositivo SDKs suporta as funções necessárias para que dispositivos e clientes se conectem e acessem AWS IoT os serviços. O dispositivo SDKs suporta os protocolos de autenticação exigidos pelos AWS IoT serviços e os requisitos de ID de conexão exigidos pelos protocolos MQTT e MQTT over WSS. Para obter informações sobre como se conectar ao AWS IoT uso do AWS dispositivo SDKs e links para exemplos AWS IoT nos idiomas suportados, consultethe section called “Conectando-se ao MQTT usando o dispositivo AWS IoT SDKs”. Para obter mais informações sobre métodos de autenticação e os mapeamentos de porta para mensagens MQTT, consulte ???.

Embora seja recomendável usar o AWS IoT dispositivo SDKs para se conectar AWS IoT, eles não são obrigatórios. No entanto SDKs, se você não usar o AWS IoT dispositivo, deverá fornecer a

segurança necessária de conexão e comunicação. Os clientes devem enviar a extensão TLS de Server Name Indication (SNI) na solicitação de conexão. As tentativas de conexão que não incluem o SNI são recusadas. Para obter mais informações, consulte Segurança de transporte em AWS IoT. Clientes que usam usuários e AWS credenciais do IAM para autenticar clientes devem fornecer a autenticação correta do Signature versão 4.

Depois que seus clientes estiverem conectados, você poderá monitorar e gerenciar suas conexões de clientes MQTT usando APIs. Para obter mais informações, consulte the section called “Gerenciar conexões do MQTT”.

Neste tópico:

- Conectando-se ao MQTT usando o dispositivo AWS IoT SDKs
- Opções de Qualidade de serviço (QoS) do MQTT
- Sessões persistentes do MQTT
- Mensagens retidas do MQTT
- Mensagens de Último testamento e Testamento (LWT, Last Will and Testament) do MQTT
- Uso do ConnectAttributes
- Recursos compatíveis com o MQTT 5
- MQTT 5 propriedades
- Códigos de motivo do MQTT
- AWS IoT diferenças das especificações do MQTT
- Gerenciar conexões do MQTT

Conectando-se ao MQTT usando o dispositivo AWS IoT SDKs

Esta seção contém links para o AWS IoT dispositivo SDKs e para o código-fonte de programas de amostra que ilustram como conectar um dispositivo a. AWS IoT Os aplicativos de amostra vinculados aqui mostram como se conectar AWS IoT usando o protocolo MQTT e o MQTT pelo WSS.

> **Note**
>
> O AWS IoT dispositivo lançou SDKs um cliente MQTT 5.

C++

Usando o SDK de dispositivos AWS IoT C++ para conectar dispositivos

- Código-fonte de um aplicativo de amostra que apresenta um exemplo de conexão MQTT em C++
- AWS IoT SDK de dispositivo para C++ v2 ativado GitHub

Python

Usando o AWS IoT Device SDK para Python para conectar dispositivos

- Código-fonte de um aplicativo de amostra que apresenta um exemplo de conexão MQTT em Python
- AWS IoT SDK de dispositivo v2 para Python em GitHub

JavaScript

Usando o AWS IoT Device SDK para JavaScript conectar dispositivos

- Código-fonte de um aplicativo de amostra que mostra um exemplo de conexão MQTT em JavaScript
- AWS IoT SDK do dispositivo para JavaScript v2 ativado GitHub

Java

Usando o AWS IoT Device SDK for Java para conectar dispositivos

> **Note**
>
> O AWS IoT Device SDK for Java v2 agora oferece suporte ao desenvolvimento para Android. Para obter mais informações, consulte SDK de dispositivos AWS IoT para Android.

- Código-fonte de um aplicativo de amostra que apresenta um exemplo de conexão MQTT em Java
- AWS IoT SDK de dispositivo para Java v2 ativado GitHub

Embedded C

Usando o AWS IoT Device SDK for Embedded C para conectar dispositivos

> **Important**
>
> Esse SDK é destinado ao uso por desenvolvedores experientes de software incorporado.

- Código-fonte de um aplicativo de amostra que apresenta um exemplo de conexão MQTT em Embedded C
- AWS IoT SDK de dispositivo para C incorporado em GitHub

Opções de Qualidade de serviço (QoS) do MQTT

AWS IoT e o AWS IoT dispositivo SDKs suporta os níveis de Qualidade de Serviço (QoS) 0 do MQTT e. 1 O protocolo MQTT define um terceiro nível de QoS, 2 nível, AWS IoT mas não o suporta. Apenas o protocolo MQTT dá suporte ao atributo de QoS. HTTPS dá suporte a QoS passando um parâmetro de string de consulta `?qos=qos` em que o valor pode ser 0 ou 1.

Essa tabela descreve como cada nível de QoS afeta as mensagens publicadas para e pelo agente de mensagens.

Com um nível de QoS de...	A mensagem é…	Comentários
QoS nível 0	Enviado zero ou mais vezes	Esse nível deve ser usado para mensagens enviadas por links de comunicação confiáveis ou que podem ser perdidas sem problemas.
QoS nível 1	Enviado pelo menos uma vez e, em seguida, de modo repetido até que uma resposta PUBACK seja recebida	A mensagem não é considera da completa até que o remetente receba uma resposta PUBACK indicando a entrega bem-sucedida.

Sessões persistentes do MQTT

As sessões persistentes armazenam as assinaturas e mensagens de um cliente, com uma Qualidade de Serviço (QoS) de 1, que não foram reconhecidas pelo cliente. Quando o dispositivo se reconecta a uma sessão persistente, a sessão é retomada, as assinaturas são restabelecidas e as mensagens assinadas não confirmadas que são recebidas e armazenadas antes da reconexão são enviadas ao cliente.

O processamento das mensagens armazenadas é registrado em CloudWatch métricas e CloudWatch registros. Para obter informações sobre as entradas gravadas em CloudWatch e CloudWatch Logs, consulte Métricas do agente de mensagens Entrada de log em fila e.

Criar uma sessão persistente

No MQTT 3, você cria uma sessão persistente do MQTT enviando uma mensagem `CONNECT` e definindo o sinalizador `cleanSession` como `0`. Se nenhuma sessão existir para o cliente enviar a mensagem `CONNECT`, uma nova sessão persistente será criada. Se já existir uma sessão para o cliente, o cliente retomará a sessão existente. Para criar uma sessão limpa, você envia uma mensagem `CONNECT` e define o sinalizador `cleanSession` como `1`, e o broker não armazenará nenhum estado da sessão quando o cliente se desconectar.

No MQTT 5, você lida com sessões persistentes definindo o sinalizador `Clean Start` e `Session Expiry Interval`. O Clean Start controla o início da sessão de conexão e o final da sessão anterior. Quando você define `Clean Start` =`1`, uma nova sessão é criada e uma sessão anterior é encerrada, se existir. Quando você define `Clean Start` =`0`, a sessão de conexão retoma uma sessão anterior, se ela existir. O intervalo de expiração da sessão controla o final da sessão de conexão. O intervalo de expiração da sessão especifica o tempo, em segundos (número inteiro de 4 bytes), em que uma sessão persistirá após a desconexão. Configuração `Session Expiry interval` = `0` faz com que a sessão seja encerrada imediatamente após a desconexão. Se o intervalo de expiração da sessão não for especificado na mensagem CONNECT, o padrão será 0.

Início limpo e expiração da sessão do MQTT 5

Valor da propriedade	Description
`Clean Start= 1`	Cria uma sessão e encerra uma sessão anterior, se houver.
`Clean Start= 0`	Retoma uma sessão se existir uma sessão anterior.

Valor da propriedade	Description
`Session Expiry Interval> 0`	Persistir uma sessão.
`Session Expiry interval= 0`	Não persistir uma sessão.

No MQTT 5, se você define `Clean Start = 1` e `Session Expiry Interval =0`, isso equivale a uma sessão limpa do MQTT 3. Se você define `Clean Start = 0` e `Session Expiry Interval> 0`, isso equivale a uma sessão persistente do MQTT 3.

> **Note**
>
> Não há suporte para as sessões persistentes da versão MQTT cruzada (MQTT 3 e MQTT 5). Uma sessão persistente do MQTT 3 não pode ser retomada como uma sessão do MQTT 5 e vice-versa.

Operações durante uma sessão persistente

Os clientes usam o atributo `sessionPresent` na mensagem de conexão confirmada (`CONNACK`) para determinar se uma sessão persistente está presente. Se `sessionPresent` for `1`, uma sessão persistente estará presente e todas as mensagens armazenadas para o cliente serão entregues ao cliente após o cliente receber a `CONNACK`, conforme descrito em Tráfego de mensagens após a reconexão com uma sessão persistente. Se `sessionPresent` for `1`, o cliente não precisará se inscrever novamente. Porém, se `sessionPresent` for `0`, nenhuma sessão persistente estará presente, e o cliente deverá se inscrever novamente nos filtros de tópico.

Depois que o cliente ingressa em uma sessão persistente, ele pode publicar mensagens e assinar filtros de tópico sem quaisquer sinalizadores adicionais em cada operação.

Tráfego de mensagens após a reconexão com uma sessão persistente

Uma sessão persistente representa uma conexão contínua entre um cliente e um agente de mensagens MQTT. Quando um cliente se conecta ao agente de mensagens usando uma sessão persistente, o agente de mensagens salva todas as inscrições que o cliente faz durante a conexão. Quando o cliente se desconecta, o agente de mensagens armazena mensagens de QoS não

confirmadas e novas mensagens de QoS 1 publicadas em tópicos nos quais o cliente está inscrito. As mensagens são armazenadas conforme o limite da conta. As mensagens que excedem o limite serão descartadas. Para obter mais informações sobre os limites de mensagem persistente, consulte endpoints e cotas do AWS IoT Core. Quando o cliente se conecta novamente à sua sessão persistente, todas as inscrições são restabelecidas e todas as mensagens armazenadas são enviadas para o cliente com uma taxa máxima de 10 mensagens por segundo. No MQTT 5, se uma QoS 1 de saída com o intervalo de expiração da mensagem expirar quando um cliente estiver offline, depois que a conexão for retomada, o cliente não receberá a mensagem expirada.

Após a reconexão, as mensagens armazenadas são enviadas ao cliente, a uma taxa limitada a 10 mensagens armazenadas por segundo, junto com qualquer tráfego de mensagens atual até que o limite `Publish requests per second per connection` seja atingido. Como a taxa de entrega das mensagens armazenadas é limitada, serão necessários vários segundos para entregar todas as mensagens armazenadas se uma sessão tiver mais de 10 mensagens armazenadas para entregar após a reconexão.

Para assinantes compartilhados, as mensagens serão colocadas na fila se pelo menos um assinante de um grupo usar uma sessão persistente e nenhum assinante estiver on-line para receber a mensagem de QoS 1. A retirada de mensagens da fila é realizado em uma velocidade máxima de vinte mensagens por segundo por assinante ativo em um grupo. Para acessar mais informações, consulte Enfileiramento de mensagens de assinaturas compartilhadas.

Encerrar uma sessão persistente

As sessões persistentes podem terminar das seguintes maneiras:

- O tempo de expiração persistente da sessão expira. O cronômetro persistente de expiração da sessão começa quando o agente de mensagens detecta que um cliente se desconectou, seja pela desconexão do cliente ou pelo tempo limite da conexão.
- O cliente envia uma mensagem `CONNECT` que define o sinalizador `cleanSession` como `1`.
- Você desconecta manualmente o cliente e limpa a sessão usando a API `DeleteConnection`. Para obter mais informações, consulte the section called “Gerenciar conexões do MQTT”.

No MQTT 3, o valor padrão do tempo de expiração das sessões persistentes é de uma hora, e isso se aplica a todas as sessões na conta.

No MQTT 5, você pode definir o intervalo de expiração da sessão para cada sessão nos pacotes CONNECT e DISCONNECT.

Para o intervalo de expiração da sessão no pacote DISCONNECT:

- Se a sessão atual tiver um intervalo de expiração de sessão de 0, você não poderá definir o intervalo de expiração da sessão como maior que 0 no pacote DISCONNECT.
- Se a sessão atual tiver um intervalo de expiração de sessão maior que 0 e você definir o intervalo de expiração da sessão como 0 no pacote DISCONNECT, a sessão será encerrada em DISCONNECT.
- Caso contrário, o intervalo de expiração da sessão no pacote DISCONNECT atualizará o intervalo de expiração da sessão atual.

> **Note**
>
> As mensagens armazenadas que aguardam para serem enviadas ao cliente quando a sessão termina são descartadas; porém, elas ainda são cobradas conforme a taxa de mensagens padrão, mesmo que não tenham sido enviadas. Para obter mais informações sobre os preços das mensagens, consulte Preços do AWS IoT Core. Você pode configurar o intervalo de tempo de expiração.

Reconexão após a expiração de uma sessão persistente

Se um cliente não se reconectar à sessão persistente antes que ela expire, a sessão terminará e as mensagens armazenadas serão descartadas. Quando um cliente se reconecta depois que a sessão expira com um sinalizador `cleanSession` para `0`, o serviço cria uma nova sessão persistente. As assinaturas ou mensagens da sessão anterior não estão disponíveis para esta sessão porque foram descartadas quando a sessão anterior expirou.

Cobranças persistentes de mensagens de sessão

As mensagens são cobradas de você Conta da AWS quando o agente de mensagens envia uma mensagem para um cliente ou para uma sessão persistente offline. Quando um dispositivo offline com uma sessão persistente se reconecta e retoma a sessão, as mensagens armazenadas são entregues ao dispositivo e cobradas novamente em sua conta. Para obter mais informações sobre a definição de preço de mensagem, consulte Preços do AWS IoT Core – Mensagens.

O tempo de expiração padrão da sessão persistente de uma hora pode ser aumentado usando o processo padrão de aumento de limite. Observe que aumentar o tempo de expiração da sessão

pode aumentar suas cobranças de mensagens, pois o tempo adicional pode permitir que mais mensagens sejam armazenadas no dispositivo off-line e essas mensagens adicionais seriam cobradas em sua conta conforme a taxa de mensagens padrão. O tempo de expiração da sessão é aproximado e uma sessão pode persistir por até 30 minutos a mais do que o limite da conta; porém, uma sessão não será menor que o limite da conta. Para obter mais informações sobre limites de sessão, consulte AWS Service Quotas.

Mensagens retidas do MQTT

AWS IoT Core suporta a `RETAIN` bandeira descrita no protocolo MQTT. Quando um cliente define o `RETAIN` sinalizador em uma mensagem MQTT que ele publica, AWS IoT Core salva a mensagem. Em seguida, ele pode ser enviado para novos assinantes, recuperado chamando a operação `GetRetainedMessage` e visualizado no console do AWS IoT.

Exemplos de uso de mensagens retidas do MQTT

- Como uma mensagem de configuração inicial

  As mensagens retidas do MQTT são enviadas a um cliente depois que o cliente se inscreve em um tópico. Se você quiser que todos os clientes que assinam um tópico recebam a mensagem retida do MQTT logo após a assinatura, publique uma mensagem de configuração com o sinalizador `RETAIN` definido. Os clientes assinantes também recebem atualizações dessa configuração sempre que uma nova mensagem de configuração é publicada.
- Como uma última mensagem de estado conhecida

  Os dispositivos podem definir o sinalizador `RETAIN` nas mensagens do estado atual para que o AWS IoT Core as salve. Quando os aplicativos se conectam ou se reconectam, eles podem se inscrever nesse tópico e obter o último estado relatado logo após se inscreverem no tópico da mensagem retida. Dessa forma, eles podem evitar ter que esperar até a próxima mensagem do dispositivo para ver o estado atual.

Nesta seção:

- Tarefas comuns com mensagens retidas pelo MQTT em AWS IoT Core
- Faturamento e mensagens retidas
- Comparar mensagens retidas do MQTT e sessões persistentes do MQTT
- O MQTT reteve mensagens e AWS IoT sombras do dispositivo

Tarefas comuns com mensagens retidas pelo MQTT em AWS IoT Core

AWS IoT Core salva mensagens MQTT com o `RETAIN` sinalizador definido. Essas mensagens retidas são enviadas a todos os clientes que se inscreveram no tópico, como uma mensagem MQTT normal, e também são armazenadas para serem enviadas aos novos assinantes do tópico.

As mensagens retidas pelo MQTT exigem ações políticas específicas para autorizar os clientes a acessá-las. Para obter exemplos de uso de políticas de mensagens retidas, consulte Exemplos de políticas de mensagens retidas.

Esta seção descreve operações comuns que envolvem mensagens retidas.

- Criar uma mensagem retida

  O cliente determina se uma mensagem é retida quando publica uma mensagem MQTT. Os clientes podem definir o sinalizador `RETAIN` ao publicar uma mensagem usando um SDK de dispositivo. Aplicações e serviços podem definir o sinalizador `RETAIN` quando usam a ação `Publish` para publicar uma mensagem do MQTT.

  Apenas uma mensagem por nome de tópico é retida. Uma nova mensagem com o sinalizador RETAIN definido publicada em um tópico substitui qualquer mensagem retida que tenha sido enviada ao tópico antes.

  > **Note**
  >
  > Você não pode publicar em um tópico reservado com o sinalizador `RETAIN` definido.

- Assinar um tópico de mensagem retida

  Os clientes assinam os tópicos de mensagens retidos como fariam com qualquer outro tópico de mensagem do MQTT. As mensagens retidas recebidas ao assinar um tópico desse tipo têm o sinalizador `RETAIN` definido.

  As mensagens retidas são excluídas AWS IoT Core quando um cliente publica uma mensagem retida com uma carga útil de mensagem de 0 bytes no tópico da mensagem retida. Os clientes que se inscreveram no tópico da mensagem retida também receberão a mensagem de 0 byte.

  Inscrever-se em um filtro de tópico curinga que inclui um tópico de mensagem retida permite que o cliente receba mensagens subsequentes publicadas no tópico da mensagem retida, mas não entrega a mensagem retida após a assinatura.

> **Note**
>
> Para receber uma mensagem retida após a assinatura, o filtro de tópicos na solicitação de assinatura deve corresponder exatamente ao tópico da mensagem retida.

As mensagens retidas recebidas ao assinar um tópico desse tipo têm o sinalizador `RETAIN` definido. As mensagens retidas que são recebidas por um cliente assinante após a assinatura não têm.

- Recuperar uma mensagem retida

  As mensagens retidas são entregues de modo automático aos clientes quando eles se inscrevem no tópico com a mensagem retida. Para que um cliente receba a mensagem retida após a assinatura, ele deve assinar o nome exato do tópico da mensagem retida. Inscrever-se em um filtro de tópico curinga que inclui um tópico de mensagem retida permite que o cliente receba mensagens subsequentes publicadas no tópico da mensagem retida, mas não entrega a mensagem retida após a assinatura.

  Serviços e aplicativos podem listar e recuperar mensagens retidas chamando `ListRetainedMessages` e `GetRetainedMessage`.

  Um cliente não é impedido de publicar mensagens em um tópico de mensagem retida sem definir o sinalizador `RETAIN`. Isso pode causar resultados inesperados, como a mensagem retida não corresponder à mensagem recebida ao se inscrever no tópico.

  Com o MQTT 5, se uma mensagem retida tiver o intervalo de expiração da mensagem definido e a mensagem retida expirar, um novo assinante que assine esse tópico não receberá a mensagem retida após a assinatura bem-sucedida.

- Listando tópicos de mensagens retidas

  Você pode listar as mensagens retidas chamando `ListRetainedMessages` e as mensagens retidas podem ser visualizadas no console do AWS IoT.

- Obter detalhes da mensagem retida

  Você pode obter detalhes das mensagens retidas chamando `GetRetainedMessage` e elas podem ser visualizadas no console do AWS IoT.

- Reter de uma mensagem de Testamento

As mensagens do MQTT de Testamento criadas quando um dispositivo se conecta podem ser retidas definindo o sinalizador `Will Retain` no campo `Connect Flag bits`.

- Excluir uma mensagem retida

  Dispositivos, aplicações e serviços podem excluir uma mensagem retida publicando uma mensagem com o sinalizador `RETAIN` definido e uma carga útil vazia (0 byte) no nome do tópico da mensagem retida a ser excluída. Essas mensagens excluem a mensagem retida de AWS IoT Core, são enviadas aos clientes com uma assinatura do tópico, mas não são retidas por. AWS IoT Core

  As mensagens retidas também podem ser excluídas interativamente acessando a mensagem retida no console do AWS IoT. As mensagens retidas excluídas usando o console do AWS IoT também enviam uma mensagem de 0 byte aos clientes que se inscreveram no tópico da mensagem retida.

  As mensagens retidas não podem ser restauradas após serem excluídas. Um cliente precisaria publicar uma nova mensagem retida para substituir a mensagem excluída.

- Depuração e solução de problemas de mensagens retidas

  O console do AWS IoT fornece várias ferramentas para ajudá-lo a solucionar problemas de mensagens retidas:

  - A página Mensagens retidas

    A página Mensagens retidas no console do AWS IoT fornece uma lista paginada das mensagens retidas armazenadas pela sua Conta na região atual. Nesta página, você pode:

    - Veja os detalhes de cada mensagem retida, como a carga útil da mensagem, o QoS e a hora em que ela foi recebida.
    - Atualize o conteúdo de uma mensagem retida.
    - Exclua uma mensagem retida.

  - O cliente de teste MQTT

    A página do cliente de teste do MQTT no console do AWS IoT pode se inscrever e publicar nos tópicos do MQTT. A opção de publicação permite que você defina o sinalizador RETAIN nas mensagens que você publica para simular como seus dispositivos podem se comportar. Também é possível usar o cliente de teste do MQTT para monitorar mensagens de clientes conectados gerenciados pela interface de conexões de clientes. Para acessar mais informações

sobre o gerenciamento de configurações de clientes, consulte the section called “Gerenciar conexões do MQTT”.

Alguns resultados inesperados podem ser o resultado desses aspectos de como as mensagens retidas são implementadas em AWS IoT Core.

- Limites de mensagens retidas

  Quando uma conta armazena o número máximo de mensagens retidas, AWS IoT Core retorna uma resposta limitada às mensagens publicadas com RETAIN definido e cargas superiores a 0 bytes até que algumas mensagens retidas sejam excluídas e a contagem de mensagens retidas fique abaixo do limite.

- Ordem de entrega de mensagens retidas

  A sequência da mensagem retida e da entrega da mensagem assinada não é garantida.

Faturamento e mensagens retidas

A publicação de mensagens com o sinalizador `RETAIN` definido por um cliente, usando o console da AWS IoT ou chamando `Publish` gera cobranças adicionais de mensagens descritas em Preços do AWS IoT Core : mensagens.

A recuperação de mensagens retidas por um cliente, usando o AWS IoT console ou ligando `GetRetainedMessage`gera cobranças de mensagens, além das cobranças normais de uso da API. As cobranças adicionais estão descritas em AWS IoT Core Preços – Mensagens.

As mensagens de Testamento do MQTT publicadas quando um dispositivo se desconecta inesperadamente geram cobranças de mensagens descritas em AWS IoT Core Preços - Mensagens.

Para obter mais informações sobre custos de mensagens, consulte Preços do AWS IoT Core – Mensagens.

Comparar mensagens retidas do MQTT e sessões persistentes do MQTT

Mensagens retidas e sessões persistentes são recursos padrão do MQTT que possibilitam que os dispositivos recebam mensagens publicadas enquanto eles estavam off-line. As mensagens retidas podem ser publicadas de sessões persistentes. Esta seção descreve os principais aspectos desses recursos e como eles funcionam juntos.

	Mensagens retidas	Sessões persistentes
Recursos principais	As mensagens retidas podem ser usadas para configurar ou notificar grandes grupos de dispositivos após a conexão. As mensagens retidas também podem ser usadas quando você deseja que os dispositivos recebam apenas a última mensagem publicada em um tópico após uma reconexão.	As sessões persistentes são úteis para dispositivos que têm conectividade intermitente e podem perder várias mensagens importantes. Os dispositivos podem se conectar a uma sessão persistente para receber mensagens enviadas enquanto estão off-line.
Exemplos	As mensagens retidas podem fornecer aos dispositivos informações de configuração sobre seu ambiente quando eles ficam on-line. A configuração inicial pode incluir uma lista de outros tópicos de mensagens nos quais ele deve se inscrever ou informações sobre como ele deve configurar seu fuso horário local.	Dispositivos que se conectam por uma rede celular com conectividade intermitente podem usar sessões persistentes para evitar a perda de mensagens importantes enviadas enquanto um dispositivo está fora da cobertura da rede ou precisa desligar o rádio do celular.
Mensagens recebidas na assinatura inicial de um tópico	Depois de assinar um tópico com uma mensagem retida, a mensagem retida mais recente é recebida.	Depois de assinar um tópico sem uma mensagem retida, nenhuma mensagem é recebida até que uma seja publicada no tópico.
Tópicos assinados após a reconexão	Sem uma sessão persistente, o cliente deve se inscrever nos tópicos após a reconexão.	Os tópicos inscritos são restaurados após a reconexão.

	Mensagens retidas	Sessões persistentes
Mensagens recebidas após a reconexão	Depois de assinar um tópico com uma mensagem retida, a mensagem retida mais recente é recebida.	Todas as mensagens publicadas com uma QOS = 1 e assinadas com uma QOS =1 enquanto o dispositivo estava desconectado são enviadas após a reconexão do dispositivo.
Data/expiração da sessão	No MQTT 3, as mensagens retidas não expiram. Elas são armazenados até serem substituídas ou excluídas. No MQTT 5, as mensagens retidas expiram após o intervalo de expiração da mensagem que você definiu. Para obter mais informações, consulte Expiração da mensagem.	As sessões persistentes expiram se o cliente não se reconectar dentro do tempo limite. Depois que uma sessão persistente expira, as assinaturas e mensagens salvas do cliente publicadas com uma QOS = 1 e assinadas com uma QOS = 1 enquanto o dispositivo estava desconectado são excluídas. As mensagens expiradas não serão entregues. Para obter mais informações sobre expirações de sessões persistentes, consulte the section called “Sessões persistentes do MQTT”.

Para obter mais informações sobre sessões persistentes, consulte the section called “Sessões persistentes do MQTT”.

Com as mensagens retidas, o cliente de publicação determina se uma mensagem deve ser retida e entregue a um dispositivo após a conexão, independentemente de ter tido uma sessão anterior ou não. A escolha de armazenar uma mensagem é feita pelo publicador e a mensagem armazenada é entregue a todos os clientes atuais e futuros que usam assinaturas de QoS 0 ou QoS 1. As mensagens retidas mantêm apenas uma mensagem sobre um determinado tópico por vez.

Quando uma conta armazena o número máximo de mensagens retidas, AWS IoT Core retorna uma resposta limitada às mensagens publicadas com RETAIN definido e cargas superiores a 0 bytes até que algumas mensagens retidas sejam excluídas e a contagem de mensagens retidas fique abaixo do limite.

O MQTT reteve mensagens e AWS IoT sombras do dispositivo

Tanto as mensagens retidas quanto as sombras do dispositivo retêm dados de um dispositivo, mas se comportam de maneira diferente e servem a propósitos diferentes. Esta seção descreve suas semelhanças e diferenças.

	Mensagens retidas	Device Shadows
A carga útil da mensagem tem uma estrutura ou esquema predefinido	Conforme definido pela implementação. O MQTT não especifica uma estrutura ou esquema para sua carga de mensagens.	AWS IoT suporta uma estrutura de dados específica.
A atualização da carga útil da mensagem gera mensagens de eventos	A publicação de uma mensagem retida envia a mensagem aos clientes inscritos, mas não gera mensagens de atualização adicionais.	A atualização de uma Sombra do dispositivo produz mensagens de atualização que descrevem a alteração.
As atualizações de mensagens são numeradas	As mensagens retidas não são numeradas automatic amente.	Os documentos de Sombra do dispositivo têm números de versão e carimbos de data e hora automáticos.
A carga útil da mensagem é anexada a um recurso	As mensagens retidas não são anexadas a um recurso.	Sombras do dispositivo são anexadas a um recurso de objeto.
Atualização de elementos individuais da carga útil da mensagem	Elementos individuais da mensagem não podem ser alterados sem atualizar toda a carga útil da mensagem.	Elementos individuais de um documento de Sombra do dispositivo podem ser atualizados sem a necessida

	Mensagens retidas	Device Shadows
		de de atualizar todo o documento de Sombra do dispositivo.
O cliente recebe dados da mensagem no momento da assinatura	O cliente recebe automatic amente uma mensagem retida depois de assinar um tópico com uma mensagem retida.	Os clientes podem assinar as atualizações de Sombra do dispositivo, mas devem solicitar o estado atual deliberadamente.
Indexação e capacidade de pesquisa	As mensagens retidas não são indexadas para pesquisa.	A indexação de frotas indexa dados de Sombra do dispositivo para pesquisa e agregação .

Mensagens de Último testamento e Testamento (LWT, Last Will and Testament) do MQTT

Último testamento e Testamento (LWT, Last Will and Testament) é um atributo do MQTT. Com o LWT, os clientes podem especificar uma mensagem que o corretor publicará em um tópico definido pelo cliente e enviará a todos os clientes que se inscreveram no tópico quando ocorrer uma desconexão não iniciada. A mensagem que os clientes especificam é chamada de mensagem LWT ou Mensagem de testamento, e o tópico que os clientes definem é chamado de Tópico de testamento. Você pode especificar uma mensagem LWT quando um dispositivo se conecta ao agente. Essas mensagens podem ser retidas definindo o sinalizador `Will Retain` no campo `Connect Flag bits` durante a conexão. Por exemplo, se o sinalizador `Will Retain` estiver definido como 1, uma mensagem de testamento será armazenada no corretor no tópico de testamento associado. Para obter mais informações, consulte Mensagens de Will.

Ao gerenciar conexões de clientes, você pode controlar se as mensagens LWT são publicadas quando você desconecta um cliente. Para obter mais informações, consulte the section called “Gerenciar conexões do MQTT”.

O corretor armazenará as mensagens de testamento até que ocorra uma desconexão não iniciada. Quando isso acontecer, o corretor publicará as mensagens para todos os clientes que se inscreveram no Tópico de testamento para notificar a desconexão. Se o cliente se desconectar do agente com uma desconexão iniciada pelo cliente usando a mensagem MQTT DISCONNECT, o

broker não publicará as mensagens LWT armazenadas. Em todos os outros casos, as mensagens LWT serão enviadas. Devido à natureza assíncrona do processamento de desconexão, não é garantido que as mensagens LWT sejam enviadas em ordem durante a reconexão. Recomendamos que você use eventos de ciclo de vida para melhorar a precisão da detecção do estado de conectividade, pois esses eventos fornecem atributos como registros de data e hora e números de versão para gerenciar eventos. out-of-order Para obter uma lista completa dos cenários de desconexão em que o agente enviará as mensagens LWT, confira Eventos de conexão/desconexão.

Uso do ConnectAttributes

O `ConnectAttributes` permite que você especifique quais atributos deseja usar na mensagem de conexão em suas políticas do IAM, como `PersistentConnect` e `LastWill`. Com o `ConnectAttributes`, você pode criar políticas que não dão aos dispositivos acesso a novos recursos por padrão, o que pode ser útil se um dispositivo for comprometido.

O `connectAttributes` oferece suporte aos seguintes recursos:

`PersistentConnect`

Use o atributo `PersistentConnect` para salvar todas as assinaturas que o cliente faz durante a conexão quando a conexão entre o cliente e o corretor é interrompida.

`LastWill`

Use o atributo `LastWill` para publicar uma mensagem no `LastWillTopic` quando um cliente se desconectar inesperadamente.

Por padrão, Sua política tem uma conexão não persistente e não há atributos passados para essa conexão. Você deve especificar uma conexão persistente em sua política do IAM se quiser ter uma.

Ao gerenciar conexões de clientes, você pode visualizar os atributos de conexão e a configuração da sessão para clientes conectados. Para obter mais informações, consulte the section called “Gerenciar conexões do MQTT”.

Para exemplos de `ConnectAttributes`, consulte Exemplos da política de conexão.

Recursos compatíveis com o MQTT 5

AWS IoT Core o suporte para o MQTT 5 é baseado na especificação MQTT v5.0, com algumas diferenças, conforme documentado em. the section called “AWS IoT diferenças das especificações do MQTT”

AWS IoT Core suporta os seguintes recursos do MQTT 5:

- Assinaturas compartilhadas
- Início limpo e expiração da sessão
- Código de motivo em todos ACKs
- Aliases de tópicos
- Validade da mensagem
- Outros recursos do MQTT 5

Assinaturas compartilhadas

AWS IoT Core suporta assinaturas compartilhadas para o MQTT 3.1.1 e o MQTT 5. As assinaturas compartilhadas permitem que vários clientes compartilhem uma assinatura de um tópico e somente um cliente receberá mensagens publicadas nesse tópico usando uma distribuição randomizada. As assinaturas compartilhadas podem efetivamente balancear a carga das mensagens do MQTT entre vários assinantes. Por exemplo, digamos que você tenha 1.000 dispositivos publicando no mesmo tópico e 10 aplicativos de backend processando essas mensagens. Nesse caso, as aplicações de backend podem assinar o mesmo tópico e cada um receberá mensagens publicadas aleatoriamente pelos dispositivos no tópico compartilhado. Isso é efetivamente “compartilhar” a carga dessas mensagens. As assinaturas compartilhadas também permitem uma melhor resiliência. Quando qualquer aplicativo de backend é desconectado, o agente distribui a carga para os assinantes restantes no grupo. Quando todos os assinantes se desconectam, as mensagens ficam na fila.

Os recursos de enfileiramento de mensagens estão disponíveis para assinaturas compartilhadas nas conexões MQTT 3.1.1 e MQTT 5 para melhorar a confiabilidade da entrega de mensagens.

Para usar assinaturas compartilhadas, os clientes assinam um filtro de tópicos de uma assinatura compartilhada da seguinte maneira:

```
$share/{ShareName}/{TopicFilter}
```

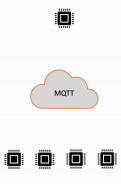
- `$share` é uma string literal para indicar o filtro de tópicos de uma Assinatura Compartilhada, que deve começar com `$share`.
- `{ShareName}` é uma cadeia de caracteres para especificar o nome compartilhado usado por um grupo de assinantes. O filtro de tópicos de uma assinatura compartilhada deve conter um `ShareName` e ser seguido pelo caractere `/`. O `{ShareName}` não deve incluir os seguintes caracteres:`/`, `+` ou `#`. O tamanho máximo de `{ShareName}` é de 128 caracteres UTF-8.

- O `{TopicFilter}` segue a mesma sintaxe que o filtro de tópicos de uma assinatura não compartilhada. O tamanho máximo de `{TopicFilter}` é de 256 caracteres UTF-8.
- As duas barras obrigatórias (`/`) para `$share/{ShareName}/{TopicFilter}` não estão incluídas no limite de Número máximo de barras no tópico e filtro de tópicos.

As assinaturas que têm o mesmo `{ShareName}/{TopicFilter}` pertencem ao mesmo grupo de assinaturas compartilhadas. Você pode criar vários grupos de assinaturas compartilhadas e não exceder o limite de assinaturas compartilhadas por grupo. Para obter mais informações, consulte Endpoints e cotas do AWS IoT Core na Referência geral da AWS .

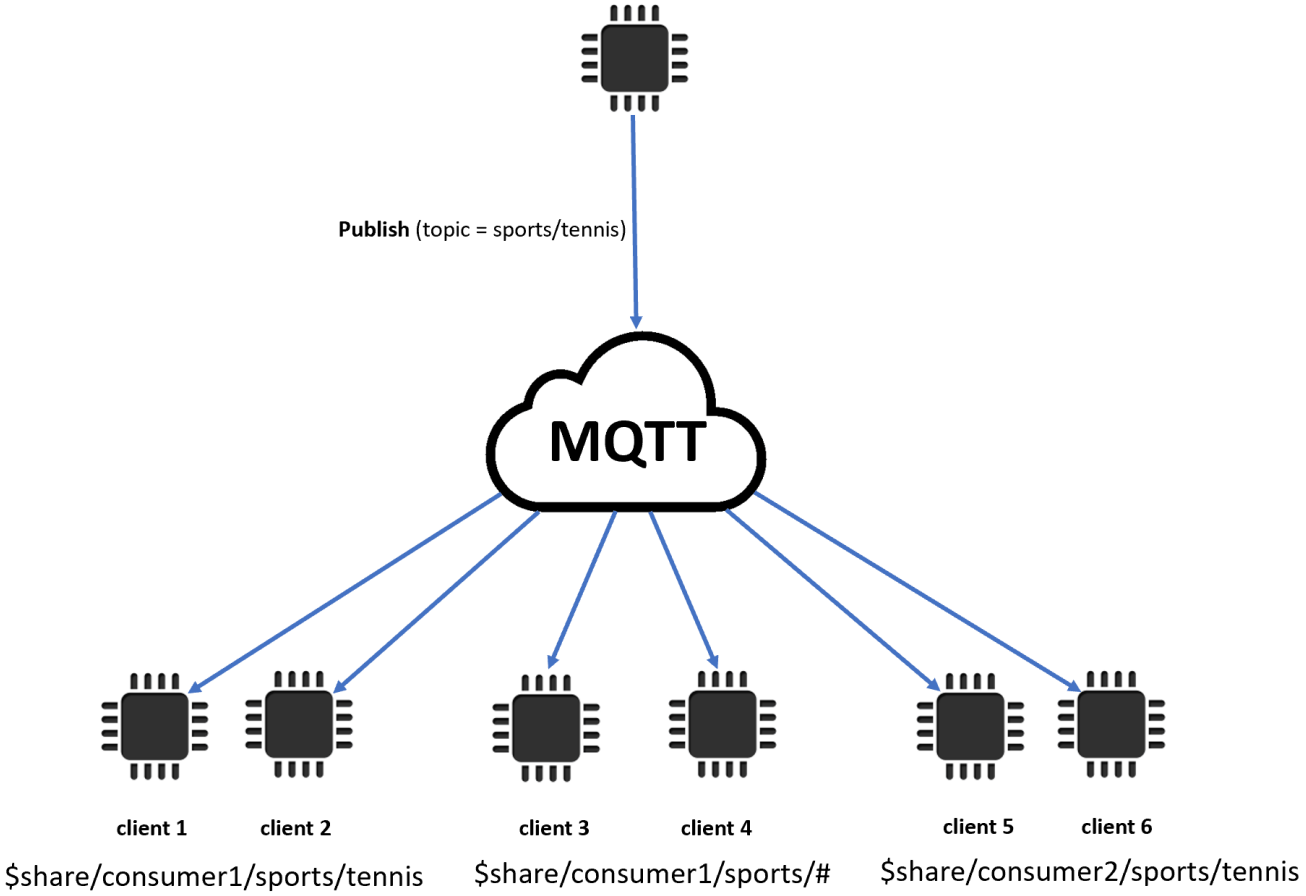
As tabelas a seguir comparam assinaturas não compartilhadas e as assinaturas compartilhadas:

Assinatura	Description	Exemplos de filtro do tópico
Assinaturas não compartilhadas	Cada cliente cria uma assinatura separada para receber as mensagens publicadas. Quando uma mensagem é publicada em um tópico, todos os assinantes desse tópico recebem uma cópia da mensagem.	`sports/tennis` `sports/#`
Assinaturas compartilhadas	Vários clientes podem compartilhar uma assinatura de um tópico e somente um cliente receberá mensagens publicadas nesse tópico em uma distribuição aleatória.	`$share/consumer/sports/tennis` `$share/consumer/sports/#`

Fluxo de assinaturas não compartilhadas	Fluxo de assinaturas compartilhadas		
			

Observações importantes sobre o uso de assinaturas compartilhadas

- Se o grupo de assinantes compartilhado consistir em qualquer assinante de sessão persistente, quando todos os assinantes do grupo compartilhado estiverem desconectados ou se algum assinante violar o limite de solicitações de publicação por segundo por conexão, todas as mensagens de QoS 1 não confirmadas e mensagens de QoS 1 não entregues publicadas em um grupo de assinatura compartilhado serão colocadas em fila. Para acessar mais informações, consulte Enfileiramento de mensagens de assinaturas compartilhadas.
- As mensagens de QoS 0 publicadas em um grupo de assinatura compartilhado serão descartadas em caso de falha.
- As assinaturas compartilhadas não recebem mensagens retidas ao assinar padrões de tópicos como parte de um grupo compartilhado de assinantes. As mensagens publicadas em tópicos com assinantes compartilhados e com o sinalizador `RETAIN` definido são entregues aos assinantes compartilhados como qualquer outra mensagem publicada.
- Quando as assinaturas compartilhadas contêm caracteres curinga (# ou +), pode haver várias assinaturas compartilhadas correspondentes a um tópico. Se isso acontecer, o agente de mensagens copiará a mensagem de publicação e a enviará a um cliente aleatório em cada assinatura compartilhada correspondente. O comportamento curinga das assinaturas compartilhadas pode ser explicado no diagrama a seguir.



Neste exemplo, há três assinaturas compartilhadas correspondentes ao tópico de publicação do MQTT `sports/tennis`. O agente de mensagens copia a mensagem publicada e a envia para um cliente aleatório em cada grupo correspondente.

O cliente 1 e o cliente 2 compartilham a assinatura: `$share/consumer1/sports/tennis`

O cliente 3 e o cliente 4 compartilham a assinatura: `$share/consumer1/sports/#`

O cliente 5 e o cliente 6 compartilham a assinatura: `$share/consumer2/sports/tennis`

Para acessar mais informações sobre limites de assinaturas compartilhadas, consulte AWS IoT Core endpoints and quotas na Referência geral da AWS . Para testar assinaturas compartilhadas usando o cliente AWS IoT MQTT no AWS IoT console, consulte. ??? Você também pode ver em quais tópicos os clientes conectados assinaram, incluindo assinaturas compartilhadas, usando os recursos de gerenciamento de conexão do cliente. Para obter mais informações, consulte the section called

"Gerenciar conexões do MQTT". Para obter mais informações sobre assinaturas compartilhadas, consulte Assinaturas compartilhadas da especificação 4.0. MQTTv5

Enfileiramento de mensagens de assinaturas compartilhadas

Para aumentar a confiabilidade da entrega de mensagens, as assinaturas compartilhadas incluem recursos de enfileiramento de mensagens que armazenam mensagens quando não há assinantes on-line disponíveis. Quando um grupo de assinatura compartilhado contém pelo menos um membro com uma sessão persistente, o recurso de enfileiramento é habilitado para o grupo. Ao distribuir mensagens, os membros on-line são selecionados como destinatários. As mensagens de QoS 1 são colocadas na fila quando nenhum membro é encontrado on-line ou quando os assinantes excedem o limite `Publish requests per second per connection`. As mensagens em fila são entregues quando os membros existentes retomam suas sessões persistentes ou quando novos membros ingressam no grupo. As mensagens em fila são entregues em até vinte mensagens em fila por segundo por assinante do grupo ativo, junto com quaisquer outras mensagens entregues ao assinante de acordo com as assinaturas.

Por padrão, a retenção de mensagens em fila segue a cota `Persistent Session expiry period`. No entanto, se um intervalo de expiração de mensagens (MEI) for definido na mensagem de publicação de entrada, o MEI terá precedência. Quando o MEI está presente, ele determina o período de retenção da mensagem, independentemente do período de expiração da sessão persistente.

As taxas de fila de mensagens são limitadas de acordo com a cota `Queued messages per second per account`, e o número de mensagens é limitado pela cota `Maximum number of queued messages per shared subscription group`. Para visualizar e gerenciar suas cotas, acesse o console Service Quotas.

Você pode monitorar a fila CloudWatch pesquisando no `AWS/Usage` namespace ou pode usar o seguinte comando da CLI para `ApproximateQueueDepth` listar as métricas associadas à profundidade da fila de cada grupo de assinaturas compartilhadas.

```
aws cloudwatch list-metrics --namespace "AWS/Usage" --dimensions
 Name=Resource,Value='ApproximateQueueDepth'
```

Quando esses limites são excedidos, somente as mensagens que já estavam na fila antes de atingir o limite são retidas. Novas mensagens recebidas que excederiam os limites são descartadas. O sistema não substitui mensagens antigas em fila por mensagens mais novas.

O enfileiramento de mensagens é registrado em CloudWatch métricas e CloudWatch registros. Para obter informações sobre as entradas gravadas em CloudWatch e CloudWatch Logs, consulte Métricas do agente de mensagens Entrada de log em fila e. Ainda ocorre cobrança pelas mensagens em fila de acordo com a taxa de mensagens padrão. Para obter mais informações sobre os preços das mensagens, consulte Preços do AWS IoT Core.

Ciclo de vida da sessão em um grupo de assinatura compartilhado

Quando uma sessão limpa assina um grupo, ela se torna um membro on-line do grupo. Quando cancela a assinatura ou se desconecta, a sessão limpa sai do grupo.

Quando uma sessão persistente assina um grupo, ela se torna um membro on-line do grupo. Quando ela se desconecta, ainda permanece no grupo, mas se torna um membro offline do grupo. Quando ela se reconecta, se torna um membro on-line novamente. A sessão persistente deixa o grupo quando ela cancela explicitamente a assinatura ou quando expira após a desconexão.

Início limpo e expiração da sessão

Você pode usar o Clean Start e o Session Expiry para lidar com suas sessões persistentes com mais flexibilidade. Um sinalizador Clean Start indica se a sessão deve começar sem usar uma sessão existente. O intervalo de expiração da sessão indica por quanto tempo manter a sessão após uma desconexão. O intervalo de expiração da sessão pode ser modificado na desconexão. Para obter mais informações, consulte the section called “Sessões persistentes do MQTT”.

Código de motivo em todos ACKs

Você pode depurar ou processar mensagens de erro com mais facilidade usando os códigos de motivo. Os códigos de motivo são retornados pelo agente de mensagens com base no tipo de interação com o corretor (Assinar, Publicar, Confirmar). Para obter mais informações, consulte Códigos de motivo do MQTT. Para obter uma lista completa dos códigos de motivo do MQTT, consulte a especificação do MQTT v5.

Aliases de tópicos

Você pode substituir o nome de um tópico por um alias de tópico, que é um número inteiro de dois bytes. O uso de aliases de tópicos pode otimizar a transmissão de nomes de tópicos para reduzir potencialmente os custos de dados em serviços de dados medidos. AWS IoT Core tem um limite padrão de 8 aliases de tópicos. Para obter mais informações, consulte Endpoints e cotas do AWS IoT Core na Referência geral da AWS .

Validade da mensagem

Você pode adicionar valores de expiração de mensagens às mensagens publicadas. Esses valores representam o intervalo de expiração da mensagem (MEI) em segundos. Se as mensagens não tiverem sido enviadas aos assinantes dentro desse intervalo, a mensagem expirará e será removida. Se você não definir o valor de expiração da mensagem, ela não expirará.

Na saída, o assinante receberá uma mensagem com o tempo restante no intervalo de expiração. Por exemplo, se uma mensagem de publicação de entrada tiver uma expiração de 30 segundos e for encaminhada para o assinante após 20 segundos, o campo de expiração da mensagem será atualizado para 10. É possível que a mensagem recebida pelo assinante tenha um MEI atualizado de 0. Isso porque, assim que o tempo restante for de 999 ms ou menos, ele será atualizado para 0.

Em AWS IoT Core, o MEI mínimo é 1. Se o intervalo for definido como 0 do lado do cliente, ele será ajustado para 1. O intervalo máximo de expiração da mensagem é 604800 (7 dias). Qualquer valor maior que isso será ajustado para o valor máximo.

Na comunicação entre versões, o comportamento da expiração da mensagem é decidido pela versão MQTT da mensagem de publicação de entrada. Por exemplo, uma mensagem com expiração de mensagem enviada por uma sessão conectada via MQTT5 pode expirar para dispositivos inscritos com sessões. MQTT3 A tabela abaixo mostra como a expiração de mensagens oferece suporte aos seguintes tipos de mensagens publicadas:

Publicar tipo de mensagem	Intervalo de expiração da mensagem
Publicação regular	Se um servidor não entregar a mensagem dentro do prazo especificado, a mensagem expirada será removida e o assinante não a receberá. Isso inclui situações como quando um dispositivo não está fazendo backup de suas mensagens do QoS 1.
Reter	Se uma mensagem retida expirar e um novo cliente se inscrever no tópico, o cliente não receberá a mensagem após a assinatura.
Último testamento	O intervalo para as mensagens de último testamento começa depois que o cliente se desconecta e o servidor tenta entregar a mensagem de último testamento aos assinantes.

Publicar tipo de mensagem	Intervalo de expiração da mensagem
Mensagens na fila	Se uma QoS 1 de saída com intervalo de expiração de mensagens expirar quando um cliente estiver offline, após a retomada da sessão persistente, o cliente não receberá a mensagem expirada.

Outros recursos do MQTT 5

Desconexão do servidor

Quando ocorre uma desconexão, o servidor pode enviar proativamente ao cliente um DISCONNECT para notificar o encerramento da conexão com um código de motivo para a desconexão.

Resposta/solicitação

Os editores podem solicitar que uma resposta seja enviada pelo destinatário para um tópico especificado pelo publicador no momento do recebimento.

Tamanho máximo do pacote

O cliente e o servidor podem especificar de modo independente o tamanho máximo do pacote ao qual eles dão suporte.

Formato de carga útil e tipo de conteúdo

Você pode especificar o formato da carga útil (binário, texto) e o tipo de conteúdo quando uma mensagem é publicada. Eles são encaminhados para o destinatário da mensagem.

MQTT 5 propriedades

As propriedades do MQTT 5 são adições importantes ao padrão MQTT para oferecer suporte aos novos recursos do MQTT 5, como a expiração da sessão e o padrão. Request/Response Em AWS IoT Core, você pode criar regras que podem encaminhar as propriedades em mensagens de saída ou usar HTTP Publish para publicar mensagens MQTT com algumas das novas propriedades.

A tabela a seguir lista todas as propriedades do MQTT 5 que AWS IoT Core suportam.

Propriedade	Description	Tipo de entrad	Pacote
Indicador de formato da carga útil	Um valor booliano que indica se a carga útil está formatada como UTF-8.	Byte	PUBLICAR, CONECTAR
Tipo de conteúdo	Uma string UTF-8 que descreve o conteúdo da carga útil.	String UTF-8	PUBLICAR, CONECTAR
Tópico de resposta	Uma string UTF-8 que descreve o tópico que o destinatário deve publicar como parte do fluxo solicitação-resposta. O tópico não deve conter caracteres curinga.	String UTF-8	PUBLICAR, CONECTAR
Dados de correlação	Os dados binários usados pelo remetente da mensagem de solicitação para identificar para qual solicitação é a mensagem de resposta.	Binário	PUBLICAR, CONECTAR
Propriedades do usuário	Um par de strings UTF-8. Essa propriedade pode aparecer várias vezes em um pacote. Os destinatários receberão os pares de chave-valor na mesma ordem em que são enviados.	Par de strings UTF-8	CONECTAR, PUBLICAR, Propriedades, ASSINAR, DESCONECTAR, REMOVER ASSINATURA
Intervalo de expiração da mensagem	Um valor inteiro de 4 bytes que representa o intervalo de expiração da mensagem em segundos. Se ausente, a mensagem não expira.	inteiro de 4 bytes	PUBLICAR, CONECTAR
Intervalo de expiração da sessão	Um número inteiro de 4 bytes que representa o intervalo de expiração da sessão em segundos. AWS IoT Core suporta no máximo 7 dias, com um máximo padrão de uma hora. Se o valor definido exceder o máximo da	inteiro de 4 bytes	CONECTAR, CONNACK, DESCONECTAR

Propriedade	Description	Tipo de entrad	Pacote
	sua conta, AWS IoT Core retornará o valor ajustado no CONNACK.		
Identificador de cliente atribuído	Um ID de cliente aleatório gerado AWS IoT Core quando um ID de cliente não é especific ado pelos dispositivos. O ID aleatório do cliente deve ser um novo identificador de cliente que não seja usado por nenhuma outra sessão atualmente gerenciada pelo corretor.	String UTF-8	CONNACK
Servidor Keep Alive	Um número inteiro de 2 bytes que represent a o tempo de manutenção de atividade atribuído pelo servidor. O servidor desconect ará o cliente se o cliente ficar inativo por mais do que o tempo de manutenção de atividade.	Inteiro de 2 bytes	CONNACK
Solicitar informaçõ es sobre o problema	Um valor booliano que indica se a string do motivo ou as propriedades do usuário são enviadas no caso de falhas.	Byte	CONECTAR
Máximo de recebimento	Um inteiro de 2 bytes que representa o número máximo de pacotes PUBLICAR QOS > 0 que podem ser enviados sem receber um PUBACK.	Inteiro de 2 bytes	CONNECT, CONNACK
Tópico: máximo do alias	Esse valor indica o valor mais alto que será aceito como um alias de tópico. O padrão é 0.	Inteiro de 2 bytes	CONNECT, CONNACK
QoS máximo	O valor máximo de QoS que AWS IoT Core suporta. O padrão é 1. O AWS IoT Core não comporta QoS 2.	Byte	CONNACK

Propriedade	Description	Tipo de entrad	Pacote
Reter disponível	Um valor booleano que indica se o agente de AWS IoT Core mensagens suporta mensagens retidas. O padrão é um.	Byte	CONNACK
Tamanho máximo do pacote	O tamanho máximo do pacote que AWS IoT Core aceita e envia. Não pode exceder 128 KB.	inteiro de 4 bytes	CONNECT, CONNACK
Assinatura de curinga disponível	Um valor booleano que indica se o agente de AWS IoT Core mensagens oferece suporte à assinatura Wildcard Available. O padrão é um.	Byte	CONNACK
Identificador de assinatura disponível	Um valor booleano que indica se o agente de AWS IoT Core mensagens oferece suporte ao Identificador de Assinatura Disponível. O padrão é 0.	Byte	CONNACK

Códigos de motivo do MQTT

O MQTT 5 introduz relatórios de erros aprimorados com respostas de código de motivo. AWS IoT Core pode retornar códigos de motivo, incluindo, mas não se limitando aos seguintes, agrupados por pacotes. Para obter uma lista completa dos códigos de motivo com suporte no MQTT 5, confira especificações do MQTT v5.

Códigos de motivo do CONNACK

Valor	Hex	Nome do código de motivo	Description
0	0x00	Bem-sucedida	A conexão é criptografada.
128	0x80	Erro não especificado	O servidor não deseja revelar o motivo da falha, ou nenhum dos outros códigos de motivo se aplica.

Valor	Hex	Nome do código de motivo	Description
133	0x85	O identificador de cliente não é válido	O identificador do cliente é uma string válida, mas não é permitido pelo servidor.
134	0x86	Nome de usuário ou senha incorretos	O servidor não aceita o nome de usuário ou a senha especificados pelo cliente.
135	0x87	Não autorizado	O cliente não está autorizado a se conectar.
144	0x90	Nome do tópico inválido	O nome do tópico do testamento está formado corretame nte, mas não é aceito pelo servidor.
151	0x97	Cota excedida	Um limite imposto administrativo ou de implementação foi excedido.
155	0x9B	Não há suporte ao QoS	O servidor não dá suporte a QoS definida na QoS do Testamento.

Códigos de motivo PUBACK

Valor	Hex	Nome do código de motivo	Description
0	0x00	Bem-sucedida	A mensagem foi aceita. A publicação da mensagem de QoS 1 continua.
128	0x80	Erro não especificado	O destinatário não aceita a publicação, mas não quer revelar o motivo ou ele não corresponde a um dos outros valores.
135	0x87	Não autorizad o	A ação PUBLICAR não é autorizada.

Valor	Hex	Nome do código de motivo	Description
144	0x90	Nome do tópico inválido	O nome do tópico não está mal formado, mas não é aceito pelo cliente ou servidor.
145	0x91	Identificador de pacote em uso	O identificador do pacote já está em uso. Isso pode indicar uma incompatibilidade no estado da sessão entre o cliente e o servidor.
151	0x97	Cota excedida	Um limite imposto administrativo ou de implementação foi excedido.

Código do motivo da ação DESCONECTAR

Valor	Hex	Nome do código de motivo	Description
129	0x81	Pacote malformado	O pacote recebido não está em conformidade com essa especificação.
130	0x82	Erro de protocolo	Um pacote inesperado ou fora de ordem foi recebido.
135	0x87	Não autorizado	A solicitação não está autorizada.
139	0x8B	Encerramento do servidor	O servidor está sendo desligado.
141	0x8D	Tempo limite do Keep Alive	A conexão está fechada porque nenhum pacote foi recebido por 1,5 veze o tempo de Keep Alive.
142	0x8E	Sessão retomada	Outra conexão usando o mesmo ClientID foi conectada, fazendo com que essa conexão seja fechada.

Valor	Hex	Nome do código de motivo	Description
143	0x8F	Filtro de tópicos inválido	O filtro de tópicos está formado corretamente, mas não é aceito pelo servidor.
144	0x90	Nome do tópico inválido	O nome do tópico está formado corretamente, mas não é aceito por esse cliente ou servidor.
147	0x93	Máximo de recebimento excedido	O cliente ou o servidor recebeu mais do que a publicação Máximo de recebimento para a qual não enviou PUBACK ou PUBCOMP.
148	0x94	Aliases do tópico inválido	O cliente ou servidor recebeu um pacote PUBLICAR contendo um alias de tópico maior que o alias de tópico máximo enviado no pacote CONECTAR ou CONNACK.
151	0x97	Cota excedida	Um limite imposto administrativo ou de implementação foi excedido.
152	0x98	Ação administrativa	A conexão foi encerrada devido a uma ação administrativa.
155	0x9B	Não há suporte ao QoS	O cliente especificou uma QoS maior que a QoS especificada em uma QoS máxima no CONNACK.
161	0xA1	Identificadores de assinatura sem suporte	O servidor não tem suporte para identificadores de assinatura; a assinatura não é aceita.

Códigos de motivo SUBACK

Valor	Hex	Nome do código de motivo	Description
0	0x00	QoS concedida 0	A assinatura é aceita e a QoS máxima enviada será QoS 0. Pode ser uma QoS menor do que a solicitada.
1	0x01	QoS concedida 1	A assinatura é aceita e a QoS máxima enviada será QoS 1. Pode ser uma QoS menor do que a solicitada.
128	0x80	Erro não especificado	A assinatura não é aceita e o Servidor não deseja revelar o motivo ou nenhum dos outros Códigos de Motivo se aplica.
135	0x87	Não autorizado	O Cliente não está autorizado a fazer essa assinatura.
143	0x8F	Filtro de tópicos inválido	O Filtro de tópicos está formado corretamente, mas não é permitido para este Cliente.
145	0x91	Identificador de pacote em uso	O identificador do pacote já está em uso.
151	0x97	Cota excedida	Um limite imposto administrativo ou de implementação foi excedido.

Códigos de motivo UNSUBACK

Valor	Hex	Nome do código de motivo	Description
0	0x00	Bem-sucedida	A assinatura é excluída.
128	0x80	Erro não especificado	Não foi possível concluir a assinatura e o Servidor não deseja revelar o motivo ou nenhum dos outros Códigos de Motivo se aplica.
143	0x8F	Filtro de tópicos inválido	O Filtro de tópicos está formado corretamente, mas não é permitido para este Cliente.

Valor	Hex	Nome do código de motivo	Description
145	0x91	Identificador de pacote em uso	O identificador do pacote já está em uso.

AWS IoT diferenças das especificações do MQTT

A implementação do agente de mensagens é baseada na especificação MQTT v3.1.1 e na Especificação MQTT v5.0, mas difere das especificações das seguintes maneiras:

- AWS IoT não suporta os seguintes pacotes para o MQTT 3: PUBREC, PUBREL e PUBCOMP.
- AWS IoT não suporta os seguintes pacotes para o MQTT 5: PUBREC, PUBREL, PUBCOMP e AUTH.
- AWS IoT não suporta o redirecionamento do servidor MQTT 5.
- AWS IoT suporta somente os níveis 0 e 1 de qualidade de serviço (QoS) do MQTT. AWS IoT não suporta publicação ou assinatura com QoS nível 2. Quando o nível 2 da QoS é solicitado, o agente de mensagens do não envia um PUBACK ou SUBACK.
- Em AWS IoT, assinar um tópico com QoS nível 0 significa que uma mensagem é entregue zero ou mais vezes. Uma mensagem pode ser entregue mais de uma vez. As mensagens entregues mais de uma vez podem ser enviadas com outro ID de pacote. Nesses casos, o sinalizador DUP não é definido.
- Ao responder a uma solicitação de conexão, o agente de mensagens envia uma mensagem CONNACK. Essa mensagem contém um sinalizador para indicar se a conexão está retomando uma sessão anterior.
- Antes de enviar pacotes de controle adicionais ou uma solicitação de desconexão, o cliente deve esperar que a mensagem CONNACK seja recebida em seu dispositivo pelo agente de mensagens AWS IoT .
- Quando um cliente se inscreve em um tópico, pode haver uma demora entre o momento em que o operador envia uma mensagem SUBACK e o momento em que o cliente começa a receber novas mensagens correspondentes.
- Quando um cliente usa o caractere curinga # no filtro de tópicos para assinar um tópico, há correspondência de todas as sequências de caracteres em e abaixo de seu nível na hierarquia de tópicos. Porém, o tópico principal não corresponde. Por exemplo, uma assinatura do tópico `sensor/#` recebe mensagens publicadas para os tópicos `sensor/`, `sensor/temperature`,

`sensor/temperature/room1`, mas não as mensagens publicadas para `sensor`. Para obter mais informações sobre curingas, consulte Filtros de nomes de tópicos.

- O agente de mensagens usa o ID do cliente para identificar cada cliente. O ID do cliente é transmitido do cliente para o agente de mensagens como parte da carga útil do MQTT. Dois clientes com o mesmo ID de cliente não podem ser conectados simultaneamente ao agente de mensagens. Quando um cliente se conecta ao agente de mensagens usando um ID que outro cliente está usando, a nova conexão do cliente é aceita e o cliente conectado anteriormente é desconectado. Você também pode desconectar manualmente os clientes usando o. APIs Para obter mais informações, consulte the section called “Gerenciar conexões do MQTT”.
- Em raras ocasiões, o agente de mensagens pode reenviar a mesma mensagem lógica PUBLICAR com um ID de pacote diferente.
- A assinatura de filtros de tópicos que contêm um caractere curinga não pode receber mensagens retidas. Para receber uma mensagem retida, a solicitação de assinatura deve conter um filtro de tópico que corresponda exatamente ao tópico da mensagem retida.
- O agente de mensagens não garante a ordem em que as mensagens e o ACK são recebidos.
- AWS IoT podem ter limites diferentes das especificações. Para obter mais informações, consulte agente de mensagens AWS IoT Core e limites e cotas do protocolo no Guia de referência do AWS IoT .
- Não há suporte para o sinalizador MQTT DUP.

Gerenciar conexões do MQTT

AWS IoT Core fornece ajuda APIs para gerenciar conexões MQTT, incluindo a capacidade de desconectar clientes e gerenciar suas sessões. Esses recursos oferecem mais controle sobre sua frota de AWS IoT clientes e ajudam na solução de problemas de conexão.

DeleteConnection API

Use a `DeleteConnection` API para desconectar dispositivos MQTT AWS IoT Core especificando seu cliente. IDs Quando você desconecta um cliente, AWS IoT Core desconecta o cliente do agente de mensagens e, opcionalmente, pode limpar o estado da sessão e suprimir a AWS IoT Core mensagem Last Will and Testament (LWT).

Quando você chama a `DeleteConnection` API, AWS IoT Core executa várias ações para garantir uma desconexão limpa. AWS IoT Core primeiro envia uma mensagem de desconexão do MQTT ao cliente para encerrar a sessão do MQTT. Em seguida, o serviço fecha o TCP/TLS soquete subjacente.

O agente de mensagens envia um pacote `DISCONNECT` ao dispositivo e publica um evento de ciclo de vida com o motivo da desconexão `API_INITIATED_DISCONNECT`. Isso ajuda a identificar quando uma desconexão foi iniciada por meio da API e não pelo cliente ou devido a problemas de rede. Você pode monitorar esses eventos para fins de visibilidade, solução de problemas e auditoria. Por exemplo, você pode usar AWS IoT regras para processar esses eventos para rastrear quando e por que os clientes foram desconectados.

Se você definir o `cleanSession` parâmetro como`true`, AWS IoT Core remove o estado da sessão do cliente, incluindo todas as assinaturas e mensagens em fila. Se você limpar uma sessão, a sessão persistente será encerrada. Se o cliente tiver uma sessão persistente e o parâmetro `preventWillMessage` estiver definido como `false`, o serviço enviará a mensagem LWT, se disponível, o que é útil durante as operações de manutenção planejadas.

Quando você chama a API `DeleteConnection`, o processo de desconexão começa imediatamente, mas o momento exato em que o cliente reconhece a desconexão pode variar com base nas condições da rede e na implementação do cliente. A maioria dos clientes detectará a desconexão em alguns segundos, mas, em alguns casos, com conectividade de rede inadequada, pode levar mais tempo para que o cliente reconheça que foi desconectado.

> **Note**
>
> Ao forçar a desconexão, o cliente precisa se autenticar e reautorizar com um novo estado de sessão. A chamada de API em si não impede a reconexão dos clientes. Se isso for desejado, além disso, a credencial ou a política do cliente deverão ser modificadas antes de emitir a chamada de API `DeleteConnection`.

Para obter mais informações sobre preços, consulte Preços do AWS IoT Core.

Casos de uso

A API `DeleteConnection` é útil para gerenciar clientes que apresentam comportamento problemático ou consomem recursos excessivos. Ao forçar a desconexão, você pode garantir que os clientes restabeleçam suas conexões com autenticação e autorização adequadas, o que pode ajudar a resolver problemas de consumo de recursos.

Os cenários de redirecionamento de clientes também se beneficiam dessa API. Quando você precisa redirecionar clientes para endpoints diferentes ou Regiões da AWS pode desconectá-los programaticamente e fazer com que eles se reconectem a um endpoint diferente AWS IoT Core

alterando as configurações de DNS. Essa API pode ajudar a resolver conexões bloqueadas ou eliminar estados de sessão problemáticos que podem estar impedindo a operação normal.

Parâmetros de API

A API `DeleteConnection` aceita os seguintes parâmetros:

clientId (obrigatório)

O identificador exclusivo do cliente do MQTT a ser desconectado. Isso é especificado no caminho do URL. O ID do cliente não pode começar com cifrão ($).

> **Note**
>
> O cliente MQTT IDs pode conter caracteres que não são válidos em solicitações HTTP. Ao usar a API `DeleteConnection`, você deve codificar em URL (codificação percentual) quaisquer caracteres no ID do cliente que sejam válidos no MQTT, mas não em HTTP. Isso inclui caracteres especiais, como espaços, barras (/) e caracteres UTF-8. Por exemplo, um espaço se torna %20, uma barra invertida se torna %2F e o caractere UTF-8 ü se torna %C3%BC. A codificação adequada garante que o cliente IDs MQTT seja transmitido corretamente na chamada de API baseada em HTTP.

cleanSession (opcional)

Especifica se o estado da sessão do cliente deve ser removido ao se desconectar. Defina como `true` para excluir todas as informações da sessão, incluindo assinaturas e mensagens em fila. Defina como `false` para preservar o estado da sessão. Por padrão, isso é definido como `false` (preserva o estado da sessão). Para sessões limpas, esse parâmetro será ignorado.

preventWillMessage (opcional)

Controla se a mensagem AWS IoT Core Last Will and Testament (LWT) é enviada se estiver disponível após a desconexão. Defina como `true` para evitar o envio da mensagem LWT. Defina como `false` para permitir o envio. Por padrão, isso é definido como `false` (envia o LWT, se disponível).

Sintaxe da API

A API `DeleteConnection` usa o seguinte formato de solicitação HTTP:

```
DELETE /connections/<clientId>?
cleanSession=<cleanSession>&preventWillMessage=<preventWillMessage> HTTP/1.1
```

Exemplo de solicitações:

```
// Basic disconnect (preserves session, allows LWT message)
DELETE /connections/myDevice123 HTTP/1.1

// Disconnect and clear session
DELETE /connections/myDevice123?cleanSession=TRUE HTTP/1.1

// Disconnect, clear session, and prevent LWT message
DELETE /connections/myDevice123?cleanSession=TRUE&preventWillMessage=TRUE HTTP/1.1
```

Solicitações bem-sucedidas exibem HTTP 200 OK sem corpo de resposta.

> **Note**
>
> O nome do serviço usado pelo AWS Signature Version 4 para assinar solicitações é: iotdevicegateway. Você pode encontrar o endpoint usando o comando `aws iot describe-endpoint --endpoint-type iot:Data-ATS`.

Permissões obrigatórias

Para usar a API `DeleteConnection`, é necessário ter a seguinte permissão do IAM:

```
iot:DeleteConnection
```

Você pode definir essa permissão para um cliente específico IDs usando políticas baseadas em recursos. Por exemplo:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:DeleteConnection",
            "Resource": "arn:aws:iot:region:account:client/myDevice*"
        }
```

```
    ]
}
```

Considerações importantes

Clientes desconectados podem tentar se reconectar imediatamente, a menos que você tenha implementado uma lógica adicional para evitar a reconexão. A operação de desconexão apenas encerra a conexão atual e não impede que uma conexão se estabeleça novamente. Se você precisar evitar a reconexão, pense em implementar a lógica do lado do cliente ou desabilitar a credencial de um dispositivo.

Os limites de taxa se aplicam à API como parte da limitação de taxa padrão da AWS IoT Core API. Ao planejar operações de desconexão em massa, leve em conta esses limites e implemente a lógica de nova tentativa e as estratégias de agrupamento em lote apropriadas para evitar o controle de utilização. Para saber mais, consulte Endpoints e cotas do AWS IoT Core.

Respostas de erro

A API `DeleteConnection` pode exibir as seguintes respostas de erro:

InvalidRequestException

A solicitação não é válida. Isso poderá ocorrer se o formato do ID do cliente for inválido, contiver um prefixo de cifrão ($) ou se os parâmetros necessários estiverem ausentes.

ResourceNotFoundException

O ID do cliente especificado não existe ou não está conectado no momento e não tem uma sessão persistente.

UnauthorizedException

Você não tem autorização para executar esta operação. Garanta que você tenha as permissões `iot:DeleteConnection` necessárias.

ForbiddenException

O chamador não está autorizado a fazer a solicitação. Isso pode ocorrer devido a permissões insuficientes do IAM ou restrições de políticas baseadas em recursos.

ThrottlingException

A taxa excede o limite. Reduza a frequência de suas chamadas de API e implemente a lógica de nova tentativa apropriada com recuo exponencial.

InternalFailureException

Ocorreu um erro inesperado. Tente fazer a solicitação novamente após um breve atraso.

ServiceUnavailableException

O serviço está temporariamente indisponível. Tente fazer a solicitação novamente após um breve atraso.

Publicação HTTPS

Os clientes podem publicar mensagens fazendo solicitações para a API REST usando os protocolos HTTP 1.0 ou 1.1. Para os mapeamentos de porta e autenticação usados por solicitações HTTP, consulte ???.

> **Note**
>
> O HTTPS é compatível com um valor de `clientId` como o MQTT. O `clientId` está disponível ao usar o MQTT, mas não está disponível ao usar HTTPS.

URL da mensagem HTTPS

Dispositivos e clientes publicam suas mensagens fazendo solicitações POST para um endpoint específico do cliente e um URL específico do tópico:

```
https://IoT_data_endpoint/topics/url_encoded_topic_name?qos=1
```

- *IoT_data_endpoint*é o endpoint de dados do AWS IoT dispositivo. Você pode encontrar o endpoint no AWS IoT console na página de detalhes da coisa ou no cliente usando o AWS CLI comando:

  ```
  aws iot describe-endpoint --endpoint-type iot:Data-ATS
  ```

  O endpoint deve ser algo parecido com isto: `a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com`
- *url_encoded_topic_name*é o nome completo do tópico da mensagem que está sendo enviada.

Exemplos de código de mensagem HTTPS

Estes são alguns exemplos de como enviar uma mensagem HTTPS à AWS IoT.

Python (port 8443)

```
import requests
import argparse

# define command-line parameters
parser = argparse.ArgumentParser(description="Send messages through an HTTPS
 connection.")
parser.add_argument('--endpoint', required=True, help="Your AWS IoT data custom
 endpoint, not including a port. " +
                                                       "Ex: \"abcdEXAMPLExyz-
ats.iot.us-east-1.amazonaws.com\"")
parser.add_argument('--cert', required=True, help="File path to your client
 certificate, in PEM format.")
parser.add_argument('--key', required=True, help="File path to your private key, in
 PEM format.")
parser.add_argument('--topic', required=True, default="test/topic", help="Topic to
 publish messages to.")
parser.add_argument('--message', default="Hello World!", help="Message to publish. "
 +
                                                       "Specify empty string to
 publish nothing.")

# parse and load command-line parameter values
args = parser.parse_args()

# create and format values for HTTPS request
publish_url = 'https://' + args.endpoint + ':8443/topics/' + args.topic + '?qos=1'
publish_msg = args.message.encode('utf-8')

# make request
publish = requests.request('POST',
            publish_url,
            data=publish_msg,
            cert=[args.cert, args.key])

# print results
print("Response status: ", str(publish.status_code))
if publish.status_code == 200:
```

```
        print("Response body:", publish.text)
```

Python (port 443)

```
import requests
import http.client
import json
import ssl

ssl_context = ssl.SSLContext(protocol=ssl.PROTOCOL_TLS_CLIENT)
ssl_context.minimum_version = ssl.TLSVersion.TLSv1_2

# note the use of ALPN
ssl_context.set_alpn_protocols(["x-amzn-http-ca"])
ssl_context.load_verify_locations(cafile="./<root_certificate>")

# update the certificate and the AWS endpoint
ssl_context.load_cert_chain("./<certificate_in_PEM_Format>",
 "<private_key_in_PEM_format>")
connection = http.client.HTTPSConnection('<the ats IoT endpoint>', 443,
 context=ssl_context)
message = {'data': 'Hello, I'm using TLS Client authentication!'}
json_data = json.dumps(message)
connection.request('POST', '/topics/device%2Fmessage?qos=1', json_data)

# make request
response = connection.getresponse()

# print results
print(response.read().decode())
```

CURL

Você pode usar o curl de um cliente ou dispositivo para enviar uma mensagem para a AWS IoT.

Para usar o curl para enviar uma mensagem de um dispositivo AWS IoT cliente

1. Verifique a versão do curl.

   a. No cliente, execute esse comando em um prompt de comando.

      curl --help

No texto de ajuda, procure as opções de TLS. Você deve ver a opção `--tlsv1.2`.

b. Se você vir a opção `--tlsv1.2`, continue.

c. Se você não vir a opção `--tlsv1.2` ou receber um erro `command not found`, pode ser necessário atualizar ou instalar o curl no cliente ou instalar o `openssl` antes de continuar.

2. Instale os certificados no cliente.

   Copie os arquivos de certificado que você criou quando registrou seu cliente (coisa) no AWS IoT console. Verifique se você tem esses três arquivos de certificado no cliente antes de continuar.

   - O arquivo de certificado CA (*`Amazon-root-CA-1.pem`* neste exemplo).
   - O arquivo de certificado do cliente (*`device.pem.crt`* neste exemplo).
   - O arquivo de chave privada do cliente (*`private.pem.key`* neste exemplo).

3. Crie a linha de comando curl, substituindo os valores substituíveis pelos da sua conta e do sistema.

```
curl --tlsv1.2 \
    --cacert Amazon-root-CA-1.pem \
    --cert device.pem.crt \
    --key private.pem.key \
    --request POST \
    --data "{ \"message\": \"Hello, world\" }" \
    "https://IoT_data_endpoint:8443/topics/topic?qos=1"
```

--tlsv1.2

Use TLS 1.2 (SSL).

--cacert *`Amazon-root-CA-1.pem`*

O nome e o caminho do arquivo, se necessário, do certificado CA para verificar o peer.

--certificado *`device.pem.crt`*

O nome e o caminho do arquivo de certificado do cliente, se necessário.

--chave *`private.pem.key`*

O nome e o caminho do arquivo de chave privada do cliente, se necessário.

--solicitação POST

O tipo de solicitação HTTP (nesse caso, POST).

--dados "" *{ \"message\": \"Hello, world\" }*

Os dados de HTTP POST que você deseja publicar. Nesse caso, é uma string JSON, com as aspas internas escapadas com o caractere de barra invertida (\).

“https: *IoT_data_endpoint* //:8443/topics/? *topic* qos = 1"

A URL do endpoint de dados do AWS IoT dispositivo do seu cliente, seguida pela porta HTTPS`:8443`, seguida pela palavra-chave `/topics/` e pelo nome do tópico`topic`, nesse caso. Especifique a qualidade do serviço como o parâmetro de consulta, `?qos=1`.

4. Abra o cliente de teste MQTT no AWS IoT console.

   Siga as instruções Exibir mensagens MQTT com o cliente AWS IoT MQTT e configure o console para assinar mensagens com o nome do tópico *topic* usado em seu curl comando ou use o filtro de tópico curinga de#.

5. Teste o comando.

   Ao monitorar o tópico no cliente de teste do console do AWS IoT , acesse o cliente e emita a linha de comando curl criada na etapa 3. Você deve ver as mensagens do cliente no console.

Tópicos do MQTT

Os tópicos do MQTT identificam AWS IoT mensagens. AWS IoT os clientes identificam as mensagens que publicam dando nomes de tópicos às mensagens. Os clientes identificam as mensagens que desejam assinar (receber) registrando um filtro de tópico com o AWS IoT Core. O agente de mensagens usa nomes de tópicos e filtros de tópicos para rotear mensagens de clientes de publicação a clientes assinatura.

O agente de mensagens usa tópicos para identificar mensagens enviadas usando MQTT e HTTP para o URL da mensagem HTTPS.

Embora AWS IoT ofereça suporte a alguns tópicos reservados do sistema, a maioria dos tópicos do MQTT é criada e gerenciada por você, o designer do sistema. AWS IoT usa tópicos para identificar mensagens recebidas de clientes de publicação e selecionar mensagens para enviar aos clientes assinantes, conforme descrito nas seções a seguir. Antes de criar um namespace de tópico para seu

sistema, revise as características dos tópicos MQTT para criar a hierarquia de nomes de tópicos que funciona melhor para seu sistema de IoT.

Nomes de tópicos

Os nomes de tópicos e os filtros de tópicos são strings codificadas em UTF-8. Eles podem representar uma hierarquia de informações usando o caractere barra (/) para separar os níveis da hierarquia. Por exemplo, o nome deste tópico pode se referir a um sensor de temperatura no compartimento 1:

- `sensor/temperature/room1`

Neste exemplo, também pode haver outros tipos de sensores em outros compartimentos com nomes de tópicos, como:

- `sensor/temperature/room2`
- `sensor/humidity/room1`
- `sensor/humidity/room2`

> **Note**
>
> Ao considerar os nomes de tópicos para as mensagens no sistema, lembre-se de que:
>
> - Os nomes de tópicos e os filtros de tópico diferenciam letras maiúsculas de minúsculas.
> - Os nomes de tópicos não devem conter informações de identificação pessoal.
> - Os nomes de tópicos que começam com $ são tópicos reservados para serem usados somente pelo AWS IoT Core.
> - AWS IoT Core não consigo enviar ou receber mensagens entre Conta da AWS s ou regiões.

Para obter mais informações sobre como criar seus nomes de tópicos e namespace, consulte nosso whitepaper, Projetando tópicos em MQTT para AWS IoT Core.

Para exemplos de como os aplicativos podem publicar e assinar mensagens, comece com Introdução aos AWS IoT Core tutoriais e AWS IoT Device SDKs, Mobile SDKs e AWS IoT Device Client.

> **Important**
>
> O namespace do tópico é limitado a uma região Conta da AWS e. Por exemplo, o `sensor/temp/room1` tópico usado por um Conta da AWS em uma região é diferente do `sensor/temp/room1` tópico usado pela mesma AWS conta em outra região ou usado por qualquer outra Conta da AWS em qualquer região.

Tópico ARN

Todos os tópicos ARNs (Amazon Resource Names) têm o seguinte formato:

```
arn:aws:iot:aws-region:AWS-account-ID:topic/Topic
```

Por exemplo, `arn:aws:iot:us-west-2:123EXAMPLE456:topic/application/topic/device/sensor` é um ARN para o tópico `application/topic/device/sensor`.

Filtros de nomes de tópicos

Os clientes assinantes registram filtros de tópico no agente de mensagens a fim de especificar os tópicos que o agente de mensagens deve enviar a eles. Um filtro de nome de tópico pode corresponder a apenas um tópico ou incluir caracteres curinga para permitir a assinatura de vários nomes de tópico simultaneamente.

Os clientes de publicação não podem usar caracteres curinga nos nomes de tópicos que publicam.

A tabela a seguir lista os caracteres curinga que podem ser usados em um filtro de tópico.

Curingas de tópicos

Caractere curinga	Correspondências	Observações
#	Todas as strings de caracteres em e abaixo de seu nível na hierarquia de tópicos.	Deve ser o último caractere no filtro de tópico. Deve ser o único caractere no nível da hierarquia de tópicos.

Caractere curinga	Correspondências	Observações
		Pode ser usado em um filtro de tópico que também contém o caractere curinga +.
+	Qualquer string no nível que contém o caractere.	Deve ser o único caractere no nível da hierarquia de tópicos. Pode ser usado em vários níveis de um filtro de tópico.

Exemplos de uso de curingas com o nome de tópico do sensor anterior:

- Uma assinatura de `sensor/#` recebe mensagens publicadas em `sensor/`, `sensor/temperature`, `sensor/temperature/room1`, mas não mensagens publicadas em `sensor`.
- Uma assinatura de `sensor/+/room1` recebe mensagens publicadas em `sensor/temperature/room1` e `sensor/humidity/room1`, mas não mensagens enviadas para `sensor/temperature/room2` ou `sensor/humidity/room2`.

ARN do filtro do tópico

Todos os filtros de tópicos ARNs (Amazon Resource Names) têm o seguinte formato:

```
arn:aws:iot:aws-region:AWS-account-ID:topicfilter/TopicFilter
```

Por exemplo, `arn:aws:iot:us-west-2:123EXAMPLE456:topicfilter/application/topic/+/sensor` é um ARN para o filtro de tópicos `application/topic/+/sensor`.

Carga útil da mensagem do MQTT

A carga útil da mensagem enviada em suas mensagens do MQTT não é especificada por AWS IoT, a menos que seja para uma das. the section called “Tópicos reservados” Para acomodar as necessidades do seu aplicativo, recomendamos que você defina a carga útil da mensagem para seus tópicos dentro das restrições das AWS IoT Core Service Quotas para protocolos.

Usar um formato JSON para sua carga de mensagens permite que o mecanismo de AWS IoT regras analise suas mensagens e aplique consultas SQL a elas. Se seu aplicativo não precisar

do mecanismo de regras para aplicar consultas SQL às cargas úteis de mensagens, você poderá usar qualquer formato de dados que seu aplicativo exija. Para obter informações sobre limitações e caracteres reservados em um documento JSON usado em consultas SQL, consulte Extensões JSON.

Para obter mais informações sobre como criar seus tópicos do MQTT e suas cargas de mensagens correspondentes, consulte Como criar tópicos do MQTT para AWS IoT Core.

Se o limite de tamanho de uma mensagem exceder as service quotas, isso resultará em um `CLIENT_ERROR` motivo `PAYLOAD_LIMIT_EXCEEDED` e é exibida a mensagem “A carga útil da mensagem excede o limite de tamanho para o tipo de mensagem”. Para obter mais informações sobre o limite de tamanho de mensagens, consulte AWS IoT Core limites e cotas do agente de mensagens.

Tópicos reservados

Os tópicos que começam com um cifrão ($) são reservados para uso por AWS IoT. É possível assinar e publicar nesses tópicos reservados conforme permitido. No entanto, não é possível criar tópicos que comecem com um cifrão. Operações de publicação ou assinatura sem suporte em tópicos reservados podem resultar no encerramento de uma conexão.

Tópicos de modelos de ativos

Tópico	Operações do cliente permitidas	Description
$ aws/sitewise/asset - models/ /assets/ /properties/ *assetModelId assetId propertyId*	Assinar	AWS IoT SiteWise publica notificações de proprieda des de ativos neste tópico. Para obter mais informaçõ es, consulte Interagindo com outros AWS serviços no Guia do AWS IoT SiteWise usuário.

AWS IoT Device Defender tópicos

Essas mensagens oferecem suporte a buffers de resposta no formato Concise Binary Object Representation (CBOR) e em JavaScript Object Notation (JSON), dependendo do tópico. *payload-format* AWS IoT Device Defender os tópicos suportam somente a publicação em MQTT.

payload-format	Tipo de dados do formato de resposta
cbor	Representação Concisa de Objetos Binários (CBOR)
json	JavaScript Notação de objeto (JSON)

Para obter mais informações, consulte Enviar métricas de dispositivos.

Tópico	Operações permitidas	Description
thingName $aws/things/ /defender/metrics/ *payload-format*	Publicar	AWS IoT Device Defender agentes publicam métricas para esse tópico. Para obter mais informações, consulte Enviar métricas de dispositivos.
$aws/things/ /defender /metrics/ /accepted *thingName* *payload-format*	Assinar	AWS IoT publica neste tópico depois que um AWS IoT Device Defender agente publica uma mensagem bem-sucedida em *thingName* $aws/things/ /defender /metrics/. *payload-format* Para obter mais informações, consulte Enviar métricas de dispositivos.
$aws/things/ /defender /metrics/ /rejeitado *thingName* *payload-format*	Assinar	AWS IoT publica neste tópico depois que um AWS IoT Device Defender agente publica uma mensagem malsucedida em $aws/things/ /defender/metrics/. *thingName* *payload-format* Para obter mais informações, consulte Enviar métricas de dispositivos.

AWS IoT Core Tópicos de localização do dispositivo

AWS IoT Core A localização do dispositivo pode resolver os dados de medição do seu dispositivo e fornecer uma localização estimada dos seus dispositivos de IoT. Os dados de medição do dispositivo podem incluir GNSS, Wi-Fi, celular e endereço IP. AWS IoT Core Em seguida, a localização do dispositivo escolhe o tipo de medição que fornece a melhor precisão e resolve as informações de localização do dispositivo. Para obter mais informações, consulte AWS IoT Core Localização do dispositivo e Resolvendo a localização do dispositivo usando a localização do AWS IoT Core dispositivo (tópicos do MQTT).

Tópico	Operações permitidas	Description
$ aws/device_location/ / get_position_estimate *customer_device_id*	Publicar	Um dispositivo publica este tópico para que os dados brutos de medição digitalizados sejam resolvidos pela localização AWS IoT Core do dispositivo.
$ aws/device_location/ / get_position_estimate/ accepted *customer_device_id*	Assinar	AWS IoT Core A localização do dispositivo é publicada neste tópico depois de resolver a localização do dispositivo com êxito.
$ aws/device_location/ / get_position_estimate/ rejeitado *customer_device_id*	Assinar	AWS IoT Core A localização do dispositivo é publicada neste tópico quando não é possível resolver a localização do dispositivo com êxito devido a erros 4xx.

Tópicos de eventos

As mensagens de evento são publicadas quando determinados eventos acontecem. Por exemplo, os eventos são gerados pelo registro quando as objetos são adicionadas, atualizadas ou excluídas. A tabela mostra os vários AWS IoT eventos e seus tópicos reservados.

Tópico	Operações do cliente permitidas	Description
$aws/events/certificates/registered/*caCertificateId*	Assinar	AWS IoT publica essa mensagem ao registrar AWS IoT automaticamente um certificado e quando um cliente apresenta um certificado com o `PENDING_ACTIVATION` status. Para obter mais informações, consulte the section called "Configurar a primeira conexão feita por um cliente para registro automático".
$aws/events/job/*jobID*/cancelado	Assinar	AWS IoT publica essa mensagem quando um trabalho é cancelado. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/job/*jobID*/cancelamento_em_andamento	Assinar	AWS IoT publica essa mensagem quando um cancelamento de trabalho está em andamento. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/job/*jobID*/concluído	Assinar	AWS IoT publica essa mensagem quando um trabalho é concluído. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/job/*jobID*/excluído	Assinar	AWS IoT publica essa mensagem quando um trabalho é excluído. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/job/*jobID*/deletion_in_progress	Assinar	AWS IoT publica essa mensagem quando a exclusão de um trabalho está em andamento. Para obter mais informações, consulte Eventos de trabalho.

Tópico	Operações do cliente permitidas	Description
$aws/events/jobExecution/*jobID*/cancelado	Assinar	AWS IoT publica essa mensagem quando a execução de um trabalho é cancelada. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/jobExecution/*jobID*/excluído	Assinar	AWS IoT publica essa mensagem quando uma execução de trabalho é excluída. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/jobExecution/*jobID*/falhou	Assinar	AWS IoT publica essa mensagem quando a execução de um trabalho falha. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/jobExecution/*jobID*/rejeitado	Assinar	AWS IoT publica essa mensagem quando a execução de um trabalho é rejeitada. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/jobExecution/*jobID*/removido	Assinar	AWS IoT publica essa mensagem quando a execução de um trabalho é removida. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/jobExecution/*jobID*/foi bem sucedido	Assinar	AWS IoT publica essa mensagem quando a execução de um trabalho é bem-sucedida. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/jobExecution/*jobID*/timed_out	Assinar	AWS IoT publica essa mensagem quando a execução de um trabalho atinge o tempo limite. Para obter mais informações, consulte Eventos de trabalho.

Tópico	Operações do cliente permitidas	Description
$aws/events/presence/connected/*clientId*	Assinar	AWS IoT publica neste tópico quando um cliente MQTT com o ID do cliente especificado se conecta a. AWS IoT Para obter mais informações, consulte Eventos de conexão/desconexão.
$aws/events/presence/disconnected/*clientId*	Assinar	AWS IoT publica neste tópico quando um cliente MQTT com a ID de cliente especificada se desconecta a. AWS IoT Para obter mais informações, consulte Eventos de conexão/desconexão.
$aws/events/subscriptions/subscribed/*clientId*	Assinar	AWS IoT publica neste tópico quando um cliente MQTT com o ID de cliente especificado se inscreve em um tópico do MQTT. Para obter mais informações, consulte Eventos de assinatura/cancelamento de assinatura.
$aws/events/subscriptions/unsubscribed/*clientId*	Assinar	AWS IoT publica neste tópico quando um cliente MQTT com o ID de cliente especificado cancela a assinatura de um tópico MQTT. Para obter mais informações, consulte Eventos de assinatura/cancelamento de assinatura.
$aws/events/thing/*thingName* /criado	Assinar	AWS IoT publica neste tópico quando a *thingName* coisa é criada. Para obter mais informações, consulte the section called “Eventos de registro”.
$aws/events/thing/*thingName* /atualizado	Assinar	AWS IoT publica neste tópico quando a *thingName* coisa é atualizada. Para obter mais informações, consulte the section called “Eventos de registro”.

Tópico	Operações do cliente permitidas	Description
$aws/events/thing/ *thingName* /excluído	Assinar	AWS IoT publica neste tópico quando a *thingName* coisa é excluída. Para obter mais informações, consulte the section called “Eventos de registro”.
$aws/events/thingGroup/*thingGroupName* /criado	Assinar	AWS IoT publica neste tópico quando um grupo de coisas *thingGroupName* é criado. Para obter mais informações, consulte the section called “Eventos de registro”.
$aws/events/thingGroup/*thingGroupName* /atualizado	Assinar	AWS IoT publica neste tópico quando o grupo de coisas *thingGroupName* é atualizado. Para obter mais informações, consulte the section called “Eventos de registro”.
$aws/events/thingGroup/*thingGroupName* /excluído	Assinar	AWS IoT publica neste tópico quando um grupo de coisas *thingGroupName* é excluído. Para obter mais informações, consulte the section called “Eventos de registro”.
$aws/events/thingType/*thingTypeName* /criado	Assinar	AWS IoT publica neste tópico quando o tipo de *thingTypeName* coisa é criado. Para obter mais informações, consulte the section called “Eventos de registro”.
$aws/events/thingType/*thingTypeName* /atualizado	Assinar	AWS IoT publica neste tópico quando o tipo de *thingTypeName* coisa é atualizado. Para obter mais informações, consulte the section called “Eventos de registro”.

Tópico	Operações do cliente permitidas	Description
$aws/events/thingType/*thingTypeName*/excluído	Assinar	AWS IoT publica neste tópico quando o tipo de *thingTypeName* coisa é excluído. Para obter mais informações, consulte the section called “Eventos de registro”.
$aws/events/thingTypeAssociation/thing/*thingName*/*thingTypeName*	Assinar	AWS IoT publica neste tópico quando algo *thingName* está associado ou desassociado do tipo de coisa. *thingTypeName* Para obter mais informações, consulte the section called “Eventos de registro”.
$aws/events/thingGroupMembership/thingGroup/*thingGroupName*/thing/*thingName*/adicionado	Assinar	AWS IoT publica neste tópico quando algo *thingName* é adicionado ao grupo *thingGroupName* de coisas. Para obter mais informações, consulte the section called “Eventos de registro”.
$aws/events/thingGroupMembership/thingGroup/*thingGroupName*/thing/*thingName*/removido	Assinar	AWS IoT publica neste tópico quando algo *thingName* é removido do grupo *thingGroupName* de coisas. Para obter mais informações, consulte the section called “Eventos de registro”.
$aws/events/thingGroupHierarchy/thingGroup//*parentThingGroupName* childThingGroup/*childThingGroupName* /adicionado	Assinar	AWS IoT publica neste tópico quando um grupo de coisas *childThingGroupName* é adicionado ao grupo *parentThingGroupName* de coisas. Para obter mais informações, consulte the section called “Eventos de registro”.

Tópico	Operações do cliente permitidas	Description
$aws/events/thingGroupHierarchy/thingGroup//*parentThingGroupName* childThingGroup/*childThingGroupName* /removido	Assinar	AWS IoT publica neste tópico quando o grupo de coisas *childThingGroupName* é removido do grupo *parentThingGroupName* de coisas. Para obter mais informações, consulte the section called “Eventos de registro”.

Tópicos de provisionamento de frota

> **Note**
>
> As operações do cliente anotadas como Receber nesta tabela indicam tópicos que AWS IoT são publicados diretamente no cliente que os solicitou, independentemente de o cliente ter se inscrito no tópico ou não. Os clientes devem esperar receber essas mensagens de resposta mesmo que não sejam assinantes delas. Essas mensagens de resposta não passam pelo agente de mensagens e não podem ser assinadas por outros clientes ou regras.

Essas mensagens oferecem suporte a buffers de resposta no formato Concise Binary Object Representation (CBOR) e em JavaScript Object Notation (JSON), dependendo do tópico. *payload-format*

payload-format	Tipo de dados do formato de resposta
cbor	Representação Concisa de Objetos Binários (CBOR)
json	JavaScript Notação de objeto (JSON)

Para obter mais informações, consulte API MQTT de provisionamento de dispositivos.

Tópico	Operações do cliente permitidas	Description
$aws/certificates/create/*payload-format*	Publicar	Publique nesse tópico para criar um certificado usando uma solicitação de assinatura de certificado (CSR).
$aws/certificates/create/*payload-format* /aceito	Assine, receba	AWS IoT publica neste tópico após uma chamada bem-sucedida para $aws/certificates/create/*payload-format* .
$aws/certificates/create/*payload-format* /rejeitado	Assine, receba	AWS IoT publica neste tópico após uma chamada malsucedida para $aws/certificates/create/. *payload-format*
$ aws/certificates/create-de-csr/ *payload-format*	Publicar	Publica nesse tópico para criar um certificado a partir de uma CSR.
$ aws/certificates/create *payload-format* -from-csr/ /aceito	Assine, receba	AWS IoT publica neste tópico uma chamada bem-sucedida para $ aws/certificates/create *payload-format* -from-csr/.
$ aws/certificates/create *payload-format* -from-csr/ /rejeitado	Assine, receba	AWS IoT publica neste tópico uma chamada malsucedida para $ aws/certificates/create -from-csr/. *payload-format*
$aws/provisioning-templates/ /provision/ *templateName* *payload-format*	Publicar	Publique nesse tópico para registrar um objeto.
$aws/provisioning-templates/ /provision/ /accepted *templateName* *payload-format*	Assine, receba	AWS IoT publica neste tópico após uma chamada bem-sucedida para *templateName* $aws/provisioning-templates/ /provision/. *payload-format*

Tópico	Operações do cliente permitidas	Description
$aws/provisioning-templates/ /provision/ / rejeitado *templateName* *payload-format*	Assine, receba	AWS IoT publica neste tópico após uma chamada malsucedida para *templateName* $aws/provisioning-templates/ / provision/. *payload-format*

Tópicos de trabalhos

> **Note**
>
> As operações do cliente anotadas como Receber nesta tabela indicam tópicos que AWS IoT são publicados diretamente no cliente que os solicitou, independentemente de o cliente ter se inscrito no tópico ou não. Os clientes devem esperar receber essas mensagens de resposta mesmo que não sejam assinantes delas.
> Essas mensagens de resposta não passam pelo agente de mensagens e não podem ser assinadas por outros clientes ou regras. Para assinar mensagens relacionadas a atividades de trabalhos, use os tópicos `notify` e `notify-next`.
> Ao assinar os tópicos de tarefas e `jobExecution` eventos de sua solução de monitoramento de frota, você deve primeiro habilitar os eventos de tarefas e execução de tarefas para receber quaisquer eventos no lado da nuvem.
> Para obter mais informações, consulte Operações da API MQTT do dispositivo de trabalhos.

Tópico	Operações do cliente permitidas	Description
$ *thingName* aws/things/ /jobs/get	Publicar	Os dispositivos publicam uma mensagem neste tópico para fazer uma solicitação `GetPendingJobExecutions` . Para obter mais informações, consulte Operações da API MQTT do dispositivo de trabalhos.

Tópico	Operações do cliente permitidas	Description
$ aws/coisas//*thingName* jobs/get/accepted	Assine, receba	Os dispositivos se inscrevem nesse tópico para receber respostas bem-sucedidas de uma solicitação `GetPendingJobExecutions` . Para obter mais informações, consulte Operações da API MQTT do dispositivo de trabalhos.
$ aws/coisas//*thingName* jobs/get/rejected	Assine, receba	Os dispositivos assinam este tópico para receber uma resposta quando uma solicitação `GetPendingJobExecutions` é rejeitada. Para obter mais informações, consulte Operações da API MQTT do dispositivo de trabalhos.
thingName $aws/things/ / jobs/start-next	Publicar	Os dispositivos publicam uma mensagem neste tópico para fazer uma solicitação `StartNextPendingJobExecution` . Para obter mais informações, consulte Operações da API MQTT do dispositivo de trabalhos.
$ aws/coisas//*thingName* jobs/start-next/accepted	Assine, receba	Os dispositivos assinam este tópico para receber respostas bem-sucedidas para uma solicitação `StartNextPendingJobExecution` . Para obter mais informações, consulte Operações da API MQTT do dispositivo de trabalhos.

Tópico	Operações do cliente permitidas	Description
$ aws/coisas//*thingName* jobs/start-next/rejected	Assine, receba	Os dispositivos assinam este tópico para receber uma resposta quando uma solicitação `StartNextPendingJobExecution` é rejeitada. Para obter mais informações, consulte Operações da API MQTT do dispositivo de trabalhos.
$aws/things/ /jobs/ /get *thingName* *jobId*	Publicar	Os dispositivos publicam uma mensagem neste tópico para fazer uma solicitação `DescribeJobExecution`. Para obter mais informações, consulte Operações da API MQTT do dispositivo de trabalhos.
$aws/things/ /jobs/ /get/accepted *thingName* *jobId*	Assine, receba	Os dispositivos assinam este tópico para receber respostas bem-sucedidas para uma solicitação `DescribeJobExecution`. Para obter mais informações, consulte Operações da API MQTT do dispositivo de trabalhos.
$aws/things/ /jobs/ /get/rejeitado *thingName* *jobId*	Assine, receba	Os dispositivos assinam este tópico para receber uma resposta quando uma solicitação `DescribeJobExecution` é rejeitada. Para obter mais informações, consulte Operações da API MQTT do dispositivo de trabalhos.

Tópico	Operações do cliente permitidas	Description
$aws/things/ /jobs/ /update *thingName jobId*	Publicar	Os dispositivos publicam uma mensagem neste tópico para fazer uma solicitação `UpdateJobExecution` . Para obter mais informações, consulte Operações da API MQTT do dispositivo de trabalhos.
$aws/things/ /jobs/ /update/ accepted *thingName jobId*	Assine, receba	Os dispositivos assinam este tópico para receber respostas de êxito de uma solicitação `UpdateJobExecution` . Para obter mais informações, consulte Operações da API MQTT do dispositivo de trabalhos. **Observação** Somente o dispositivo que publica em $aws/things/ *thingName* /jobs/ *jobId* / update recebe mensagens sobre esse tópico.

Tópico	Operações do cliente permitidas	Description
$aws/things/ /jobs/ /update/ rejeitado *thingName* *jobId*	Assine, receba	Os dispositivos assinam este tópico para receber uma resposta quando uma solicitação `UpdateJobExecution` é rejeitada. Para obter mais informaçõ es, consulte Operações da API MQTT do dispositivo de trabalhos. **Observação** Somente o dispositivo que publica em $aws/things/ *thingName* /jobs/ *jobId* / update recebe mensagens sobre esse tópico.
thingName $aws/things/ / jobs/notify	Assine, receba	Os dispositivos assinam este tópico para receber notificações quando uma execução de tarefa é adicionada à lista de execuções pendentes de um objeto ou removida dela. Para obter mais informações, consulte Operações da API MQTT do dispositivo de trabalhos.
thingName $aws/things/ / jobs/notify-next	Assine, receba	Os dispositivos assinam este tópico para receber notificações quando a próxima execução de tarefa pendente para o objeto é alterada. Para obter mais informações, consulte Operações da API MQTT do dispositivo de trabalhos.
$aws/events/job/*jobId*/ concluído	Assinar	O serviço Tarefas publica um evento neste tópico quando uma tarefa é concluída. Para obter mais informações, consulte Eventos de trabalho.

Tópico	Operações do cliente permitidas	Description
$aws/events/job/*jobId*/cancelado	Assinar	O serviço Tarefas publica um evento neste tópico quando uma tarefa é cancelada. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/job/*jobId*/excluído	Assinar	O serviço Tarefas publica um evento neste tópico quando uma tarefa é excluída. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/job/*jobId*/cancelamento_em_andamento	Assinar	O serviço Tarefas publica um evento neste tópico quando o cancelamento de uma tarefa começa. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/job/*jobId*/deletion_in_progress	Assinar	O serviço Tarefas publica um evento neste tópico quando a exclusão de tarefa começa. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/jobExecution/*jobId*/foi bem sucedido	Assinar	O serviço Tarefas publica um evento neste tópico quando a execução da tarefa é bem-sucedida. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/jobExecution/*jobId*/falhou	Assinar	O serviço Tarefas publica um evento neste tópico quando uma execução de tarefa falha. Para obter mais informações, consulte Eventos de trabalho.

Tópico	Operações do cliente permitidas	Description
$aws/events/jobExecution/*jobId*/rejeitado	Assinar	O serviço Tarefas publica um evento neste tópico quando uma execução de tarefa é rejeitada. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/jobExecution/*jobId*/cancelado	Assinar	O serviço Tarefas publica um evento neste tópico quando a execução de uma tarefa é cancelada. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/jobExecution/*jobId*/timed_out	Assinar	O serviço Tarefas publica um evento neste tópico quando a execução de uma tarefa atinge o tempo limite. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/jobExecution/*jobId*/removido	Assinar	O serviço Tarefas publica um evento neste tópico quando a execução de uma tarefa é removida. Para obter mais informações, consulte Eventos de trabalho.
$aws/events/jobExecution/*jobId*/excluído	Assinar	O serviço Tarefas publica um evento neste tópico quando a execução de uma tarefa é excluída. Para obter mais informações, consulte Eventos de trabalho.

Tópicos de comandos

> **Note**
>
> As operações do cliente anotadas como Receber nesta tabela indicam tópicos que AWS IoT são publicados diretamente no cliente que os solicitou, independentemente de o cliente ter se inscrito no tópico ou não. Os clientes devem esperar receber essas mensagens de resposta mesmo que não sejam assinantes delas.
> Essas mensagens de resposta não passam pelo agente de mensagens e não podem ser assinadas por outros clientes ou regras.

Tópico	Operações do cliente permitidas	Description
$aws/commands/// executions/ /request/ *<devices> <DeviceID> <ExecutionId> <PayloadFormat>* $aws/commands/// executions/ /request *<devices> <DeviceID> <ExecutionId>*	Assine, receba	Os dispositivos recebem uma mensagem sobre esse tópico quando é feita uma solicitação para iniciar uma execução de comando no console ou usando a API `StartCommandExecution` . Nesse caso, *<devices>* podem ser coisas de IoT ou clientes MQTT e *<DeviceID>* podem ser o nome da coisa de IoT ou o ID do cliente MQTT.
$aws/commands///executions/ /response/ *<devices> <DeviceID> <ExecutionId> <PayloadFormat>*	Publicar	Os dispositivos usam a API `UpdateCommandExecution` do MQTT para publicar uma mensagem nesse tópico sobre a execução do comando. A mensagem é publicada como uma resposta à solicitação para iniciar uma execução de comando no console ou usando a API `StartCommandExecution` . A mensagem publicada usará JSON ou CBOR como o. *<PayloadFormat>*

Tópico	Operações do cliente permitidas	Description
$aws/commands///executions/ /response/accepted/ *<devices> <DeviceID> <ExecutionId> <PayloadFormat>* $aws/commands///executions/ /response/accepted *<devices> <DeviceID> <ExecutionId>*	Assine, receba	Se o serviço de nuvem tiver processado com êxito o resultado da execução do comando, o AWS IoT Device Management publicará uma resposta no tópico /accepted.
$aws/commands///executions/ /response/rejected/ *<devices> <DeviceID> <ExecutionId> <PayloadFormat>* $aws/commands///executions/ /response/rejeitado *<devices> <DeviceID> <ExecutionId>*	Publicar	Se o serviço de nuvem não conseguir processar o resultado da execução do comando, o AWS IoT Device Management publicará uma resposta no tópico /rejected.

Tópicos de regras

Tópico	Operações do cliente permitidas	Description
$ aws/rules/ *ruleName*	Publicar	Um dispositivo ou um aplicativo faz publicações nesse tópico para acionar regras diretamente. Para obter mais

Tópico	Operações do cliente permitidas	Description
		informações, consulte Reduzir custos do sistema de mensagens com Ingestão básica.

Proteger tópicos de túneis

Tópico	Operações do cliente permitidas	Description
thing-name $aws/things/ /tunnels/notify	Assinar	AWS IoT publica essa mensagem para que um agente de IoT inicie um proxy local no dispositivo remoto. Para obter mais informações, consulte the section called “Snippet de atendente de IoT”.

Tópicos de sombra

Os tópicos desta seção são usados por sombras nomeadas e sem nome. Os tópicos usados por cada uma diferem apenas no prefixo do tópico. Esta tabela mostra o prefixo do tópico usado em cada tipo de sombra.

Valor do ***ShadowTopicPrefix***	Tipo de sombra
thingName $aws/things/ /shadow	Sombra sem nome (clássica)
$aws/things/ /shadow/name/ *thingName* *shadowName*	Sombra nomeada

Para criar um tópico completo, selecione o *ShadowTopicPrefix* tipo de sombra ao qual você deseja se referir, substitua *thingName* e, se aplicável,*shadowName*, pelos valores correspondentes e, em seguida, anexe-o ao esboço do tópico, conforme mostrado na tabela a seguir. Lembre-se de que os tópicos diferenciam maiúsculas de minúsculas.

Tópico	Operações do cliente permitidas	Description
ShadowTopicPrefix / excluir	Publicar/assinar	Um dispositivo ou um aplicativo faz publicações nesse tópico para excluir uma sombra. Para obter mais informaçõ es, consulte /delete.
ShadowTopicPrefix / excluir/aceito	Assinar	O serviço Sombra do dispositivo envia mensagens para esse tópico quando uma sombra é excluída. Para obter mais informações, consulte /delete/accepted.
ShadowTopicPrefix / excluir/rejeitado	Assinar	O serviço Sombra do dispositivo envia mensagens para esse tópico quando uma solicitação de exclusão de uma sombra é rejeitada. Para obter mais informações, consulte /delete/rejected.
ShadowTopicPrefix / obter	Publicar/assinar	Um aplicativo ou um objeto publica uma mensagem vazia nesse tópico para obter uma sombra. Para obter mais informações, consulte Tópicos MQTT da Sombra do Dispositivo.
ShadowTopicPrefix / obter/aceito	Assinar	O serviço Sombra do dispositivo envia mensagens para esse tópico quando uma solicitação para uma sombra é feita com êxito. Para obter mais informações, consulte /get/accepted.
ShadowTopicPrefix / get/rejeitado	Assinar	O serviço Sombra do dispositivo envia mensagens para esse tópico quando uma solicitação de sombra é rejeitada. Para obter mais informações, consulte /get/rejected.

Tópico	Operações do cliente permitidas	Description
ShadowTopicPrefix / atualizar	Publicar/assinar	Umo objeto ou um aplicativo faz publicações nesse tópico para atualizar uma sombra. Para obter mais informaçõ es, consulte /update.
ShadowTopicPrefix / update/aceito	Assinar	O serviço Sombra do dispositivo envia mensagens para esse tópico quando uma atualização é feita com êxito em uma sombra. Para obter mais informaçõ es, consulte /update/accepted.
ShadowTopicPrefix / update/rejeitado	Assinar	O serviço Sombra do dispositivo envia mensagens para esse tópico quando a atualização de uma sombra é rejeitada. Para obter mais informações, consulte /update/rejected.
ShadowTopicPrefix / update/delta	Assinar	O serviço Sombra do dispositivo envia mensagens para esse tópico quando uma diferença é detectada entre as seções desejadas e relatadas de uma sombra. Para obter mais informações, consulte /update/delta.
ShadowTopicPrefix / update/documents	Assinar	AWS IoT publica um documento de estado para esse tópico sempre que uma atualização na sombra for executada com êxito. Para obter mais informações, consulte /update/documents.

Tópicos de entrega de arquivos baseados em MQTT

> **Note**
>
> As operações do cliente anotadas como Receber nesta tabela indicam tópicos que AWS IoT são publicados diretamente no cliente que os solicitou, independentemente de o cliente ter se inscrito no tópico ou não. Os clientes devem esperar receber essas mensagens de resposta mesmo que não sejam assinantes delas. Essas mensagens de resposta não passam pelo agente de mensagens e não podem ser assinadas por outros clientes ou regras.

Essas mensagens oferecem suporte a buffers de resposta no formato Concise Binary Object Representation (CBOR) e em JavaScript Object Notation (JSON), dependendo do tópico. *payload-format*

payload-format	Tipo de dados do formato de resposta
cbor	Representação Concisa de Objetos Binários (CBOR)
json	JavaScript Notação de objeto (JSON)

Tópico	Operações do cliente permitidas	Description
$ aws/things/ /streams/ / data/ *ThingName StreamId payload-format*	Assine, receba	AWS A entrega de arquivos baseada em MQTT é publicada neste tópico se a solicitação "GetStream" de um dispositivo for aceita. A carga útil contém os dados do fluxo. Para obter mais informações, consulte Usando a entrega de arquivos AWS IoT baseada em MQTT em dispositivos.

Tópico	Operações do cliente permitidas	Description
$ aws/things/ /streams/ /get/ *ThingName StreamId payload-format*	Publicar	Um dispositivo publica neste tópico para realizar uma solicitação GetStream "". Para obter mais informações, consulte Usando a entrega de arquivos AWS IoT baseada em MQTT em dispositivos.
$aws/things/ /streams/ / description/ *ThingName StreamId payload-format*	Assine, receba	AWS A entrega de arquivos baseada em MQTT é publicada neste tópico se a solicitação "DescribeStream" de um dispositivo for aceita. A carga útil contém a descrição do fluxo. Para obter mais informações, consulte Usando a entrega de arquivos AWS IoT baseada em MQTT em dispositivos.
$ aws/things/ /streams/ / describe/ *ThingName StreamId payload-format*	Publicar	Um dispositivo publica neste tópico para realizar uma solicitação DescribeStream "". Para obter mais informações, consulte Usando a entrega de arquivos AWS IoT baseada em MQTT em dispositivos.
$aws/things/ /streams/ / rejeted/ *ThingName StreamId payload-format*	Assine, receba	AWS A entrega de arquivos baseada em MQTT é publicada neste tópico se uma solicitação "DescribeStream" ou "GetStream" de um dispositivo for rejeitada. Para obter mais informações, consulte Usando a entrega de arquivos AWS IoT baseada em MQTT em dispositivos.

ARN de tópicos reservados

Todos os tópicos reservados ARNs (Amazon Resource Names) têm o seguinte formato:

```
arn:aws:iot:aws-region:AWS-account-ID:topic/Topic
```

Por exemplo, `arn:aws:iot:us-west-2:123EXAMPLE456:topic/$aws/things/thingName/jobs/get/accepted` é um ARN para o tópico reservado `$aws/things/thingName/jobs/get/accepted`.

Configurações do domínio

Em AWS IoT Core, você pode usar configurações de domínio para configurar e gerenciar os comportamentos dos seus endpoints de dados. Com as configurações de domínio, você pode gerar vários endpoints de AWS IoT Core dados, personalizá-los com seus próprios nomes de domínio totalmente qualificados (FQDN) e certificados de servidor associados, além de associar um autorizador personalizado. Para obter mais informações, consulte Autenticação e autorização personalizadas.

> **Note**
>
> Esse recurso não está disponível em AWS GovCloud (US) Regiões da AWS.

Neste capítulo:

- O que é a configuração de um domínio?
- Criação e configuração de domínios AWS gerenciados
- Criar e configurar domínios gerenciados do cliente
- Gerenciar configurações de domínio
- Definir configurações de TLS nas configurações de domínio
- Configuração de certificado de servidor para grampeamento de OCSP

O que é a configuração de um domínio?

Em AWS IoT Core, uma configuração de domínio se refere à instalação e configuração de um domínio (domínio gerenciado ou domínio AWS gerenciado pelo cliente) para seus endpoints de

AWS IoT Core dados. AWS IoT Core também fornece um endpoint padrão para sua AWS conta (`iot:Data-ATS`) para os dispositivos se AWS IoT Core comunicarem.

Neste tópico:

- Casos de uso
- Principais conceitos
- Observações importantes

Casos de uso

É possível usar configurações de domínio para simplificar tarefas, como a seguinte:

- Migre dispositivos para o. AWS IoT Core
- Suporte a frotas de dispositivos heterogêneos mantendo configurações de domínio separadas para tipos de dispositivos separados.
- Mantenha a identidade da marca (por exemplo, por meio do nome de domínio) enquanto migra a infraestrutura de aplicativos para o. AWS IoT Core

Principais conceitos

Os conceitos a seguir dão detalhes das configurações de domínio e de conceitos relacionados.

- Configuração do domínio

  A instalação e configuração de um domínio para seus AWS IoT Core endpoints.

- Domínio de endpoint padrão

  O domínio que AWS IoT fornece o endpoint padrão, como`iot:Data-ATS`. Para encontrar o endpoint padrão, execute o comando describe-endpoint ou CLI. describe-domain-configuration Como alternativa, acesse o console do AWS IoT Core e escolha Configurações de domínio em Conectar no painel de navegação à esquerda. O endpoint padrão é listado com o nome `iot:Data-ATS`.

- AWS domínio gerenciado

O domínio que AWS será gerenciado. Escolher um domínio AWS gerenciado significa que seus dispositivos se conectarão usando um endpoint de dados fornecido pela AWS. AWS gerenciará o domínio e os certificados.

- Domínio gerenciado pelo cliente

  O domínio que você gerenciará. Também conhecido como domínio personalizado. Escolher um domínio gerenciado pelo cliente significa que seus dispositivos serão conectados usando um endpoint de dados de domínio personalizado. Você gerenciará o domínio e os certificados. O domínio gerenciado pelo cliente permite que você personalize o endpoint de URLs acordo com suas necessidades. Por exemplo, você pode usar um nome de domínio personalizado (`your-domain-name.com`) ou aplicar políticas de acesso específicas.

- Tipo de autenticação

  O tipo de autenticação escolhido para autenticar seus dispositivos ao se conectar ao AWS IoT Core. Ao criar uma configuração de domínio, é necessário especificar um tipo de autenticação. Para obter mais informações, consulte ???.

- Protocolo de aplicativo

  Os protocolos da camada de aplicação que seus dispositivos usam ao se conectar ao AWS IoT Core. Ao criar uma configuração de domínio, é necessário especificar um protocolo de aplicativo. Para obter mais informações, consulte ???.

Observações importantes

AWS IoT Core usa a extensão TLS de indicação de nome do servidor (SNI) para aplicar configurações de domínio. Ao conectar dispositivos a AWS IoT Core, os clientes podem enviar a extensão Server Name Indication (SNI), que é necessária para recursos como registro de várias contas, endpoints configuráveis, domínios personalizados e endpoints VPC. Eles também devem enviar um nome de servidor idêntico ao nome de domínio especificado na configuração do domínio. Para testar esse serviço, use a versão v2 do AWS IoT dispositivo SDKs em GitHub.

Se você criar vários endpoints de dados no seu Conta da AWS, eles compartilharão AWS IoT Core recursos como tópicos do MQTT, sombras de dispositivos e regras.

Quando você fornece os certificados do servidor para configuração de domínio AWS IoT Core personalizada, os certificados têm no máximo quatro nomes de domínio. Para saber mais, consulte Endpoints e cotas do AWS IoT Core.

Criação e configuração de domínios AWS gerenciados

Você cria um endpoint configurável em um domínio AWS gerenciado usando a CreateDomainConfigurationAPI. Uma configuração de domínio para um domínio AWS gerenciado consiste no seguinte:

- `domainConfigurationName`

  Um nome definido pelo usuário que identifica a configuração do domínio e o valor deve ser exclusivo para você. Região da AWS Não é possível usar nomes de configuração de domínio que comecem com `IoT:` porque eles são reservados para endpoints padrão.
- `defaultAuthorizerName` (opcional)

  O nome do autorizador personalizado que deve ser usado no endpoint.
- `allowAuthorizerOverride` (opcional)

  Um valor booliano que especifica se os dispositivos podem substituir o autorizador padrão especificando outro autorizador no cabeçalho HTTP da solicitação. Esse valor será exigido se um valor para `defaultAuthorizerName` for especificado.
- `serviceType` (opcional)

  O tipo de serviço que o endpoint fornece. AWS IoT Core suporta apenas o tipo `DATA` de serviço. Ao especificar `DATA`, o AWS IoT Core retornará um endpoint com um tipo de endpoint de `iot:Data-ATS`. Você não pode criar um endpoint configurável `iot:Data` (VeriSign).
- `TlsConfig` (opcional)

  Um objeto que especifica a configuração TLS para um domínio. Para obter mais informações, consulte ???.

O AWS CLI comando de exemplo a seguir cria uma configuração de domínio para um `Data` endpoint.

```
aws iot create-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" --service-type "DATA"
```

A saída do comando pode ser semelhante à seguinte.

```
{
    "domainConfigurationName": "myDomainConfigurationName",
    "domainConfigurationArn": "arn:aws:iot:us-east-1:123456789012:domainconfiguration/
myDomainConfigurationName/itihw"
}
```

Criar e configurar domínios gerenciados do cliente

As configurações de domínio permitem especificar um nome de domínio totalmente qualificado (FQDN) para conectar-se ao AWS IoT Core. Há muitos benefícios em usar domínios gerenciados pelo cliente (também conhecidos como domínios personalizados): você pode expor seu próprio domínio ou o domínio da sua empresa aos clientes para fins de marca; você pode facilmente alterar seu próprio domínio para apontar para um novo corretor; você pode oferecer suporte à multilocação para atender clientes com domínios diferentes dentro do mesmo Conta da AWS; e você pode gerenciar os detalhes dos seus próprios certificados de servidor, como a autoridade de certificação raiz (CA) usada para assinar o certificado, o algoritmo de assinatura, a profundidade da cadeia de certificados e o ciclo de vida do certificado.

O fluxo de trabalho para definir uma configuração de domínio com um domínio personalizado consiste nos três estágios a seguir.

1. Registrando certificados de servidor em AWS Certificate Manager
2. Criar uma configuração de domínio
3. Criar registros DNS

Registrando certificados de servidor no gerenciador AWS de certificados

Antes de criar uma configuração de domínio com um domínio personalizado, é necessário registrar a cadeia de certificados do servidor no AWS Certificate Manager (ACM). É possível usar os seguintes três tipos de certificados de servidor.

- Certificados públicos gerados pelo ACM
- Certificados externos assinados por uma CA pública
- Certificados externos assinados por uma CA privada

> **Note**
>
> AWS IoT Core considera que um certificado é assinado por uma CA pública se estiver incluído no pacote ca confiável da Mozilla.

Requisitos de certificado

Consulte Pré-requisitos para importar certificados para obter os requisitos de importação de certificados para o ACM. Além desses requisitos, o AWS IoT Core adiciona os seguintes requisitos.

- O certificado de entidade final deve incluir a extensão Extended Key Usage x509 v3 com um valor de serverAuth (Autenticação de servidor Web TLS). Se você solicitar o certificado do ACM, essa extensão será adicionada automaticamente.
- A profundidade máxima da cadeia de certificados é de 5 certificados.
- O tamanho máximo da cadeia de certificados é de 16 KB.
- Os algoritmos criptográficos e os tamanhos de chave compatíveis incluem RSA de 2048 bits (RSA_2048) e ECDSA de 256 bits (EC_Prime256v1).

Usar um certificado para vários domínios

Se você planeja usar um certificado para abranger vários subdomínios, use um domínio curinga no campo nome comum (CN) ou Nomes alternativos do assunto (SAN). Por exemplo, use **`*.iot.example.com`** para abranger dev.iot.example.com, qa.iot.example.com e prod.iot.example.com. Cada FQDN exige sua própria configuração de domínio, mas mais de uma configuração de domínio pode usar o mesmo valor curinga. O CN ou o SAN devem abranger o FQDN que você deseja usar como domínio personalizado. Se SANs estiverem presentes, o CN é ignorado e uma SAN deve cobrir o FQDN que você deseja usar como domínio personalizado. Essa abrangência pode ser uma correspondência exata ou uma correspondência curinga. Depois que um certificado curinga é validado e registrado em uma conta, outras contas na região são impedidas de criar domínios personalizados que se sobreponham ao certificado.

As seções a seguir descrevem como obter cada tipo de certificado. Cada recurso de certificado requer um nome do recurso da Amazon (ARN) registrado no ACM usado ao criar a configuração do domínio.

Certificados públicos gerados pelo ACM

Você pode gerar um certificado público para seu domínio personalizado usando a RequestCertificateAPI. Ao gerar um certificado dessa maneira, o ACM valida a sua propriedade do domínio personalizado. Para obter mais informações, consulte Solicitar um certificado público no Guia do usuário do AWS Certificate Manager .

Certificados externos assinados por uma CA pública

Se você já tem um certificado de servidor assinado por uma CA pública (uma CA incluída no pacote ca confiável da Mozilla), você pode importar a cadeia de certificados diretamente para o ACM usando a API. ImportCertificate Para saber mais sobre essa tarefa, sobre os pré-requisitos e sobre as exigências de formato do certificado, consulte Importar certificados.

Certificados externos assinados por uma CA privada

Se você já tiver um certificado de servidor assinado por uma CA privada ou autoassinado, será possível usar o certificado para criar a configuração de domínio, mas também será necessário criar um certificado público adicional no ACM para validar a propriedade do seu domínio. Para fazer isso, registre sua cadeia de certificados de servidor no ACM usando a ImportCertificateAPI. Para saber mais sobre essa tarefa, sobre os pré-requisitos e sobre as exigências de formato do certificado, consulte Importar certificados.

Criar um certificado de validação

Depois de importar seu certificado para o ACM, gere um certificado público para seu domínio personalizado usando a RequestCertificateAPI. Ao gerar um certificado dessa maneira, o ACM valida a sua propriedade do domínio personalizado. Para obter mais informações, consulte Solicitar um certificado público. Ao criar a configuração de domínio, utilize esse certificado público como certificado de validação.

Criar uma configuração de domínio

Você cria um endpoint configurável em um domínio personalizado usando a CreateDomainConfigurationAPI. Uma configuração de domínio para um domínio personalizado consiste no seguinte:

- `domainConfigurationName`

Um nome definido pelo usuário que identifica a configuração de domínio. Os nomes da configuração de domínio que começam com `IoT:` são reservados para endpoints padrão e não podem ser usados. Além disso, esse valor deve ser exclusivo para você Região da AWS.

- `domainName`

  O FQDN ao qual seus dispositivos usam para se conectar. AWS IoT Core AWS IoT Core aproveita a extensão TLS de indicação de nome do servidor (SNI) para aplicar configurações de domínio. Os dispositivos devem usar essa extensão ao conectar e transmitir um nome de servidor idêntico ao nome de domínio especificado na configuração de domínio.

- `serverCertificateArns`

  O ARN da cadeia de certificados do servidor que você registrou no ACM. AWS IoT Core atualmente oferece suporte a apenas um certificado de servidor.

- `validationCertificateArn`

  O ARN do certificado público gerado no ACM para validar a propriedade do domínio personalizado. Esse argumento não será necessário se você usar um certificado de servidor gerado pelo ACM ou assinado publicamente.

- `defaultAuthorizerName (optional)`

  O nome do autorizador personalizado que deve ser usado no endpoint.

- `allowAuthorizerOverride`

  Um valor booliano que especifica se os dispositivos podem substituir o autorizador padrão especificando outro autorizador no cabeçalho HTTP da solicitação. Esse valor será exigido se um valor para `defaultAuthorizerName` for especificado.

- `serviceType`

  AWS IoT Core atualmente suporta somente o tipo `DATA` de serviço. Quando você especifica`DATA`, AWS IoT retorna um endpoint com um tipo de endpoint de. `iot:Data-ATS`

- `TlsConfig` (opcional)

  Um objeto que especifica a configuração TLS para um domínio. Para obter mais informações, consulte ???.

- `serverCertificateConfig` (opcional)

Um objeto que especifica a configuração de certificado de servidor para um domínio. Para obter mais informações, consulte ???.

O AWS CLI comando a seguir cria uma configuração de domínio para iot.example.com.

```
aws iot create-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" --service-type "DATA"
--domain-name "iot.example.com" --server-certificate-arns serverCertARN --validation-
certificate-arn validationCertArn
```

> **Note**
>
> Depois de criar sua configuração de domínio, pode levar até 60 minutos até que seus certificados AWS IoT Core de servidor personalizados sejam entregues.

Para obter mais informações, consulte ???.

Criar registros DNS

Depois de registrar sua cadeia de certificados e criar sua configuração de domínio, crie um registro de DNS para que o domínio personalizado aponte para um domínio da AWS IoT . Esse registro deve apontar para um AWS IoT ponto final do tipo`iot:Data-ATS`. Você pode obter seu endpoint usando a DescribeEndpointAPI.

O AWS CLI comando a seguir mostra como obter seu endpoint.

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

Depois de obter seu `iot:Data-ATS` endpoint, crie um `CNAME` registro do seu domínio personalizado para esse AWS IoT endpoint. Se você criar vários domínios personalizados no mesmo Conta da AWS, atribua um alias a esse mesmo `iot:Data-ATS` endpoint.

Solução de problemas

Se você tiver problemas para conectar dispositivos a um domínio personalizado, verifique se ele AWS IoT Core aceitou e aplicou seu certificado de servidor. Você pode verificar AWS IoT Core se aceitou seu certificado usando o AWS IoT Core console ou AWS CLI o.

Para usar o AWS IoT Core console, navegue até a página Configurações do domínio e selecione o nome da configuração do domínio. Na seção Detalhes do certificado do servidor, verifique o status e os detalhes do status. Se o certificado for inválido, substitua-o no ACM por um certificado que atenda aos requisitos de certificado listados na seção anterior. Se o certificado tiver o mesmo ARN, ele AWS IoT Core será retirado e aplicado automaticamente.

Para verificar o status do certificado usando o AWS CLI, chame a DescribeDomainConfigurationAPI e especifique seu nome de configuração de domínio.

> **Note**
>
> Se seu certificado for inválido, AWS IoT Core continuará a fornecer o último certificado válido.

Você pode verificar qual certificado está sendo fornecido em seu endpoint usando o seguinte comando openssl.

```
openssl s_client -connect custom-domain-name:8883 -showcerts -servername custom-domain-name
```

Gerenciar configurações de domínio

Este tópico aborda as principais operações para você gerenciar seus recursos de configuração de domínio. Você também pode gerenciar os ciclos de vida das configurações existentes usando o seguinte APIs: ListDomainConfigurations,, e. DescribeDomainConfigurationUpdateDomainConfigurationDeleteDomainConfiguration

Neste tópico:

- Visualizar configurações de domínio
- Atualizar configurações de domínio
- Excluir configurações de domínio
- Renovação de certificados em domínios personalizados

Visualizar configurações de domínio

Para retornar uma lista paginada de todas as configurações de domínio em seu Conta da AWS, use a API. ListDomainConfigurations Você pode ver os detalhes de uma configuração de domínio

específica usando a DescribeDomainConfigurationAPI. Essa API usa um único parâmetro `domainConfigurationName` e retorna os detalhes da configuração especificada.

Exemplo

Atualizar configurações de domínio

Para atualizar o status ou o autorizador personalizado da configuração do seu domínio, use a UpdateDomainConfigurationAPI. É possível definir o status como `ENABLED` ou `DISABLED`. Se você desativar a configuração de domínio, os dispositivos conectados a esse domínio receberão um erro de autenticação. No momento, não é possível atualizar o certificado de servidor da configuração de domínio. Para alterar o certificado de uma configuração de domínio, é necessário excluí-la e recriá-la.

Exemplo

Excluir configurações de domínio

Antes de excluir uma configuração de domínio, use a UpdateDomainConfigurationAPI para definir o status como`DISABLED`. Isso ajuda a evitar a exclusão acidental do endpoint. Depois de desativar a configuração do domínio, exclua-a usando a DeleteDomainConfigurationAPI. Você deve colocar os domínios AWS gerenciados em `DISABLED` status por 7 dias antes de poder excluí-los. É possível colocar domínios personalizados no status `DISABLED` e depois excluí-los de uma só vez.

Exemplo

Depois de excluir uma configuração de domínio, o certificado do servidor associado a esse domínio personalizado AWS IoT Core não é mais exibido.

Renovação de certificados em domínios personalizados

Talvez seja necessário substituir periodicamente o certificado do servidor por um atualizado. A taxa na qual você faz isso depende do período de validade do certificado. Se você gerou o certificado de servidor usando o AWS Certificate Manager (ACM), é possível configurar o certificado para ser renovado automaticamente. Quando o ACM renova seu certificado, pega AWS IoT Core automaticamente o novo certificado. Não é necessário realizar nenhuma ação adicional. Se você importou o certificado do servidor de uma origem diferente, poderá alterná-lo reimportando-o para o ACM. Para ver informações sobre a reimportação de certificados, consulte Reimportar um certificado.

> **Note**
>
> AWS IoT Core só coleta atualizações de certificados sob as seguintes condições.

- O novo certificado tem o mesmo ARN do antigo.
- O novo certificado tem o mesmo algoritmo de assinatura, nome comum ou nome alternativo do assunto que o antigo.

Definir configurações de TLS nas configurações de domínio

AWS IoT Core fornece políticas de segurança predefinidas para você personalizar suas configurações de Transport Layer Security (TLS) para TLS 1.2 e TLS 1.3 nas configurações de domínio. Uma política de segurança é uma combinação de protocolos TLS e suas cifras que determinam os protocolos e cifras compatíveis durante as negociações de TLS entre um cliente e um servidor. Com as políticas de segurança suportadas, você pode gerenciar as configurações de TLS de seus dispositivos com mais flexibilidade, aplicar o máximo de medidas de up-to-date segurança ao conectar novos dispositivos e manter configurações TLS consistentes para dispositivos existentes.

A tabela a seguir descreve as políticas de segurança, suas versões de TLS e as regiões compatíveis:

Nome da política de segurança	Suportado Regiões da AWS
TSecurityPolítica de IP_ _1_3_2022_10 TLS13	Tudo Regiões da AWS
TSecurityPolítica de IO_ _1_2_2022_10 TLS13	Tudo Regiões da AWS
TSecurityPolítica de IO_ _1_2_2022_10 TLS12	Tudo Regiões da AWS
TSecurityPolítica de Io_ _1_0_2016_01 TLS12	ap-east-1, ap-northeast-2, ap-south-1, ap-southeast-2, ca-central-1, cn-north-1, cn-northwest-1, eu-north-1, eu-west-2, eu-west-3, me-south-1, sa-east-1, us-east-2, us-west-1

Nome da política de segurança	Suportado Regiões da AWS
TSecurityPolítica de Io_ _1_0_2015_01 TLS12	ap-northeast-1, ap-southeast-1, eu-central-1, eu-west-1, us-east-1, us-west-2

Os nomes das políticas de segurança AWS IoT Core incluem informações de versão com base no ano e mês em que foram lançadas. Se você criar uma nova configuração de domínio, a política de segurança assumirá `IoTSecurityPolicy_TLS13_1_2_2022_10` como padrão. Para obter uma tabela completa de políticas de segurança com detalhes de protocolos, portas TCP e cifras, consulte Políticas de segurança. AWS IoT Core não oferece suporte a políticas de segurança personalizadas. Para obter mais informações, consulte ???.

Para definir as configurações de TLS nas configurações de domínio, você pode usar o AWS IoT console ou o. AWS CLI

Conteúdo

- Definir as configurações de TLS nas configurações de domínio (console)
- Definir as configurações de TLS nas configurações de domínio (CLI)

Definir as configurações de TLS nas configurações de domínio (console)

Para definir as configurações de TLS usando o console AWS IoT

1. Faça login no Console de gerenciamento da AWS e abra o AWS IoT console.
2. Para definir as configurações de TLS ao criar uma nova configuração de domínio, siga estas etapas.
    1. No painel de navegação esquerdo, escolha Configurações de domínio e, em seguida, escolha Criar configuração de domínio.
    2. Na página Criar configuração de domínio, na seção Configurações de domínio personalizado - opcional, escolha uma política de segurança em Selecionar política de segurança.
    3. Siga o widget e conclua o restante das etapas. Selecione Criar configuração de domínio.
3. Para atualizar as configurações de TLS em uma configuração de domínio existente, siga estas etapas.

1. No painel de navegação esquerdo, escolha Configurações de domínio e, em seguida, escolha uma configuração de domínio.
2. Na página de Detalhes da configuração do domínio, escolha Editar. Em seguida, na seção Configurações de domínio personalizado - opcional, em Selecionar política de segurança, escolha uma política de segurança.
3. Selecione Atualizar configuração de domínio.

Para obter mais informações, consulte Criar uma configuração de domínio e Gerenciar configurações de domínio.

Definir as configurações de TLS nas configurações de domínio (CLI)

É possível usar os comandos CLI create-domain-configuration e update-domain-configuration para definir suas configurações de TLS em configurações de domínio.

1. Para especificar as configurações de TLS usando o comando CLI create-domain-configuration:

```
aws iot create-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

   A saída desse comando pode ser semelhante à seguinte:

```
{
"domainConfigurationName": "test",
"domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
test/34ga9"
}
```

   Se você criar uma nova configuração de domínio sem especificar a política de segurança, o valor será definido por padrão como: `IoTSecurityPolicy_TLS13_1_2_2022_10`.

2. Para descrever as configurações de TLS usando o comando CLI describe-domain-configuration:

```
aws iot describe-domain-configuration \
    --domain-configuration-name domainConfigurationName
```

Esse comando pode retornar os detalhes da configuração do domínio que incluem as configurações de TLS, como as seguintes:

```
{
 "tlsConfig": {
 "securityPolicy": "IoTSecurityPolicy_TLS13_1_2_2022_10"
 },
 "domainConfigurationStatus": "ENABLED",
 "serviceType": "DATA",
 "domainType": "AWS_MANAGED",
 "domainName": "d1234567890abcdefghij-ats.iot.us-west-2.amazonaws.com",
 "serverCertificates": [],
 "lastStatusChangeDate": 1678750928.997,
 "domainConfigurationName": "test",
 "domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
test/34ga9"
}
```

3. Para atualizar as configurações de TLS usando o comando CLI update-domain-configuration:

```
aws iot update-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

A saída desse comando pode ser semelhante à seguinte:

```
{
"domainConfigurationName": "test",
"domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
test/34ga9"
}
```

4. Para atualizar as configurações de TLS do endpoint do ATS, execute o comando CLI update-domain-configuration. O nome de configuração do domínio para o endpoint do ATS é `iot:Data-ATS`.

```
aws iot update-domain-configuration \
    --domain-configuration-name "iot:Data-ATS" \
    --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

A saída do comando pode ser semelhante à seguinte:

```
{
"domainConfigurationName": "iot:Data-ATS",
"domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
iot:Data-ATS"
}
```

Para obter mais informações, consulte CreateDomainConfiguration e UpdateDomainConfiguration na Referência da API do AWS .

Configuração de certificado de servidor para grampeamento de OCSP

AWS IoT Core suporta grampeamento OCSP (Online Certificate Status Protocol) para certificado de servidor, também conhecido como grampeamento OCSP de certificado de servidor ou grampeamento OCSP. É um mecanismo de segurança usado para verificar o status de revogação no certificado do servidor em um handshake de Transport Layer Security (TLS). O grampeamento OCSP AWS IoT Core permite que você adicione uma camada adicional de verificação à validade do certificado do servidor do seu domínio personalizado.

Você pode ativar o grampeamento OCSP do certificado do servidor AWS IoT Core para verificar a validade do certificado consultando o respondente OCSP periodicamente. A configuração de grampeamento de OCSP faz parte do processo para criar ou atualizar uma configuração de domínio com um domínio personalizado. O grampeamento de OCSP verifica continuamente o status de revogação no certificado do servidor. Isso ajuda a verificar se os certificados que foram revogados pela CA não são mais confiáveis para os clientes que se conectam aos seus domínios personalizados. Para obter mais informações, consulte ???.

O grampeamento de OCSP do certificado de servidor fornece verificação do status da revogação em tempo real, reduz a latência associada à verificação do status da revogação e melhora a privacidade e a confiabilidade das conexões seguras. Para obter mais informações sobre os benefícios de usar o grampeamento de OCSP, consulte ???.

> **Note**
>
> Esse recurso não está disponível em AWS GovCloud (US) Regions.

Neste tópico:

- O que é o OCSP?
- Como funciona o grampeamento de OCSP
- Habilitando o certificado de servidor OCSP em AWS IoT Core
- Configurando o certificado de servidor OCSP para endpoints privados em AWS IoT Core
- Notas importantes sobre o uso do grampeamento OCSP do certificado de servidor em AWS IoT Core
- Solucionando problemas de grampeamento do certificado OCSP do servidor AWS IoT Core

O que é o OCSP?

O Protocolo de status de certificado online (OCSP) ajuda a fornecer o status de revogação de um certificado de servidor para um handshake Transport Layer Security (TLS).

Principais conceitos

Os conceitos-chave a seguir fornecem detalhes sobre o protocolo de status de certificados online (OCSP).

OCSP

O OCSP é usado para verificar o status de revogação do certificado durante o handshake de Transport Layer Security (TLS). O OCSP permite a validação de certificados em tempo real. Isso confirma que o certificado não foi revogado nem expirou desde que foi emitido. O OCSP também é mais escalável em comparação com as tradicionais Listas de Revogação de Certificados (). CRLs As respostas OCSP são menores e podem ser geradas com eficiência, tornando-as mais adequadas para infraestruturas de chave privada de grande escala (). PKIs

Respondente OCSP

Um respondente OCSP (também conhecido como servidor OCSP) recebe e responde às solicitações OCSP de clientes que buscam verificar o status de revogação dos certificados.

OCSP do lado do cliente

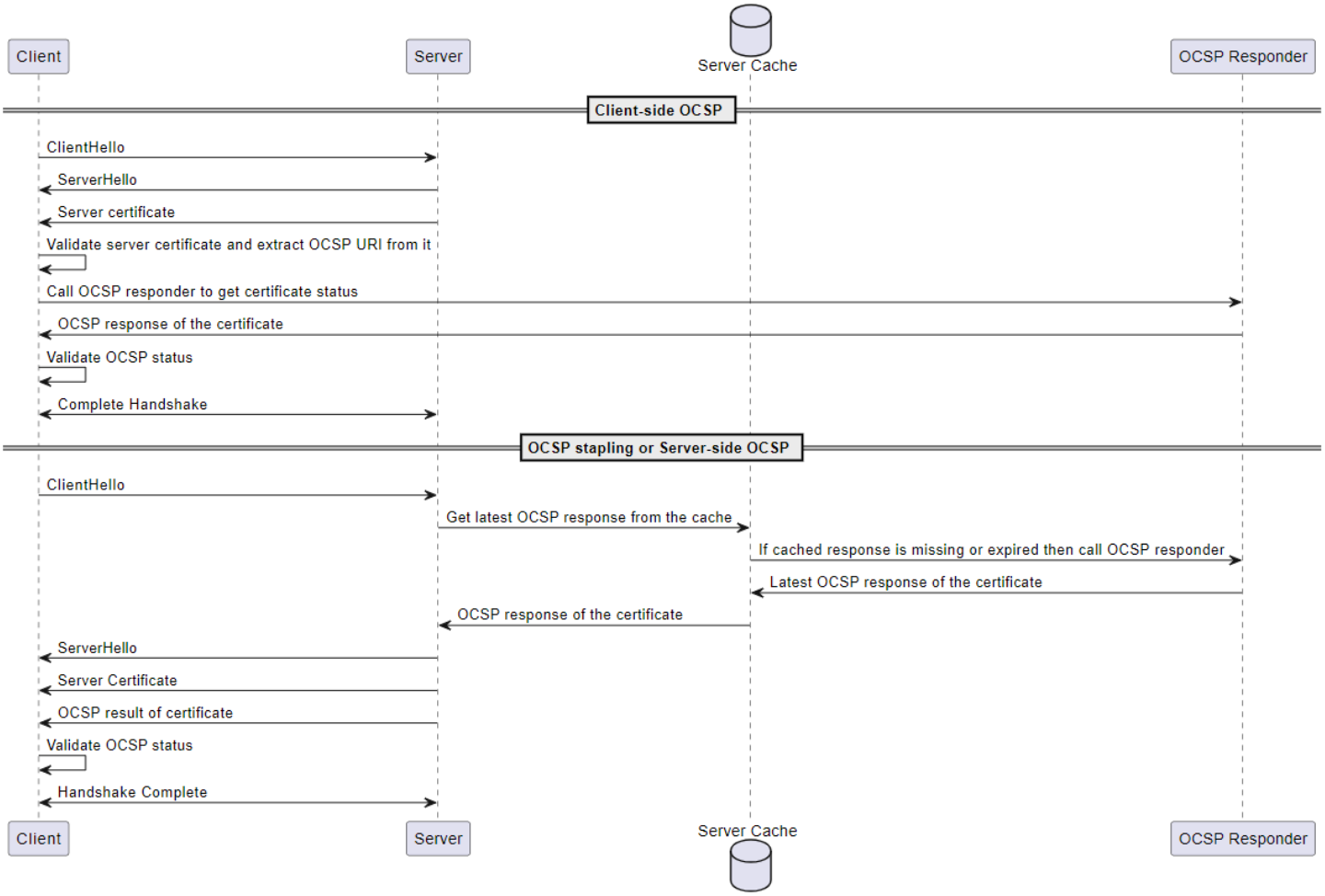
No OCSP do lado do cliente, o cliente usa o OCSP para contatar um respondente OCSP a fim de conferir o status de revogação do certificado durante o handshake de TLS.

OCSP do lado do servidor

No OCSP do lado do servidor (também conhecido como grampeamento de OCSP), o servidor está habilitado (em vez do cliente) para fazer a solicitação ao respondente OCSP. O servidor grampeia a resposta OCSP ao certificado e a devolve ao cliente durante o handshake de TLS.

Diagramas do OCSP

O diagrama a seguir ilustra como o OCSP do lado do cliente e o OCSP do lado do servidor funcionam.



OCSP do lado do cliente

1. O cliente envia uma mensagem `ClientHello` para iniciar o handshake TLS com o servidor.
2. O servidor recebe a mensagem e responde com uma mensagem `ServerHello`. O servidor também envia o certificado do servidor para o cliente.
3. O cliente valida o certificado do servidor e extrai um URI OCSP dele.

4. O cliente envia uma solicitação de verificação de revogação de certificado para o respondente OCSP.
5. O respondente OCSP envia uma resposta OCSP.
6. O cliente valida o status do certificado com base na resposta do OCSP.
7. O handshake TLS é concluído.

OCSP do lado do servidor

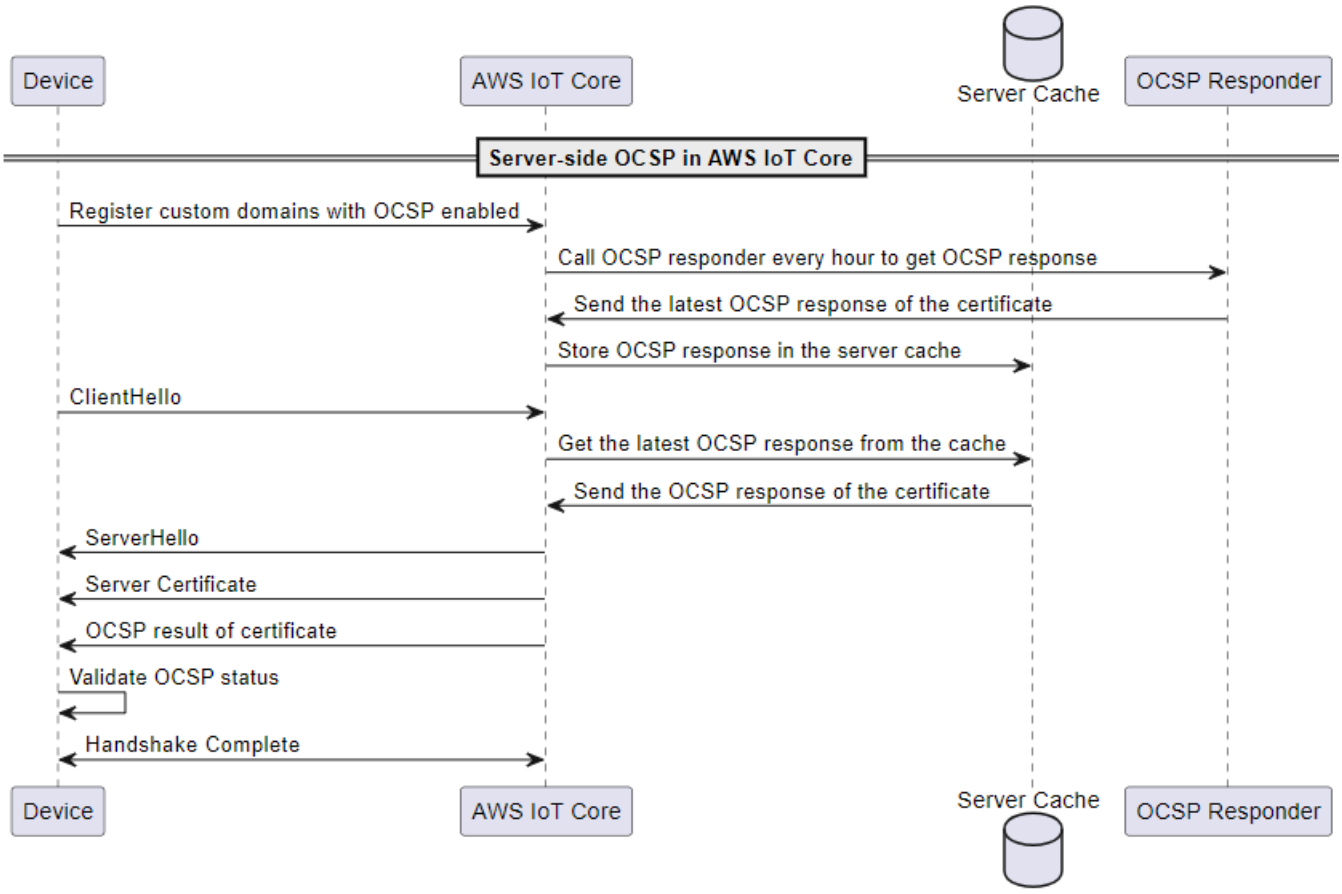
1. O cliente envia uma mensagem `ClientHello` para iniciar o handshake TLS com o servidor.
2. O servidor recebe a mensagem e obtém a resposta OCSP em cache mais recente. Se a resposta em cache estiver ausente ou expirada, o servidor chamará o respondente OCSP para obter o status do certificado.
3. O respondente OCSP envia uma resposta OCSP ao servidor.
4. O servidor envia uma mensagem `ServerHello`. O servidor também envia o certificado do servidor e o status do certificado para o cliente.
5. O cliente valida o status do certificado OCSP.
6. O handshake TLS é concluído.

Como funciona o grampeamento de OCSP

O OCSP stapling é usado durante o handshake de TLS entre o cliente e o servidor para conferir o status de revogação do certificado do servidor. O servidor faz a solicitação OCSP ao respondente OCSP e grampeia as respostas OCSP aos certificados retornados ao cliente. Quando o servidor faz a solicitação ao respondente OCSP, as respostas podem ser armazenadas em cache e usadas várias vezes para muitos clientes.

Como funciona o grampeamento OCSP em AWS IoT Core

O diagrama a seguir mostra como o grampeamento de OCSP do lado do servidor funciona no AWS IoT Core.



1. O dispositivo precisa ser registrado com os domínios personalizados com o grampeamento de OCSP ativado.
2. AWS IoT Core chama o respondente OCSP a cada hora para obter o status do certificado.
3. O respondente OCSP recebe a solicitação, envia a resposta OCSP mais recente e armazena a resposta OCSP em cache.
4. O dispositivo envia uma `ClientHello` mensagem para iniciar o handshake TLS. AWS IoT Core
5. AWS IoT Core obtém a resposta OCSP mais recente do cache do servidor, que responde com uma resposta OCSP do certificado.
6. O servidor envia uma `ServerHello` mensagem para o dispositivo. O servidor também envia o certificado do servidor e o status do certificado para o cliente.
7. O dispositivo valida o status do certificado OCSP.
8. O handshake TLS é concluído.

Benefícios de usar o grampeamento OCSP em comparação com as verificações OCSP do lado do cliente

Algumas vantagens de usar o OCSP stapling de certificado de servidor são as seguintes:

Privacidade aprimorada

Sem o grampeamento de OCSP, o dispositivo do cliente pode expor informações a respondedores OCSP de terceiros, potencialmente comprometendo a privacidade do usuário. O grampeamento de OCSP atenua esse problema fazendo com que o servidor obtenha a resposta OCSP e a entregue diretamente ao cliente.

Confiabilidade aprimorada

O grampeamento de OCSP pode melhorar a confiabilidade de conexões seguras porque reduz o risco de interrupções do servidor OCSP. Quando as respostas OCSP são grampeadas, o servidor inclui a resposta mais recente com o certificado. Isso é para que os clientes tenham acesso ao status de revogação, mesmo que o respondente OCSP esteja temporariamente indisponível. O OCSP stapling ajuda a atenuar esses problemas porque o servidor busca respostas OCSP periodicamente e as inclui em cache no handshake TLS. Isso reduz a dependência da disponibilidade em tempo real dos respondentes OCSP.

Carga reduzida do servidor

O grampeamento de OCSP alivia a carga de responder às solicitações OCSP dos respondentes OCSP para o servidor. Isso pode ajudar a distribuir a carga de maneira mais uniforme, tornando o processo de validação do certificado mais eficiente e escalável.

Latência reduzida

O grampeamento de OCSP reduz a latência associada à verificação do status de revogação de um certificado durante o handshake TLS. Em vez de o cliente precisar consultar um servidor OCSP separadamente, o servidor envia a solicitação e anexa a resposta OCSP ao certificado do servidor durante o handshake.

Habilitando o certificado de servidor OCSP em AWS IoT Core

Para habilitar o grampeamento OCSP do certificado de servidor AWS IoT Core, crie uma configuração de domínio para um domínio personalizado ou atualize uma configuração de domínio personalizado existente. Para obter informações gerais sobre como criar a configuração de um domínio com um domínio personalizado, consulte ???.

Use as instruções a seguir para habilitar o grampeamento do servidor OCSP usando ou. Console de gerenciamento da AWS AWS CLI

Console

Para habilitar o grampeamento OCSP do certificado do servidor usando o console: AWS IoT

1. No menu de navegação, escolha Configurações e selecione Criar configuração de domínio ou escolha uma configuração de domínio existente para um domínio personalizado.
2. Se você optar por criar uma configuração de domínio na etapa anterior, verá a página Criar configuração de domínio. Na seção Propriedades de configuração do domínio, escolha Domínio personalizado. Insira as informações para criar a configuração de um domínio.

   Se você optar por atualizar uma configuração de domínio existente para um domínio personalizado, verá a página de Detalhes da configuração do domínio. Escolha Editar.
3. Para habilitar o grampeamento de servidor OCSP, escolha Ativar grampeamento de OCSP do certificado do servidor na subseção Configurações do certificado do servidor.
4. Escolha Criar configuração de domínio ou Atualizar configuração de domínio.

AWS CLI

Para habilitar o grampeamento de OCSP do certificado do servidor usando a AWS CLI:

1. Se você criar uma nova configuração de domínio para um domínio personalizado, o comando para habilitar o grampeamento de servidor OCSP poderá ser assim:

```
aws iot create-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" \
        --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
        --server-certificate-config "enableOCSPCheck=true|false"
```

2. Se você atualizar uma configuração de domínio existente para um domínio personalizado, o comando para habilitar o grampeamento de servidor OCSP poderá ser assim:

```
aws iot update-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" \
```

```
        --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
        --server-certificate-config "enableOCSPCheck=true|false"
```

Para obter mais informações, consulte CreateDomainConfiguratione a UpdateDomainConfigurationpartir da Referência AWS IoT da API.

Configurando o certificado de servidor OCSP para endpoints privados em AWS IoT Core

O OCSP para endpoints privados permite que você use seus recursos OCSP privados em sua Amazon Virtual Private Cloud (Amazon VPC) para operações. AWS IoT Core O processo envolve a configuração de uma função do Lambda que atue como um respondente OCSP. A função Lambda pode usar seus recursos OCSP privados para criar respostas OCSP que serão usadas. AWS IoT Core

Função do Lambda

Antes de configurar o servidor OCSP para um endpoint privado, crie uma função do Lambda que atue como um respondente OCSP compatível com RFC 6960, aceitando respostas OCSP básicas. A função do Lambda aceita uma codificação base64 da solicitação OCSP no formato DER. A resposta da função do Lambda também é uma resposta OCSP codificada em base64 no formato DER. O tamanho da resposta não deve exceder 4 kilobytes (KiB). A função Lambda deve estar na Região da AWS mesma Conta da AWS configuração do domínio. Os exemplos a seguir são de funções do Lambda.

Exemplo de funções do Lambda

JavaScript

```
import * as pkijs from 'pkijs';
console.log('Loading function');

export const handler = async (event, context) => {
    const requestBytes = decodeBase64(event);
    const ocspRequest = pkijs.OCSPRequest.fromBER(requestBytes);

    console.log("Here is a better look at the OCSP request");
    console.log(ocspRequest.toJSON());
```

```
    const ocspResponse = getOcspResponse();

    console.log("Here is a better look at the OCSP response");
    console.log(ocspResponse.toJSON());

   const responseBytes = ocspResponse.toSchema().toBER();
   return encodeBase64(responseBytes);
};

function getOcspResponse() {
    const responseString = "MIIC/
woBAKCCAvgwggL0BgkrBgEFBQcwAQEEggLlMIIC4TCByqFkMGIxJzAlBgNVBAoMHlJpY2hhcmQncyBEaXNjb3VudCBMY
p5w7WOtPjp3otNtVgIBAYAAGA8yMDI0MDQyMzE4NTMyNVowDQYJKoZIhvcNAQELBQADggIBAJFRyjDAHfazNejo704Ra
+s82R1spDarr3k7Pzkod9jJhwsZ2YgushlS4Npfe4lHCdwFyZR75WXrW55aXFddy03KLz01ZLNYyxkleW3f5dgrUcRU3
DEBiyS7ZsyhKo6igWU/SY7YMSKgwBvFsqSDcOa/hRYQkxWKWJ19gcz8CIkWN7NvfIxCs6VrAdzEJwmE7y3v
+jdfhxW9JmI4xStE4K0tAR9vVOOfKs7NvxXj7oc9pCSG60xl96kaEE6PaY1YsfNTsKQ7pyCJ0s7/2q
+ieZ4AtNyzw1XBadPzPJNv6E0LvI24yQZqN5wACvtut5prMMRxAHbOy
+abLZR58wloFSELtGJ7UD96LFv1GgtC5s
+2QlzPc4bEEof7Lo1EISt3j2ibNch8LxhqTQ4ufrbhsMkpSOTFYEJVMJF6aKj/OGXBUUqgc0Jx6jjJXNQd
+l5KCY9pQFeb/wVUYC6mYqZOkNNMMJxPbHHbFnqb68yO+g5BE9011N44YXoPVJYoXxBLFX+OpRu9cqPkT9/
vlkKd+SYXQknwZ81agKzhf1HsBKabtJwNVMlBKaI8g5UGa7Bxi6ewH3ezdWiERRUK7F56OM53wto/";
    const responseBytes = decodeBase64(responseString);
    return pkijs.OCSPResponse.fromBER(responseBytes);
}

function decodeBase64(input) {
    const binaryString = atob(input);

    const byteArray = new Uint8Array(binaryString.length);
    for (var i = 0; i < binaryString.length; i++) {
        byteArray[i] = binaryString.charCodeAt(i);
    }

    return byteArray.buffer;
}

function encodeBase64(buffer) {
    var binary = '';
    const bytes = new Uint8Array( buffer );
    const len = bytes.byteLength;

    for (var i = 0; i < len; i++) {
        binary += String.fromCharCode( bytes[ i ] );
    }
```

```
    return btoa(binary);
}
```

Java

```
package com.example.ocsp.responder;
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import org.bouncycastle.cert.ocsp.OCSPReq;
import org.bouncycastle.cert.ocsp.OCSPResp;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Base64;

public class LambdaResponderApplication implements RequestHandler<String, String> {
    @Override
    public String handleRequest(final String input, final Context context) {
        LambdaLogger logger = context.getLogger();

        byte[] decodedInput = Base64.getDecoder().decode(input);

        OCSPReq req;
        try {
            req = new OCSPReq(decodedInput);
        } catch (IOException e) {
            logger.log("Got an IOException creating the OCSP request: " +
 e.getMessage());
            throw new RuntimeException(e);
        }

        try {
            OCSPResp response = businessLogic.getMyResponse();
            String toReturn =
 Base64.getEncoder().encodeToString(response.getEncoded());
            return toReturn;
        } catch (Exception e) {
            logger.log("Got an exception creating the response: " + e.getMessage());
            return "";
        }
    }
```

```
}
```

Autorizando AWS IoT a invocação da função Lambda

No processo de criação da configuração do domínio com um respondente Lambda OCSP, você deve conceder AWS IoT permissão para invocar a função Lambda após a criação da função. Para conceder a permissão, você pode usar o comando da CLI add-permission.

Conceda permissão para sua função Lambda usando o AWS CLI

1. Depois de inserir os valores, digite o seguinte comando. Observe que o valor `statement-id` deve ser exclusivo. Substitua *Id-1234* pelo valor exato que você tem, caso contrário, você poderá receber um erro de `ResourceConflictException`.

   ```
   aws lambda add-permission  \
   --function-name "ocsp-function" \
   --principal "iot.amazonaws.com" \
   --action "lambda:InvokeFunction" \
   --statement-id "Id-1234" \
   --source-arn arn:aws:iot:us-east-1:123456789012:domainconfiguration/<domain-config-name>/*
   --source-account 123456789012
   ```

   A configuração do domínio de IoT ARNs seguirá o seguinte padrão. O sufixo gerado pelo serviço não será conhecido antes do momento da criação, portanto, você deve substituí-lo por um *. Você pode atualizar a permissão depois que a configuração do domínio for criada e o ARN exato for conhecido.

   `arn:aws:iot:use-east-1:123456789012:domainconfiguration/domain-config-name/service-generated-suffix`

2. Se o comando for executado com êxito, ele retornará uma declaração de permissão, como neste exemplo. Você pode prosseguir para a próxima seção para configurar o OCSP stapling para endpoints privados.

   ```
   {
       "Statement": "{\"Sid\":\"Id-1234\",\"Effect\":\"Allow\",\"Principal
   \":{\"Service\":\"iot.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction
   \",\"Resource\":\"arn:aws:lambda:us-east-1:123456789012:function:ocsp-
   ```

```
function\",\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":\"arn:aws:iot:us-
east-1:123456789012:domainconfiguration/domain-config-name/*\"}}}"
}
```

Se o comando não for executado com êxito, ele retornará um erro, como neste exemplo. Será necessário verificar e corrigir o erro antes de continuar.

```
An error occurred (AccessDeniedException) when calling the AddPermission operation:
 User: arn:aws:iam::57EXAMPLE833:user/EXAMPLE-1 is not authorized to perform:
 lambda:AddPer
mission on resource: arn:aws:lambda:us-east-1:123456789012:function:ocsp-function
```

Configurar o OCSP stapling do servidor para endpoints privados

Console

Para configurar o grampeamento OCSP do certificado do servidor usando o console: AWS IoT

1. No menu de navegação, escolha Configurações e selecione Criar configuração de domínio ou uma configuração de domínio existente para um domínio personalizado.
2. Se você optar por criar uma configuração de domínio na etapa anterior, verá a página Criar configuração de domínio. Na seção Propriedades de configuração do domínio, escolha Domínio personalizado. Insira as informações para criar a configuração de um domínio.

   Se você optar por atualizar uma configuração de domínio existente para um domínio personalizado, verá a página de Detalhes da configuração do domínio. Escolha Editar.
3. Para habilitar o grampeamento de servidor OCSP, escolha Ativar grampeamento de OCSP do certificado do servidor na subseção Configurações do certificado do servidor.
4. Escolha Criar configuração de domínio ou Atualizar configuração de domínio.

AWS CLI

Para configurar o OCSP stapling do certificado do servidor usando a AWS CLI:

1. Se você criar uma configuração para um domínio personalizado, o comando para configurar o OCSP do certificado do servidor para endpoints privados poderá ser assim:

```
aws iot create-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" \
        --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
        --server-certificate-config "enableOCSPCheck=true,
 ocspAuthorizedResponderArn=arn:aws:acm:us-
east-1:123456789012:certificate/certificate_ID, ocspLambdaArn=arn:aws:lambda:us-
east-1:123456789012:function:my-function"
```

2. Se você atualizar uma configuração de domínio existente para um domínio personalizado, o comando para configurar o OCSP do certificado do servidor para endpoints privados poderá ser assim:

```
aws iot update-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" \
        --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
        --server-certificate-config "enableOCSPCheck=true,
 ocspAuthorizedResponderArn=arn:aws:acm:us-
east-1:123456789012:certificate/certificate_ID, ocspLambdaArn=arn:aws:lambda:us-
east-1:123456789012:function:my-function"
```

enableOCSPCheck

Esse é um valor booliano que indica se a verificação do OCSP stapling do servidor está habilitada ou não. Para habilitar o OCSP stapling do certificado do servidor, esse valor deve ser verdadeiro.

ocspAuthorizedResponderArn

Esse é um valor de string do Amazon Resource Name (ARN) para um certificado X.509 armazenado em AWS Certificate Manager (ACM). Se fornecido, AWS IoT Core usará esse certificado para validar a assinatura da resposta OCSP recebida. Se não for fornecido, AWS IoT Core usará o certificado de emissão para validar as respostas. O certificado deve estar na mesma configuração Conta da AWS e na Região da AWS mesma configuração do domínio. Para ter mais informações sobre como registrar seu certificado de respondente autorizado, consulte Importar certificados para o AWS Certificate Manager.

ocspLambdaArn

Esse é um valor de string do nome do recurso da Amazon (ARN) para uma função do Lambda que atua como um respondente OCSP compatível com RFC 6960, aceitando respostas OCSP básicas. A função do Lambda aceita uma codificação base64 da solicitação OCSP, que é codificada usando o formato DER. A resposta da função do Lambda também é uma resposta OCSP codificada em base64 no formato DER. O tamanho da resposta não deve exceder 4 kilobytes (KiB). A função Lambda deve estar na Região da AWS mesma Conta da AWS configuração do domínio.

Para obter mais informações, consulte CreateDomainConfiguratione a UpdateDomainConfigurationpartir da Referência AWS IoT da API.

Notas importantes sobre o uso do grampeamento OCSP do certificado de servidor em AWS IoT Core

Ao usar o certificado de servidor OCSP em AWS IoT Core, lembre-se do seguinte:

1. AWS IoT Core suporta somente os respondedores OCSP que podem ser acessados por endereços públicos. IPv4
2. O recurso de grampeamento OCSP AWS IoT Core não oferece suporte a respondedores autorizados. Todas as respostas do OCSP devem ser assinadas pela CA que assinou o certificado, e a CA deve fazer parte da cadeia de certificados do domínio personalizado.
3. O recurso de grampeamento OCSP AWS IoT Core não oferece suporte a domínios personalizados criados usando certificados autoassinados.
4. AWS IoT Core chama um respondente OCSP a cada hora e armazena a resposta em cache. Se a chamada para o respondente falhar, AWS IoT Core grampeará a resposta válida mais recente.
5. Se não `nextUpdateTime` for mais válido, AWS IoT Core removerá a resposta do cache e o handshake TLS não incluirá os dados da resposta OCSP até a próxima chamada bem-sucedida para o respondente OCSP. Isso pode acontecer quando a resposta em cache expira antes que o servidor receba uma resposta válida do respondente OCSP. O valor de `nextUpdateTime` sugere que a resposta do OCSP é válida até esse momento. Para saber mais sobre o `nextUpdateTime`, consulte ???.
6. Às vezes, AWS IoT Core falha ao receber a resposta OCSP ou remove a resposta OCSP existente porque ela expirou. Se situações como essas acontecerem, AWS IoT Core continuará usando o certificado de servidor fornecido pelo domínio personalizado sem a resposta OCSP.
7. O tamanho da resposta OCSP não pode exceder 4 KiB.

Solucionando problemas de grampeamento do certificado OCSP do servidor AWS IoT Core

AWS IoT Core emite a `RetrieveOCSPStapleData.Success` métrica e as entradas de `RetrieveOCSPStapleData` registro para CloudWatch. A métrica e as entradas de log podem ajudar a detectar problemas relacionados a recuperar respostas do OCSP. Para obter mais informações, consulte ??? e ???.

Conecte-se aos AWS IoT endpoints FIPS

AWS IoT fornece endpoints que suportam o Padrão Federal de Processamento de Informações (FIPS) 140-2. Os endpoints compatíveis com FIPS são diferentes dos endpoints padrão. AWS Para interagir com a AWS IoT de maneira compatível com FIPS, é necessário usar os endpoints descritos abaixo com seu cliente compatível com FIPS. O AWS IoT console não é compatível com FIPS.

As seções a seguir descrevem como acessar os AWS IoT endpoints compatíveis com FIPS usando a API REST, um SDK ou o. AWS CLI

Tópicos

- AWS IoT Core - endpoints do ambiente de gerenciamento
- Endpoints do plano de dados do AWS IoT Core
- AWS IoT Core: endpoints do provedor de credenciais
- Endpoints de dados de trabalhos do AWS IoT Device Management
- Endpoints do Fleet Hub do AWS IoT Device Management
- Endpoints de encapsulamento seguro do AWS IoT Device Management
- AWS IoT Device Management - Endpoints de integrações gerenciadas

AWS IoT Core - endpoints do ambiente de gerenciamento

Os endpoints do ambiente de gerenciamento do AWS IoT Core compatíveis com FIPS que oferecem suporte às operações de AWS IoT e seus comandos CLI relacionados estão listados em Endpoints FIPS por serviço. Em Endpoints FIPS por serviço, encontre o serviço do ambiente de gerenciamento do AWS IoT Core e procure o endpoint para a sua Região da AWS.

Para usar o endpoint compatível com FIPS ao acessar AWS IoTas operações, use o AWS SDK ou a API REST com o endpoint apropriado para você. Região da AWS

Para usar o endpoint compatível com FIPS ao executar comandos CLI da aws iot, adicione ao comando o parâmetro --endpoint com o endpoint apropriado para sua Região da AWS .

Endpoints do plano de dados do AWS IoT Core

Os endpoints do plano de dados do AWS IoT Core compatíveis com FIPS estão listados em Endpoints FIPS por serviço. Em Endpoints FIPS por serviço, encontre o serviço do plano de dados do AWS IoT Core e procure o endpoint para a sua Região da AWS.

Você pode usar o endpoint compatível com FIPS para você Região da AWS com um cliente compatível com FIPS usando o AWS IoT Device SDK e fornecendo o endpoint para a função de conexão do SDK no lugar do endpoint padrão do plano de dados da sua conta.AWS IoT Core A função de conexão é específica para o AWS IoT Device SDK. Para ver um exemplo de função de conexão, consulte a função Connection no AWS IoT Device SDK para Python.

> **Note**
>
> AWS IoT não oferece suporte a endpoints Conta da AWS de plano de dados específicos AWS IoT Core que sejam compatíveis com FIPS. Os recursos do serviço que exigem um endpoint Conta da AWS específico na Indicação de Nome do Servidor (SNI) não podem ser usados. Os endpoints do plano de dados do AWS IoT Core compatíveis com FIPS não podem oferecer suporte a certificados de registro de várias contas, domínios personalizados, autorizadores personalizados e endpoints configuráveis (incluindo Políticas TLS compatíveis).

AWS IoT Core: endpoints do provedor de credenciais

Os endpoints do provedor de credenciais do AWS IoT Core compatíveis com FIPS estão listados em Endpoints FIPS por serviço. Em Endpoints FIPS por serviço, encontre o serviço provedor de credenciais do AWS IoT Core e procure o endpoint para sua Região da AWS.

> **Note**
>
> AWS IoT não oferece suporte a endpoints Conta da AWS de provedores de credenciais específicos AWS IoT Core que sejam compatíveis com FIPS. Os recursos do serviço que exigem um endpoint Conta da AWS específico na Indicação de Nome do Servidor (SNI) não podem ser usados. Os endpoints do provedor de credenciais do AWS IoT Core compatíveis com FIPS não podem oferecer suporte a certificados de registro de várias contas, domínios

personalizados, autorizadores personalizados e endpoints configuráveis (incluindo Políticas TLS compatíveis).

Endpoints de dados de trabalhos do AWS IoT Device Management

Os endpoints dos dados de trabalhos do AWS IoT Device Management compatíveis com FIPS estão listados em Endpoints FIPS por serviço. Em Endpoints FIPS por serviço, encontre o serviço dos dados de trabalhos do AWS IoT Device Management e procure o endpoint para a sua Região da AWS.

Para usar o endpoint de dados de trabalhos do AWS IoT Device Management compatível com FIPS ao executar comandos aws iot-jobs-data da CLI, adicione ao comando o parâmetro --endpoint com o endpoint apropriado para sua Região da AWS . Você também pode usar a API REST com esse endpoint.

Recomendamos usar `Data-ATS` em vez de`iot:Jobs`. `iot:Data-ATS`oferece suporte a endpoints de pilha dupla (IPv4 e IPv6), embora ofereça suporte apenas. `iot:Jobs` IPv4

Você pode usar o endpoint compatível com FIPS para você Região da AWS com um cliente compatível com FIPS usando o AWS IoT Device SDK e fornecendo o endpoint para a função de conexão do SDK no lugar do endpoint de dados de trabalhos padrão da sua conta.AWS IoT Device Management A função de conexão é específica do SDK do dispositivo de AWS IoT . Para ver um exemplo de função de conexão, consulte a função Connection no AWS IoT Device SDK para Python.

Endpoints do Fleet Hub do AWS IoT Device Management

Os endpoints Fleet Hub compatíveis AWS IoT Device Management com FIPS a serem usados com os comandos CLI do Fleet Hub para AWS IoT gerenciamento de dispositivos estão listados em FIPS Endpoints by Service. Em Endpoints FIPS por serviço, encontre o serviço do Fleet Hub do AWS IoT Device Management e procure o endpoint para a sua Região da AWS.

Para usar o endpoint Fleet Hub compatível com AWS IoT Device Management FIPS ao executar comandos aws iotfleethub CLI, adicione o --endpoint parâmetro com o endpoint apropriado para você ao comando. Região da AWS Você também pode usar a API REST com esse endpoint.

Endpoints de encapsulamento seguro do AWS IoT Device Management

Os endpoints de encapsulamento seguro do AWS IoT Device Management compatíveis com FIPS para a API de encapsulamento seguro de AWS IoT e os comandos CLI correspondentes estão

listados em Endpoints FIPS por serviço. Em Endpoints FIPS por serviço, encontre o serviço de encapsulamento seguro do AWS IoT Device Management e procure o endpoint para a sua Região da AWS.

Para usar o endpoint de secure tunneling do AWS IoT Device Management compatível com FIPS ao executar comandos aws iotsecuretunneling da CLI, adicione ao comando o parâmetro --endpoint com o endpoint apropriado para sua Região da AWS . Você também pode usar a API REST com esse endpoint.

AWS IoT Device Management - Endpoints de integrações gerenciadas

Os endpoints do plano de controle compatíveis com FIPS que suportam as operações de integrações gerenciadas e seus AWS CLI comandos relacionados estão listados em FIPS Endpoints by Service. Em Endpoints FIPS por serviço, encontre o serviço AWS IoT Device Management : integrações gerenciadas e procure o endpoint para a sua Região da AWS.

Para usar o endpoint compatível com FIPS ao acessar as operações de integrações gerenciadas, use o AWS SDK ou a API REST com o endpoint apropriado para você. Região da AWS

Para usar o endpoint compatível com FIPS ao executar comandos da CLI de integrações gerenciadas, adicione ao comando o parâmetro --endpoint com o endpoint apropriado para sua Região da AWS .

Gerenciando dispositivos com AWS IoT

AWS IoT fornece um registro que ajuda você a gerenciar coisas. Um item é uma representação de um dispositivo específico ou entidade lógica. Ela pode ser um dispositivo físico ou sensor (por exemplo, uma lâmpada ou um interruptor em uma parede). Também pode ser uma entidade lógica, como uma instância de um aplicativo ou entidade física que não se conecta, AWS IoT mas está relacionada a outros dispositivos que o fazem (por exemplo, um carro com sensores de motor ou um painel de controle).

As informações sobre um objeto estão armazenadas no registro como dados JSON. Veja um exemplo de objeto:

```
{
     "version": 3,
    "thingName": "MyLightBulb",
    "defaultClientId": "MyLightBulb",
    "thingTypeName": "LightBulb",
    "attributes": {
        "model": "123",
        "wattage": "75"
    }
}
```

As objetos são identificadas por um nome. As objetos também podem ter atributos, que são pares de nome-valor que você pode usar para armazenar informações sobre o objeto, como seu número de série ou fabricante.

Um caso típico de uso de dispositivos envolve o uso do nome do objeto como o ID de cliente MQTT padrão. Embora não imponhamos um mapeamento entre o nome de registro de uma coisa e o uso do cliente MQTT IDs, dos certificados ou do estado sombra, recomendamos que você escolha um nome de coisa e o use como ID do cliente MQTT tanto para o registro quanto para o serviço Device Shadow. Isso proporciona organização e conveniência para sua frota de IoT sem remover a flexibilidade do modelo de certificado do dispositivo ou sombras subjacente.

Não é necessário criar um objeto no registro para conectar um dispositivo à AWS IoT. Adicione objetos ao registro permite que você gerencie e faça pesquisa por dispositivos com mais facilidade.

Os dados do registro de coisas (incluindo atributos, tipos de coisas e associações de grupos) também podem ser recuperados dinamicamente no mecanismo de regras para uso

no processamento e roteamento de mensagens. Para obter mais informações, consulte `get_registry_data`.

Gerenciar objetos com o Registro

Você usa o AWS IoT console, a AWS IoT API ou o AWS CLI para interagir com o registro. As seções a seguir mostram como usar a CLI para trabalhar com o registro.

Ao nomear objetos do objeto:

- Não use informações de identificação pessoal nos nomes de objetos. O nome do objeto pode surgir em comunicações e relatórios não criptografados.

Tópicos

- Criar um objeto
- Listar objetos
- Descreva as objetos
- Atualizar um objeto
- Excluir um objeto
- Anexar uma entidade principal a um objeto
- Listar coisas associadas a uma entidade principal
- Listar as entidades principais associadas a uma coisa
- Listar as coisas associadas a uma entidade principal V2
- Listar as entidades principais associadas a uma coisa V2
- Desanexar uma entidade principal de um objeto

Criar um objeto

O comando a seguir mostra como usar o AWS IoT CreateThing comando da CLI para criar algo. Você não pode mudar o nome de um objeto depois de criá-la. Para alterar o nome de um objeto, crie um objeto, dê o novo nome e exclua o objeto antigo.

```
$ aws iot create-thing \
    --thing-type-name "MyLightBulb" \
    --attribute-payload "{\"attributes\": {\"wattage\":\"75\", \"model\":\"123\"}}"
```

O comando CreateThing exibe o nome e o nome do recurso da Amazon (ARN) do novo objeto:

```
{
    "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyLightBulb",
    "thingName": "MyLightBulb",
    "thingId": "12345678abcdefgh12345678ijklmnop12345678"
}
```

> **Note**
>
> Não recomendamos o uso de informações pessoais identificáveis nos nomes das objetos.

Para obter mais informações, consulte create-cluster na Referência do comando AWS CLI .

Listar objetos

Você pode usar o comando ListThings para listar todas as objetos em sua conta:

```
$ aws iot list-things
```

```
{
    "things": [
       {
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MyLightBulb"
        },
        {
            "attributes": {
                "numOfStates":"3"
             },
            "version": 11,
            "thingName": "MyWallSwitch"
        }
    ]
}
```

Você pode usar o comando ListThings para pesquisar todas as objetos associadas a um tipo de objeto específica:

```
$  aws iot list-things --thing-type-name "LightBulb"
```

```
{
    "things": [
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MyRGBLight"
        },
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MySecondLightBulb"
        }
    ]
}
```

Você pode usar o comando ListThings para pesquisar todas as objetos que tenham um atributo com um valor específico. Esse comando pesquisa até três atributos.

```
$  aws iot list-things --attribute-name "wattage" --attribute-value "75"
```

```
{
    "things": [
        {
            "thingTypeName": "StopLight",
            "attributes": {
                "model": "123",
                "wattage": "75"
```

```
            },
            "version": 3,
            "thingName": "MyLightBulb"
        },
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MyRGBLight"
        },
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MySecondLightBulb"
        }
    ]
}
```

Para obter mais informações, consulte list-domains na Referência do comando AWS CLI .

Descreva as objetos

Você pode usar o comando DescribeThing para exibir informações mais detalhadas sobre um objeto:

```
$ aws iot describe-thing --thing-name "MyLightBulb"
{
    "version": 3,
    "thingName": "MyLightBulb",
    "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyLightBulb",
    "thingId": "12345678abcdefgh12345678ijklmnop12345678",
    "defaultClientId": "MyLightBulb",
    "thingTypeName": "StopLight",
    "attributes": {
        "model": "123",
        "wattage": "75"
    }
```

```
}
```

Você também pode acessar essa API dentro do mecanismo de regras usando a função `get_registry_data()` embutida. Você pode usar essa função para acessar e utilizar dinamicamente as informações do registro de coisas (incluindo atributos, tipos de coisas e associações de grupos) ligando e `ListThingGroupsForThing` APIs diretamente dentro AWS IoT das regras, permitindo o processamento `DescribeThing` e o roteamento de mensagens em tempo real com base nos dados de registro do seu dispositivo. Para obter mais informações, consulte `get_registry_data`.

Para obter mais informações, consulte describe-thing na Referência de Comandos. AWS CLI

Atualizar um objeto

Você pode usar o comando UpdateThing para atualizar um objeto. Esse comando atualiza apenas os atributos do objeto. Você não pode mudar o nome de um objeto. Para alterar o nome de um objeto, crie um objeto, dê o novo nome e exclua o objeto antigo.

```
$ aws iot update-thing --thing-name "MyLightBulb" --attribute-payload "{\"attributes\":
 {\"wattage\":\"150\", \"model\":\"456\"}}"
```

O comando UpdateThing não produz saída. Você pode usar o comando DescribeThing para exibir o resultado:

```
$ aws iot describe-thing --thing-name "MyLightBulb"
{
    "attributes": {
        "model": "456",
        "wattage": "150"
    },
    "version": 2,
    "thingName": "MyLightBulb"
}
```

Para obter mais informações, consulte update-bucket na Referência do comando AWS CLI .

Excluir um objeto

Você pode usar o comando DeleteThing para excluir um objeto:

```
$ aws iot delete-thing --thing-name "MyThing"
```

Esse comando retornará com êxito sem erros se a exclusão for bem-sucedida ou se você especificar algo que não existe.

Para obter mais informações, consulte delete-objects na Referência do comando AWS CLI .

Anexar uma entidade principal a um objeto

Um dispositivo físico pode usar um principal para se comunicar com AWS IoT. Uma entidade principal pode ser um certificado X.509 ou um ID do Amazon Cognito. Você pode associar um certificado ou um ID do Amazon Cognito à coisa no registro que representa seu dispositivo, executando o attach-thing-principalcomando.

Para anexar um certificado ou um ID do Amazon Cognito à sua coisa, use o attach-thing-principalcomando:

```
$ aws iot attach-thing-principal \
    --thing-name "MyLightBulb1" \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

Para anexar um certificado à sua coisa com um tipo de anexo (anexo exclusivo ou anexo não exclusivo), use o attach-thing-principalcomando e especifique um tipo no `--thing-principal-type` campo. Um anexo exclusivo significa que sua coisa de IoT é a única coisa anexada ao certificado, e este não pode ser associado a nenhuma outra coisa. Um anexo não exclusivo significa que sua coisa de IoT está anexada ao certificado, e este pode ser associado a outras coisas. Para obter mais informações, consulte ???.

> **Note**
>
> Para o recurso ???, só é possível usar o certificado X.509 como entidade principal.

```
$ aws iot attach-thing-principal \
    --thing-name "MyLightBulb2" \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847" \
```

```
    --thing-principal-type "EXCLUSIVE_THING"
```

Se o anexo for bem-sucedido, o comando AttachThingPrincipal não produzirá nenhuma saída. Para descrever o anexo, use list-thing-principals-v 2 comandos CLI.

Para obter mais informações, consulte AttachThingPrincipal na Referência de APIs de AWS IoT Core .

Listar coisas associadas a uma entidade principal

Para listar as coisas associadas à entidade principal especificada, execute o comando `list-principal-things`. Observe que esse comando não lista o tipo de anexo entre a coisa e o certificado. Para listar o tipo de anexo, use o comando `list-principal-things-v2`. Para obter mais informações, consulte ???.

```
$ aws iot list-principal-things \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
```

A saída poderá ser parecida com a seguinte:

```
{
    "things": [
        "MyLightBulb1",
        "MyLightBulb2"
    ]
}
```

Para obter mais informações, consulte ListPrincipalThings na Referência de APIs de AWS IoT Core .

Listar as entidades principais associadas a uma coisa

Para listar as entidades principais associadas à coisa especificada, execute o comando `list-thing-principals`. Observe que esse comando não lista o tipo de anexo entre a coisa e o certificado. Para listar o tipo de anexo, use o comando `list-thing-principals-v2`. Para obter mais informações, consulte ???.

```
$ aws iot list-thing-principals \
    --thing-name "MyLightBulb1"
```

A saída poderá ser parecida com a seguinte:

```
{
    "principals": [
        "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8",
        "arn:aws:iot:us-
east-1:123456789012:cert/
1a234b39b4b68278f2e9d84bf97eac2cbf4a1c28b23ea29a44559b9bcf8d395b"
    ]
}
```

Para obter mais informações, consulte ListThingPrincipals na Referência de APIs de AWS IoT Core .

Listar as coisas associadas a uma entidade principal V2

Para listar as coisas associadas ao certificado especificado, junto com o tipo de anexo, execute o comando `list-principal-things-v2`. O tipo de anexo se refere à forma como o certificado é anexado à coisa.

```
$ aws iot list-principal-things-v2 \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
```

A saída poderá ser parecida com a seguinte:

```
{
    "PrincipalThingObjects": [
        {
            "thingPrincipalType": "NON_EXCLUSIVE_THING",
            "thing": "arn:aws:iot:us-east-1:123456789012:thing/thing_1"
        },
        {
            "thingPrincipalType": "NON_EXCLUSIVE_THING",
            "thing": "arn:aws:iot:us-east-1:123456789012:thing/thing_2"
        }

    ]
}
```

Para obter mais informações, consulte ListPrincipalThingsV2 da Referência da AWS IoT Core API.

Listar as entidades principais associadas a uma coisa V2

Para listar os certificados associados à coisa especificada, junto com o tipo de anexo, execute o comando `list-thing-principals-V2`. O tipo de anexo se refere à forma como o certificado é anexado à coisa.

```
$ aws iot list-thing-principals-v2 \
    --thing-name "thing_1"
```

A saída poderá ser parecida com a seguinte:

```
{
    "ThingPrincipalObjects": [
        {
            "thingPrincipalType": "NON_EXCLUSIVE_THING",
            "principal": "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
        },
        {
            "thingPrincipalType": "NON_EXCLUSIVE_THING",
            "principal": "arn:aws:iot:us-
east-1:123456789012:cert/
1a234b39b4b68278f2e9d84bf97eac2cbf4a1c28b23ea29a44559b9bcf8d395b"
        }
    ]
}
```

Para obter mais informações, consulte ListThingsPrincipalV2 da Referência da AWS IoT Core API.

Desanexar uma entidade principal de um objeto

Você pode usar o comando `DetachThingPrincipal` para desanexar um certificado de um objeto:

```
$ aws iot detach-thing-principal \
    --thing-name "MyLightBulb" \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
```

O comando DetachThingPrincipal não produz saída.

Para obter mais informações, consulte detach-thing-principal na Referência de APIs de AWS IoT Core .

Tipos de objeto

Os tipos de objeto permitem que você armazene descrições e informações de configuração que sejam comuns a todas as objetos associadas com o mesmo tipo. Isso simplifica o gerenciamento de objetos no registro. Por exemplo, você pode definir um tipo de LightBulb coisa. Todas as coisas associadas ao tipo de LightBulb coisa compartilham um conjunto de atributos: número de série, fabricante e potência. Quando você cria uma coisa do tipo LightBulb (ou altera o tipo de uma coisa existente para LightBulb), você pode especificar valores para cada um dos atributos definidos no tipo da LightBulb coisa.

Embora os tipos de objeto sejam opcionais, seu uso facilita a descobrir objetos.

- As objetos com um tipo de objeto podem ter até 50 atributos.
- As objetos sem um tipo de objeto podem ter até três atributos.
- Umo objeto só pode ser associada a um tipo de objeto.
- Não há limite para o número de tipos de objeto que você pode criar em sua conta.

Não é possível alterar um nome do tipo de objeto após sua criação. Você pode reprovar um tipo de objeto a qualquer momento para impedir que novas objetos sejam associadas a ele. Você também pode excluir tipos de objeto que não tenham objetos associadas a eles.

Tópicos:

- Criar um tipo de objeto
- Listar tipos de objeto
- Descrever um tipo de objeto
- Associar um tipo de objeto a um objeto
- Atualizar um tipo de coisa
- Reprovar um tipo de objeto
- Excluir um tipo de objeto

Criar um tipo de objeto

Você pode usar o comando CreateThingType para criar um tipo de objeto:

```
$ aws iot create-thing-type

                --thing-type-name "LightBulb" --thing-type-properties
 "thingTypeDescription=light bulb type, searchableAttributes=wattage,model"
```

O comando CreateThingType retorna uma resposta que contém o tipo de objeto e seu ARN:

```
{
    "thingTypeName": "LightBulb",
    "thingTypeId": "df9c2d8c-894d-46a9-8192-9068d01b2886",
    "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb"
}
```

Listar tipos de objeto

Você pode usar o comando ListThingTypes para listar tipos de objeto:

```
$ aws iot list-thing-types
```

O ListThingTypes comando retorna uma lista dos tipos de coisas definidos em seu Conta da AWS:

```
{
    "thingTypes": [
        {
            "thingTypeName": "LightBulb",
            "thingTypeProperties": {
                "searchableAttributes": [
                    "wattage",
                    "model"
                ],
                "thingTypeDescription": "light bulb type"
            },
            "thingTypeMetadata": {
                "deprecated": false,
                "creationDate": 1468423800950
            }
        }
```

```
    ]
}
```

Descrever um tipo de objeto

Você pode usar o comando DescribeThingType para obter informações sobre um tipo de objeto:

```
$ aws iot describe-thing-type --thing-type-name "LightBulb"
```

O comando DescribeThingType retorna informações sobre o tipo especificado:

```
{
    "thingTypeProperties": {
        "searchableAttributes": [
            "model",
            "wattage"
        ],
        "thingTypeDescription": "light bulb type"
    },
    "thingTypeId": "df9c2d8c-894d-46a9-8192-9068d01b2886",
    "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb",
    "thingTypeName": "LightBulb",
    "thingTypeMetadata": {
        "deprecated": false,
        "creationDate": 1544466338.399
    }
}
```

Associar um tipo de objeto a um objeto

Você pode usar o comando CreateThing para especificar um tipo de objeto ao criar um objeto:

```
$ aws iot create-thing --thing-name "MyLightBulb" --thing-type-name "LightBulb" --
attribute-payload "{\"attributes\": {\"wattage\":\"75\", \"model\":\"123\"}}"
```

Você pode usar o comando UpdateThing a qualquer momento para alterar o tipo de objeto associado a um objeto:

```
$ aws iot update-thing --thing-name "MyLightBulb"
               --thing-type-name "LightBulb" --attribute-payload  "{\"attributes\":
 {\"wattage\":\"75\", \"model\":\"123\"}}"
```

Você também pode usar o comando UpdateThing para desassociar um objeto de um tipo de objeto.

Atualizar um tipo de coisa

Você pode usar o comando UpdateThingType para atualizar um tipo de coisa ao criá-la:

```
$ aws iot create-thing --thing-name "MyLightBulb" --thing-type-name "LightBulb" --
attribute-payload "{\"attributes\": {\"wattage\":\"75\", \"model\":\"123\"}}"
```

Você pode usar o comando UpdateThing a qualquer momento para alterar o tipo de objeto associado a um objeto:

```
$ aws iot update-thing --thing-name "MyLightBulb"
                --thing-type-name "LightBulb" --attribute-payload  "{\"attributes\":
 {\"wattage\":\"75\", \"model\":\"123\"}}"
```

Você também pode usar o comando UpdateThing para desassociar um objeto de um tipo de objeto.

Reprovar um tipo de objeto

Os tipos de objeto são imutáveis. Eles não podem ser alterados depois que são definidos. No entanto, você pode reprovar um tipo de objeto para impedir que os usuários associem novas objetos a ele. Todas as objetos existentes associadas ao tipo de objeto são inalteradas.

Para tornar um tipo de objeto obsoleto, use o comando DeprecateThingType:

```
$ aws iot deprecate-thing-type --thing-type-name "myThingType"
```

Você pode usar o comando DescribeThingType para exibir o resultado:

```
$ aws iot describe-thing-type --thing-type-name "StopLight":
```

```
{
    "thingTypeName": "StopLight",
    "thingTypeProperties": {
        "searchableAttributes": [
            "wattage",
            "numOfLights",
            "model"
```

```
        ],
        "thingTypeDescription": "traffic light type",
    },
    "thingTypeMetadata": {
        "deprecated": true,
        "creationDate": 1468425854308,
        "deprecationDate": 1468446026349
    }
}
```

Reprovar um tipo de objeto é uma operação reversível. Você pode desfazer uma reprovação usando o sinalizador `--undo-deprecate` com o comando CLI DeprecateThingType:

```
$ aws iot deprecate-thing-type --thing-type-name "myThingType" --undo-deprecate
```

Você pode usar o comando CLI DescribeThingType para exibir o resultado:

```
$ aws iot describe-thing-type --thing-type-name "StopLight":
```

```
{
    "thingTypeName": "StopLight",
    "thingTypeArn": "arn:aws:iot:us-east-1:123456789012:thingtype/StopLight",
    "thingTypeId": "12345678abcdefgh12345678ijklmnop12345678"
    "thingTypeProperties": {
        "searchableAttributes": [
            "wattage",
            "numOfLights",
            "model"
        ],
        "thingTypeDescription": "traffic light type"
    },
    "thingTypeMetadata": {
        "deprecated": false,
        "creationDate": 1468425854308,
    }
}
```

Excluir um tipo de objeto

É possível excluir tipos de objeto somente depois que eles forem preteridos. Para excluir um tipo de objeto, use o comando DeleteThingType:

```
$ aws iot delete-thing-type --thing-type-name "StopLight"
```

> **Note**
>
> Antes de excluir um tipo de objeto, espere cinco minutos após descontinuá-lo.

Grupos de objetos estáticas

Os grupos de objetos estáticas permitem gerenciar várias objetos simultaneamente, categorizando-as em grupos. Os grupos de objetos estáticas contêm um grupo de objetos que são gerenciadas usando o console, a CLI ou a API. Grupos de objetos dinâmicos, por outro lado, contêm objetos que correspondem a uma consulta especificada. Os grupos de itens estáticos também podem conter outros grupos do mesmo tipo. Então, você pode criar uma hierarquia de grupos. Você pode anexar uma política a um grupo pai, e ela será herdada pelos grupos filho, e por todas as objetos do grupo e também dos grupos filho. Isso facilita o controle de permissões para grandes números de objetos.

> **Note**
>
> As políticas do Thing Group não permitem acesso às operações do plano de AWS IoT Greengrass dados. Para permitir que uma coisa acesse uma operação de plano de AWS IoT Greengrass dados, adicione a permissão a uma AWS IoT política que você anexa ao certificado da coisa. Para obter mais informações, consulte Autenticação e autorização de dispositivos no AWS IoT Greengrass guia do desenvolvedor.

Veja o que você pode fazer com grupos de objetos estáticas:

- Criar, descrever ou excluir um grupo.
- Adicione um objeto a um grupo ou a mais de um grupo.
- Remover um objeto de um grupo.
- Listar os grupos que você criou.
- Listar todos os grupos filho de um grupo (seus descendentes diretos e indiretos).
- Listar as objetos em um grupo, incluindo todas as objetos em seus grupos filho.
- Listar todos os grupos ancestrais de um grupo (seus pais diretos e indiretos).

- Adicionar, excluir ou atualizar os atributos de um grupo. (Os atributos são pares nome-valor que você pode usar para armazenar informações sobre um grupo.)
- Anexar ou desanexar uma política de ou para um grupo.
- Listar as políticas anexadas a um grupo.
- Listar as políticas herdadas por um objeto (em virtude das políticas anexadas a seu grupo ou um de seus grupos pais).
- Configure as opções de registro em log para as objetos em um grupo. Consulte Configurar o AWS IoT registro.
- Crie trabalhos que serão enviados para e executados em cada objeto em um grupo e nos respectivos grupos filho. Consulte AWS IoT Empregos.

> **Note**
>
> Quando uma coisa é anexada a um grupo estático ao qual uma AWS IoT Core política está anexada, o nome da coisa deve corresponder à ID do cliente.

Estas são algumas limitações de grupos de objetos estáticas:

- Um grupo pode ter no máximo um pai direto.
- Se um grupo for filho de outro grupo, especifique isso no momento em que for criado.
- Você não pode alterar o pai de um grupo mais tarde, portanto, certifique-se de planejar sua hierarquia de grupos e criar um grupo pai antes de qualquer grupo filho que ele contenha.
- O número de grupos aos quais um objeto pode pertencer é limitado.
- Você não pode adicionar um objeto a mais de um grupo na mesma hierarquia. (Ou seja, não é possível adicionar um objeto a dois grupos que compartilham o mesmo pai.)
- Não é possível renomear um grupo.
- Nomes de objetos não podem conter caracteres internacionais, como, û, é e ñ.
- Não use informações de identificação pessoal nos nomes de grupo de objetos. O nome do grupo de objetos pode aparecer em comunicações e relatórios não criptografados.

Anexar e desanexar políticas em grupos pode aprimorar a segurança das operações da AWS IoT de várias maneiras significativas. O método por dispositivo de anexar uma política a um certificado,

que, depois, é anexado a um objeto, é demorado e dificulta a atualização ou a alteração rápida de políticas em toda a frota de dispositivos. Ter uma política anexada ao grupo do objeto economiza etapas na hora de alternar os certificados em um objeto. Além disso, as políticas são aplicadas dinamicamente o objetos quando suas associações a grupos é alterada, portanto, não é necessário recriar um conjunto complexo de permissões toda vez que a associação de um dispositivo é alterada em um grupo.

Criar um grupo de objetos estáticas

Use o comando CreateThingGroup para criar um grupo de objetos estáticas:

```
$ aws iot create-thing-group --thing-group-name LightBulbs
```

O comando CreateThingGroup retorna uma resposta que contém o nome, ID e ARN do grupo de objetos estáticas:

```
{
    "thingGroupName": "LightBulbs",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
}
```

> **Note**
>
> Não recomendamos o uso de informações pessoais identificáveis nos nomes dos grupos de objetos.

Este é um exemplo que especifica um pai do grupo de objetos estáticas quando ele é criado:

```
$ aws iot create-thing-group --thing-group-name RedLights --parent-group-name
 LightBulbs
```

Como anteriormente, o comando CreateThingGroup retorna uma resposta que contém o grupo de objetos, seu ID e ARN:

```
{
    "thingGroupName": "RedLights",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx",
```

```
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
}
```

> **Important**
>
> Lembre-se dos seguintes limites ao criar as hierarquias de grupo de objetos:
>
> - Um grupo de objetos pode ter apenas um pai direto.
> - O número de grupos filhos diretos que um grupo pode ter é limitado.
> - A profundidade máxima de uma hierarquia de grupos é limitada.
> - O número de atributos que um grupo de objetos pode ter é limitado. (Os atributos são pares nome-valor que você pode usar para armazenar informações sobre um grupo.) Os comprimentos de cada nome de atributo e cada valor também são limitados.

Descrever um grupo de objetos

Você pode usar o comando DescribeThingGroup para obter informações sobre um grupo de objetos:

```
$ aws iot describe-thing-group --thing-group-name RedLights
```

O comando DescribeThingGroup retorna informações sobre o grupo especificado:

```
{
    "thingGroupName": "RedLights",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
    "thingGroupId": "12345678abcdefgh12345678ijklmnop12345678",
    "version": 1,
    "thingGroupMetadata": {
        "creationDate": 1478299948.882
        "parentGroupName": "Lights",
        "rootToParentThingGroups": [
            {
                "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
ShinyObjects",
                "groupName": "ShinyObjects"
            },
            {
                "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs",
                "groupName": "LightBulbs"
```

```
            }
        ]
    },
    "thingGroupProperties": {
        "attributePayload": {
            "attributes": {
                "brightness": "3400_lumens"
            },
        },
        "thingGroupDescription": "string"
    },
}
```

Adicionar um objeto a um grupo de objetos estáticas

Você pode usar o comando AddThingToThingGroup para adicionar um objeto a um grupo de objetos estáticas:

```
$ aws iot add-thing-to-thing-group --thing-name MyLightBulb --thing-group-name
 RedLights
```

O comando AddThingToThingGroup não produz nenhuma saída.

> **Important**
>
> Você pode adicionar um objeto a um máximo de 10 grupos. Mas você não pode adicionar um objeto a mais de um grupo na mesma hierarquia. (Ou seja, não é possível adicionar um objeto a dois grupos que compartilham o mesmo pai.)
> Se um objeto pertencer ao máximo de grupos de objetos possível, e um ou mais desses grupos for um grupo de objetos dinâmicas, você poderá usar o sinalizador overrideDynamicGroups para fazer com que grupos estáticos tenham prioridade sobre grupos dinâmicos.

Remover um objeto de um grupo de objetos estáticas

Você pode usar o comando RemoveThingFromThingGroup para remover um objeto de um grupo:

```
$ aws iot remove-thing-from-thing-group --thing-name MyLightBulb --thing-group-name
 RedLights
```

O comando RemoveThingFromThingGroup não produz nenhuma saída.

Listar objetos em um grupo de objetos

Você pode usar o comando ListThingsInThingGroup para listar objetos que pertencem a um grupo:

```
$ aws iot list-things-in-thing-group --thing-group-name LightBulbs
```

O comando ListThingsInThingGroup retorna uma lista das objetos no grupo especificado:

```
{
    "things":[
        "TestThingA"
    ]
}
```

Com o parâmetro `--recursive`, você pode listar objetos que pertencem a um grupo e a qualquer um dos grupos filho:

```
$ aws iot list-things-in-thing-group --thing-group-name LightBulbs --recursive
```

```
{
    "things":[
        "TestThingA",
        "MyLightBulb"
    ]
}
```

> **Note**
>
> Essa operação é eventualmente consistente. Em outras palavras, alterações no grupo de objetos podem não ser refletidas imediatamente.

Listar grupos de objetos

Você pode usar o comando ListThingGroups para listar grupos de objetos da sua conta:

```
$ aws iot list-thing-groups
```

O ListThingGroups comando retorna uma lista dos grupos de coisas em seu Conta da AWS:

```
{
    "thingGroups": [
        {
            "groupName": "LightBulbs",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
        },
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "RedLEDLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLEDLights"
        },
        {
            "groupName": "RedIncandescentLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
RedIncandescentLights"
        }
        {
            "groupName": "ReplaceableObjects",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
ReplaceableObjects"
        }
    ]
}
```

Use os filtros opcionais para listar os grupos que têm um determinado grupo como pai (`--parent-group`) ou grupos cujos nomes começam com um determinado prefixo (`--name-prefix-filter`). O parâmetro `--recursive` permite que você liste todos os grupos filho, não apenas grupos filho diretos de um grupo de objetos:

```
$ aws iot list-thing-groups --parent-group LightBulbs
```

Nesse caso, o ListThingGroups comando retorna uma lista dos grupos secundários diretos do grupo de coisas definido em seu Conta da AWS:

```
{
    "childGroups":[
        {
```

```
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        }
    ]
}
```

Use o parâmetro `--recursive` com o comando ListThingGroups para listar todos os grupos filho de um grupo de objetos, não apenas os filhos diretos:

```
$ aws iot list-thing-groups --parent-group LightBulbs --recursive
```

O comando ListThingGroups retorna uma lista de todos os grupos filho de um grupo de objetos:

```
{
    "childGroups":[
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "RedLEDLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLEDLights"
        },
        {
            "groupName": "RedIncandescentLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
RedIncandescentLights"
        }
    ]
}
```

> **Note**
>
> Essa operação é eventualmente consistente. Em outras palavras, alterações no grupo de objetos podem não ser refletidas imediatamente.

Listar grupos para um objeto

Você pode usar o comando ListThingGroupsForThing para listar grupos diretos a que um objeto pertence:

```
$ aws iot list-thing-groups-for-thing --thing-name MyLightBulb
```

O comando ListThingGroupsForThing retorna uma lista dos grupos diretos de objetos aos quais o objeto pertence, incluindo todos os grupos pai:

```
{
    "thingGroups":[
        {
            "groupName": "LightBulbs",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
        },
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "ReplaceableObjects",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
ReplaceableObjects"
        }
    ]
}
```

Você também pode acessar essa API dentro do mecanismo de regras usando a função `get_registry_data()` embutida. Você pode usar essa função para acessar e utilizar dinamicamente as informações do registro de coisas (incluindo atributos, tipos de coisas e associações de grupos) ligando e `ListThingGroupsForThing` APIs diretamente dentro AWS IoT das regras, permitindo o processamento `DescribeThing` e o roteamento de mensagens em tempo real com base nos dados de registro do seu dispositivo. Para obter mais informações, consulte [get_registry_data].

Atualizar um grupo de objetos estáticas

Você pode usar o comando UpdateThingGroup para atualizar os atributos de um grupo de objetos estáticas:

```
$ aws iot update-thing-group --thing-group-name "LightBulbs" --thing-group-properties
 "thingGroupDescription=\"this is a test group\", attributePayload=\"{\"attributes
\"={\"Owner\"=\"150\",\"modelNames\"=\"456\"}}"
```

O comando UpdateThingGroup retorna uma resposta que contém o número da versão do grupo depois da atualização:

```
{
    "version": 4
}
```

> **Note**
>
> O número de atributos que um objeto pode ter é limitado.

Excluir um grupo de objetos

Para excluir um grupo de objetos, use o comando DeleteThingGroup:

```
$ aws iot delete-thing-group --thing-group-name "RedLights"
```

O comando DeleteThingGroup não produz nenhuma saída.

> **Important**
>
> Se você tentar excluir um grupo de objetos que tem grupos de objetos filho, você receberá um erro:
>
> ```
> A client error (InvalidRequestException) occurred when calling the
>  DeleteThingGroup
> operation: Cannot delete thing group : RedLights when there are still child
>  groups attached to it.
> ```
>
> Antes de excluir o grupo, exclua todos os grupos filho.

Você pode excluir um grupo de objetos filho, mas as permissões concedidas para as objetos por associação no grupo não se aplicarão mais. Antes de excluir um grupo que tem uma política anexada, verifique cuidadosamente se a remoção dessas permissões não fará com que as objetos no grupo não funcionem corretamente. Além disso, os comandos que mostram a quais grupos um objeto pertence (por exemplo, ListGroupsForThing) podem continuar a mostrar o grupo enquanto os registros na nuvem estão sendo atualizados.

Anexar uma política a um grupo de objetos estáticas

Você pode usar o comando AttachPolicy para anexar uma política a um grupo de objetos estáticas e, por extensão, a todas as objetos daquele grupo e objetos em qualquer um dos grupos filho:

```
$ aws iot attach-policy \
  --target "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs" \
  --policy-name "myLightBulbPolicy"
```

O comando AttachPolicy não produz nenhuma saída

> **Important**
>
> Você pode anexar a quantidade máxima de duas políticas a um grupo.

> **Note**
>
> Não recomendamos o uso de informações pessoais identificáveis nos nomes das políticas.

O parâmetro `--target` pode ser o ARN de um grupo de objetos (conforme acima), o ARN de um certificado ou uma identidade do Amazon Cognito. Para obter mais informações sobre políticas, certificados e autenticação, consulte Autenticação.

Para obter mais informações, consulte Políticas AWS IoT Core.

Desanexar uma política de um grupo de objetos estáticas

Você pode usar o comando DetachPolicy para desanexar uma política de um grupo de objetos e, portanto, por extensão, de todas as objetos daquele grupo e objetos em qualquer um dos grupos filho:

```
$ aws iot detach-policy --target "arn:aws:iot:us-west-2:123456789012:thinggroup/
LightBulbs" --policy-name "myLightBulbPolicy"
```

O comando DetachPolicy não produz nenhuma saída.

Listar as políticas anexadas a um grupo de objetos estáticas

Você pode usar o comando ListAttachedPolicies para listar as políticas anexadas a um grupo de objetos estáticas:

```
$ aws iot list-attached-policies --target "arn:aws:iot:us-
west-2:123456789012:thinggroup/RedLights"
```

O parâmetro `--target` pode ser o ARN de um grupo de objetos (conforme acima), o ARN de um certificado ou uma identidade do Amazon Cognito.

Adicione o parâmetro `--recursive` opcional para incluir todas as políticas anexadas aos grupos pai do grupo.

O comando ListAttachedPolicies retorna uma lista de políticas:

```
{
    "policies": [
        "MyLightBulbPolicy"
        ...
    ]
}
```

Listar os grupos para uma política

Você pode usar o comando ListTargetsForPolicy para listar os destinos, incluindo todos os grupos, aos quais uma política está anexada:

```
$ aws iot list-targets-for-policy --policy-name "MyLightBulbPolicy"
```

Adicione o parâmetro `--page-size` *`number`* opcional para especificar o número máximo de resultados a serem retornados em cada consulta, e o parâmetro `--marker` *`string`* em chamadas subsequentes para recuperar o próximo conjunto de resultados, se houver.

O comando ListTargetsForPolicy retorna uma lista de destinos e o token a ser usado para recuperar mais resultados:

```
{
    "nextMarker": "string",
```

```
    "targets": [ "string" ... ]
}
```

Obter políticas efetivas para um objeto

Você pode usar o comando GetEffectivePolicies para listar as políticas em vigor para um objeto, incluindo as políticas anexadas a todos os grupos aos quais o objeto pertence (se o grupo for um pai direto ou um ancestral indireto):

```
$ aws iot get-effective-policies \
  --thing-name "MyLightBulb" \
  --principal "arn:aws:iot:us-east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

Use o parâmetro `--principal` para especificar o ARN do certificado anexado ao objeto. Se você estiver usando a autenticação de identidades do Amazon Cognito, use o parâmetro `--cognito-identity-pool-id` e, opcionalmente, adicione o parâmetro `--principal` para especificar uma identidade específica do Amazon Cognito. Se você especificar apenas o `--cognito-identity-pool-id`, as políticas associadas a essa função do banco de identidades para usuários não autenticados serão retornadas. Se usar os dois, as políticas associadas a essa função do banco de identidades para usuários autenticados serão retornadas.

O parâmetro `--thing-name` é opcional e pode ser usado em vez do parâmetro `--principal`. Quando o parâmetro é usado, as políticas anexadas a qualquer grupo ao qual o objeto pertence e as políticas anexadas a qualquer grupo pai desses grupos (até o grupo raiz na hierarquia) são retornadas.

O comando GetEffectivePolicies retorna uma lista de políticas:

```
{
    "effectivePolicies": [
        {
            "policyArn": "string",
            "policyDocument": "string",
            "policyName": "string"
        }
        ...
    ]
}
```

Testar a autorização de ações MQTT

Você pode usar o comando TestAuthorization para testar se uma ação MQTT (`Publish`, `Subscribe`) é permitida para um objeto:

```
aws iot test-authorization \
    --principal "arn:aws:iot:us-east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847" \
    --auth-infos "{\"actionType\": \"PUBLISH\", \"resources\": [ \"arn:aws:iot:us-
east-1:123456789012:topic/my/topic\"]}"
```

Use o parâmetro `--principal` para especificar o ARN do certificado anexado ao objeto. Se estiver usando a autenticação de identidade do Amazon Cognito, especifique uma identidade do Cognito como `--principal` ou use o parâmetro `--cognito-identity-pool-id` ou ambos. (Se você especificar apenas o `--cognito-identity-pool-id`, as políticas associadas a essa função do banco de identidades para usuários não autenticados serão consideradas. Se você usar os dois parâmetros, as políticas associadas a essa função do banco de identidades para usuários autenticados serão consideradas.

Especifique uma ou mais ações MQTT que você deseja testar listando conjuntos de recursos e tipos de ação depois do parâmetro `--auth-infos`. O campo `actionType` deve conter as opções "PUBLICAR", "ASSINAR", "RECEBER" ou "CONECTAR". O `resources` campo deve conter uma lista de recursos ARNs. Consulte AWS IoT Core políticas para obter mais informações.

Você pode testar os efeitos da adição de políticas especificando-as com o parâmetro `--policy-names-to-add`. Ou pode testar os efeitos da remoção de políticas especificando-as com o parâmetro `--policy-names-to-skip`.

Você pode usar o parâmetro `--client-id` opcional para refinar ainda mais os resultados.

O comando TestAuthorization retorna detalhes das ações que foram permitidas ou negadas para cada conjunto de consultas `--auth-infos` que você especificou:

```
{
    "authResults": [
        {
            "allowed": {
                "policies": [
                    {
```

```
                        "policyArn": "string",
                        "policyName": "string"
                    }
                ]
            },
            "authDecision": "string",
            "authInfo": {
                "actionType": "string",
                "resources": [ "string" ]
            },
            "denied": {
                "explicitDeny": {
                    "policies": [
                        {
                            "policyArn": "string",
                            "policyName": "string"
                        }
                    ]
                },
                "implicitDeny": {
                    "policies": [
                        {
                            "policyArn": "string",
                            "policyName": "string"
                        }
                    ]
                }
            },
            "missingContextValues": [ "string" ]
        }
    ]
}
```

Grupos de objetos dinâmicos

Grupos dinâmicos de objetos são criados com base em consultas de pesquisa específicas no registro. Parâmetros de consulta de pesquisa, como conectividade do dispositivo, criação de sombra do dispositivo e dados de violações de AWS IoT Device Defender dão suporte a isso. Grupos de objetos dinâmicos exigem que a indexação de frotas seja habilitada para indexar, pesquisar e agregar os dados dos dispositivos. Você pode visualizar as objetos em um grupo de objetos dinâmico usando uma consulta de pesquisa de indexação de frota antes de criá-los. Para obter mais informações, consulte Indexação de frotas e Sintaxe de consulta.

> **Note**
>
> As operações dinâmicas de grupos de objetos são medidas em operações de registro. Para obter mais informações, consulte AWS IoT Core Detalhes de medição adicionais.

Os grupos de objetos dinâmicas diferem dos grupos de objetos estáticas das seguintes maneiras:

- A associação de objetos não é explicitamente definida. Para criar um grupo de objetos dinâmico, defina uma string de consulta de pesquisa para determinara a associação ao grupo.
- Grupos de objetos dinâmicos não podem fazer parte de uma hierarquia.
- Grupos de objetos dinâmicos não podem ter políticas aplicadas a eles.
- Use um conjunto diferente de comandos para criar, atualizar e excluir grupos de objetos dinâmicas. Para todas as outras operações, você usa os mesmos comandos para os dois tipos de grupos de objetos.
- O número de grupos dinâmicos por grupo Conta da AWS é limitado.

- Não use informações de identificação pessoal nos nomes de grupo de objetos. O nome do grupo de objetos pode aparecer em comunicações e relatórios não criptografados.

Para obter mais informações sobre grupos de objetos estáticas, consulte Grupos de objetos estáticas.

Casos de uso de grupos objetos dinâmicos

Você pode usar grupos de objetos dinâmicos para os seguintes casos de uso:

Especifique um grupo de objetos dinâmico como o destino para um trabalho

Criar um trabalho contínuo com um grupo dinâmico como destino permite que você direcione automaticamente os dispositivos quando eles atenderem aos critérios desejados. Os critérios podem ser o estado de conectividade ou qualquer critério armazenado no registro ou na sombra, como versão ou modelo do software. Se um objeto não aparecer no grupo de objetos dinâmico, ele não receberá do trabalho o documento de trabalho.

Por exemplo, se a frota de dispositivos precisar de uma atualização de firmware para minimizar o risco de interrupção durante o processo de atualização, e você só quiser atualizar o firmware em

dispositivos com duração de bateria maior que 80%. Você pode criar um grupo dinâmico chamado 80, PercentBatteryLife que inclui apenas dispositivos com duração de bateria acima de 80% e usá-lo como alvo para seu trabalho. Somente dispositivos que atenderem aos critérios de duração da bateria receberão a atualização do firmware. Conforme os dispositivos atingem os critérios de 80% de duração da bateria, eles são adicionados automaticamente ao grupo de coisas dinâmico e receberão a atualização do firmware.

Você também pode ter vários modelos de dispositivos com firmware ou sistema operacional diferentes, exigindo versões diferentes de novas atualizações de software. Esse é o caso de uso mais comum para grupos dinâmicos com trabalhos contínuos, em que você pode criar um grupo dinâmico para cada combinação de modelo de dispositivo, firmware e sistema operacional. Em seguida, você pode configurar trabalhos contínuos para cada um desses grupos dinâmicos para fazer push de atualizações de software à medida que os dispositivos se tornam automaticamente membros desses grupos com base nos critérios definidos.

Para obter mais informações sobre como especificar grupos de coisas como destinos de trabalho, consulte CreateJob.

Use alterações dinâmicas de associação ao grupo para realizar as ações desejadas

Sempre que um dispositivo é adicionado ou removido de um grupo dinâmico, uma notificação é enviada para um tópico do MQTT como parte das atualizações de eventos do Registro. Você pode configurar AWS IoT Core regras para interagir com os AWS serviços com base nas atualizações dinâmicas dos membros do grupo e realizar as ações desejadas. Exemplos de ação incluem gravar em Amazon DynamoDB, invocar uma função do Lambda ou enviar uma notificação ao Amazon SNS.

Adicione dispositivos a um grupo dinâmico para detecção automática de violações

AWS IoT Device Defender Os clientes do Detect podem definir um perfil de segurança em um grupo dinâmico. Os dispositivos do grupo de objetos dinâmico são detectados automaticamente por violações pelo perfil de segurança definido no grupo.

Defina níveis de log em grupos de objetos dinâmicos para observar dispositivos com registro em log detalhado

Você pode especificar um nível de log em um grupo de objetos dinâmico. Isso é útil se você quer personalizar apenas o nível e os detalhes do registro para dispositivos que atendam a determinados critérios. Por exemplo, se você suspeitar que dispositivos com determinada versão de firmware estão causando erros no tópico publicado de uma regra específica, convém definir um registro detalhado

para depurar esses problemas. Nesse caso, você pode criar um grupo dinâmico para todos os dispositivos que tenham essa versão de firmware, que presumimos que esteja armazenada como um atributo do registro ou em uma sombra do dispositivo. Em seguida, você pode definir um nível de depuração, com o destino de registro definido como esse grupo de objetos dinâmico. Para obter mais informações sobre registros detalhados, consulte Monitorar AWS IoT usando registros. CloudWatch Para obter mais informações sobre como especificar um nível de log para um grupo de objetos específico, consulte Configure resource-specific logging in AWS IoT.

Criar grupo de objetos dinâmicas

Use o comando CreateDynamicThingGroup para criar um grupo de objetos dinâmicas. Para criar um grupo dinâmico de coisas para o PercentBatteryLife cenário 80, use o comando create-dynamic-thing-group CLI:

```
$ aws iot create-dynamic-thing-group --thing-group-name "80PercentBatteryLife" --query-string "attributes.batterylife80"
```

> **Note**
>
> Não use informações de identificação pessoal nos nomes de grupo de objetos dinâmico.

O comando CreateDynamicThingGroup retorna uma resposta. A resposta contém o nome do índice, a string de consulta, a versão de consulta, o nome do grupo de objetos, o ID do grupo de objetos e o nome do recurso da Amazon (ARN) do grupo de objetos:

```
{
    "indexName": "AWS_Things",
    "queryVersion": "2017-09-30",
    "thingGroupName": "80PercentBatteryLife",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/80PercentBatteryLife",
    "queryString": "attributes.batterylife80\n",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx"
}
```

A criação de grupos de objetos dinâmico não acontece de uma só vez. A alocação do grupo de objetos dinâmicas leva tempo para ser concluída. Quando você cria um grupo de objetos dinâmico, o status do grupo é definido como `BUILDING`. Quando a alocação é concluída, o

status muda para `ACTIVE`. Para verificar o status do seu grupo dinâmico de coisas, use o DescribeThingGroupcomando.

Descrever um grupo de objetos dinâmicas

Use o comando DescribeThingGroup para obter informações sobre um grupo de objetos dinâmicas:

```
$ aws iot describe-thing-group --thing-group-name "80PercentBatteryLife"
```

O comando DescribeThingGroup retorna informações sobre o grupo especificado:

```
{
    "status": "ACTIVE",
    "indexName": "AWS_Things",
    "thingGroupName": "80PercentBatteryLife",
    "thingGroupArn": "arn:aws:iot:us-
west-2:123456789012:thinggroup/80PercentBatteryLife",
    "queryString": "attributes.batterylife80\n",
    "version": 1,
    "thingGroupMetadata": {
        "creationDate": 1548716921.289
    },
    "thingGroupProperties": {},
    "queryVersion": "2017-09-30",
    "thingGroupId": "84dd9b5b-2b98-4c65-84e4-be0e1ecf4fd8"
}
```

Executar DescribeThingGroup em um grupo de objetos dinâmico retorna atributos específicos para grupos de objetos dinâmicos. Exemplos de atributos de retorno são queryString e o status.

O status de um grupo de objetos dinâmicas pode ter os seguintes valores:

`ACTIVE`

O grupo de objetos dinâmicas está pronto para uso.

`BUILDING`

O grupo de objetos dinâmicas está sendo criado e a associação de objetos está sendo processada.

`REBUILDING`

A associação do grupo de objetos dinâmicas está sendo atualizada, seguindo o ajuste da consulta de pesquisa do grupo.

> **Note**
>
> Depois de criar um grupo de objetos dinâmicas, use-o independentemente do status. Somente grupos de objetos dinâmicas com status `ACTIVE` incluem todas as objetos que correspondem à consulta de pesquisa para esse grupo de objetos dinâmicas. Grupos de objetos dinâmicos com status `BUILDING` e `REBUILDING` podem não incluir todas as objetos que correspondem à consulta de pesquisa.

Atualizar um grupo de objetos dinâmicas

Use o comando UpdateDynamicThingGroup para atualizar os atributos de um grupo de objetos dinâmicas, incluindo a consulta de pesquisa do grupo. O comando a seguir atualiza dois atributos. Um é a descrição do grupo de objetos e o outra é a sequência de caracteres de consulta que altera os critérios de associação para duração da bateria >85:

```
$ aws iot update-dynamic-thing-group --thing-group-name "80PercentBatteryLife" --thing-
group-properties "thingGroupDescription=\"This thing group contains devices with a
 battery life greater than 85 percent.\"" --query-string "attributes.batterylife85"
```

O comando UpdateDynamicThingGroup retorna uma resposta que contém o número da versão do grupo depois da atualização:

```
{
    "version": 2
}
```

A atualização de um grupo de objetos dinâmico não acontece de uma só vez. A alocação do grupo de objetos dinâmicas leva tempo para ser concluída. Ao atualizar um grupo de objetos dinâmico, o status do grupo muda para `REBUILDING` e o grupo atualiza sua associação. Quando a alocação é concluída, o status muda para `ACTIVE`. Para verificar o status do seu grupo dinâmico de coisas, use o DescribeThingGroupcomando.

Excluir um grupo de objetos dinâmicas

Use o comando DeleteDynamicThingGroup para excluir um grupo de objetos dinâmicas:

```
$ aws iot delete-dynamic-thing-group --thing-group-name "80PercentBatteryLife"
```

O comando DeleteDynamicThingGroup não produz saída.

Os comandos que mostram a quais grupos um objeto pertence (por exemplo, ListGroupsForThing) podem continuar a mostrar o grupo enquanto os registros na nuvem estão sendo atualizados.

Limitações do grupo de objetos estático e dinâmico

Grupos de objetos dinâmicos e grupos de objetos estáticos compartilham as mesmas limitações:

- O número de atributos que um grupo de objetos pode ter é limitado.
- O número de grupos aos quais um objeto pode pertencer é limitado.
- Não é possível renomear os grupos de objetos.
- Nomes de objetos não podem conter caracteres internacionais, como, û, é e ñ.

Limitações do grupo de objetos dinâmico

Grupos de objetos dinâmicos têm as seguintes limitações:

Indexação de frotas

Com o serviço de indexação de frotas ativado, é possível realizar consultas de pesquisa em sua frota de dispositivos. Você pode criar e gerenciar grupos dinâmicos depois de preencher a indexação da frota. O tamanho da frota de dispositivos registrada no Nuvem AWS afeta diretamente o tempo de conclusão do processo de preenchimento. Depois de ativar o serviço de indexação da frota para grupos de objetos dinâmicas, você não poderá desativá-lo até que todos os seus grupos de objetos dinâmicas sejam excluídos.

> **Note**
>
> Se você tiver permissões para consultar o índice de frota, você poderá acessar os dados das objetos em toda a frota.

O número de grupos de objetos dinâmicas é limitado

O número de grupos de objetos dinâmicas é limitado.

Comandos bem-sucedidos podem registrar erros

Ao criar ou atualizar um grupo de objetos dinâmico, é possível que alguns objetos possam ser qualificados para inclusão em um grupo de objetos dinâmico, mas ainda não tenham sido adicionados a ele. Esse cenário causará um comando de criação ou atualização bem-sucedido ao registrar um erro e gerar uma métrica de `AddThingToDynamicThingGroupsFailed`. Uma única métrica pode representar várias entradas de log.

Uma entrada de registro de erros no CloudWatch registro é criada quando ocorre o seguinte:

- Um objeto elegível não pode ser adicionado a um grupo de objetos dinâmico.
- Um objeto é removido de um grupo dinâmico para ser adicionado a outro grupo.

Quando um objeto se qualifica para ser adicionado a um grupo de objetos dinâmico, considere o seguinte:

- A objeto já está no máximo possível de grupos? (Consulte limites)
  - NÃO: o objeto é adicionada ao grupo de objetos dinâmicas.
  - SIM: o objeto é um membro de algum grupo dinâmico de objetos?
    - NÃO: o objeto não pode ser adicionada ao grupo dinâmico, um erro é registrado e uma métrica `AddThingToDynamicThingGroupsFailed` é gerada.
    - SIM: o grupo de objetos dinâmicas a ingressar é mais antigo do que qualquer grupo de objetos dinâmicas do qual o objeto já é membro?
      - NÃO: o objeto não pode ser adicionada ao grupo dinâmico, um erro é registrado e uma métrica `AddThingToDynamicThingGroupsFailed` é gerada.
      - SIM: remova o objeto do grupo de objetos dinâmico mais recente, registre um erro e adicione o objeto ao grupo de objetos dinâmicas. Isso gera um erro e uma métrica `AddThingToDynamicThingGroupsFailed` para o grupo dinâmico do qual o objeto foi removida.

Quando um objeto em um grupo de objetos dinâmico não satisfaz mais a consulta de pesquisa, o objeto é removido do grupo de objetos dinâmico. Da mesma forma, quando um objeto é atualizado

para atender a consulta de pesquisa de um grupo de objetos dinâmico, ele é adicionado ao grupo como já descrito. Essas adições e remoções são normais e não produzem entradas de log de erros.

Com **`overrideDynamicGroups`** habilitado, os grupos estáticos terão prioridade sobre os grupos dinâmicos

O número de grupos aos quais um objeto pode pertencer é limitado. Quando você usa os UpdateThingGroupsForThingcomandos AddThingToThingGroupou para atualizar a associação de coisas, adicionar o `--overrideDynamicGroups` parâmetro dá prioridade aos grupos de coisas estáticos sobre grupos de coisas dinâmicas.

Ao adicionar um objeto a um grupo de objetos estáticas, considere o seguinte:

- A objeto já pertence ao número máximo de grupos?
  - NÃO: o objeto é adicionada ao grupo de objetos estáticas.
  - SIM: o objeto está em algum grupo dinâmico?
    - NÃO: o objeto não pode ser adicionada ao grupo de objetos. O comando gera uma exceção.
    - SIM: --overrideDynamicGroups foi ativado?
      - NÃO: o objeto não pode ser adicionada ao grupo de objetos. O comando gera uma exceção.
      - SIM: o objeto é removida do grupo dinâmico criado mais recentemente, um erro é registrado e uma métrica `AddThingToDynamicThingGroupsFailed` é gerada para o grupo de objetos dinâmicas do qual o objeto foi removida. Depois, o objeto é adicionada ao grupo de objetos estáticas.

Os grupos de objetos dinâmicas mais antigos têm prioridade sobre os mais recentes

O número de grupos aos quais um objeto pode pertencer é limitado. Quando uma operação de criação ou atualização cria elegibilidade de grupo adicional para um objeto e o objeto atinge o limite de grupo, é possível remover de outro grupo de objetos dinâmico para possibilitar essa adição. Para obter mais informações sobre como isso ocorre, consulte Comandos bem-sucedidos podem registrar erros e Com `overrideDynamicGroups` habilitado, os grupos estáticos terão prioridade sobre os grupos dinâmicos para obter exemplos.

Quando um objeto é removida de um grupo dinâmico, um erro é registrado e um evento é gerado.

Você não pode aplicar políticas a grupos de objetos dinâmicas

A tentativa de aplicar uma política a um grupo dinâmico gera uma exceção.

A associação do grupo de objetos dinâmicas é eventualmente consistente

Somente o estado final de um objeto é avaliada para o registro. Os estados intermediários podem ser ignorados se forem atualizados rapidamente. Evite associar uma regra ou um trabalho, a um grupo de objetos dinâmicas cuja associação depende de um estado intermediário.

Associando qualquer AWS IoT coisa a uma conexão de cliente MQTT

Uma associação de coisa exclusiva é quando você anexa um certificado X.509 a uma única coisa da AWS IoT . Nesse caso, o certificado não pode ser usado com outras coisas. Garantir que um certificado seja usado somente por uma única coisa da IoT ajuda a evitar vulnerabilidades de segurança.

Em AWS IoT, o ID do cliente é um identificador exclusivo de uma coisa ou dispositivo quando ele se conecta ao corretor AWS IoT Core MQTT. Se você usa uma associação não exclusiva, é possível anexar várias coisas ao mesmo certificado. Quando existe uma associação de coisa não exclusiva, para manter uma associação clara e evitar possíveis conflitos, você deve combinar seu ID de cliente com o nome da coisa.

Neste tópico

- Casos de uso
- Como associar uma coisa a uma conexão

Casos de uso

A associação de uma coisa a uma conexão oferece os recursos a seguir.

> **Note**
>
> Observe que, se sua conexão da IoT com o seu cliente tiver uma associação não exclusiva, você poderá usar todos os recursos a seguir, exceto o de eventos de ciclo de vida. Para

incluir o nome da sua coisa nas mensagens do evento do ciclo de vida, sua conexão da IoT com o cliente deve ter uma associação exclusiva.

Variáveis de política de coisas - Você pode usar variáveis de política de coisas para autorizar o acesso do dispositivo às operações AWS IoT da API. Essas variáveis permitem que você crie AWS IoT Core políticas que concedam ou neguem permissões com base nas propriedades das coisas, como nomes, tipos e valores de atributos. Ao usar variáveis de política de coisas, você pode aplicar a mesma política para controlar vários AWS IoT Core dispositivos. Isso permite simplificar o gerenciamento de políticas e reduzir a duplicação de recursos. Para acessar mais informações, consulte Variáveis de política de coisas.

Eventos de ciclo de vida: você pode receber o nome da coisa em eventos de ciclo de vida (por exemplo, conectar, desconectar, assinar e cancelar a assinatura). Isso permite o processamento do nome da coisa incluída nas mensagens, como nas regras. Para acessar mais informações, consulte Eventos de ciclo de vida.

Registro em log específico de recursos: você pode configurar o registro em log específico de recursos para grupos de coisas e aplicar facilmente a configuração de registro em log desejada para todas as coisas dentro do grupo definido. Para obter mais informações, consulte ???.

Alocação de custos: é possível criar grupos de cobrança com tags personalizadas para alocação de custos e adicionar as coisas a esses grupos. Para acessar mais informações, consulte Grupos de cobrança.

Como associar uma coisa a uma conexão

Se o ID do seu cliente corresponder ao nome da sua coisa no registro, depois de anexar um certificado X.509 a essa coisa de IoT, a conexão do cliente AWS IoT Core será associada à coisa. Se o ID do seu cliente não corresponder ao nome da coisa no registro, você poderá anexar exclusivamente um certificado X.509 à coisa para estabelecer essa associação. A coisa com esse anexo exclusivo é chamada de exclusiva. Caso contrário, é chamada de não exclusiva. Quando um certificado é associado a uma coisa exclusiva, ele só poderá ser associado a outras coisas se você separá-lo da coisa exclusiva. Nesta seção, escolha Console de gerenciamento da AWS ou AWS CLI associe algo a uma conexão.

Console de gerenciamento da AWS

Como anexar um certificado a uma coisa usando exclusivamente o Console de gerenciamento da AWS.

1. Abra a página AWS IoT inicial no AWS IoT console. Na navegação esquerda, em Segurança, escolha Certificados.
2. Na página Certificados, escolha um certificado ao qual você deseja anexar uma coisa. Depois, escolha Anexar a coisas em Ações no canto superior direito da página.

   Como alternativa, escolha um certificado e navegue até a página de detalhes do certificado. Escolha a guia Coisas e, depois, selecione Anexar a coisas.
3. Na página Anexar certificado à(s) coisa(s), marque a caixa de seleção Associar coisa à conexão. Depois, selecione uma coisa à qual anexar esse certificado na lista suspensa Coisas.
4. Escolha Anexar coisa(s). Se a ação for bem-sucedida, você verá um banner indicando: “Anexou com êxito uma coisa ao seu certificado”, e a coisa será adicionada à guia Coisas.

Para separar um certificado de algo exclusivo usando o Console de gerenciamento da AWS

1. Abra a página AWS IoT inicial no AWS IoT console. Na navegação esquerda, em Segurança, escolha Certificados.
2. Na página Certificados, escolha um certificado e navegue até a página de detalhes do certificado.
3. Na página de detalhes do certificado, escolha a guia Coisas. Então, escolha uma coisa cujo certificado você deseja separar. Escolha Separar coisas.
4. Na janela Separar coisas, confirme sua ação. Escolha Desassociar. Se a ação for bem-sucedida, você verá um banner indicando: “Separou com êxito uma coisa do seu certificado”, e a coisa deixará de aparecer na guia Coisas.

AWS CLI

1. Para anexar um certificado a qualquer coisa usando AWS CLI, execute o attach-thing-principalcomando. Para especificar o certificate-to-thing anexo exclusivo, você deve especificar `EXCLUSIVE_THING` no `--thing-principal-type` campo. O comando a seguir é um exemplo.

```
aws iot attach-thing-principal \
    --thing-name "thing_1" \
```

```
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8" \
    --thing-principal-type "EXCLUSIVE_THING"
```

Esse comando não retorna nenhuma saída. Para obter mais informações, consulte ???.

2. Para listar as coisas associadas ao certificado especificado, junto com o tipo de anexo, execute o comando `list-principal-things-v2`. O tipo de anexo se refere à forma como o certificado é anexado à coisa. O comando a seguir é um exemplo.

```
$ aws iot list-principal-things-v2 \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
```

A saída poderá ser parecida com a seguinte:

```
{
    "PrincipalThingObjects": [
        {
            "thingPrincipalType": "EXCLUSIVE_THING",
            "thing": "arn:aws:iot:us-east-1:123456789012:thing/thing_1"
        }
    ]
}
```

Para obter mais informações, consulte ???.

3. Para listar as entidades principais associadas à coisa especificada, junto com o tipo de anexo, execute o comando `list-thing-principals-v2`. O tipo de anexo se refere à forma como o certificado é anexado à coisa. O comando a seguir é um exemplo.

```
$ aws iot list-thing-principals-v2 \
    --thing-name "thing_1"
```

A saída poderá ser parecida com a seguinte:

```
{
    "ThingPrincipalObjects": [
        {
```

```
            "thingPrincipalType": "EXCLUSIVE_THING",
            "principal": "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
        }
    ]
}
```

Para obter mais informações, consulte ???.

4. Para separar um certificado de uma coisa, execute o detach-thing-principalcomando.

```
aws iot detach-thing-principal \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8" \
    --thing-name "thing_1"
```

Esse comando não retorna nenhuma saída. Para obter mais informações, consulte ???.

Adicionar atributos de propagação para enriquecimento da mensagem

Em AWS IoT Core, você pode enriquecer mensagens MQTT de dispositivos adicionando atributos de propagação, que são metadados contextuais de atributos de coisas ou detalhes de conexão. Esse processo, conhecido como enriquecimento de mensagens, pode ser útil em vários cenários. Por exemplo, é possível enriquecer mensagens para cada operação de publicação de entrada sem fazer alterações no dispositivo nem precisar usar regras. Ao utilizar os atributos de propagação, você pode se beneficiar de uma maneira mais eficiente e econômica de enriquecer seus dados da IoT sem as complexidades de configurar regras ou gerenciar configurações de republicação.

O recurso de enriquecimento de mensagens está disponível para AWS IoT Core clientes que usam ingestão básica e agente de mensagens. É importante observar que, embora os dispositivos de publicação possam usar qualquer versão do MQTT, os assinantes (aplicações ou serviços que consomem mensagens) devem comportar o MQTT 5 para receber as mensagens enriquecidas com atributos de propagação. As mensagens enriquecidas serão adicionadas como propriedades do usuário do MQTT 5 a todas as mensagens publicadas dos dispositivos. Se você usar regras, poderá utilizar a função get_user_properties para recuperar os dados enriquecidos para roteamento ou processamento de mensagens com base nos dados.

Em AWS IoT Core, você pode adicionar atributos de propagação ao criar ou atualizar um tipo de coisa, usando o Console de gerenciamento da AWS ou o. AWS CLI

> **Important**
>
> Ao adicionar atributos de propagação, você deve garantir que o cliente que está publicando a mensagem tenha sido autenticado com um certificado. Para obter mais informações, consulte Autenticação de cliente.

> **Note**
>
> Se você tentar testar esse recurso usando o cliente de teste do MQTT no console, pode ser que ele não funcione, pois esse recurso precisa de clientes do MQTT autenticados com um certificado associado.

Console de gerenciamento da AWS

Para adicionar atributos de propagação para enriquecimento de mensagens usando o Console de gerenciamento da AWS

1. Abra a página AWS IoT inicial no AWS IoT console. No painel de navegação esquerdo, em Gerenciar, escolha Todos os dispositivos. Depois, selecione Tipos de coisas.
2. Na página Tipos de coisas, selecione Criar tipo de coisa.

   Para configurar o enriquecimento de mensagens atualizando um tipo de coisa, escolha um tipo de coisa. Depois, na página de detalhes do tipo de coisa, escolha Atualizar.
3. Na página Criar tipo de coisa, selecione ou insira as informações do tipo de coisa em Propriedades do tipo de coisa.

   Se você optar por atualizar um tipo de coisa, verá as propriedades do tipo de coisa depois de selecionar Atualizar na etapa anterior.
4. Em Configuração adicional, expanda Atributos de propagação. Em seguida, escolha o atributo Coisa e insira o atributo da coisa que você deseja preencher nas MQTT5 mensagens publicadas. Usando o console, você pode adicionar até outros três atributos de coisa.

Na seção Atributos de propagação, escolha Atributo de conexão e insira o tipo de atributo e, opcionalmente, o nome do atributo.

5. Opcionalmente, adicione tags. Depois, escolha Criar tipo de coisa.

   Se você optar por atualizar um tipo de coisa, escolha Atualizar tipo de coisa.

AWS CLI

1. Para adicionar atributos de propagação para enriquecimento de mensagens criando um novo tipo de coisa usando o AWS CLI, execute o create-thing-typecomando. O comando a seguir é um exemplo.

```
aws iot create-thing-type \
    --thing-type-name "LightBulb" \
    --thing-type-properties "{\"mqtt5Configuration\":{\"propagatingAttributes\":
[{\"userPropertyKey\":\"iot:ClientId\", \"connectionAttribute\":\"iot:ClientId\"},
 {\"userPropertyKey\":\"test\", \"thingAttribute\":\"A\"}]}}" \
```

   A saída do comando pode ser semelhante à seguinte.

```
{
 "thingTypeName": "LightBulb",
 "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb",
 "thingTypeId": "ce3573b0-0a3c-45a7-ac93-4e0ce14cd190"
}
```

2. Para configurar o enriquecimento de mensagens atualizando um tipo de coisa usando AWS CLI, execute o update-thing-typecomando. Observe que você só pode atualizar a `mqtt5Configuration` quando executar esse comando. O comando a seguir é um exemplo.

```
aws iot update-thing-type \
    --thing-type-name "MyThingType" \
    --thing-type-properties "{\"mqtt5Configuration\":{\"propagatingAttributes\":
[{\"userPropertyKey\":\"iot:ClientId\", \"connectionAttribute\":\"iot:ClientId\"},
 {\"userPropertyKey\":\"test\", \"thingAttribute\":\"A\"}]}}" \
```

   Esse comando não retorna nenhuma saída.

3. Para descrever um tipo de coisa, execute o comando `describe-thing-type`. Esse comando produzirá uma saída com informações de configuração de enriquecimento de mensagens no campo `thing-type-properties`. O comando a seguir é um exemplo.

```
aws iot describe-thing-type \
    --thing-type-name "LightBulb"
```

A saída poderá ser parecida com a seguinte:

```
{
 "thingTypeName": "LightBulb",
 "thingTypeId": "bdf72512-0116-4392-8d79-bf39b17ef73d",
 "thingTypeArn": "arn:aws:iot:us-east-1:123456789012:thingtype/LightBulb",
 "thingTypeProperties": {
  "mqtt5Configuration": {
   "propagatingAttributes": [
    {
     "userPropertyKey": "iot:ClientId",
     "connectionAttribute": "iot:ClientId"
    },
    {
     "userPropertyKey": "test",
     "thingAttribute": "attribute"
    }
   ]
  }
 },
 "thingTypeMetadata": {
  "deprecated": false,
  "creationDate": "2024-10-18T17:37:46.656000+00:00"
 }
}
```

Para obter mais informações, consulte ???.

Marcando seus recursos AWS IoT

Para ajudá-lo a gerenciar e organizar seus grupos de objetos, tipos de objetos, regras de tópicos, trabalhos, auditorias programadas e perfis de segurança, opcionalmente, você pode atribuir seus próprios metadados a cada um desses recursos na forma de tags. Esta seção descreve tags e mostra a você como criá-las.

Para ajudá-lo a gerenciar seus custos relacionados o objetos, você pode criar grupos de faturamento que contenham objetos. Depois, você pode atribuir tags que contenham metadados a cada um desses grupos de faturamento. Esta seção também discute os grupos de faturamento e os comandos disponíveis para criá-los e gerenciá-los.

Conceitos Básicos de Tags

Você pode usar tags para categorizar seus AWS IoT recursos de maneiras diferentes (por exemplo, por finalidade, proprietário ou ambiente). Isso é útil quando você tem muitos recursos do mesmo tipo — é possível identificar rapidamente um recurso baseado nas tags que você atribuiu a ele. Cada tag consiste em uma chave e em um valor opcional, ambos definidos por você. Por exemplo, você pode definir um conjunto de tags para os seus tipos de objetos, o que ajuda a monitorar dispositivos por tipo. Recomendamos que você crie um conjunto de chave de tags que atenda às suas necessidades para cada tipo de recurso. Usar um conjunto consistente de chaves de tags facilita para você gerenciar seus recursos.

Você pode pesquisar e filtrar recursos conforme as tags que adicionar ou aplicar. Você também pode usar tags de grupo de faturamento para categorizar e rastrear seus custos. Também é possível usar tags para controlar o acesso aos recursos, conforme descrito em Utilização de tags com políticas do IAM.

Para facilitar o uso, o Editor de tags no AWS Management Console fornece uma forma central e unificada de criar e gerenciar suas tags. Para obter mais informações, consulte Trabalhando com o Tag Editor em Trabalhando com o AWS Management Console.

Você também pode trabalhar com tags usando a AWS CLI e a AWS IoT API. Você pode associar tags a grupos de objetos, tipos de objetos, regras de tópico, trabalhos, perfis de segurança, políticas, grupos de faturamento e aos pacotes e versões associados a itens ao criá-los usando o campo `Tags` nos comandos a seguir:

- CreateBillingGroup

- CreateDestination
- CreateDeviceProfile
- CreateDynamicThingGroup
- CreateJob
- CriarOTAUpdate
- CreatePolicy
- CreateScheduledAudit
- CreateSecurityProfile
- CreateServiceProfile
- CreateStream
- CreateThingGroup
- CreateThingType
- CreateTopicRule
- CreateWirelessGateway
- CreateWirelessDevice

Você pode adicionar, modificar ou excluir tags de recursos existentes que oferecem suporte a marcação, usando os seguintes comandos:

- TagResource
- ListTagsForResource
- UntagResource

É possível editar chaves de tags e valores, e é possível remover as tags de um recurso a qualquer momento. É possível definir o valor de uma tag a uma string vazia, mas não pode configurar o valor de um tag como nula. Caso adicione uma tag com a mesma chave de outra existente no recurso, o novo valor substituirá o antigo. Se você excluir um recurso, todas as tags associadas ao recurso também serão excluídas.

Restrições e limitações de tags

As restrições básicas a seguir se aplicam a tags:

- Número máximo de tags por recurso — 50

- Comprimento máximo da chave — 127 caracteres Unicode em UTF-8
- Valor máximo da chave — 255 caracteres Unicode em UTF-8
- As chaves e valores das tags diferenciam maiúsculas de minúsculas.
- Não use o prefixo `aws:` no nome nem no valor das suas tags. Está reservado para AWS uso. Você não pode editar nem excluir nomes ou valores de tag com esse prefixo. As tags com esse prefixo não contam para as tags por limite de recurso.
- Se o seu esquema de tags é usado em vários serviços e recursos , lembre-se de que outros serviços talvez tenham restrições em caracteres permitidos. Os caracteres permitidos incluem letras, espaços e números representáveis em UTF-8, além dos seguintes caracteres especiais: + - = . _ : / @.

Utilização de tags com políticas do IAM

É possível aplicar permissões em nível de recurso baseadas em tags às políticas do IAM que você usa para as ações de API do AWS IoT . Isso oferece a você mais controle sobre quais recursos um usuário pode criar, modificar ou usar. Você pode usar o elemento `Condition` (também chamado bloco `Condition`) juntamente com os seguintes valores e chaves de contexto de condição em uma política do IAM para controlar o acesso do usuário (permissões) baseado em tags de um recurso:

- Use `aws:ResourceTag/`*`tag-key`*`: `*`tag-value`*, para permitir ou negar ações do usuário em recursos com tags específicas.
- Use `aws:RequestTag/`*`tag-key`*`: `*`tag-value`* para exigir que uma tag específica seja (ou não seja) usada ao fazer uma solicitação de API para criar ou modificar um recurso que permite tags.
- Use `aws:TagKeys: [`*`tag-key`*`, ...]` para exigir que um conjunto específico de chaves de tag seja (ou não seja) usado ao fazer uma solicitação de API para criar ou modificar um recurso que permite tags.

> **Note**
>
> As chaves e valores de contexto da condição em uma política do IAM se aplicam somente às AWS IoT ações em que um identificador de um recurso capaz de ser marcado é um parâmetro obrigatório. Por exemplo, o uso de não DescribeEndpointé permitido ou negado com base nas chaves e valores do contexto de condição porque nenhum recurso marcável (grupos de coisas, tipos de coisas, regras de tópicos, trabalhos ou perfil de segurança) é referenciado nessa solicitação. Para obter mais informações sobre AWS IoT os recursos que

são marcáveis e as chaves de condição que eles suportam, leia Ações, recursos e chaves de condição para. AWS IoT

Para obter mais informações sobre o uso de tags, consulte Controlar o acesso usando tags no Guia do usuário do AWS Identity and Access Management . A seção Referência de política JSON do IAM desse guia detalhou a sintaxe, as descrições e os exemplos dos elementos, variáveis e lógica de avaliação das políticas JSON no IAM.

O exemplo de política a seguir aplica duas restrições baseadas em tags para as ações `ThingGroup`. Um usuário do IAM restrito por essa política:

- Não é possível criar um grupo de objetos com a tag “env=prod” (no exemplo, veja a linha `"aws:RequestTag/env" : "prod"`).
- Não é possível modificar ou acessar um grupo de objetos que possui uma tag "env=prod" existente (no exemplo, consulte a linha `"aws:ResourceTag/env" : "prod"`).

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "iot:CreateThingGroup",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/env": "prod"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:UpdateThingGroup"
      ],
      "Resource": "*",
      "Condition": {
```

```
        "StringEquals": {
          "aws:ResourceTag/env": "prod"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:UpdateThingGroup"
      ],
      "Resource": "*"
    }
  ]
}
```

Você também pode especificar vários valores de tag para uma determinada chave de tag, colocando-as em uma lista como esta:

```
          "StringEquals" : {
            "aws:ResourceTag/env" : ["dev", "test"]
          }
```

> **Note**
>
> Se você permitir ou negar aos usuários o acesso a recursos com base em tags, considere negar explicitamente aos usuários a capacidade de adicionar essas tags ou removê-las dos mesmos recursos. Caso contrário, é possível que um usuário contorne suas restrições e obtenha acesso a um recurso modificando as tags.

Grupos de cobrança

AWS IoT não permite que você aplique tags diretamente a itens individuais, mas permite que você coloque itens em grupos de cobrança e aplique tags a eles. Pois AWS IoT, a alocação de dados de custo e uso com base em tags é limitada aos grupos de cobrança.

AWS IoT Core pois recursos de LoRa WAN, como dispositivos sem fio e gateways, não podem ser adicionados aos grupos de cobrança. No entanto, eles podem ser associados a AWS IoT itens, que podem ser adicionados aos grupos de cobrança.

Os seguintes comandos estão disponíveis:

- AddThingToBillingGroup adiciona uma coisa a um grupo de faturamento.
- CreateBillingGroup cria um grupo de faturamento.
- DeleteBillingGroup exclui o grupo de faturamento.
- DescribeBillingGroup retorna informações sobre um grupo de faturamento.
- ListBillingGroups lista os grupos de faturamento que você criou.
- ListThingsInBillingGroup lista as coisas que você adicionou a determinado grupo de faturamento.
- RemoveThingFromBillingGroup remove a coisa específica do grupo de faturamento.
- UpdateBillingGroup atualiza Informações sobre o grupo de faturamento.
- CreateThing permite que você especifique um grupo de faturamento para a coisa ao criá-la.
- DescribeThing retorna a descrição de uma coisa, incluindo o grupo de faturamento ao qual ela pertence, se houver.

A API AWS IoT Wireless fornece essas ações para associar dispositivos e gateways sem fio a AWS IoT coisas.

- AssociateWirelessDeviceWithThing
- AssociateWirelessGatewayWithThing

Visualizar alocação de custos e dados de uso

Você pode usar tags de grupo de faturamento para categorizar e rastrear seus custos. Quando você aplica tags aos grupos de cobrança (e também aos itens que eles incluem), AWS gera um relatório de alocação de custos como um arquivo de valores separados por vírgula (CSV) com seu uso e custos agregados por suas tags. É possível aplicar tags que representem categorias de negócios (como centros de custos, nomes da aplicações ou proprietários) para organizar seus custos de vários serviços. Para obter mais informações sobre como usar tags de alocação de custos, consulte Usar etiquetas de alocação de custos no Guia do usuário de gerenciamento de custos e faturamento da AWS.

> **Note**
>
> Para associar com precisão os dados de uso e custo a essas objetos que você colocou em grupos de faturamento, cada dispositivo ou aplicativo deve:
>
> - Seja registrado como uma coisa em AWS IoT. Para obter mais informações, consulte Gerenciando dispositivos com AWS IoT.
> - Conecte-se ao agente de AWS IoT mensagens por meio do MQTT usando apenas o nome da coisa como ID do cliente. Para obter mais informações, consulte the section called "Protocolos de comunicação do dispositivo". Se o ID do cliente não corresponder ao nome da coisa, você poderá habilitar o anexo de coisa exclusiva para estabelecer a associação. Para obter mais informações, consulte ???.
> - Autenticar-se usando um certificado do cliente associado ao objeto.

As seguintes definições de preço estão disponíveis para grupos de faturamento (com base na atividade das objetos associadas ao grupo de faturamento):

- Conectividade (com base no nome do objeto usado como ID do cliente para se conectar).
- Sistema de mensagens (com base em mensagens de entrada e saída de um objeto; MQTT apenas).
- Operações Shadow (com base no objeto cuja mensagem acionou uma atualização de shadow).
- Regras acionadas (com base no objeto cuja mensagem de entrada acionou a regra; não se aplica às regras acionadas por eventos de ciclo de vida de MQTT).
- Atualizações do índice de objetos (com base no objeto adicionada ao índice).
- Ações remotas (com base no objeto atualizada).
- Relatórios de detecção de AWS IoT Device Defender (com base no objeto cuja atividade é reportada).

Os dados de custo e uso com base em tags (e reportados a um grupo de faturamento) não refletem as seguintes atividades:

- Operações de registro de dispositivos (incluindo atualizações de objetos, grupos de objetos e tipos de objetos). Para obter mais informações, consulte Gerenciando dispositivos com AWS IoT).
- Atualizações de índice do grupo de objetos (ao adicionar um grupo de objetos).

- Consultas de pesquisa de índice.
- Provisionamento de dispositivos.
- Relatórios de auditoria de AWS IoT Device Defender.

Segurança em AWS IoT

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de uma arquitetura de data center e rede criada para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O modelo de responsabilidade compartilhada descreve isto como segurança da nuvem e segurança na nuvem.

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa AWS os serviços na AWS nuvem. AWS também fornece serviços que você pode usar com segurança. Auditores de terceiros testam e verificam regularmente a eficácia da nossa segurança como parte dos programas de conformidade da AWS. Para saber mais sobre os programas de conformidade aplicáveis AWS IoT, consulte AWS Serviços no escopo por programa de conformidade.
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da sua empresa e as leis e normas aplicáveis.

Esta documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usar AWS IoT. Os tópicos a seguir mostram como configurar para atender AWS IoT aos seus objetivos de segurança e conformidade. Você também aprenderá a usar outros AWS serviços que ajudam a monitorar e proteger seus AWS IoT recursos.

Tópicos

- AWS IoT segurança
- Autenticação
- Autorização
- Proteção de dados em AWS IoT Core
- Gerenciamento de identidade e acesso para AWS IoT
- Registro em log e monitoramento
- Validação de conformidade para AWS IoT Core
- Resiliência no AWS IoT Core
- Usando AWS IoT Core com interface VPC endpoints
- Segurança da infraestrutura em AWS IoT

- Monitoramento de segurança de frotas ou dispositivos de produção com Core AWS IoT
- Melhores práticas de segurança em AWS IoT Core
- AWS treinamento e certificação

AWS IoT segurança

Cada dispositivo ou cliente conectado deve ter uma credencial para interagir com o AWS IoT. Todo o tráfego de ida e volta AWS IoT é enviado com segurança pelo Transport Layer Security (TLS). AWS os mecanismos de segurança na nuvem protegem os dados à medida que eles se movem entre AWS IoT outros AWS serviços.

- Você é responsável por gerenciar credenciais de dispositivo (certificados X.509, credenciais da AWS , identidades do Amazon Cognito, identidades federadas ou tokens de autenticação personalizada) e políticas no AWS IoT. Você também é responsável por atribuir identidades exclusivas a cada dispositivo e gerenciar as permissões para cada um ou para o grupo de dispositivos.
- Seus dispositivos se conectam AWS IoT usando certificados X.509 ou identidades do Amazon Cognito por meio de uma conexão TLS segura. Durante a pesquisa e o desenvolvimento, e para alguns aplicativos que fazem chamadas ou usam API WebSockets, você também pode se autenticar usando usuários e grupos do IAM ou tokens de autenticação personalizados. Para obter mais informações, consulte Usuários, grupos e funções do IAM.

- Ao usar a AWS IoT autenticação, o agente de mensagens é responsável por autenticar seus dispositivos, ingerir dados do dispositivo com segurança e conceder ou negar as permissões de acesso que você especifica para seus dispositivos usando políticas. AWS IoT
- Ao usar a autenticação personalizada, um autorizador personalizado é responsável por autenticar seus dispositivos e conceder ou negar as permissões de acesso que você especifica para seus dispositivos usando AWS IoT nossas políticas do IAM.
- O mecanismo de AWS IoT regras encaminha os dados do dispositivo para outros dispositivos ou outros AWS serviços de acordo com as regras definidas por você. Ele é usado AWS Identity and Access Management para transferir dados com segurança para seu destino final. Para obter mais informações, consulte Gerenciamento de identidade e acesso para AWS IoT.

Autenticação

A autenticação é um mecanismo no qual você verifica a identidade de um cliente ou servidor. A autenticação do servidor é o processo em que dispositivos ou outros clientes garantem que estão se comunicando com um AWS IoT endpoint real. A autenticação do cliente é o processo com AWS IoT o qual dispositivos ou outros clientes se autenticam.

Visão geral do certificado X.509

Os certificados X.509 são certificados digitais que usam a infraestrutura de chave pública X.509 padrão para associar uma chave pública a uma identidade contida em um certificado. Os certificados X.509 são emitidos por uma entidade confiável chamada CA (autoridade de certificação). A CA mantém um ou mais certificados especiais chamados certificados CA que são usados para emitir certificados X.509. Somente a autoridade de certificação tem acesso aos certificados CA. As cadeias de certificados X.509 são usadas tanto para autenticação do servidor por clientes quanto para autenticação do cliente pelo servidor.

Autenticação do servidor

Quando seu dispositivo ou outro cliente tentar se conectar AWS IoT Core, o AWS IoT Core servidor enviará um certificado X.509 que seu dispositivo usa para autenticar o servidor. A autenticação ocorre na camada TLS por meio da validação da cadeia de certificados X.509. Esse é o mesmo método usado pelo seu navegador ao acessar uma URL HTTPS. Para usar certificados de sua própria autoridade de certificação, consulte Gerenciar os certificados CA.

Quando seus dispositivos ou outros clientes estabelecem uma conexão TLS com um AWS IoT Core endpoint, AWS IoT Core apresenta uma cadeia de certificados que os dispositivos usam para verificar se estão se comunicando AWS IoT Core e não se passando por outro servidor. AWS IoT Core A cadeia apresentada depende de uma combinação do tipo de endpoint ao qual o dispositivo está se conectando e do conjunto de cifras que o cliente AWS IoT Core negociou durante o handshake TLS.

Tipos de endpoint

AWS IoT Core suportesiot:Data-ATS. iot:Data-ATSos endpoints apresentam um certificado de servidor assinado por uma CA da Amazon Trust Services.

Os certificados apresentados pelos endpoints ATS são assinados por Starfield. Algumas implementações de cliente TLS exigem a validação da raiz da confiança e exigem que os certificados da CA Starfield estejam instalados nos armazenamentos de confiança do cliente.

> **Warning**
>
> O uso de um método de fixação de certificado que adiciona hash a todo o certificado (incluindo o nome do emissor, etc.) não é recomendado porque isso fará com que a verificação do certificado falha, pois os certificados ATS que fornecemos são de assinatura cruzada Starfield e têm um nome de emissor diferente.

> **Important**
>
> Use endpoints iot:Data-ATS. Os certificados da Symantec e da Verisign foram descontinuados e não são mais suportados pelo. AWS IoT Core

Você pode usar o comando describe-endpoint para criar seu endpoint ATS.

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

O comando describe-endpoint retorna um endpoint no seguinte formato.

```
account-specific-prefix.iot.your-region.amazonaws.com
```

> **Note**
>
> Na primeira vez que `describe-endpoint` for chamado, um endpoint será criado. Todas as chamadas subsequentes para `describe-endpoint` retornam o mesmo endpoint.

> **Note**
>
> Para ver seu `iot:Data-ATS` endpoint no AWS IoT Core console, escolha Configurações. O console exibe somente o endpoint `iot:Data-ATS`.

Criando um **`IotDataPlaneClient`** com o AWS SDK for Java

Para criar um `IotDataPlaneClient` que use um endpoint `iot:Data-ATS`, faça o seguinte.

- Crie um `iot:Data-ATS` endpoint usando a DescribeEndpointAPI.
- Especifique esse endpoint ao criar o `IotDataPlaneClient`.

O exemplo a seguir executa as duas operações.

```
public void setup() throws Exception {
        IotClient client =
 IotClient.builder().credentialsProvider(CREDENTIALS_PROVIDER_CHAIN).region(Region.US_EAST_1).b
        String endpoint = client.describeEndpoint(r -> r.endpointType("iot:Data-
ATS")).endpointAddress();
        iot = IotDataPlaneClient.builder()
                                .credentialsProvider(CREDENTIALS_PROVIDER_CHAIN)
                                .endpointOverride(URI.create("https://" + endpoint))
                                .region(Region.US_EAST_1)
                                .build();
}
```

Certificados CA para autenticação do servidor

Dependendo do tipo de endpoint de dados que você está usando e do pacote de criptografia negociado, os certificados de autenticação AWS IoT Core do servidor são assinados por um dos seguintes certificados CA raiz:

Endpoints do Amazon Trust Services (preferencial)

> **Note**
>
> Talvez seja necessário clicar com o botão direito do mouse nesses links e selecionar Salvar link como... para salvar esses certificados como arquivos.

- Chave RSA de 2.048 bits: Amazon Root CA 1.
- Chave RSA de 4.096 bits: Amazon Root CA 2. Reservado para uso futuro.
- Chave ECC de 256 bits: Amazon Root CA 3.
- Chave ECC de 384 bits: Amazon Root CA 4. Reservado para uso futuro.

Esses certificados são todos assinados pelo Certificado da CA raiz Starfield. Todas as novas AWS IoT Core regiões, começando com o lançamento em 9 de maio de 2018 AWS IoT Core na região Ásia-Pacífico (Mumbai), oferecem somente certificados ATS.

VeriSign Endpoints (antigos)

- Chave RSA de 2048 bits: certificado CA VeriSign raiz G5 público primário de classe 3

Diretrizes de autenticação do servidor

Há muitas variáveis que podem afetar a capacidade de um dispositivo de validar o certificado de autenticação do servidor do AWS IoT Core . Por exemplo, os dispositivos podem ter memória muito limitada para armazenar todos os certificados da CA raiz possíveis ou os dispositivos podem implementar um método não padrão de validação de certificado. Por estas razões, sugerimos seguir estas diretrizes:

- Recomendamos usar seu endpoint do ATS e instalar todos os certificados Amazon Root CA compatíveis.
- Se você não puder armazenar todos esses certificados em seu dispositivo, e se seus dispositivos não usarem a validação baseada em ECC, será possível omitir os certificados ECC Amazon Root CA 3 e Amazon Root CA 4. Se seus dispositivos implementarem a validação de certificado baseada em RSA, será possível omitir os certificados RSA Amazon Root CA 1 e Amazon Root CA 2. Talvez seja necessário clicar com o botão direito do mouse nesses links e selecionar Salvar link como... para salvar esses certificados como arquivos.

- Se você estiver tendo problemas de validação de certificado do servidor ao se conectar ao endpoint do ATS, tente adicionar o certificado da CA raiz da Amazon relevante ao armazenamento confiável. Talvez seja necessário clicar com o botão direito do mouse nesses links e selecionar Salvar link como... para salvar esses certificados como arquivos.
  - Amazon Root CA 1 de assinatura cruzada
  - Amazon Root CA 2 de assinatura cruzada - Reservado para uso futuro.
  - Amazon Root CA 3 de assinatura cruzada
  - Amazon Root CA 4 de assinatura cruzada - Reservado para uso futuro.
- Se você estiver tendo problemas de validação de certificado do servidor, talvez seja necessário confiar explicitamente na CA raiz. Tente adicionar o Starfield Root CA Certificate ao seu armazenamento confiável.
- Se você ainda tiver problemas depois de executar as etapas acima, entre em contato com o Suporte ao desenvolvedor da AWS.

> **Note**
>
> Os certificados CA têm uma data de expiração depois da qual não podem ser usados para validar um certificado de servidor. O certificados da CA podem precisar ser substituídos antes da data de expiração. Você deve verificar se é possível atualizar o certificado CA raiz em todos os seus dispositivos ou clientes para ajudar a garantir uma conectividade contínua e manter-se atualizado com as melhores práticas de segurança.

> **Note**
>
> Ao se conectar ao AWS IoT Core código do seu dispositivo, passe o certificado para a API que você está usando para se conectar. A API usada varia conforme o SDK. Para obter mais informações, consulte o AWS IoT Core Dispositivo SDKs.

Autenticação de cliente

AWS IoT oferece suporte a três tipos de princípios de identidade para autenticação de dispositivos ou clientes:

- Certificados do cliente X.509

- Usuários, grupos e funções do IAM
- Identidades do Amazon Cognito

Essas identidades podem ser usadas com dispositivos, aplicativos móveis, web ou de desktop. Eles podem até mesmo ser usados por um usuário digitando AWS IoT comandos de interface de linha de comando (CLI). Normalmente, AWS IoT os dispositivos usam certificados X.509, enquanto os aplicativos móveis usam identidades do Amazon Cognito. Os aplicativos da Web e de desktop usam identidades federadas ou do IAM. Os comandos da AWS CLI usam o IAM. Para obter mais informações sobre identidades do IAM, consulte Gerenciamento de identidade e acesso para AWS IoT.

Certificados do cliente X.509

Os certificados X.509 oferecem AWS IoT a capacidade de autenticar conexões de clientes e dispositivos. Os certificados do cliente devem ser registrados AWS IoT antes que o cliente possa se comunicar com AWS IoT. Um certificado de cliente pode ser registrado em vários Conta da AWS s no mesmo Região da AWS para facilitar a movimentação de dispositivos entre seus Conta da AWS s na mesma região. Consulte Usando certificados de cliente X.509 em vários Conta da AWS s com registro de várias contas para obter mais informações.

Recomendamos que cada dispositivo ou cliente receba um certificado exclusivo para habilitar ações de gerenciamento de clientes refinadas, incluindo a revogação de certificados. Os dispositivos e os clientes também devem oferecer suporte à rotação e à substituição de certificados para ajudar a garantir uma operação contínua conforme os certificados expiram.

Para acessar informações sobre como usar certificados X.509 para comportar alguns dispositivos, consulte Provisionamento de dispositivos para conhecer as diferentes opções de gerenciamento e provisionamento de certificados compatíveis com AWS IoT .

AWS IoT suporta esses tipos de certificados de cliente X.509:

- Certificados X.509 gerados por AWS IoT
- Certificados X.509 assinados por uma CA registrada com. AWS IoT
- Certificados X.509 assinados por uma CA que não está registrada no AWS IoT.

Esta seção descreve como gerenciar certificados X.509 no AWS IoT. Você pode usar o AWS IoT console ou AWS CLI realizar estas operações de certificado:

- Crie certificados de AWS IoT cliente
- Criar seus próprios certificados de cliente
- Registrar um certificado de cliente
- Ativar ou desativar um certificado de cliente
- Revogar um certificado de cliente

Para obter mais informações sobre os AWS CLI comandos que executam essas operações, consulte a Referência da AWS IoT CLI.

Usar certificados do cliente X.509

Os certificados X.509 autenticam conexões de clientes e dispositivos a. AWS IoT Os certificados X.509 oferecem vários benefícios em relação a outros mecanismos de identificação e autenticação. Os certificados X.509 permitem que as chaves assimétricas sejam usadas com os dispositivos. Por exemplo, é possível gravar chaves privadas em um armazenamento seguro de um dispositivo para que o material criptográfico confidencial nunca saia do dispositivo. Os certificados X.509 fornecem opções mais fortes de autenticação de cliente em outros esquemas, como nome de usuário e senha ou tokens do portador, porque a chave privada nunca deixa o dispositivo.

AWS IoT autentica certificados de cliente usando o modo de autenticação de cliente do protocolo TLS. O suporte do TLS está disponível em muitas linguagens de programação e sistemas operacionais e é comumente usado para a criptografia de dados. Na autenticação de cliente TLS, AWS IoT solicita um certificado de cliente X.509 e valida o status do certificado em Conta da AWS relação a um registro de certificados. Em seguida, desafia o cliente a obter uma prova de propriedade da chave privada que corresponde à chave pública contida no certificado. AWS IoT exige que os clientes enviem a extensão Server Name Indication (SNI) para o protocolo Transport Layer Security (TLS). Para obter mais informações sobre como configurar a extensão SNI, consulte Segurança de transporte em AWS IoT Core.

Para facilitar uma conexão segura e consistente do cliente com o AWS IoT núcleo, um certificado de cliente X.509 deve possuir o seguinte:

- Registrado no AWS IoT Core. Para obter mais informações, consulte Registrar um certificado de cliente.
- Um status de `ACTIVE`. Para obter mais informações, consulte Ativar ou desativar um certificado de cliente.
- Ainda não atingiu a data de expiração do certificado.

Você pode criar certificados de cliente que usam a CA raiz da Amazon e usar seus próprios certificados de cliente assinados por outra autoridade de certificação (CA). Para obter mais informações sobre o uso do AWS IoT console para criar certificados que usam a Amazon Root CA, consulteCrie certificados de AWS IoT cliente. Para obter mais informações sobre como usar seus próprios certificados X.509, consulte Criar seus próprios certificados de cliente.

A data e a hora de quando os certificados assinados por um certificado CA expiram são definidas quando o certificado é criado. Os certificados X.509 gerados pela AWS IoT expiram à meia-noite UTC de 31 de dezembro de 2049 (2049-12-31T 23:59:59 Z).

AWS IoT Device Defender pode realizar auditorias em você Conta da AWS e em dispositivos que oferecem suporte às melhores práticas comuns de segurança de IoT. Isso inclui gerenciar as datas de expiração dos certificados X.509 assinados pela sua CA ou pela CA raiz da Amazon. Para obter mais informações sobre como gerenciar a data de expiração de um certificado, consulte Certificado de dispositivo expirando e Certificado CA expirando.

No AWS IoT blog oficial, um mergulho mais profundo no gerenciamento da rotação de certificados de dispositivos e nas melhores práticas de segurança é explorado em Como gerenciar a rotação de certificados de dispositivos de IoT usando. AWS IoT

Usando certificados de cliente X.509 em vários Conta da AWS s com registro de várias contas

O registro de várias contas possibilita a movimentação de dispositivos entre suas Conta da AWS na mesma região ou em regiões diferentes. Você pode registrar, testar e configurar um dispositivo em uma conta de pré-produção e, em seguida, registrar e usar o mesmo dispositivo e certificado de dispositivo em uma conta de produção. Você também pode registrar o certificado do cliente no dispositivo ou os certificados do dispositivo sem uma CA registrada com AWS IoT. Para obter mais informações, consulte Registrar um certificado de cliente assinado por uma CA não registrada (CLI).

> **Note**
>
> Os certificados usados para registro de várias contas são compatíveis com os tipos de endpoint `iot:Data-ATS`, `iot:Data` (legado), `iot:Jobs` e `iot:CredentialProvider`. Para obter mais informações sobre endpoints de AWS IoT dispositivos, consulteAWS IoT dados do dispositivo e endpoints de serviço.

Os dispositivos que usam o registro de várias contas devem enviar a extensão Server Name Indication (SNI) para o protocolo Transport Layer Security (TLS) e fornecer o endereço completo

do endpoint no `host_name` campo, quando se conectarem. AWS IoT AWS IoT usa o endereço do endpoint `host_name` para rotear a conexão para a AWS IoT conta correta. Os dispositivos existentes que não enviarem um endereço de endpoint válido em `host_name` continuarão a funcionar, mas não poderão usar os recursos que exigem essas informações. Para obter mais informações sobre a extensão SNI e para saber como identificar o endereço de endpoint para o campo `host_name`, consulte Segurança de transporte em AWS IoT Core.

Como usar o registro de várias contas

1. Você pode registrar os certificados de dispositivo com uma CA. Você pode registrar a CA de assinatura em várias contas no modo `SNI_ONLY` e usar essa CA para registrar o mesmo certificado de cliente em várias contas. Para obter mais informações, consulte Registrar um certificado da CA no modo SNI_ONLY (CLI) - Recomendado.
2. Você pode registrar os certificados de dispositivo sem uma CA. Consulte Registrar um certificado de cliente assinado por uma CA não registrada (CLI). O registro de uma CA é opcional. Você não precisa registrar a CA com a qual assinou os certificados do dispositivo AWS IoT.

Algoritmos de assinatura de certificados suportados pelo AWS IoT

AWS IoT suporta os seguintes algoritmos de assinatura de certificados:

- SHA256COM A RSA
- SHA384COM A RSA
- SHA512COM A RSA
- SHA256WITHRSAANDMGF1 (RSASSA-PSS)
- SHA384WITHRSAANDMGF1 (RSASSA-PSS)
- SHA512WITHRSAANDMGF1 (RSASSA-PSS)
- DSA_COM_ SHA256
- ECDSA-COM- SHA256
- ECDSA-COM- SHA384
- ECDSA-COM- SHA512

Para obter mais informações sobre autenticação e segurança de certificados, consulte Qualidade da chave de certificado do dispositivo.

> **Note**
>
> A solicitação de assinatura de certificado (CSR) deve incluir uma chave pública. A chave pode ser uma chave RSA com um comprimento de pelo menos 2.048 bits ou uma chave ECC das curvas do NIST P-256, NIST P-384 ou NIST P-521. Para obter mais informações, consulte CreateCertificateFromCsr no AWS IoT API Reference Guide (Guia de referência da API do &CDSlong;).

Algoritmos principais suportados por AWS IoT

A tabela abaixo mostra o suporte para os principais algoritmos:

Algoritmo de chave	Algoritmo de assinatura de certificado	Versão do TLS	Compatível? Yes ou No
RSA com um tamanho de chave de pelo menos 2048 bits	Todos	TLS 1.2 TLS 1.3	Sim
ECC NIST P-256/P-384/ P-521	Todos	TLS 1.2 TLS 1.3	Sim
RSA-PSS com um tamanho de chave de pelo menos 2048 bits	Todos	TLS 1.2	Não
RSA-PSS com um tamanho de chave de pelo menos 2048 bits	Todos	TLS 1.3	Sim

Para criar um certificado usando CreateCertificateFromCSR, você pode usar um algoritmo de chave compatível para gerar uma chave pública para sua CSR. Para registrar seu próprio certificado usando RegisterCertificatenossa RegisterCertificateWithoutCA, você pode usar um algoritmo de chave compatível para gerar uma chave pública para o certificado.

Para obter mais informações, consulte Políticas de segurança.

Crie certificados de AWS IoT cliente

AWS IoT fornece certificados de cliente assinados pela autoridade de certificação (CA) raiz da Amazon.

Este tópico descreve como criar um certificado de cliente assinado pela autoridade de certificação raiz da Amazon e como fazer download dos arquivos do certificado. Depois de criar os arquivos do certificado de cliente, é necessário instalá-los no cliente.

> **Note**
>
> Cada certificado de cliente X.509 fornecido pelo AWS IoT contém os atributos do emissor e do sujeito que você definiu no momento da criação do certificado. Os atributos do certificado ficam imutáveis somente após a criação do certificado.

Você pode usar o AWS IoT console ou o AWS CLI para criar um AWS IoT certificado assinado pela autoridade de certificação Amazon Root.

Crie um AWS IoT certificado (console)

Para criar um AWS IoT certificado usando o AWS IoT console

1. Faça login no Console de gerenciamento da AWS e abra o AWS IoT console.
2. No painel de navegação, selecione Segurança, Certificados e, em seguida, Criar.
3. Selecione Criação de certificado com um clique (recomendado) – Criar certificado.
4. Na página Certificado criado, faça download dos arquivos do certificado de cliente do objeto, da chave pública e da chave privada para um local seguro. Esses certificados gerados por só AWS IoT estão disponíveis para uso com AWS IoT serviços.

   Se você também precisar do arquivo do certificado CA raiz da Amazon, essa página também tem o link da página de onde é possível fazer download dele.
5. Um certificado de cliente foi criado e registrado no AWS IoT. É necessário ativar o certificado antes de usá-lo em um cliente.

   Para ativar o certificado de cliente agora, selecione Ativar. Se você não quiser ativar o certificado agora, consulte Ativar um certificado de cliente (console) para saber como ativar o certificado posteriormente.

6. Para anexar uma política ao certificado, selecione Anexar uma política.

   Se você não quiser anexar uma política agora, selecione Concluído para concluir. É possível anexar uma política posteriormente.

Depois de concluir o procedimento, instale os arquivos do certificado no cliente.

Crie um AWS IoT certificado (CLI)

O AWS CLI fornece o create-keys-and-certificate comando para criar certificados de cliente assinados pela autoridade de certificação Amazon Root. Esse comando, no entanto, não faz download do arquivo de certificado CA raiz da Amazon. É possível fazer download do arquivo de certificado CA raiz da Amazon em Certificados CA para autenticação do servidor.

Esse comando cria arquivos de chave privada, chave pública e certificado X.509 e registra e ativa o certificado com. AWS IoT

```
aws iot create-keys-and-certificate \
    --set-as-active \
    --certificate-pem-outfile certificate_filename.pem \
    --public-key-outfile public_filename.key \
    --private-key-outfile private_filename.key
```

Se você não deseja ativar o certificado ao criá-lo e registrá-lo, esse comando cria os arquivos de chave privada, chave pública e certificado X.509 e registra o certificado, mas não o ativa. Ativar um certificado de cliente (CLI) descreve como ativar o certificado posteriormente.

```
aws iot create-keys-and-certificate \
    --no-set-as-active \
    --certificate-pem-outfile certificate_filename.pem \
    --public-key-outfile public_filename.key \
    --private-key-outfile private_filename.key
```

Instale os arquivos do certificado no cliente.

Criar seus próprios certificados de cliente

AWS IoT suporta certificados de cliente assinados por qualquer autoridade de certificação (CA) raiz ou intermediária. AWS IoT usa certificados CA para verificar a propriedade dos certificados. Para usar certificados de dispositivo assinados por uma CA que não seja a CA da Amazon, o certificado da CA deve ser registrado AWS IoT para que possamos verificar a propriedade do certificado do dispositivo.

AWS IoT suporta várias maneiras de trazer seus próprios certificados (BYOC):

- Primeiro, registre a CA usada para assinar os certificados do cliente e, em seguida, registre esses certificados individuais. Se você quiser registrar o dispositivo ou cliente em seu certificado de cliente quando ele se conectar pela primeira vez AWS IoT (também conhecido como Just-in-Time Provisioning), você deve registrar a CA de assinatura e ativar o registro automático. AWS IoT
- Se você não conseguir registrar a CA de assinatura, poderá optar por registrar certificados de cliente sem CA. Para dispositivos registrados sem CA, será necessário apresentar a Server Name Indication (SNI) ao conectá-los ao AWS IoT.

> **Note**
>
> Para registrar certificados de cliente usando a CA, você deve registrar a CA assinante com AWS IoT, e não com qualquer outra CAs na hierarquia.

> **Note**
>
> Um certificado da CA pode ser registrado no modo `DEFAULT` por apenas uma conta em uma região. Um certificado da CA pode ser registrado no modo `SNI_ONLY` por várias contas em uma região.

Para acessar mais informações sobre como usar certificados X.509 para comportar alguns dispositivos, consulte Provisionamento de dispositivos para conhecer as diferentes opções de gerenciamento e provisionamento de certificados compatíveis com AWS IoT .

Tópicos

- Gerenciar os certificados CA

- Criar um certificado de cliente usando o certificado CA

Gerenciar os certificados CA

Esta seção descreve tarefas comuns para gerenciar seus próprios certificados de autoridade de certificação (CA).

Você pode registrar sua autoridade de certificação (CA) AWS IoT se estiver usando certificados de cliente assinados por uma CA que AWS IoT não reconhece.

Se você quiser que os clientes registrem automaticamente seus certificados de cliente AWS IoT quando se conectarem pela primeira vez, a CA que assinou os certificados de cliente deve estar registrada AWS IoT. Caso contrário, não é necessário registrar o certificado CA que assinou os certificados de cliente.

> **Note**
>
> Um certificado da CA pode ser registrado no modo `DEFAULT` por apenas uma conta em uma região. Um certificado da CA pode ser registrado no modo `SNI_ONLY` por várias contas em uma região.

Tópicos:

- Criar um certificado CA
- Registrar o certificado CA
- Desativar um certificado CA

Criar um certificado CA

Se você não tiver um certificado da CA, poderá usar as ferramentas do OpenSSL v1.1.1i para criar um.

> **Note**
>
> Você não pode realizar esse procedimento no AWS IoT console.

Como criar um certificado da CA usando ferramentas do OpenSSL v1.1.1i

1. Gere um par de chaves.

```
openssl genrsa -out root_CA_key_filename.key 2048
```

2. Use a chave privada do par de chaves para gerar um certificado CA.

```
openssl req -x509 -new -nodes \
    -key root_CA_key_filename.key \
    -sha256 -days 1024 \
    -out root_CA_cert_filename.pem
```

Registrar o certificado CA

Esses procedimentos descrevem como registrar um certificado de uma autoridade de certificação (CA) que não seja a CA da Amazon. AWS IoT Core usa certificados CA para verificar a propriedade dos certificados. Para usar certificados de dispositivo assinados por uma CA que não seja a CA da Amazon, você deve registrar o certificado da CA AWS IoT Core para que ela possa verificar a propriedade do certificado do dispositivo.

Registrar um certificado CA (console)

> **Note**
>
> Para registrar um certificado da CA no console, inicie no console em Registrar certificado da CA. Você pode registrar sua CA no modo de várias contas e sem a necessidade de fornecer um certificado de verificação ou acesso à chave privada. Uma CA pode ser registrada no modo de várias contas por várias Contas da AWS na mesma Região da AWS. Você pode registrar sua CA no modo de conta única fornecendo um certificado de verificação e comprovante de propriedade da chave privada da CA.

Registrar um certificado CA (CLI)

Você pode registrar um certificado da CA no modo `DEFAULT` ou `SNI_ONLY`. Uma CA pode ser registrada no `DEFAULT` modo um Conta da AWS em um Região da AWS. Uma CA pode ser registrada no `SNI_ONLY` modo por várias Contas da AWS na mesma Região da AWS. Para obter mais informações sobre o modo de certificado da CA, consulte certificateMode.

> **Note**
>
> Recomendamos que você registre uma CA no modo `SNI_ONLY`. Você não precisa fornecer um certificado de verificação ou acesso à chave privada e pode registrar a CA por várias vezes Contas da AWS na mesma Região da AWS.

Registrar um certificado da CA no modo SNI_ONLY (CLI) - Recomendado

Pré-requisitos

Certifique-se de que você tenha o seguinte no seu computador antes de continuar:

- O arquivo de certificado da CA raiz (referenciado no exemplo a seguir como *root_CA_cert_filename.pem*)
- O OpenSSL v1.1.1i ou mais recente

Para registrar um certificado CA no **SNI_ONLY** modo usando o AWS CLI

1. Registre o certificado CA com AWS IoT. Usando o comando register-ca-certificate, insira o nome do arquivo do certificado da CA. Para obter mais informações, consulte register-ca-certificate na Referência de comandos da AWS CLI .

   ```
   aws iot register-ca-certificate \
       --ca-certificate file://root_CA_cert_filename.pem \
       --certificate-mode SNI_ONLY
   ```

   Se for bem-sucedido, esse comando retornará *certificateId* o.

2. Neste ponto, o certificado CA foi registrado, AWS IoT mas está inativo. O certificado da CA deve ser ativado antes do registro de qualquer certificado de cliente assinado por ele.

   Esta etapa ativa o certificado CA.

   Para ativar o certificado da CA, use o comando update-certificate da seguinte forma. Para obter mais informações, consulte update-certificate na Referência de comandos da AWS CLI .

   ```
   aws iot update-ca-certificate \
       --certificate-id certificateId \
   ```

```
    --new-status ACTIVE
```

Para ver o status do certificado da CA, use o comando describe-ca-certificate. Para obter mais informações, consulte describe-ca-certificate na Referência de comandos da AWS CLI .

Registrar um certificado da CA no modo **DEFAULT** (CLI)

Pré-requisitos

Certifique-se de que você tenha o seguinte no seu computador antes de continuar:

- O arquivo de certificado da CA raiz (referenciado no exemplo a seguir como *root_CA_cert_filename.pem*)
- O arquivo de chave privada do certificado da CA raiz (referenciado no exemplo a seguir como *root_CA_key_filename.key*)
- O OpenSSL v1.1.1i ou mais recente

Para registrar um certificado CA no **DEFAULT** modo usando o AWS CLI

1. Para obter um código de registro de AWS IoT, useget-registration-code. Salve o `registrationCode` retornado para usar como o `Common Name` do certificado de verificação da chave privada. Para obter mais informações, consulte get-registration-code na Referência de comandos da AWS CLI .

   ```
   aws iot get-registration-code
   ```

2. Gere um par de chaves para o certificado de verificação de chave privada:

   ```
   openssl genrsa -out verification_cert_key_filename.key 2048
   ```

3. Crie uma solicitação de assinatura de certificado (CSR) para o certificado de verificação da chave privada. Defina o campo `Common Name` do certificado como o `registrationCode` retornado por get-registration-code.

   ```
   openssl req -new \
       -key verification_cert_key_filename.key \
       -out verification_cert_csr_filename.csr
   ```

Você será solicitado a fornecer algumas informações, incluindo o Common Name para o certificado.

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
    State or Province Name (full name) []:
    Locality Name (for example, city) []:
    Organization Name (for example, company) []:
    Organizational Unit Name (for example, section) []:
    Common Name (e.g. server FQDN or YOUR name) []:your_registration_code
    Email Address []:

    Please enter the following 'extra' attributes
    to be sent with your certificate request
    A challenge password []:
    An optional company name []:
```

4. Use uma CSR para criar o certificado de verificação de chave privada.

```
openssl x509 -req \
    -in verification_cert_csr_filename.csr \
    -CA root_CA_cert_filename.pem \
    -CAkey root_CA_key_filename.key \
    -CAcreateserial \
    -out verification_cert_filename.pem \
    -days 500 -sha256
```

5. Registre o certificado CA com AWS IoT. Transmita o nome do arquivo do certificado da CA e o nome do arquivo do certificado de verificação da chave privada para o comando register-ca-certificate, da seguinte maneira. Para obter mais informações, consulte register-ca-certificate na Referência de comandos da AWS CLI .

```
aws iot register-ca-certificate \
    --ca-certificate file://root_CA_cert_filename.pem \
    --verification-cert file://verification_cert_filename.pem
```

Esse comando retorna o*certificateId*, se for bem-sucedido.

6. Neste momento, o certificado CA foi registrado AWS IoT , mas não está ativo. O certificado da CA deve ser ativado antes do registro de qualquer certificado de cliente assinado por ele.

   Esta etapa ativa o certificado CA.

   Para ativar o certificado da CA, use o comando update-certificate da seguinte forma. Para obter mais informações, consulte update-certificate na Referência de comandos da AWS CLI .

   ```
   aws iot update-ca-certificate \
       --certificate-id certificateId \
       --new-status ACTIVE
   ```

Para ver o status do certificado da CA, use o comando describe-ca-certificate. Para obter mais informações, consulte describe-ca-certificate na Referência de comandos da AWS CLI .

Crie um certificado de verificação de CA para registrar o certificado da CA no console

> **Note**
>
> Esse procedimento só deve ser usado se você estiver registrando um certificado CA no AWS IoT console.
> Se você não chegou a esse procedimento a partir do AWS IoT console, inicie o processo de registro do certificado CA no console em Registrar certificado CA.

Certifique-se de que você tenha o seguinte no mesmo computador antes de continuar:

- O arquivo de certificado da CA raiz (referenciado no exemplo a seguir como *root_CA_cert_filename.pem*)
- O arquivo de chave privada do certificado da CA raiz (referenciado no exemplo a seguir como *root_CA_key_filename.key*)
- O OpenSSL v1.1.1i ou mais recente

Como usar a interface de linha de comando para criar um certificado de verificação de CA a fim de registrar seu certificado da CA no console

1. Substitua *verification_cert_key_filename.key* pelo nome do arquivo da chave do certificado de verificação que você deseja criar (por exemplo, **verification_cert.key**). Em seguida, execute esse comando para gerar um par de chaves para o certificado de verificação de chave privada:

   ```
   openssl genrsa -out verification_cert_key_filename.key 2048
   ```

2. Substitua *verification_cert_key_filename.key* pelo nome do arquivo da chave criado na etapa 1.

   Substitua *verification_cert_csr_filename.csr* pelo nome do arquivo de solicitação de assinatura de certificado (CSR) que você deseja criar. Por exemplo, .**verification_cert.csr**

   Execute esse comando para criar o arquivo CRS.

   ```
   openssl req -new \
       -key verification_cert_key_filename.key \
       -out verification_cert_csr_filename.csr
   ```

   O comando solicita que você forneça informações adicionais que serão explicadas posteriormente.
3. No AWS IoT console, no contêiner do certificado de verificação, copie o código de registro.
4. As informações solicitadas pelo comando openssl são mostradas no exemplo a seguir. Com exceção do campo `Common Name`, você pode inserir seus próprios valores ou mantê-los em branco.

   No campo `Common Name`, cole o código de registro que você copiou na etapa anterior.

   ```
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----
   Country Name (2 letter code) [AU]:
   ```

```
    State or Province Name (full name) []:
    Locality Name (for example, city) []:
    Organization Name (for example, company) []:
    Organizational Unit Name (for example, section) []:
    Common Name (e.g. server FQDN or YOUR name) []:your_registration_code
    Email Address []:

    Please enter the following 'extra' attributes
    to be sent with your certificate request
    A challenge password []:
    An optional company name []:
```

Depois de terminar, o comando cria o arquivo CSR.

5. Substitua o *verification_cert_csr_filename.csr* pelo *verification_cert_csr_filename.csr* que você usou na etapa anterior.

   Substitua *root_CA_cert_filename.pem* pelo nome do arquivo do certificado da CA que você deseja registrar.

   Substitua *root_CA_key_filename.key* pelo nome do arquivo da chave privada do certificado da CA.

   Substitua *verification_cert_filename.pem* pelo nome do arquivo do certificado de verificação que você deseja criar. Por exemplo, .**verification_cert.pem**

   ```
   openssl x509 -req \
       -in verification_cert_csr_filename.csr \
       -CA root_CA_cert_filename.pem \
       -CAkey root_CA_key_filename.key \
       -CAcreateserial \
       -out verification_cert_filename.pem \
       -days 500 -sha256
   ```

6. Depois que o comando OpenSSL for concluído, você deverá ter esses arquivos prontos para uso quando retornar ao console.

   - Seu arquivo de certificado da CA (*root_CA_cert_filename.pem* usado no comando anterior)
   - O certificado de verificação que você criou na etapa anterior (*verification_cert_filename.pem*usado no comando anterior)

Desativar um certificado CA

Quando um certificado de autoridade de certificação (CA) está habilitado para registro automático de certificados de clientes, AWS IoT verifica o certificado CA para garantir que a CA estejaACTIVE. Se o certificado CA forINACTIVE, AWS IoT não permite que o certificado do cliente seja registrado.

Ao definir o certificado da CA como INACTIVE, você impede que os novos certificados de cliente emitidos pela CA sejam registrados automaticamente.

> **Note**
>
> Qualquer certificado de cliente registrado que foi assinado pelo certificado CA comprometido continuará a funcionar até que você revogue explicitamente cada um deles.

Desativar um certificado CA (console)

Para desativar um certificado CA usando o console AWS IoT

1. Faça login no Console de gerenciamento da AWS e abra o AWS IoT console.
2. No painel de navegação esquerdo, escolha Seguro, escolha CAs.
3. Na lista de autoridades de certificação, localize a que pretende desativar e selecione o ícone de reticências para abrir o menu de opções.
4. No menu de opções, selecione Desativar.

A autoridade de certificação deve aparecer como Inativa na lista.

> **Note**
>
> O AWS IoT console não fornece uma maneira de listar os certificados que foram assinados pela CA que você desativou. Para obter uma opção da AWS CLI para listar esses certificados, consulte Desativar um certificado CA (CLI).

Desativar um certificado CA (CLI)

AWS CLI Fornece o update-ca-certificatecomando para desativar um certificado CA.

```
aws iot update-ca-certificate \
```

```
    --certificate-id certificateId \
    --new-status INACTIVE
```

Use o comando list-certificates-by-ca para obter uma lista de todos os certificados de cliente registrados que foram assinados pela CA especificada. Para cada certificado de cliente assinado pelo certificado CA especificado, use o comando update-certificate para revogar o certificado de cliente a fim de impedir que ele seja usado.

Use o comando describe-ca-certificate para ver o status do certificado CA.

Criar um certificado de cliente usando o certificado CA

É possível usar sua própria autoridade de certificação (CA) para criar certificados de cliente. O certificado do cliente deve ser registrado AWS IoT antes do uso. Para obter informações sobre as opções de registro para os certificados de cliente, consulte Registrar um certificado de cliente.

Criar um certificado de cliente (CLI)

> **Note**
>
> Você não pode realizar esse procedimento no AWS IoT console.

Para criar um certificado de cliente usando o AWS CLI

1. Gere um par de chaves.

   ```
   openssl genrsa -out device_cert_key_filename.key 2048
   ```

2. Crie uma CSR para o certificado de cliente.

   ```
   openssl req -new \
       -key device_cert_key_filename.key \
       -out device_cert_csr_filename.csr
   ```

   Você será solicitado a fornecer algumas informações, conforme mostrado aqui.

   ```
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   ```

```
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
    State or Province Name (full name) []:
    Locality Name (for example, city) []:
    Organization Name (for example, company) []:
    Organizational Unit Name (for example, section) []:
    Common Name (e.g. server FQDN or YOUR name) []:
    Email Address []:

    Please enter the following 'extra' attributes
    to be sent with your certificate request
    A challenge password []:
    An optional company name []:
```

3. Crie um certificado de cliente usando a CSR.

```
openssl x509 -req \
    -in device_cert_csr_filename.csr \
    -CA root_CA_cert_filename.pem \
    -CAkey root_CA_key_filename.key \
    -CAcreateserial \
    -out device_cert_filename.pem \
    -days 500 -sha256
```

Neste momento, o certificado do cliente foi criado, mas ainda não foi registrado no AWS IoT. Para obter informações sobre como e quando registrar o certificado de cliente, consulte Registrar um certificado de cliente.

Registrar um certificado de cliente

Os certificados do cliente devem ser registrados AWS IoT para permitir a comunicação entre o cliente AWS IoT e. Você pode registrar cada certificado de cliente manualmente ou configurar os certificados de cliente para se registrarem automaticamente quando o cliente se conectar AWS IoT pela primeira vez.

Se você quiser que os clientes e os dispositivos registrem seus certificados de cliente quando se conectarem pela primeira vez, deverá Registrar o certificado CA usado para assinar o certificado de cliente no AWS IoT nas regiões nas quais deseja usá-lo. O Amazon Root CA é registrado automaticamente no AWS IoT.

Os certificados do cliente podem ser Contas da AWS compartilhados por regiões. Os procedimentos desses tópicos devem ser executados em cada conta e região em que você deseja usar o certificado de cliente. O registro de um certificado de cliente em uma conta ou região não é automaticamente reconhecido por outra.

> **Note**
>
> Os clientes que usam o protocolo Transport Layer Security (TLS) para se conectar ao AWS IoT devem oferecer suporte à Server Name Indication (SNI) extension para o TLS. Para obter mais informações, consulte Segurança de transporte em AWS IoT Core.

Tópicos

- Registrar um certificado de cliente manualmente
- Registre um certificado de cliente quando o cliente se conecta ao AWS IoT just-in-time registro (JITR)

Registrar um certificado de cliente manualmente

Você pode registrar um certificado de cliente manualmente usando o AWS IoT console AWS CLI e.

O procedimento de registro a ser usado depende se o certificado será compartilhado por Conta da AWS s e regiões. O registro de um certificado de cliente em uma conta ou região não é automaticamente reconhecido por outra.

Os procedimentos deste tópico devem ser executados em cada conta e região em que você deseja usar o certificado de cliente. Os certificados do cliente podem ser compartilhados por Conta da AWS s e regiões.

Registrar um certificado de cliente assinado por uma CA registrada (console)

> **Note**
>
> Antes de executar esse procedimento, verifique se você tem o arquivo.pem do certificado do cliente e se o certificado do cliente foi assinado por uma CA na qual você se registrou. AWS IoT

Para registrar um certificado existente AWS IoT usando o console

1. Faça login no AWS Management Console e abra o AWS IoT console.
2. No painel de navegação, na seção Gerenciar, selecione Segurança e, em seguida, Certificados.
3. Na página Certificados, na caixa de diálogo Certificados, selecione Adicionar certificado e, em seguida, Registrar certificados.
4. Na página Registrar certificado na caixa de diálogo Certificados a serem carregados, faça o seguinte:

   - Escolha a CA registrada no AWS IoT.
   - Em Escolher certificado de CA, selecione sua Autoridade de certificação.
     - Selecione Registrar uma nova CA para registrar uma nova autoridade de certificação que não esteja registrada no AWS IoT.
     - Deixe Escolher certificado de CA em branco se a autoridade de certificação raiz da Amazon for sua autoridade de certificação.
   - Selecione até 10 certificados para carregar e se registrar AWS IoT.
     - Use os arquivos de certificado que você criou em Crie certificados de AWS IoT cliente e Criar um certificado de cliente usando o certificado CA.
   - Selecione Ativar ou Desativar. Se você selecionar Desativar, Ativar ou desativar um certificado de cliente explica como ativar o certificado após o seu registro.
   - Escolha Registrar.

Na página Certificados na caixa de diálogo Certificados, seus certificados registrados agora serão exibidos.

Registrar um certificado de cliente assinado por uma CA não registrada (console)

> **Note**
>
> Antes de executar esse procedimento, verifique se você tem o arquivo .pem do certificado de cliente.

Para registrar um certificado existente AWS IoT usando o console

1. Faça login no AWS Management Console e abra o AWS IoT console.

2. No painel de navegação à esquerda, selecione Proteger, Certificados e Criar.
3. Em Criar um certificado, localize a entrada Usar meu certificado e selecione Começar.
4. Em Selecionar uma CA, selecione Próximo.
5. Em Registrar certificados de dispositivos existentes, escolha Selecionar certificados e escolha até 10 arquivos de certificado para registrar.
6. Depois de fechar a caixa de diálogo do arquivo, escolha se deseja ativar ou revogar os certificados de cliente ao registrá-los.

   Se você não ativar um certificado quando ele for registrado, Ativar um certificado de cliente (console) descreve como ativá-lo posteriormente.

   Se um certificado for revogado quando for registrado, ele não poderá ser ativado posteriormente.

   Depois de selecionar os arquivos de certificado a serem registrados e escolher as ações a serem executadas após o registro, selecione Registrar certificados.

Os certificados de cliente registrados com êxito aparecem na lista de certificados.

Registrar um certificado de cliente assinado por uma CA registrada (CLI)

> **Note**
>
> Antes de executar esse procedimento, verifique se você tem o arquivo .pem da autoridade de certificação (CA) e o arquivo .pem do certificado de cliente. O certificado do cliente deve ser assinado por uma autoridade de certificação (CA) na qual você tenha se registrado AWS IoT.

Use o comando register-certificate para registrar, mas não ativar, um certificado de cliente.

```
aws iot register-certificate \
    --certificate-pem file://device_cert_filename.pem \
    --ca-certificate-pem file://ca_cert_filename.pem
```

O certificado do cliente está registrado com AWS IoT, mas ainda não está ativo. Consulte Ativar um certificado de cliente (CLI) para acessar informações sobre como ativá-lo posteriormente.

Também é possível ativar o certificado de cliente ao registrá-lo usando esse comando.

```
aws iot register-certificate \
```

```
    --set-as-active \
    --certificate-pem file://device_cert_filename.pem \
    --ca-certificate-pem file://ca_cert_filename.pem
```

Para obter mais informações sobre como ativar o certificado para que ele possa ser usado para se conectar AWS IoT, consulte Ativar ou desativar um certificado de cliente

Registrar um certificado de cliente assinado por uma CA não registrada (CLI)

> **Note**
>
> Antes de executar esse procedimento, verifique se você tem o arquivo .pem do certificado.

Use o comando register-certificate-without-ca para registrar, mas não ativar, um certificado de cliente.

```
aws iot register-certificate-without-ca \
    --certificate-pem file://device_cert_filename.pem
```

O certificado do cliente está registrado com AWS IoT, mas ainda não está ativo. Consulte Ativar um certificado de cliente (CLI) para acessar informações sobre como ativá-lo posteriormente.

Também é possível ativar o certificado de cliente ao registrá-lo usando esse comando.

```
aws iot register-certificate-without-ca \
    --status ACTIVE \
    --certificate-pem file://device_cert_filename.pem
```

Para obter mais informações sobre como ativar o certificado para que ele possa ser usado para se conectar AWS IoT, consulteAtivar ou desativar um certificado de cliente.

Registre um certificado de cliente quando o cliente se conecta ao AWS IoT just-in-time registro (JITR)

Você pode configurar um certificado CA para permitir que os certificados do cliente que ele assinou sejam registrados AWS IoT automaticamente na primeira vez em que o cliente se conectar AWS IoT.

Para registrar certificados de cliente quando um cliente se conecta AWS IoT pela primeira vez, você deve habilitar o certificado CA para registro automático e configurar a primeira conexão do cliente para fornecer os certificados necessários.

Configurar um certificado CA para oferecer suporte ao registro automático (console)

Para configurar um certificado CA para oferecer suporte ao registro automático de certificados de clientes usando o AWS IoT console

1. Faça login no AWS Management Console e abra o AWS IoT console.
2. No painel de navegação esquerdo, escolha Seguro, escolha CAs.
3. Na lista de autoridades de certificação, localize aquela para a qual deseja ativar o registro automático e abra o menu de opções usando o ícone de reticências.
4. No menu de opções, selecione Habilitar o registro automático.

> **Note**
>
> O status do registro automático não é mostrado na lista de autoridades de certificação. Para ver o status do registro automático de uma autoridade de certificação, é necessário abrir a página Detalhes da autoridade de certificação.

Configurar um certificado CA para oferecer suporte ao registro automático (CLI)

Se você já registrou seu certificado CA com AWS IoT, use o update-ca-certificatecomando para definir `autoRegistrationStatus` o certificado CA como`ENABLE`.

```
aws iot update-ca-certificate \
--certificate-id caCertificateId \
--new-auto-registration-status ENABLE
```

Se deseja habilitar `autoRegistrationStatus` quando registrar o certificado CA, use o comando register-ca-certificate.

```
aws iot register-ca-certificate \
--allow-auto-registration  \
--ca-certificate file://root_CA_cert_filename.pem \
--verification-cert file://verification_cert_filename.pem
```

Use o comando describe-ca-certificate para ver o status do certificado CA.

Configurar a primeira conexão feita por um cliente para registro automático

Quando um cliente tenta se conectar AWS IoT pela primeira vez, o certificado do cliente assinado pelo seu certificado CA deve estar presente no cliente durante o handshake do Transport Layer Security (TLS).

Quando o cliente se conectar AWS IoT, use o certificado de cliente que você criou em Criar certificados de AWS IoT cliente ou Criar seus próprios certificados de cliente. AWS IoT reconhece o certificado CA como um certificado CA registrado, registra o certificado do cliente e define seu status como. `PENDING_ACTIVATION` Isso indica que o certificado de cliente foi registrado automaticamente e está aguardando ativação. O estado do certificado de cliente deve ser `ACTIVE` para que ele possa ser usado para se conectar ao AWS IoT. Consulte Ativar ou desativar um certificado de cliente para ver mais informações sobre como ativar um certificado de cliente.

> **Note**
>
> Você pode provisionar dispositivos usando o recurso de AWS IoT Core just-in-time registro (JITR) sem precisar enviar toda a cadeia de confiança na primeira conexão dos dispositivos. AWS IoT Core A apresentação do certificado da CA é opcional, mas é necessário que o dispositivo envie a extensão Server Name Indication (SNI) ao se conectar.

Quando registra AWS IoT automaticamente um certificado ou quando um cliente apresenta um certificado no `PENDING_ACTIVATION` status, AWS IoT publica uma mensagem no seguinte tópico do MQTT:

`$aws/events/certificates/registered/`*`caCertificateId`*

Em que *`caCertificateId`* é o ID do certificado CA que emitiu o certificado de cliente.

A mensagem publicada para este tópico tem a seguinte estrutura:

```
{
        "certificateId": "certificateId",
        "caCertificateId": "caCertificateId",
        "timestamp": timestamp,
        "certificateStatus": "PENDING_ACTIVATION",
        "awsAccountId": "awsAccountId",
        "certificateRegistrationTimestamp": "certificateRegistrationTimestamp"
}
```

Você pode criar uma regra que ouça esse tópico e execute algumas ações. Recomendamos que você crie uma regra do Lambda que verifique se o certificado de cliente não está em uma lista de revogação de certificados (CRL), ative o certificado e crie e anexe uma política para o certificado. A política determina quais recursos o cliente pode acessar. Se a política que você está criando exigir o ID do cliente dos dispositivos conectados, você poderá usar a função clientid() da regra para recuperar o ID do cliente. Este é um exemplo de definição de regra:

```
SELECT *,
   clientid() as clientid
from $aws/events/certificates/registered/caCertificateId
```

Neste exemplo, a regra assina o tópico de JITR `$aws/events/certificates/registered/`*`caCertificateID`* e usa a função clientid() para recuperar o ID do cliente. Depois, a regra anexa o ID do cliente à carga útil de JITR. Para ter mais informações sobre a função clientid() da regra, consulte clientid().

Para obter mais informações sobre como criar uma regra Lambda que escuta o `$aws/events/certificates/registered/`*`caCertificateID`* tópico e executa essas ações, consulte just-in-time registro de certificados de cliente em. AWS IoT

Se ocorrer algum erro ou exceção durante o registro automático dos certificados do cliente, AWS IoT envia eventos ou mensagens para seus CloudWatch registros no Logs. Para obter mais informações sobre como configurar os registros da sua conta, consulte a CloudWatch documentação da Amazon.

Gerenciar certificados do cliente

AWS IoT fornece recursos para você gerenciar certificados de clientes.

Neste tópico:

- Ativar ou desativar um certificado de cliente
- Anexar um objeto ou política a um certificado de cliente
- Revogar um certificado de cliente
- Transferir um certificado para outra conta

Ativar ou desativar um certificado de cliente

AWS IoT verifica se um certificado de cliente está ativo ao autenticar uma conexão.

É possível criar e registrar certificados de cliente sem ativá-los, para que eles não possam ser usados até que você queira usá-los. Também é possível desativar certificados de cliente ativos para desabilitá-los temporariamente. Por fim, é possível revogar certificados de cliente para impedir qualquer uso futuro deles.

Ativar um certificado de cliente (console)

Para ativar um certificado de cliente usando o AWS IoT console

1. Faça login no AWS Management Console e abra o AWS IoT console.
2. No painel de navegação à esquerda, selecione Proteger e Certificados.
3. Na lista de certificados, localize o certificado que você deseja ativar e abra o menu de opções usando o ícone de reticências.
4. No menu de opções, selecione Ativar.

O certificado deve ser exibido como Ativo na lista de certificados.

Desativar um certificado de cliente (console)

Para desativar um certificado de cliente usando o console AWS IoT

1. Faça login no AWS Management Console e abra o AWS IoT console.
2. No painel de navegação à esquerda, selecione Proteger e Certificados.
3. Na lista de certificados, localize o certificado que você deseja desativar e abra o menu de opções usando o ícone de reticências.
4. No menu de opções, selecione Desativar.

O certificado deve ser exibido como Inativo na lista de certificados.

Ativar um certificado de cliente (CLI)

O AWS CLI fornece o update-certificatecomando para ativar um certificado.

```
aws iot update-certificate \
    --certificate-id certificateId \
    --new-status ACTIVE
```

Se o comando tiver sido bem-sucedido, o status do certificado será `ACTIVE`. Execute describe-certificate para ver o status do certificado.

```
aws iot describe-certificate \
    --certificate-id certificateId
```

Desativar um certificado de cliente (CLI)

O AWS CLI fornece o update-certificatecomando para desativar um certificado.

```
aws iot update-certificate \
    --certificate-id certificateId \
    --new-status INACTIVE
```

Se o comando tiver sido bem-sucedido, o status do certificado será `INACTIVE`. Execute describe-certificate para ver o status do certificado.

```
aws iot describe-certificate \
    --certificate-id certificateId
```

Anexar um objeto ou política a um certificado de cliente

Quando você cria e registra um certificado separado de AWS IoT algo, ele não terá nenhuma política que autorize nenhuma AWS IoT operação, nem será associado a nenhum AWS IoT objeto. Esta seção descreve como adicionar esses relacionamentos a um certificado registrado.

> **Important**
>
> Para concluir esses procedimentos, você já deve ter criado o objeto ou a política que quer anexar ao certificado.

O certificado autentica um dispositivo AWS IoT para que ele possa se conectar. Anexar o certificado a um recurso de objeto estabelece a relação entre o dispositivo (por meio do certificado) e o recurso de objeto. Para autorizar o dispositivo a realizar AWS IoT ações, como permitir que o dispositivo se conecte e publique mensagens, uma política apropriada deve ser anexada ao certificado do dispositivo.

Anexar um objeto a um certificado de cliente (console)

Será necessário o nome do objeto para concluir este procedimento.

Como anexar um objeto a um certificado registrado

1. Faça login no AWS Management Console e abra o AWS IoT console.
2. No painel de navegação à esquerda, selecione Proteger e Certificados.
3. Na lista de certificados, localize o certificado ao qual você deseja anexar uma política, abra o menu de opções do certificado escolhendo o ícone de reticências e escolha Anexar objeto.
4. No menu pop-up, localize o nome do objeto que você deseja anexar ao certificado, marque sua caixa de seleção e escolha Anexar.

O objeto agora deve aparecer na lista de objetos na página de detalhes do certificado.

Anexar uma política a um certificado de cliente (console)

Será necessário o nome do objeto de política para concluir este procedimento.

Como anexar um objeto de política a um certificado registrado

1. Faça login no AWS Management Console e abra o AWS IoT console.
2. No painel de navegação à esquerda, selecione Proteger e Certificados.
3. Na lista de certificados, localize o certificado ao qual você deseja anexar uma política, abra o menu de opções do certificado escolhendo o ícone de reticências e escolha Anexar política.
4. No menu pop-up, localize o nome da política que você deseja anexar ao certificado, marque sua caixa de seleção e escolha Anexar.

O objeto de política agora deverá ser exibido na lista de políticas na página de detalhes do certificado.

Anexar um objeto a um certificado de cliente (CLI)

O AWS CLI fornece o attach-thing-principalcomando para anexar um objeto a um certificado.

```
aws iot attach-thing-principal \
    --principal certificateArn \
    --thing-name thingName
```

Anexar uma política a um certificado de cliente (CLI)

O AWS CLI fornece o attach-policycomando para anexar um objeto de política a um certificado.

```
aws iot attach-policy \
    --target certificateArn \
    --policy-name policyName
```

Revogar um certificado de cliente

Se detectar atividade suspeita em um certificado de cliente registrado, você poderá revogá-lo para que ele não possa ser usado novamente.

> **Note**
>
> Depois que um certificado é revogado, seu status não pode ser alterado. Ou seja, o status do certificado não pode ser alterado para `Active` nenhum outro status.

Revogar um certificado de cliente (console)

Para revogar um certificado de cliente usando o console AWS IoT

1. Faça login no AWS Management Console e abra o AWS IoT console.
2. No painel de navegação à esquerda, selecione Proteger e Certificados.
3. Na lista de certificados, localize o certificado que você deseja revogar e abra o menu de opções usando o ícone de reticências.
4. No menu de opções, selecione Revogar.

Se o certificado tiver sido revogado com êxito, ele será exibido como Revogado na lista de certificados.

Revogar um certificado de cliente (CLI)

O AWS CLI fornece o update-certificatecomando para revogar um certificado.

```
aws iot update-certificate \
    --certificate-id certificateId \
    --new-status REVOKED
```

Se o comando tiver sido bem-sucedido, o status do certificado será `REVOKED`. Execute describe-certificate para ver o status do certificado.

```
aws iot describe-certificate \
    --certificate-id certificateId
```

Transferir um certificado para outra conta

Os certificados X.509 que pertencem a um Conta da AWS podem ser transferidos para outro. Conta da AWS

Para transferir um certificado X.509 de um para outro Conta da AWS

1. the section called “Iniciar uma transferência de certificado”

   O certificado deve ser desativado e separado de todas as políticas e itens antes de a transferência ser iniciada.
2. the section called “Aceitar ou rejeitar uma transferência de certificado”

   A conta receptora deve aceitar ou rejeitar explicitamente o certificado transferido. Depois que a conta receptora aceitar o certificado, o certificado deverá ser ativado antes do uso.
3. the section called “Cancelar uma transferência de certificado”

   A conta de origem poderá cancelar uma transferência, se o certificado não tiver sido aceito.

Iniciar uma transferência de certificado

Você pode começar a transferir um certificado para outro Conta da AWS usando o AWS IoT console ou AWS CLI o.

Iniciar uma transferência de certificado (console)

Para concluir esse procedimento, você precisará da ID do certificado que deseja transferir.

Execute esse procedimento na conta com o certificado a ser transferido.

Para começar a transferir um certificado para outro Conta da AWS

1. Faça login no AWS Management Console e abra o AWS IoT console.
2. No painel de navegação à esquerda, selecione Proteger e Certificados.

Escolha o certificado com status Ativo ou Inativo que você deseja transferir e abra sua página de detalhes.

3. Na página Detalhes do certificado, no menu Ações, se a opção Desativar estiver disponível, escolha a opção Desativar para desativar o certificado.
4. Na página Detalhes do certificado, no menu à esquerda, escolha Políticas.
5. Na página Políticas do certificado, se houver alguma política anexada ao certificado, separe cada uma abrindo o menu de opções da política e escolhendo Desanexar.

   O certificado não deve possuir políticas anexadas antes de prosseguir.
6. Na página Políticas do certificado, no menu à esquerda, escolha Objetos.
7. Na página Objetos do certificado, se houver algum objeto anexado ao certificado, separe cada uma abrindo o menu de opções do objeto e escolhendo Desanexar.

   O certificado não deve possuir objetos anexados antes de prosseguir.
8. Na página Objetos do certificado, no menu à esquerda, escolha Detalhes.
9. Na página Detalhes do certificado, no menu Ações, escolha Iniciar transferência para abrir a caixa de diálogo Iniciar transferência.
10. Na caixa de diálogo Iniciar transferência, insira o Conta da AWS número da conta para receber o certificado e uma mensagem curta opcional.
11. Escolha Iniciar transferência para transferir o certificado.

O console deve exibir uma mensagem que indica o êxito ou a falha da transferência. Se a transferência foi iniciada, o status do certificado será atualizado para Transferido.

Iniciar uma transferência de certificado (CLI)

Para concluir esse procedimento, você precisará *certificateArn* do *certificateId* e do certificado que deseja transferir.

Execute esse procedimento na conta com o certificado a ser transferido.

Para começar a transferir um certificado para outra AWS conta

1. Use o comando update-certificate para desativar o certificado.

```
aws iot update-certificate --certificate-id certificateId --new-status INACTIVE
```

2. Desanexe todas as políticas.

   1. Use o comando list-attached-policies para listar as políticas anexadas ao certificado.

      ```
      aws iot list-attached-policies --target certificateArn
      ```

   2. Use o comando detach-policy para desanexar cada política anexada.

      ```
      aws iot detach-policy --target certificateArn --policy-name policy-name
      ```

3. Desanexe todas as objetos.

   1. Use o comando list-principal-things para listar as objetos anexados ao certificado.

      ```
      aws iot list-principal-things --principal certificateArn
      ```

   2. Use o comando detach-thing-principal para desanexar cada objeto anexado.

      ```
      aws iot detach-thing-principal --principal certificateArn --thing-name thing-name
      ```

4. Use o comando transfer-certificate para iniciar a transferência do certificado.

   ```
   aws iot transfer-certificate --certificate-id certificateId --target-aws-
   account account-id
   ```

Aceitar ou rejeitar uma transferência de certificado

Você pode aceitar ou rejeitar um certificado transferido para você Conta da AWS de outra pessoa Conta da AWS usando o AWS IoT console ou AWS CLI o.

Aceitar ou rejeitar uma transferência de certificado (console)

Para concluir esse procedimento, você precisará da ID do certificado que foi transferido para sua conta.

Execute esse procedimento na conta que recebeu o certificado transferido.

Para aceitar ou rejeitar um certificado que foi transferido para o seu Conta da AWS

1. Faça login no AWS Management Console e abra o AWS IoT console.
2. No painel de navegação à esquerda, selecione Proteger e Certificados.

Escolha o certificado com o status Transferência pendente que você deseja aceitar ou rejeitar e abra sua página de detalhes.

3. Na página Detalhes do certificado, no menu Ações,

   - Para aceitar o certificado, escolha Aceitar transferência.
   - Para não aceitar o certificado, escolha Rejeitar transferência.

Aceitar ou rejeitar uma transferência de certificado (CLI)

Para concluir esse procedimento, você precisará *certificateId* da transferência do certificado que deseja aceitar ou rejeitar.

Execute esse procedimento na conta que recebeu o certificado transferido.

Para aceitar ou rejeitar um certificado que foi transferido para o seu Conta da AWS

1. Use o comando accept-certificate-transfer para aceitar o certificado.

```
aws iot accept-certificate-transfer --certificate-id certificateId
```

2. Use o comando reject-certificate-transfer para rejeitar o certificado.

```
aws iot reject-certificate-transfer --certificate-id certificateId
```

Cancelar uma transferência de certificado

Você pode cancelar uma transferência de certificado antes que ela seja aceita usando o console do AWS IoT ou a AWS CLI.

Cancelar uma transferência de certificado (console)

Para concluir esse procedimento, você precisará da ID da transferência de certificado que deseja cancelar.

Execute esse procedimento na conta que iniciou a transferência de certificado.

Para cancelar uma transferência de certificado

1. Faça login no AWS Management Console e abra o AWS IoT console.

2. No painel de navegação à esquerda, selecione Proteger e Certificados.

   Escolha o certificado com status Transferido cuja transferência você deseja cancelar e abra seu menu de opções.

3. No menu de opções do certificado, escolha a opção Revogar transferência para cancelar a transferência do certificado.

> **Important**
>
> Tome cuidado para não confundir a opção Revogar transferência com a opção Revogar. A opção Revogar transferência cancela a transferência do certificado, enquanto a opção Revogar torna o certificado irreversivelmente inutilizável pelo AWS IoT.

Cancelar uma transferência de certificado (CLI)

Para concluir esse procedimento, você precisará *certificateId* da transferência do certificado que deseja cancelar.

Execute esse procedimento na conta que iniciou a transferência de certificado.

Use o comando cancel-certificate-transfer para cancelar a transferência do certificado.

```
aws iot cancel-certificate-transfer --certificate-id certificateId
```

Validação personalizada de certificados de cliente

AWS IoT Core oferece suporte à validação personalizada de certificados de cliente para certificados de cliente X.509, o que aprimora o gerenciamento da autenticação do cliente. Esse método de validação de certificado também é conhecido como verificações de certificado de pré-autenticação, nas quais você avalia os certificados do cliente com base em seus próprios critérios (definidos em uma função do Lambda) e revoga os certificados do cliente ou o certificado da autoridade certificadora (CA) de assinatura dos certificados para impedir que os clientes se conectem ao AWS IoT Core. Por exemplo, você pode criar suas próprias verificações de revogação de certificados para validar o status dos certificados em relação às autoridades de validação que oferecem suporte a endpoints do Protocolo de Status de Certificados Online (OCSP) ou de Listas de Revogação de Certificados (CRL) e impedir conexões para clientes com certificados revogados. Os critérios usados para avaliar os certificados do cliente são definidos em uma função do Lambda (também conhecida como Lambda pré-autenticação). Você deve usar os endpoints definidos nas configurações de

domínio e o tipo de autenticação deve ser certificado X.509. Além disso, os clientes devem fornecer a extensão de Indicação de Nome do Servidor (SNI) ao se conectarem a. AWS IoT Core

> **Note**
>
> Esse recurso não é suportado nas AWS GovCloud (US) regiões.

O processo de realizar a validação personalizada de certificados de cliente envolve as etapas a seguir.

- Etapa 1: registre seus certificados de cliente X.509 com AWS IoT Core
- Etapa 2: Criar uma função do Lambda
- Etapa 3: Autorizar AWS IoT a invocação da função Lambda
- Etapa 4: definir a configuração de autenticação para um domínio

Etapa 1: registre seus certificados de cliente X.509 com AWS IoT Core

Se você ainda não fez isso, registre e ative seus certificados de cliente X.509 com. AWS IoT Core Caso contrário, vá para a próxima etapa.

Para registrar e ativar seus certificados de cliente com AWS IoT Core, siga as etapas:

1. Se você criar certificados de cliente diretamente com AWS IoT. Esses certificados de cliente serão registrados automaticamente com AWS IoT Core.
2. Se você criar seus próprios certificados de cliente, siga estas instruções para registrá-los AWS IoT Core.
3. Para ativar seus certificados de cliente, siga estas instruções.

Etapa 2: Criar uma função do Lambda

Você precisa criar uma função do Lambda que realizará a verificação do certificado e será chamada para cada tentativa de conexão do cliente para o endpoint configurado. Ao criar essa função do Lambda, siga as orientações gerais para Criar sua primeira função do Lambda. Além disso, garanta que a função do Lambda cumpra os formatos de solicitação e resposta esperados da seguinte forma:

Exemplo de evento da função do Lambda

```
{
```

```
 "connectionMetadata": {
  "id": "string"
 },
 "principalId": "string",
 "serverName": "string",
 "clientCertificateChain": [
  "string",
  "string"
 ]
}
```

`connectionMetadata`

Metadados ou informações adicionais relacionadas à conexão do cliente com o AWS IoT Core.

`principalId`

O identificador de entidade principal associado ao cliente na conexão TLS.

`serverName`

A string de nome de host Indicação de nome de servidor (SNI). O AWS IoT Core requer que os dispositivos enviem a extensão SNI para o protocolo Transport Layer Security (TLS) e forneçam o endereço completo do endpoint no campo `host_name`.

`clientCertificateChain`

A matriz de strings que representa a cadeia de certificados X.509 do cliente.

Exemplo de resposta da função do Lambda

```
{
 "isAuthenticated": "boolean"
}
```

`isAuthenticated`

Um valor booliano que indica se a solicitação foi autenticada.

> **Note**
>
> Na resposta do Lambda, `isAuthenticated` precisa ser `true` para prosseguir com a autenticação e a autorização adicionais. Caso contrário, o certificado do cliente de IoT pode

ser desativado e a autenticação personalizada com certificados de cliente X.509 pode ser bloqueada para autenticação e autorização adicionais.

Etapa 3: Autorizar AWS IoT a invocação da função Lambda

Depois de criar a função Lambda, você deve conceder permissão AWS IoT para invocá-la usando o comando da CLI add-permission. Observe que essa função do Lambda será invocada para cada tentativa de conexão ao seu endpoint configurado. Para obter mais informações, consulte Autorização AWS IoT para invocar sua função Lambda.

Etapa 4: definir a configuração de autenticação para um domínio

A seção a seguir descreve como definir a configuração de autenticação para um domínio personalizado usando a AWS CLI.

Definir a configuração do certificado do cliente para um domínio (CLI)

Se você não tiver uma configuração de domínio, use o comando da CLI create-domain-configuration para criar uma. Se você já tiver uma configuração de domínio, use o comando da CLI update-domain-configuration para atualizar a configuração do certificado do cliente para um domínio. Você deve adicionar o ARN da função do Lambda que criou na etapa anterior.

```
aws iot create-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --authentication-type AWS_X509|CUSTOM_AUTH_X509 \
    --application-protocol SECURE_MQTT|HTTPS \
    --client-certificate-config 'clientCertificateCallbackArn":"arn:aws:lambda:us-
east-2:123456789012:function:my-function:1"}'
```

```
aws iot update-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --authentication-type AWS_X509|CUSTOM_AUTH_X509 \
    --application-protocol SECURE_MQTT|HTTPS \
    --client-certificate-config '{"clientCertificateCallbackArn":"arn:aws:lambda:us-
east-2:123456789012:function:my-function:1"}'
```

`domain-configuration-name`

O nome da configuração do domínio.

`authentication-type`

O tipo de autenticação da configuração do domínio. Para obter mais informações, consulte Escolher um tipo de autenticação.

`application-protocol`

O protocolo de aplicativo que os dispositivos usam para se comunicar com o AWS IoT Core. Para obter mais informações, consulte Escolhendo um protocolo de aplicativo.

`client-certificate-config`

Um objeto que especifica a configuração de autenticação do cliente para um domínio.

`clientCertificateCallbackArn`

O Amazon Resource Name (ARN) da função Lambda que é AWS IoT invocada na camada TLS quando uma nova conexão está sendo estabelecida. Para personalizar a autenticação do cliente para realizar a validação personalizada de certificados de cliente, você deve adicionar o ARN da função do Lambda que criou na etapa anterior.

Para obter mais informações, consulte CreateDomainConfiguratione a UpdateDomainConfigurationpartir da Referência AWS IoT da API. Para obter mais informações sobre configurações de domínio, consulte Configurações de domínio.

Usuários, grupos e funções do IAM

Os usuários, grupos e perfis do IAM são os mecanismos padrão para o gerenciamento de identidades e de autenticações na AWS. Você pode usá-los para se conectar a interfaces AWS IoT HTTP usando o AWS SDK e. AWS CLI

As funções do IAM também AWS IoT permitem acessar outros AWS recursos em sua conta em seu nome. Por exemplo, se você quiser que um dispositivo publique seu estado em uma tabela do DynamoDB, as funções do IAM AWS IoT permitem interagir com o Amazon DynamoDB. Para obter mais informações, consulte Perfis do IAM.

Para conexões de agente de mensagens via HTTP, AWS IoT autentica usuários, grupos e funções usando o processo de assinatura Signature versão 4. Para obter mais informações, consulte Assinatura de solicitações de AWS API.

Ao usar o AWS Signature Version 4 com AWS IoT, os clientes devem oferecer suporte ao seguinte em sua implementação de TLS:

- TLS 1.2
- Validação de assinatura do certificado SHA-256 RSA
- Um dos pacotes de criptografia na seção de suporte a pacotes de criptografia do TLS

Para mais informações, consulte Gerenciamento de identidade e acesso para AWS IoT.

Identidades do Amazon Cognito

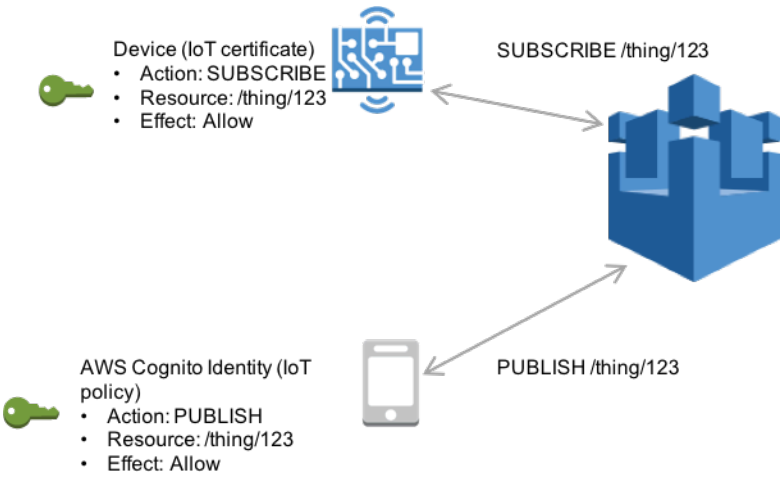
O Amazon Cognito Identity permite que você crie AWS credenciais temporárias com privilégios limitados para uso em aplicativos móveis e web. Ao usar a identidade do Amazon Cognito, são criados bancos de identidades que criam identidades exclusivas para seus usuários e os autentica com provedores de identidade como Login with Amazon, Facebook e Google. Também é possível usar identidades do Amazon Cognito com suas próprias identidades autenticadas pelo desenvolvedor. Para obter mais informações, consulte Identidade do Amazon Cognito.

Para usar a identidade do Amazon Cognito, você define um banco de identidades do Amazon Cognito associado a um perfil do IAM. A função do IAM está associada a uma política do IAM que concede às identidades do seu grupo de identidades permissão para acessar AWS recursos como AWS serviços de chamadas.

A identidade do Amazon Cognito cria identidades não autenticadas e autenticadas. As identidades não autenticadas são usadas para usuários convidados em um aplicativo móvel ou web que desejam usar o aplicativo sem fazer login. Os usuários não autenticados recebem somente as permissões especificadas na política do IAM associada ao banco de identidades.

Ao usar identidades autenticadas, além da política do IAM anexada ao grupo de identidades, você deve anexar uma AWS IoT política a uma identidade do Amazon Cognito. Para anexar uma AWS IoT política, use a AttachPolicyAPI e conceda permissões a um usuário individual do seu AWS IoT aplicativo. Você pode usar a AWS IoT política para atribuir permissões refinadas para clientes específicos e seus dispositivos.

Usuários autenticados e não autenticados são tipos de identidade diferentes. Se você não anexar uma AWS IoT política à Identidade do Amazon Cognito, um usuário autenticado falhará na autorização AWS IoT e não terá acesso aos AWS IoT recursos e ações. Para obter mais informações sobre a criação de políticas para identidades do Amazon Cognito, consulte Exemplos de política de publicação/inscrição e Autorização com identidades do Amazon Cognito.



Autenticação e autorização personalizadas

AWS IoT Core permite definir autorizadores personalizados para que você possa gerenciar sua própria autenticação e autorização do cliente. Isso é útil quando você precisa usar mecanismos de autenticação diferentes dos que oferecem suporte AWS IoT Core nativo. (Para obter mais informações sobre os mecanismos compatíveis, consulte the section called “Autenticação de cliente”).

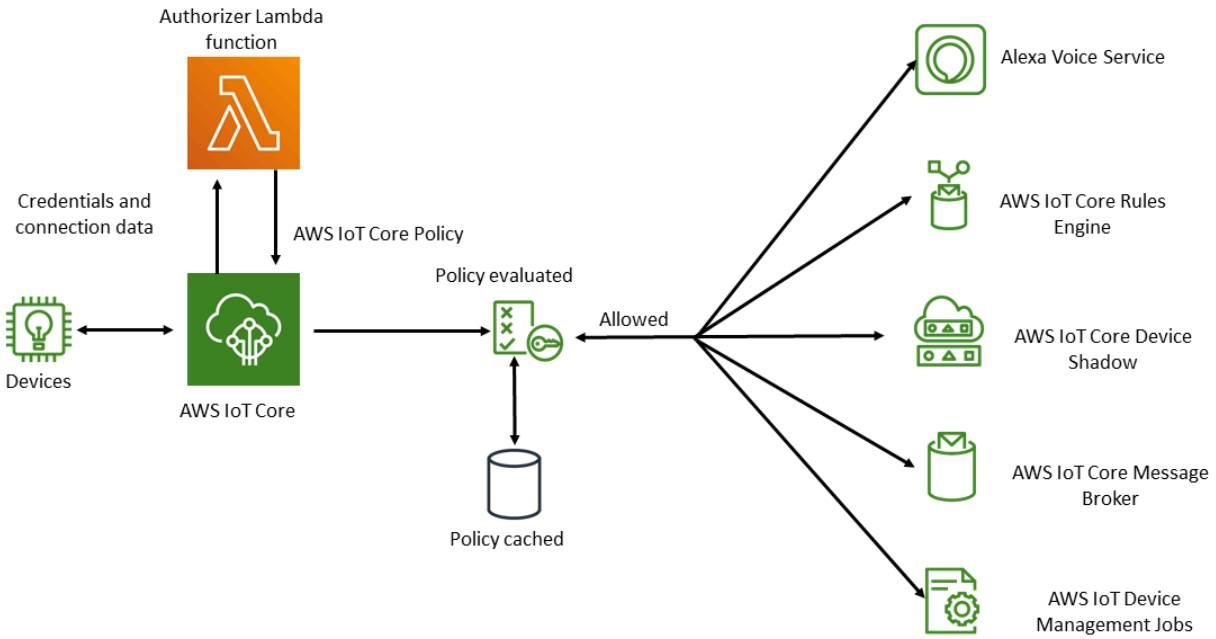
Por exemplo, se você estiver migrando dispositivos existentes no campo para AWS IoT Core e esses dispositivos usarem um token de portador personalizado ou nome de usuário e senha do MQTT para autenticação, você poderá migrá-los AWS IoT Core sem precisar provisionar novas identidades para eles. Você pode usar a autenticação personalizada com qualquer um dos protocolos de comunicação AWS IoT Core compatíveis. Para obter mais informações sobre protocolos compatíveis com o AWS IoT Core , consulte the section called “Protocolos de comunicação do dispositivo”.

Tópicos

- Entender o fluxo de trabalho de autenticação personalizada
- Criação e gerenciamento de autorizadores personalizados (CLI)
- Autenticação personalizada com certificados de cliente X.509
- AWS IoT Core Conectando-se usando autenticação personalizada
- Solução de problemas dos autorizadores

Entender o fluxo de trabalho de autenticação personalizada

A autenticação personalizada permite que você defina como autenticar e autorizar clientes usando recursos do autorizador. Cada autorizador contém uma referência a uma função do Lambda gerenciada pelo cliente, uma chave pública opcional para validar as credenciais do dispositivo e informações adicionais de configuração. O diagrama a seguir ilustra o fluxo de trabalho de autorização para autenticação personalizada no AWS IoT Core.



AWS IoT Core fluxo de trabalho personalizado de autenticação e autorização

A lista a seguir explica cada etapa do fluxo de trabalho de autenticação e autorização personalizadas.

1. Um dispositivo se conecta ao endpoint de AWS IoT Core dados do cliente usando um dos compatíveisthe section called “Protocolos de comunicação do dispositivo”. O dispositivo passa as credenciais nos campos de cabeçalho ou nos parâmetros de consulta da solicitação (para HTTP Publish ou MQTT sobre WebSockets protocolos) ou no campo de nome de usuário e senha da mensagem MQTT CONNECT (para os protocolos MQTT e MQTT over). WebSockets
2. AWS IoT Core verifica uma das duas condições:
   - A solicitação recebida especifica um autorizador.

- O endpoint de AWS IoT Core dados que recebe a solicitação tem um autorizador padrão configurado para ele.

Se AWS IoT Core encontrar um autorizador de qualquer uma dessas formas, AWS IoT Core acionará a função Lambda associada ao autorizador.

3. (Opcional) Se você habilitou a assinatura de token, AWS IoT Core valida a assinatura da solicitação usando a chave pública armazenada no autorizador antes de acionar a função Lambda. Se a validação falhar, o AWS IoT Core interromperá a solicitação sem invocar a função do Lambda.
4. A função do Lambda recebe as credenciais e os metadados de conexão na solicitação e toma uma decisão de autenticação.
5. A função Lambda retorna os resultados da decisão de autenticação e um documento de AWS IoT Core política que especifica quais ações são permitidas na conexão. A função Lambda também retorna informações que especificam com que frequência AWS IoT Core revalida as credenciais na solicitação invocando a função Lambda.
6. AWS IoT Core avalia a atividade na conexão em relação à política que recebeu da função Lambda.
7. Depois que a conexão for estabelecida e o Lambda do seu autorizador personalizado for inicialmente invocado, a próxima invocação poderá ser adiada por até 5 minutos em conexões inativas sem nenhuma operação de MQTT. Depois disso, as invocações subsequentes seguirão o intervalo de atualização no Lambda do seu autorizador personalizado. Essa abordagem pode evitar invocações excessivas que podem exceder o limite de simultaneidade do Lambda do seu. Conta da AWS

Considerações sobre dimensionamento

Uma vez que uma função do Lambda gerencia a autenticação e a autorização do seu autorizador, ela está sujeita aos limites de preços e serviços do Lambda, como a taxa de execução simultânea. Para obter mais informações sobre os preços do Lambda, consulte Preços do Lambda. Você pode gerenciar a carga em sua função do Lambda ajustando os parâmetros `refreshAfterInSeconds` e `disconnectAfterInSeconds` na resposta da função do Lambda. Para obter mais informações sobre o conteúdo da resposta da função do Lambda, consulte the section called “Definição de sua função do Lambda”.

> **Note**
>
> Se você deixar a assinatura ativada, poderá evitar o acionamento excessivo do seu Lambda por clientes não reconhecidos. Considere isso antes de desativar o login no seu autorizador.

> **Note**
>
> O limite de tempo da função do Lambda para o autorizador personalizado é de 5 segundos.

Criação e gerenciamento de autorizadores personalizados (CLI)

AWS IoT Core implementa esquemas personalizados de autenticação e autorização usando autorizadores personalizados. Um autorizador personalizado é um AWS IoT Core recurso que oferece a flexibilidade de definir e implementar as regras e políticas com base em seus requisitos específicos. Para criar um autorizador personalizado com step-by-step instruções, consulte Tutorial: Criando um autorizador personalizado para. AWS IoT Core

Cada autorizador consiste nos seguintes componentes:

- Nome: uma string exclusiva definida pelo usuário que identifica o autorizador.
- ARN da função do Lambda: o nome do recurso da Amazon (ARN) da função do Lambda que implementa a lógica de autorização e autenticação.
- Nome da chave do token: o nome da chave usado para extrair o token dos cabeçalhos HTTP, dos parâmetros de consulta ou do nome de usuário do MQTT CONNECT para realizar a validação da assinatura. Esse valor será necessário se a assinatura estiver ativada em seu autorizador.
- Sinalizador de assinatura desativada (opcional): um valor booleano que especifica se a exigência de assinatura nas credenciais deve ser desativada. Isso é útil para cenários em que assinar as credenciais não faz sentido, como esquemas de autenticação que usam nome de usuário e senha do MQTT. O valor padrão é `false`; portanto, a assinatura é ativada por padrão.
- Chave pública de assinatura de token: a chave pública que o AWS IoT Core usa para validar a assinatura de token. Seu comprimento mínimo é de 2.048 bits. Esse valor será necessário se a assinatura estiver ativada em seu autorizador.

O Lambda cobra pelo número de vezes que sua função do Lambda é executada e pelo tempo necessário para que o código em sua função seja executado. Para obter mais informações sobre

preços do Lambda, consulte Preços do Lambda. Para obter mais informações sobre a criação de funções do Lambda, consulte o Guia do desenvolvedor do Lambda.

> **Note**
>
> Se você deixar a assinatura ativada, poderá evitar o acionamento excessivo do seu Lambda por clientes não reconhecidos. Considere isso antes de desativar o login no seu autorizador.

> **Note**
>
> O limite de tempo da função do Lambda para o autorizador personalizado é de 5 segundos.

Neste capítulo:

- Definição de sua função do Lambda
- Criar um autorizador de capacidade
- Autorizando AWS IoT a invocação da função Lambda
- Testar seus autorizadores
- Gerenciar autorizadores personalizados

Definição de sua função do Lambda

Quando AWS IoT Core invoca seu autorizador, ele aciona o Lambda associado ao autorizador com um evento que contém o seguinte objeto JSON. O objeto JSON de exemplo contém todos os campos possíveis. Quaisquer campos que não sejam relevantes para a solicitação de conexão não estão incluídos.

```
{
    "token" :"aToken",
    "signatureVerified": Boolean, // Indicates whether the device gateway has validated
 the signature.
    "protocols": ["tls", "http", "mqtt"], // Indicates which protocols to expect for
 the request.
    "protocolData": {
        "tls" : {
            "serverName": "serverName" // The server name indication (SNI) host_name
 string.
```

```
        },
        "http": {
            "headers": {
                "#{name}": "#{value}"
            },
            "queryString": "?#{name}=#{value}"
        },
        "mqtt": {
            "username": "myUserName",
            "password": "myPassword", // A base64-encoded string.
            "clientId": "myClientId" // Included in the event only when the device
 sends the value.
        }
    },
    "connectionMetadata": {
        "id": UUID // The connection ID. You can use this for logging.
    },
}
```

A função do Lambda deve usar essas informações para autenticar a conexão de entrada e decidir quais ações são permitidas na conexão. A função deve enviar uma resposta que contenha os seguintes valores.

- `isAuthenticated`: um valor booliano que indica se a solicitação foi autenticada.
- `principalId`: uma sequência alfanumérica que atua como um identificador para o token enviado pela solicitação de autorização personalizada. O valor deve ser uma sequência alfanumérica com pelo menos um e não mais que 128 caracteres e corresponder a este padrão de expressão regular (regex): `([a-zA-Z0-9]){1,128}`. Caracteres especiais que não sejam alfanuméricos não são permitidos para uso com a `principalId` entrada AWS IoT Core. Consulte a documentação de outros AWS serviços se caracteres especiais não alfanuméricos forem permitidos para o. `principalId`
- `policyDocuments`: uma lista de documentos de AWS IoT Core políticas formatados em JSON Para obter mais informações sobre a criação AWS IoT Core de políticas, consulte. the section called “AWS IoT Core políticas” O número máximo de documentos de política é de 10 documentos de política. Cada documento de política pode ter, no máximo, 2.048 caracteres.
- `disconnectAfterInSeconds`: um número inteiro que especifica a duração máxima (em segundos) da conexão com o gateway do AWS IoT Core . O valor mínimo é de 300 segundos e o valor máximo é de 86.400 segundos. O valor padrão é 86.400.

> **Note**
>
> O valor de `disconnectAfterInSeconds` (retornado pela função do Lambda) é definido quando a conexão é estabelecida. Esse valor não poderá ser modificado durante as invocações do Lambda subsequentes para atualização da política.

- `refreshAfterInSeconds`: um número inteiro que especifica o intervalo entre as atualizações da política. Quando esse intervalo passa, o AWS IoT Core invoca a função do Lambda para permitir atualizações de políticas. O valor mínimo é de 300 segundos e o valor máximo é de 86.400 segundos.

O objeto JSON a seguir contém um exemplo de resposta que sua função do Lambda pode enviar.

```
{ "isAuthenticated":true, //A Boolean that determines whether
client can connect. "principalId": "xxxxxxxx",  //A string that
identifies the connection in logs. "disconnectAfterInSeconds": 86400,
  "refreshAfterInSeconds": 300,   "policyDocuments": [       {
 "Version": "2012-10-17",         "Statement": [           {
    "Action": "iot:Publish",              "Effect": "Allow",
      "Resource": "arn:aws:iot:us-east-1:<your_aws_account_id>:topic/
customauthtesting"            }         ]       }     ] }
```

O `policyDocument` valor deve conter um documento AWS IoT Core de política válido. Para obter mais informações sobre AWS IoT Core políticas, consultethe section called “AWS IoT Core políticas”. Em MQTT sobre TLS e MQTT sobre WebSockets conexões, armazena em AWS IoT Core cache essa política para o intervalo especificado no valor do campo. `refreshAfterInSeconds` No caso de conexões HTTP, a função do Lambda é chamada para cada solicitação de autorização, a menos que seu dispositivo esteja usando conexões HTTP persistentes (também chamadas de keep-alive do HTTP ou reutilização de conexão HTTP). Você pode optar por ativar o armazenamento em cache ao configurar o autorizador. Durante esse intervalo, AWS IoT Core autoriza ações em uma conexão estabelecida com essa política em cache sem acionar sua função Lambda novamente. Se ocorrerem falhas durante a autenticação personalizada, a conexão AWS IoT Core será encerrada. AWS IoT Core também encerra a conexão se ela estiver aberta por mais tempo do que o valor especificado no `disconnectAfterInSeconds` parâmetro.

A seguir JavaScript está uma amostra da função Lambda do Node.js que procura uma senha na mensagem do MQTT Connect com um valor `test` de e retorna uma política que concede permissão

AWS IoT Core para se conectar com um cliente `myClientName` chamado e publicar em um tópico que contém o mesmo nome de cliente. Se a senha esperada não for encontrada, ele retornará uma política que nega essas duas ações.

```
// A simple Lambda function for an authorizer. It demonstrates
// how to parse an MQTT password and generate a response.

exports.handler = function(event, context, callback) {
    var uname = event.protocolData.mqtt.username;
    var pwd = event.protocolData.mqtt.password;
    var buff = new Buffer(pwd, 'base64');
    var passwd = buff.toString('ascii');
    switch (passwd) {
        case 'test':
            callback(null, generateAuthResponse(passwd, 'Allow'));
            break;
        default:
            callback(null, generateAuthResponse(passwd, 'Deny'));
    }
};

// Helper function to generate the authorization response.
var generateAuthResponse = function(token, effect) {
    var authResponse = {};
    authResponse.isAuthenticated = true;
    authResponse.principalId = 'TEST123';

    var policyDocument = {};
    policyDocument.Version = '2012-10-17';
    policyDocument.Statement = [];
    var publishStatement = {};
    var connectStatement = {};
    connectStatement.Action = ["iot:Connect"];
    connectStatement.Effect = effect;
    connectStatement.Resource = ["arn:aws:iot:us-east-1:123456789012:client/
myClientName"];
    publishStatement.Action = ["iot:Publish"];
    publishStatement.Effect = effect;
    publishStatement.Resource = ["arn:aws:iot:us-east-1:123456789012:topic/telemetry/
myClientName"];
    policyDocument.Statement[0] = connectStatement;
    policyDocument.Statement[1] = publishStatement;
    authResponse.policyDocuments = [policyDocument];
```

```
    authResponse.disconnectAfterInSeconds = 3600;
    authResponse.refreshAfterInSeconds = 300;

    return authResponse;
}
```

A função do Lambda anterior retorna o seguinte JSON ao receber a senha esperada de `test` na mensagem do MQTT Connect. Os valores das propriedades `password` e `principalId` serão os valores da mensagem do MQTT Connect.

```
{
  "password": "password",
  "isAuthenticated": true,
  "principalId": "principalId",
  "policyDocuments": [
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Action": "iot:Connect",
          "Effect": "Allow",
          "Resource": "*"
        },
        {
          "Action": "iot:Publish",
          "Effect": "Allow",
          "Resource": "arn:aws:iot:region:accountId:topic/telemetry/${iot:ClientId}"
        },
        {
          "Action": "iot:Subscribe",
          "Effect": "Allow",
          "Resource": "arn:aws:iot:region:accountId:topicfilter/telemetry/
${iot:ClientId}"
        },
        {
          "Action": "iot:Receive",
          "Effect": "Allow",
          "Resource": "arn:aws:iot:region:accountId:topic/telemetry/${iot:ClientId}"
        }
      ]
    }
  ],
  "disconnectAfterInSeconds": 3600,
```

```
  "refreshAfterInSeconds": 300
}
```

Criar um autorizador de capacidade

Você pode criar um autorizador usando a CreateAuthorizerAPI. O exemplo a seguir descreve o comando.

```
aws iot create-authorizer
--authorizer-name MyAuthorizer
--authorizer-function-arn arn:aws:lambda:us-
west-2:<account_id>:function:MyAuthorizerFunction  //The ARN of the Lambda function.
[--token-key-name MyAuthorizerToken //The key used to extract the token from headers.
[--token-signing-public-keys FirstKey=
 "-----BEGIN PUBLIC KEY-----
  [...insert your public key here...]
  -----END PUBLIC KEY-----"
[--status ACTIVE]
[--tags <value>]
[--signing-disabled | --no-signing-disabled]
```

Você pode usar o parâmetro `signing-disabled` para cancelar a validação da assinatura para cada invocação do seu autorizador. É altamente recomendável que você não desative a assinatura, a menos que necessário. A validação de assinatura protege você contra invocações excessivas de sua função do Lambda de dispositivos desconhecidos. Não é possível atualizar o status `signing-disabled` de um autorizador depois de criá-lo. Para alterar esse comportamento, é necessário criar outro autorizador personalizado com um valor diferente para o parâmetro `signing-disabled`.

Os valores dos parâmetros `tokenKeyName` e `tokenSigningPublicKeys` serão opcionais se você tiver desativado a assinatura. Eles são valores obrigatórios se a assinatura estiver ativada.

Depois de criar sua função Lambda e o autorizador personalizado, você deve conceder explicitamente ao AWS IoT Core serviço permissão para invocar a função em seu nome. Você pode fazer isso com o comando a seguir.

> **Note**
>
> O endpoint de IoT padrão pode não aceitar o uso de autorizadores personalizados com funções do Lambda. Em vez disso, você pode usar configurações de domínio para definir um novo endpoint e depois especificar esse endpoint para o autorizador personalizado.

```
aws lambda add-permission --function-name <lambda_function_name>
--principal iot.amazonaws.com --source-arn <authorizer_arn>
--statement-id Id-123 --action "lambda:InvokeFunction"
```

Autorizando AWS IoT a invocação da função Lambda

Nesta seção, você concederá permissão ao recurso de autorizador personalizado que acabou de criar para executar a função do Lambda. Para conceder a permissão, você pode usar o comando da CLI add-permission.

Conceda permissão para sua função Lambda usando o AWS CLI

1. Depois de inserir os valores, digite o seguinte comando. Observe que o valor `statement-id` deve ser exclusivo. Substitua *`Id-1234`* pelo valor exato que você tem, caso contrário, você poderá receber um erro de `ResourceConflictException`.

   ```
   aws lambda add-permission  \
   --function-name "custom-auth-function" \
   --principal "iot.amazonaws.com" \
   --action "lambda:InvokeFunction" \
   --statement-id "Id-1234" \
   --source-arn authorizerArn
   ```

2. Se o comando for executado com êxito, ele retornará uma declaração de permissão, como neste exemplo. Você pode continuar na próxima seção para testar o autorizador personalizado.

   ```
   {
       "Statement": "{\"Sid\":\"Id-1234\",\"Effect\":\"Allow\",\"Principal
   \":{\"Service\":\"iot.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction
   \",\"Resource\":\"arn:aws:lambda:Region:57EXAMPLE833:function:custom-
   auth-function\",\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":
   \"arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-function\"}}}"
   }
   ```

   Se o comando não for executado com êxito, ele retornará um erro, como neste exemplo. Será necessário verificar e corrigir o erro antes de continuar.

   ```
   An error occurred (AccessDeniedException) when calling the AddPermission operation:
    User: arn:aws:iam::57EXAMPLE833:user/EXAMPLE-1 is not authorized to perform:
    lambda:AddPer
   ```

```
mission on resource: arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-
function
```

Testar seus autorizadores

Você pode usar a TestInvokeAuthorizerAPI para testar a invocação e os valores de retorno do seu autorizador. Essa API permite que você especifique metadados de protocolo e teste a validação da assinatura em seu autorizador.

As guias a seguir mostram como usar o AWS CLI para testar seu autorizador.

Unix-like

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER \
--token TOKEN_VALUE --token-signature TOKEN_SIGNATURE
```

Windows CMD

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER ^
--token TOKEN_VALUE --token-signature TOKEN_SIGNATURE
```

Windows PowerShell

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER `
--token TOKEN_VALUE --token-signature TOKEN_SIGNATURE
```

O valor do parâmetro `token-signature` é o token assinado. Para saber como obter esse valor, consulte the section called “Assinatura do token”.

Se seu autorizador usar um nome de usuário e uma senha, você poderá transmitir essas informações usando o parâmetro `--mqtt-context`. As guias a seguir mostram como usar a API `TestInvokeAuthorizer` para enviar um objeto JSON que contém nome de usuário, senha e nome de cliente para o autorizador personalizado.

Unix-like

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER  \
--mqtt-context '{"username": "USER_NAME", "password": "dGVzdA==",
 "clientId":"CLIENT_NAME"}'
```

Windows CMD

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER  ^
--mqtt-context '{"username": "USER_NAME", "password": "dGVzdA==",
 "clientId":"CLIENT_NAME"}'
```

Windows PowerShell

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER  `
--mqtt-context '{"username": "USER_NAME", "password": "dGVzdA==",
 "clientId":"CLIENT_NAME"}'
```

A senha deve ser codificada por base64. O exemplo a seguir mostra como codificar uma senha em um ambiente semelhante ao Unix.

```
echo -n PASSWORD | base64
```

Gerenciar autorizadores personalizados

Você pode gerenciar seus autorizadores usando o seguinte APIs.

- ListAuthorizers: mostre todos os autorizadores em sua conta.
- DescribeAuthorizer: exibe as propriedades do autorizador especificado. Esses valores incluem data de criação, data da última modificação e outros atributos.
- SetDefaultAuthorizer: especifica o autorizador padrão para seus endpoints de AWS IoT Core dados. AWS IoT Core usa esse autorizador se um dispositivo não passar AWS IoT Core credenciais e não especificar um autorizador. Para obter mais informações sobre o uso de AWS IoT Core credenciais, consultethe section called “Autenticação de cliente”.
- UpdateAuthorizer: altera o status, o nome da chave do token ou as chaves públicas do autorizador especificado.
- DeleteAuthorizer: exclui o autorizador especificado.

> **Note**
>
> Não é possível atualizar o requisito de assinatura de um autorizador. Assim, você não pode desativar a assinatura em um autorizador existente que exija isso. Você também não pode exigir a assinatura em um autorizador existente que não exija isso.

Autenticação personalizada com certificados de cliente X.509

Ao conectar dispositivos a AWS IoT Core, você tem vários tipos de autenticação disponíveis. Você pode usar certificados de cliente X.509 que podem ser usados para autenticar conexões de clientes e dispositivos ou definir autorizadores personalizados para gerenciar sua própria lógica de autenticação e autorização de clientes. Este tópico aborda como usar a autenticação personalizada com certificados de cliente X.509.

Usar a autenticação personalizada com certificados X.509 pode ser útil se você já autenticou seus dispositivos usando certificados X.509 e quer realizar validação adicional e autorização personalizada. Por exemplo, se você armazenar os dados de seus dispositivos, como seus números de série, no certificado do cliente X.509, depois de AWS IoT Core autenticar o certificado do cliente X.509, poderá usar um autorizador personalizado para identificar dispositivos específicos com base nas informações armazenadas no campo do certificado. CommonName O uso da autenticação personalizada com certificados X.509 pode aprimorar o gerenciamento da segurança do dispositivo ao conectar dispositivos AWS IoT Core e fornecer mais flexibilidade para gerenciar a lógica de autenticação e autorização. AWS IoT Core suporta autenticação personalizada com certificados X.509 usando o certificado X.509 e o tipo de autenticação de autorizador personalizado, que funciona com o protocolo MQTT e o protocolo HTTPS. Para obter mais informações sobre os tipos de autenticação e os protocolos de aplicativos compatíveis com os endpoints do dispositivo do AWS IoT Core , consulte Protocolos de comunicação do dispositivo.

> **Note**
>
> A autenticação personalizada com certificados de cliente X.509 não é suportada nas AWS GovCloud (US) regiões.

> **Important**
>
> Você deve usar um endpoint criado usando configurações de domínio. Além disso, os clientes devem fornecer a extensão de Indicação de Nome do Servidor (SNI) ao se conectarem a. AWS IoT Core

O processo para autenticar dispositivos usando autenticação personalizada com certificados de cliente X.509 consiste nas etapas a seguir.

- Etapa 1: registre seus certificados de cliente X.509 com AWS IoT Core
- Etapa 2: Criar uma função do Lambda
- Etapa 3: criar um autorizador personalizado
- Etapa 4: definir o tipo de autenticação e o protocolo de aplicativo em uma configuração de domínio

Etapa 1: registre seus certificados de cliente X.509 com AWS IoT Core

Se você ainda não fez isso, registre e ative seus certificados de cliente X.509 com. AWS IoT Core Caso contrário, vá para a próxima etapa.

Para registrar e ativar seus certificados de cliente com AWS IoT Core, siga as etapas:

1. Se você criar certificados de cliente diretamente com AWS IoT. Esses certificados de cliente serão registrados automaticamente com AWS IoT Core.
2. Se você criar seus próprios certificados de cliente, siga estas instruções para registrá-los AWS IoT Core.
3. Para ativar seus certificados de cliente, siga estas instruções.

Etapa 2: Criar uma função do Lambda

AWS IoT Core usa autorizadores personalizados para implementar esquemas personalizados de autenticação e autorização. Um autorizador personalizado está associado a uma função do Lambda que determina se um dispositivo está autenticado e quais operações o dispositivo tem permissão para realizar. Quando um dispositivo se conecta AWS IoT Core, AWS IoT Core recupera os detalhes do autorizador, incluindo o nome do autorizador e a função Lambda associada, e invoca a função Lambda. A função do Lambda recebe um evento que contém um objeto JSON com os dados do certificado do cliente X.509 do dispositivo. Sua função do Lambda usa esse objeto JSON do evento

para avaliar a solicitação de autenticação, decidir as ações a serem tomadas e enviar uma resposta de volta.

Exemplo de evento da função do Lambda

O objeto JSON de exemplo a seguir contém todos os campos possíveis que podem ser incluídos. O objeto JSON real conterá apenas campos relevantes para a solicitação de conexão específica.

```
{
 "token": "aToken",
 "signatureVerified": true,
 "protocols": [
  "tls",
  "mqtt"
 ],
 "protocolData": {
  "tls": {
   "serverName": "serverName",
   "x509CertificatePem": "x509CertificatePem",
   "principalId": "principalId"
  },
  "mqtt": {
   "clientId": "myClientId",
                    "username": "myUserName",
                    "password": "myPassword"
  }
 },
 "connectionMetadata": {
  "id": "UUID"
 }
}
```

`signatureVerified`

Um valor booliano que indica se a assinatura do token configurada no autorizador foi verificada ou não antes de invocar a função do Lambda do autorizador. Se o autorizador estiver configurado para desativar a assinatura do token, esse campo será falso.

`protocols`

Uma matriz que contém os protocolos esperados da solicitação.

`protocolData`

Um objeto que contém informações dos protocolos usados na conexão. Ele dá detalhes específicos do protocolo que podem ser úteis para autenticação, autorização e muito mais.

`tls`: este objeto contém informações relacionadas ao protocolo TLS (Transport Layer Security).

- `serverName`: a string de nome de host Indicação de nome de servidor (SNI). O AWS IoT Core requer que os dispositivos enviem a extensão SNI para o protocolo Transport Layer Security (TLS) e forneçam o endereço completo do endpoint no campo `host_name`.
- `x509CertificatePem`: o certificado X.509 no formato PEM, usado para autenticação do cliente na conexão TLS.
- `principalId`: o identificador da entidade principal associado ao cliente na conexão TLS.

`mqtt`: esse objeto contém informações relacionadas ao protocolo MQTT.

- `clientId`: uma string só precisa ser incluída no caso de o dispositivo enviar esse valor.
- `username`: o nome de usuário fornecido no pacote MQTT Connect.
- `password`: a senha fornecida no pacote MQTT Connect.

`connectionMetadata`

Metadados da conexão.

`id`: o ID da conexão, que você pode usar para registrar em log e solucionar problemas.

> **Note**
>
> Neste objeto JSON do evento, `x509CertificatePem` e `principalId` são dois novos campos na solicitação. O valor de `principalId` é igual ao valor de `certificateId`. Para obter mais informações, consulte Certificado.

Exemplo de resposta da função do Lambda

A função do Lambda deve usar informações do objeto JSON do evento para autenticar a conexão de entrada e decidir quais ações são permitidas na conexão.

O objeto JSON a seguir contém um exemplo de resposta que sua função do Lambda pode enviar.

```
{
 "isAuthenticated": true,
 "principalId": "xxxxxxxx",
 "disconnectAfterInSeconds": 86400,
 "refreshAfterInSeconds": 300,
 "policyDocuments": [
  {
   "Version": "2012-10-17",
   "Statement": [
    {
     "Effect": "Allow",
     "Action": "iot:Publish",
     "Resource": "arn:aws:iot:us-east-1:123456789012:topic/customauthtesting"
    }
   ]
  }
 ]
}
```

Neste exemplo, essa função deve enviar uma resposta que contenha os valores a seguir.

`isAuthenticated`

Um valor booliano que indica se a solicitação foi autenticada.

`principalId`

Uma sequência alfanumérica que atua como um identificador para o token enviado pela solicitação de autorização personalizada. O valor deve ser uma sequência alfanumérica com pelo menos um e não mais que 128 caracteres. Ele identifica a conexão nos logs. O valor de `principalId` deve ser igual ao valor de `principalId` no objeto JSON do evento (ou seja, certificateID do certificado X.509).

`policyDocuments`

Uma lista de documentos de políticas formatados em JSON AWS IoT Core . O valor é opcional e dá suporte a variáveis de política de objeto e variáveis de política de certificado. O número máximo de documentos de política é de 10. Cada documento de política pode ter, no máximo, 2.048 caracteres. Se você tiver várias políticas anexadas ao certificado de cliente e à função do Lambda, a permissão é uma coleção de todas as políticas. Para obter mais informações sobre a criação de AWS IoT Core políticas, consulte Políticas.

`disconnectAfterInSeconds`

Um número inteiro que especifica a duração máxima (em segundos) da conexão com o AWS IoT Core gateway. O valor mínimo é de 300 segundos e o valor máximo é de 86.400 segundos. `disconnectAfterInSeconds` é válido durante a vida útil de uma conexão e não é atualizado em atualizações consecutivas de políticas.

`refreshAfterInSeconds`

Um número inteiro que especifica o intervalo entre as atualizações da política. Quando esse intervalo passa, AWS IoT Core invoca a função Lambda para permitir atualizações de políticas. O valor mínimo é de 300 segundos e o valor máximo é de 86.400 segundos.

Exemplo de função do Lambda

A seguir está um exemplo de uma função do Lambda do Node.js. A função examina o certificado X.509 do cliente e extrai informações relevantes, como o número de série, a impressão digital e o nome do assunto. Se as informações extraídas corresponderem aos valores esperados, o cliente terá acesso para se conectar. Esse mecanismo garante que apenas clientes autorizados com certificados válidos possam estabelecer uma conexão.

```
const crypto = require('crypto');

exports.handler = async (event) => {

    // Extract the certificate PEM from the event
    const certPem = event.protocolData.tls.x509CertificatePem;

    // Parse the certificate using Node's crypto module
    const cert = new crypto.X509Certificate(certPem);

    var effect = "Deny";
    // Allow permissions only for a particular certificate serial, fingerprint, and
 subject
    if (cert.serialNumber === "7F8D2E4B9C1A5036DE8F7C4B2A91E5D80463BC9A1257" // This is
 a random serial
       && cert.fingerprint ===
 "F2:9A:C4:1D:B5:E7:08:3F:6B:D0:4E:92:A7:C1:5B:8D:16:0F:E3:7A" // This is a random
 fingerprint
       && cert.subject === "allow.example.com") {
      effect = "Allow";
    }
```

```
    return generateAuthResponse(event.protocolData.tls.principalId, effect);
};


// Helper function to generate the authorization response.
function generateAuthResponse(principalId, effect) {
    const authResponse = {
        isAuthenticated: true,
        principalId,
        disconnectAfterInSeconds: 3600,
        refreshAfterInSeconds: 300,
        policyDocuments: [
          {
            Version: "2012-10-17",
            Statement: [
              {
                Action: ["iot:Connect"],
                Effect: effect,
                Resource: [
                  "arn:aws:iot:us-east-1:123456789012:client/myClientName"
                ]
              },
              {
                Action: ["iot:Publish"],
                Effect: effect,
                Resource: [
                  "arn:aws:iot:us-east-1:123456789012:topic/telemetry/myClientName"
                ]
              },
              {
                Action: ["iot:Subscribe"],
                Effect: effect,
                Resource: [
                   "arn:aws:iot:us-east-1:123456789012:topicfilter/telemetry/
myClientName"
                ]
              },
              {
                Action: ["iot:Receive"],
                Effect: effect,
                Resource: [
                   "arn:aws:iot:us-east-1:123456789012:topic/telemetry/myClientName"
                ]
```

```
            }
          ]
        }
      ]
    };

  return authResponse;
}
```

A função do Lambda anterior retorna o JSON a seguir ao receber um certificado com o número de série, a impressão digital e o assunto esperados. O valor de `x509CertificatePem` será o certificado do cliente fornecido no handshake TLS. Para obter mais informações, consulte Definir sua função do Lambda.

```
{
 "isAuthenticated": true,
 "principalId": "principalId in the event JSON object",
 "policyDocuments": [
  {
   "Version": "2012-10-17",
   "Statement": [
    {
     "Action": "iot:Connect",
     "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:client/myClientName"
    },
    {
     "Action": "iot:Publish",
     "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:topic/telemetry/myClientName"
    },
    {
     "Action": "iot:Subscribe",
     "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/telemetry/
myClientName"
    },
    {
     "Action": "iot:Receive",
     "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:topic/telemetry/myClientName"
    }
   ]
```

```
  }
 ],
 "disconnectAfterInSeconds": 3600,
 "refreshAfterInSeconds": 300
}
```

Etapa 3: criar um autorizador personalizado

Depois de você definir a função do Lambda, crie um autorizador personalizado para gerenciar sua própria lógica de autenticação e autorização do cliente. É possível seguir as instruções detalhadas na Etapa 3: criar um recurso de autorizador de cliente e sua autorização. Para obter mais informações, consulte Criar um autorizador.

No processo de criação do autorizador personalizado, você deve conceder a permissão do AWS IoT para invocar a função do Lambda após sua criação. Para obter instruções detalhadas, consulte Autorização AWS IoT para invocar sua função Lambda.

Etapa 4: definir o tipo de autenticação e o protocolo de aplicativo em uma configuração de domínio

Para autenticar dispositivos usando autenticação personalizada com certificados de cliente X.509, você deve definir o tipo de autenticação e o protocolo de aplicativo em uma configuração de domínio e enviar a extensão SNI. O valor de `authenticationType` deve ser `CUSTOM_AUTH_X509` e o valor de `applicationProtocol` pode ser `SECURE_MQTT` ou `HTTPS`.

Definir o tipo de autenticação e o protocolo de aplicativo em uma configuração de domínio (CLI)

Se você não tiver uma configuração de domínio, use o comando create-domain-configuration para criar uma. O valor de `authenticationType` deve ser `CUSTOM_AUTH_X509` e o valor de `applicationProtocol` pode ser `SECURE_MQTT` ou `HTTPS`.

```
aws iot create-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --authentication-type CUSTOM_AUTH_X509 \
    --application-protocol SECURE_MQTT \
    --authorizer-config '{
        "defaultAuthorizerName": my-custom-authorizer
    }'
```

Se você já tiver uma configuração de domínio, use o comando update-domain-configuration e atualize `authenticationType` e `applicationProtocol`, se necessário. Observe que não é possível alterar o tipo de autenticação ou o protocolo no endpoint padrão (`iot:Data-ATS`).

```
aws iot update-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --authentication-type CUSTOM_AUTH_X509 \
    --application-protocol SECURE_MQTT \
    --authorizer-config '{
        "defaultAuthorizerName": my-custom-authorizer
    }'
```

`domain-configuration-name`

O nome da configuração do domínio.

`authentication-type`

O tipo de autenticação da configuração do domínio. Para obter mais informações, consulte Escolher um tipo de autenticação.

`application-protocol`

O protocolo de aplicativo que os dispositivos usam para se comunicar com o AWS IoT Core. Para obter mais informações, consulte Escolhendo um protocolo de aplicativo.

`--authorizer-config`

Um objeto que especifica a configuração do autorizador em uma configuração de domínio.

`defaultAuthorizerName`

O nome do autorizador para uma configuração de domínio.

Para obter mais informações, consulte CreateDomainConfiguratione a UpdateDomainConfigurationpartir da Referência AWS IoT da API. Para obter mais informações sobre configuração de domínio, consulte Configurações de domínio.

AWS IoT Core Conectando-se usando autenticação personalizada

Os dispositivos podem se conectar AWS IoT Core usando autenticação personalizada com qualquer protocolo AWS IoT Core compatível com mensagens de dispositivos. Para obter mais informações sobre protocolos de comunicação disponíveis, consulte the section called “Protocolos de comunicação do dispositivo”. Os dados de conexão transmitidos para a função do Lambda do autorizador dependem do protocolo usado. Para obter mais informações sobre como a função do Lambda do autorizador, consulte the section called “Definição de sua função do Lambda”. As seções a seguir explicam como se conectar para autenticar usando cada protocolo compatível.

HTTPS

Dispositivos que enviam dados AWS IoT Core usando a API HTTP Publish podem passar credenciais por meio de cabeçalhos de solicitação ou parâmetros de consulta em suas solicitações HTTP POST. Os dispositivos podem especificar um autorizador a ser invocado usando o cabeçalho `x-amz-customauthorizer-name` ou o parâmetro de consulta. Se você tiver a assinatura de token ativada no autorizador, será necessário transmitir *`token-key-name`* e `x-amz-customauthorizer-signature` nos cabeçalhos da solicitação ou nos parâmetros de consulta. Observe que o *`token-signature`* valor deve ser codificado em URL ao ser usado JavaScript de dentro do navegador.

> **Note**
>
> O autorizador do cliente para o protocolo HTTPS é compatível somente com operações de publicação. Para obter mais informações sobre o protocolo HTTP, consulte the section called "Protocolos de comunicação do dispositivo".

Os exemplos de solicitações a seguir mostram como transmitir esses parâmetros nos cabeçalhos de solicitação e nos parâmetros de consulta.

```
//Passing credentials via headers
POST /topics/topic?qos=qos HTTP/1.1
Host: your-endpoint
x-amz-customauthorizer-signature: token-signature
token-key-name: token-value
x-amz-customauthorizer-name: authorizer-name

//Passing credentials via query parameters
POST /topics/topic?qos=qos&x-amz-customauthorizer-signature=token-signature&token-key-name=token-value HTTP/1.1
```

MQTT

Os dispositivos que se conectam AWS IoT Core usando uma conexão MQTT podem passar credenciais pelos `password` campos `username` e das mensagens MQTT. Opcionalmente, o valor `username` também pode conter uma string de consulta que transmite valores adicionais (incluindo token, assinatura e nome do autorizador) ao autorizador. Você poderá usar essa string de consulta caso queira usar um esquema de autenticação baseado em tokens em vez de valores `username` e `password`.

> **Note**
>
> Os dados no campo de senha são codificados em base64 por. AWS IoT Core A sua função do Lambda deve decodificá-los.

O exemplo a seguir contém uma string `username` com parâmetros extras que especificam um token e uma assinatura.

```
username?x-amz-customauthorizer-name=authorizer-name&x-amz-customauthorizer-
signature=token-signature&token-key-name=token-value
```

Para invocar um autorizador, os dispositivos que se conectam AWS IoT Core usando o MQTT e a autenticação personalizada devem se conectar na porta 443. Eles também devem passar a extensão TLS de negociação de protocolo de camada de aplicativo (ALPN) com um valor de `mqtt` e a extensão de indicação de nome de servidor (SNI) com o nome do host de seu endpoint de dados. AWS IoT Core Para evitar possíveis erros, o valor de `x-amz-customauthorizer-signature` deve ser codificado em URL. Também é altamente recomendável que os valores de `x-amz-customauthorizer-name` e `token-key-name` sejam codificados em URL. Para obter mais informações sobre esses valores, consulte the section called “Protocolos de comunicação do dispositivo”. O AWS IoT Device SDKs, Mobile SDKs e AWS IoT Device Client V2 pode configurar essas duas extensões.

Acabou o MQTT WebSockets

Os dispositivos que se conectam AWS IoT Core usando o MQTT over WebSockets podem transmitir credenciais de uma das duas maneiras a seguir.

- Por meio de cabeçalhos de solicitação ou parâmetros de consulta na solicitação HTTP UPGRADE para estabelecer a WebSockets conexão.
- Através dos campos `username` e `password` na mensagem do MQTT CONNECT.

Se você transmitir as credenciais pela mensagem de conexão do MQTT, as extensões TLS ALPN e SNI serão necessárias. Para obter mais informações sobre essas extensões, consulte the section called “MQTT”. O exemplo a seguir demonstra como transmitir credenciais por meio da solicitação HTTP Upgrade.

```
GET /mqtt HTTP/1.1
```

```
Host: your-endpoint
Upgrade: WebSocket
Connection: Upgrade
x-amz-customauthorizer-signature: token-signature
token-key-name: token-value
sec-WebSocket-Key: any random base64 value
sec-websocket-protocol: mqtt
sec-WebSocket-Version: websocket version
```

Assinatura do token

É necessário assinar o token com a chave privada do par de chaves públicas/privadas usadas na chamada `create-authorizer`. Os exemplos a seguir mostram como criar a assinatura do token usando um comando semelhante ao Unix e. JavaScript Eles usam o algoritmo de hash SHA-256 para codificar a assinatura.

Command line

```
echo -n TOKEN_VALUE | openssl dgst -sha256 -sign PEM encoded RSA private key |
 openssl base64
```

JavaScript

```
const crypto = require('crypto')

const key = "PEM encoded RSA private key"

const k = crypto.createPrivateKey(key)
let sign = crypto.createSign('SHA256')
sign.write(t)
sign.end()
const s = sign.sign(k, 'base64')
```

Solução de problemas dos autorizadores

Este tópico aborda problemas comuns que podem causar problemas em fluxos de trabalho de autenticação personalizada e as etapas para resolvê-los. Para solucionar problemas com mais eficiência, habilite CloudWatch os registros AWS IoT Core e defina o nível do registro

como DEBUG. Você pode ativar CloudWatch os registros no AWS IoT Core console (https://console.aws.amazon.com/iot/). Para obter mais informações sobre como habilitar e configurar logs do AWS IoT Core, consulte the section called “Configurar o AWS IoT registro”.

> **Note**
>
> Se você deixar o nível de registro em DEBUG por longos períodos de tempo, CloudWatch poderá armazenar grandes quantidades de dados de registro. Isso pode aumentar suas CloudWatch cobranças. Considere usar o registro baseado em recursos para aumentar a verbosidade somente para dispositivos em um determinado grupo de objetos. Para obter mais informações sobre o registro baseado em recursos, consulte the section called “Configurar o AWS IoT registro”. Além disso, ao concluir a solução de problemas, reduza o nível do log para um nível menos detalhado.

Antes de iniciar a solução de problemas, analise the section called “Entender o fluxo de trabalho de autenticação personalizada” para obter uma visão geral do processo de autenticação personalizada. Isso ajuda você a entender onde procurar a origem de um problema.

Este tópico discute as duas áreas a seguir para você investigar.

- Problemas relacionados à função do Lambda do autorizador.
- Problemas relacionados ao dispositivo.

Verifique se há problemas na função do Lambda do autorizador

Execute as etapas a seguir para garantir que as tentativas de conexão de seus dispositivos estejam invocando a função do Lambda.

1. Verifique qual função do Lambda está associada ao seu autorizador.

   Você pode fazer isso chamando a DescribeAuthorizerAPI ou clicando no autorizador desejado na seção Seguro do AWS IoT Core console.
2. Verifique as métricas de invocação da função do Lambda. Para fazer isso, execute as seguintes etapas.

   a. Abra o AWS Lambda console (https://console.aws.amazon.com/lambda/) e selecione a função associada ao seu autorizador.

b. Escolha a guia Monitorar e visualize as métricas do período relevante para o seu problema.

3. Se você não ver nenhuma invocação, verifique se AWS IoT Core tem permissão para invocar sua função Lambda. Se você vir invocações, pule para a próxima etapa. Execute as etapas a seguir para verificar se sua função do Lambda tem as permissões necessárias.

   a. Escolha a guia Permissões para sua função no AWS Lambda console.

   b. Localize a seção Política baseada em recursos na parte inferior da página. Se a função do Lambda tiver as permissões necessárias, a política será semelhante ao exemplo a seguir.

```
{
  "Version":"2012-10-17",
  "Id": "default",
  "Statement": [
    {
      "Sid": "Id123",
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-
east-1:111111111111:function:FunctionName",
      "Condition": {
        "ArnLike": {
          "AWS:SourceArn": "arn:aws:iot:us-east-1:111111111111:authorizer/
AuthorizerName"
        },
        "StringEquals": {
          "AWS:SourceAccount": "111111111111"
        }
      }
    }
  ]
}
```

   c. Esta política concede a `InvokeFunction` permissão de sua função ao AWS IoT Core diretor. Se você não a vê, precisará adicioná-la usando a AddPermissionAPI. O exemplo a seguir mostra como fazer isso utilizando a AWS CLI.

```
aws lambda add-permission --function-name FunctionName --principal
 iot.amazonaws.com --source-arn AuthorizerARn --statement-id Id-123 --action
 "lambda:InvokeFunction"
```

4. Se você vir invocações, verifique se não há erros. Um erro pode indicar que a função Lambda não está manipulando adequadamente o evento de conexão que é AWS IoT Core enviado para ela.

   Para ver informações sobre como lidar com o evento na função do Lambda, consulte the section called “Definição de sua função do Lambda”. Você pode usar o recurso de teste no AWS Lambda console (https://console.aws.amazon.com/lambda/) para codificar os valores de teste na função para garantir que a função esteja manipulando os eventos corretamente.

5. Se você vir invocações sem erros, mas os dispositivos não conseguirem se conectar (ou publicar, assinar e receber mensagens), talvez a política retornada pela função do Lambda não conceda permissões para as ações que os dispositivos estão tentando realizar. Execute as etapas a seguir para determinar se há algo errado com a política retornada pela função.

   a. Use uma consulta do Amazon CloudWatch Logs Insights para escanear registros por um curto período de tempo para verificar se há falhas. A consulta de exemplo a seguir classifica os eventos por data e hora e procura falhas.

      ```
      display clientId, eventType, status, @timestamp | sort @timestamp desc | filter
       status = "Failure"
      ```

   b. Atualize sua função Lambda para registrar os dados aos quais ela está retornando AWS IoT Core e o evento que aciona a função. Você pode usar esses registros para inspecionar a política criada pela função.

6. Se você vir invocações sem erros, mas os dispositivos não conseguirem se conectar (ou publicar, assinar e receber mensagens), outra possível causa é que a função do Lambda excede o limite de tempo. O limite de tempo da função do Lambda para o autorizador personalizado é de 5 segundos. Você pode verificar a duração da função em CloudWatch registros ou métricas.

Investigar problemas do dispositivo

Se você não encontrar problemas ao invocar a função do Lambda ou com a política que a função retorna, procure problemas com as tentativas de conexão dos dispositivos. Solicitações de conexão

malformadas podem fazer com que seu autorizador AWS IoT Core não seja acionado. Problemas de conexão podem ocorrer tanto na camada TLS quanto na camada do aplicativo.

Possíveis problemas na camada TLS:

- Os clientes devem passar um cabeçalho de nome de host (HTTP, MQTT over WebSockets) ou a extensão TLS de indicação de nome de servidor (HTTP, MQTT over WebSockets, MQTT) em todas as solicitações de autenticação personalizadas. Em ambos os casos, o valor passado deve corresponder a um dos endpoints de AWS IoT Core dados da sua conta. Esses são os endpoints que são retornados ao executar os seguintes comandos da CLI.
  - `aws iot describe-endpoint --endpoint-type iot:Data-ATS`
  - `aws iot describe-endpoint --endpoint-type iot:Data`(para VeriSign endpoints antigos)
- Os dispositivos que usam autenticação personalizada para conexões MQTT também devem transmitir a extensão TLS Application Layer Protocol Negotiation (ALPN) com um valor de `mqtt`.
- No momento, a autenticação personalizada está disponível somente na porta 443.

Possíveis problemas na camada de aplicativo:

- Se a assinatura estiver ativada (o campo `signingDisabled` é falso em seu autorizador), procure os seguintes problemas de assinatura.
  - Certifique-se de transmitir a assinatura do token no cabeçalho `x-amz-customauthorizer-signature` ou em um parâmetro de string de consulta.
  - Certifique-se de que o serviço não esteja assinando um valor diferente do token.
  - Certifique-se de transmitir o token no cabeçalho ou no parâmetro de consulta que você especificou no campo `token-key-name` do autorizador.
- Verifique se o nome do autorizador transmitido no cabeçalho `x-amz-customauthorizer-name` ou no parâmetro do string de consulta é válido ou se você tem um autorizador padrão definido para sua conta.

Autorização

Autorização é o processo de concessão de permissões a uma identidade autenticada. Você concede permissões no AWS IoT Core uso AWS IoT Core e nas políticas do IAM. Este tópico aborda políticas

do AWS IoT Core . Para obter mais informações sobre políticas do IAM, consulte Gerenciamento de identidade e acesso para AWS IoT e Como AWS IoT funciona com o IAM.

AWS IoT Core as políticas determinam o que uma identidade autenticada pode fazer. A identidade autenticada é usada por dispositivos, aplicativos móveis, aplicativos web e aplicativos de desktop. Uma identidade autenticada pode até mesmo ser um usuário digitando comandos CLI AWS IoT Core . Uma identidade só pode executar AWS IoT Core operações se tiver uma política que conceda permissão para essas operações.

Tanto AWS IoT Core as políticas quanto as políticas do IAM são usadas AWS IoT Core para controlar as operações que uma identidade (também chamada de principal) pode realizar. O tipo de política que você usa depende do tipo de identidade com AWS IoT Core a qual você está usando para se autenticar.

AWS IoT Core as operações são divididas em dois grupos:

- A API de plano de controle permite executar tarefas administrativas, como criar ou atualizar certificados, objetos, regras, e assim por diante.
- A API do plano de dados permite que você envie e receba dados de AWS IoT Core.

O tipo de política usada depende se você estiver usando a API de plano de controle ou de plano de dados.

A tabela a seguir mostra os tipos de identidade, os protocolos usados e os tipos de política que podem ser usados para a autorização.

AWS IoT Core API do plano de dados e tipos de políticas

Mecanismo de protocolo e autenticação	SDK	Tipo de identidade	Tipo de política		
MQTT por TLS/TCP, autenticação mútua TLS (porta 8883 ou 443)[†]	AWS IoT SDK do dispositivo	Certificados X.509	AWS IoT Core política		

Mecanismo de protocolo e autenticação	SDK	Tipo de identidade	Tipo de política		
MQTT via HTTPS/ WebSocket, autenticação AWS SigV4 (porta 443)	AWS SDK móvel	Identidade autenticada do Amazon Cognito	IAM e AWS IoT Core políticas		
		Identidade não autenticada do Amazon Cognito	Política do IAM		
		Identidade do IAM ou federada	Política do IAM		
HTTPS, autenticação AWS Signature versão 4 (porta 443)	AWS CLI	Identidade do Amazon Cognito, IAM ou federada	Política do IAM		
HTTPS, autenticação mútua TLS (porta 8443)	Não há suporte para o SDK	Certificados X.509	AWS IoT Core política		
HTTPS sobre autenticação personalizada (Porta 443)	AWS IoT SDK do dispositivo	Autorizador personalizado	Política do autorizador personalizado		

AWS IoT Core API do plano de controle e tipos de políticas

Mecanismo de protocolo e autenticação	SDK	Tipo de identidade	Tipo de política		
Autenticação HTTPS AWS Signature versão 4 (porta 443)	AWS CLI	Identidade do Amazon Cognito	Política do IAM		
		Identidade do IAM ou federada	Política do IAM		

AWS IoT Core as políticas são anexadas a certificados X.509, identidades do Amazon Cognito ou grupos de coisas. As políticas do IAM são anexadas a um usuário, grupo ou perfil do IAM. Se você usa o AWS IoT console ou a AWS IoT Core CLI para anexar a política (a um certificado, Amazon Cognito Identity ou grupo de coisas), você usa uma política. AWS IoT Core Caso contrário, você usa uma política do IAM. AWS IoT Core as políticas associadas a um grupo de coisas se aplicam a qualquer coisa dentro desse grupo de coisas. Para que a AWS IoT Core política entre em vigor, o nome `clientId` e o nome da coisa devem corresponder.

A autorização com base em políticas é uma ferramenta poderosa. Ela oferece controle total sobre o que um dispositivo, um usuário ou um aplicativo pode fazer no AWS IoT Core. Por exemplo, considere um dispositivo conectado AWS IoT Core a um certificado. Você pode permitir que o dispositivo acesse todos os tópicos MQTT ou pode restringir seu acesso a um único tópico. Em outro exemplo, considere que um usuário digite comandos CLI na linha de comando. Ao usar uma política, você pode permitir ou negar o acesso a qualquer comando ou AWS IoT Core recurso para o usuário. Você também pode controlar o acesso de um aplicativo aos recursos do AWS IoT Core .

As alterações feitas em uma política podem levar alguns minutos para entrarem em vigor devido à forma como o AWS IoT armazena em cache os documentos de política. Ou seja, pode levar alguns minutos para acessar um recurso que recebeu acesso recentemente, e um recurso pode ficar acessível por vários minutos após seu acesso ter sido revogado.

AWS treinamento e certificação

Para obter informações sobre autorização AWS IoT Core, faça o curso Deep Dive into AWS IoT Core Authentication and Authorization no site de AWS Treinamento e Certificação.

AWS IoT Core políticas

AWS IoT Core políticas são documentos JSON. Elas seguem as mesmas convenções das políticas do IAM. AWS IoT Core suporta políticas nomeadas para que muitas identidades possam fazer referência ao mesmo documento de política. As políticas nomeadas são versionadas para que possam ser facilmente restabelecidas.

AWS IoT Core as políticas permitem que você controle o acesso ao plano AWS IoT Core de dados. O plano de dados do AWS IoT Core consiste em operações que permitem que você se conecte ao agente de mensagens do AWS IoT Core , envie e receba mensagens MQTT e obtenha ou atualize a sombra do dispositivo de um objeto.

Uma AWS IoT Core política é um documento JSON que contém uma ou mais declarações de política. Cada instrução contém:

- `Effect`, que especifica se a ação é permitida ou negada.
- `Action`, que especifica a ação que a política está permitindo ou negando.
- `Resource`, que especifica o recurso ou os recursos nos quais a ação é permitida ou negada.

As alterações feitas em uma política podem levar de 6 a 8 minutos para entrarem em vigor devido à forma como os documentos de política são armazenados em AWS IoT cache. Ou seja, pode levar alguns minutos para acessar um recurso que recebeu acesso recentemente, e um recurso pode ficar acessível por vários minutos após seu acesso ter sido revogado.

AWS IoT Core as políticas podem ser anexadas a certificados X.509, identidades do Amazon Cognito e grupos de coisas. As políticas associadas a um grupo de objetos se aplicam a qualquer objeto dentro desse grupo. Para que a política entre em vigor, o `clientId` e o nome do objeto devem coincidir. As políticas do AWS IoT Core seguem a mesma lógica de avaliação de políticas das políticas do IAM. Por padrão, todas as políticas são implicitamente negadas. Uma permissão explícita em qualquer política baseada em recurso ou identidade substitui o comportamento padrão. Uma negação explícita em qualquer política substitui todas as permissões. Para obter mais informações, consulte Lógica da avaliação de políticas no Guia do usuário do AWS Identity and Access Management .

Tópicos

- AWS IoT Core ações políticas
- AWS IoT Core recursos de ação
- AWS IoT Core variáveis de política
- Prevenção contra o ataque do “substituto confuso” em todos os serviços
- AWS IoT Core exemplos de políticas
- Autorização com identidades do Amazon Cognito

AWS IoT Core ações políticas

As ações de política a seguir são definidas pelo AWS IoT Core:

Ações da política MQTT

`iot:Connect`

Representa a permissão para se conectar ao agente de AWS IoT Core mensagens. A permissão `iot:Connect` é verificada sempre que uma solicitação `CONNECT` é enviada ao operador. O agente de mensagens não permite que dois clientes com o mesmo ID de cliente permaneçam conectados ao mesmo tempo. Depois que o segundo cliente se conectar, o agente fechará a conexão existente. Use a permissão `iot:Connect` para garantir que apenas clientes autorizados usando um ID de cliente específico possam se conectar.

`iot:DeleteConnection`

Representa a permissão para desconectar do AWS IoT Core um cliente do MQTT conectado. A permissão `iot:DeleteConnection` é conferida sempre que é feita uma solicitação para desconectar um cliente à força. Quando você desconecta um cliente, AWS IoT Core fecha a conexão de rede do cliente e, opcionalmente, limpa o estado da sessão.

`iot:GetRetainedMessage`

Representa a permissão para obter o conteúdo de uma única mensagem retida. As mensagens retidas são as mensagens que foram publicadas com o sinalizador RETAIN definido e armazenadas por AWS IoT Core. Para receber permissão para obter uma lista de todas as mensagens retidas da conta, consulte `iot:ListRetainedMessages`.

`iot:ListRetainedMessages`

Representa a permissão para recuperar informações resumidas sobre as mensagens retidas da conta, mas não o conteúdo das mensagens. As mensagens retidas são as mensagens que foram publicadas com o sinalizador RETAIN definido e armazenadas por AWS IoT Core. O ARN do recurso especificado para essa ação deve ser `*`. Para receber permissão para obter o conteúdo de uma única mensagem retida, consulte `iot:GetRetainedMessage`.

`iot:Publish`

Representa a permissão para publicar um tópico MQTT. Essa permissão é verificada sempre que uma solicitação PUBLICAR é enviada ao operador. Isso pode ser usado para permitir que os clientes publiquem padrões de tópicos específicos.

> **Note**
>
> Para conceder a permissão `iot:Publish`, conceda também a permissão `iot:Connect`.

`iot:Receive`

Representa a permissão para receber uma mensagem de AWS IoT Core. A permissão `iot:Receive` é confirmada sempre que uma mensagem é entregue a um cliente. Uma vez que essa permissão é verificada em cada entrega, você pode usá-la para revogar permissões para clientes que estejam inscritos em um tópico.

`iot:RetainPublish`

Representa a permissão para publicar uma mensagem MQTT com o sinalizador RETAIN definido.

> **Note**
>
> Para conceder a permissão `iot:RetainPublish`, conceda também a permissão `iot:Publish`.

`iot:Subscribe`

Representa a permissão para se inscrever em um filtro de tópico. Essa permissão é verificada sempre que uma solicitação SUBSCRIBE é enviada ao operador. Use para permitir que os clientes se inscrevam em tópicos que correspondam a padrões de tópico específicos.

> **Note**
>
> Para conceder a permissão `iot:Subscribe`, conceda também a permissão `iot:Connect`.

Ações da política de sombra do dispositivo

`iot:DeleteThingShadow`

Representa a permissão para excluir a sombra do dispositivo de um objeto. A permissão `iot:DeleteThingShadow` é verificada sempre que é feita uma solicitação para excluir o conteúdo da sombra do dispositivo de um objeto.

`iot:GetThingShadow`

Representa a permissão para recuperar a sombra do dispositivo de um objeto. A permissão `iot:GetThingShadow` é verificada sempre que é feita uma solicitação para recuperar o conteúdo da sombra do dispositivo de um objeto.

`iot:ListNamedShadowsForThing`

Representa a permissão para listar as sombras nomeadas de um objeto. A permissão `iot:ListNamedShadowsForThing` é verificada sempre que é feita uma solicitação para listar as sombras nomeadas de um objeto.

`iot:UpdateThingShadow`

Representa a permissão para atualizar uma shadow de dispositivo. A permissão `iot:UpdateThingShadow` é verificada sempre que é feita uma solicitação para atualizar o conteúdo da sombra do dispositivo de um objeto.

> **Note**
>
> As ações da política de execução de trabalhos se aplicam apenas ao endpoint HTTP TLS. Se você usar o endpoint MQTT, deverá usar as ações de política do MQTT definidas neste tópico.
> Para ver um exemplo de uma política de execução de tarefas que demonstra isso, consulte o the section called “Exemplo básico de políticas de trabalho” que funciona com o protocolo MQTT.

Ações da AWS IoT Core Política de Execução de Job

`iotjobsdata:DescribeJobExecution`

Representa a permissão para recuperar uma execução de trabalho para uma determinado objeto. A permissão `iotjobsdata:DescribeJobExecution` é verificada sempre que é feita uma solicitação para obter uma execução de tarefa.

`iotjobsdata:GetPendingJobExecutions`

Representa a permissão para recuperar a lista de trabalhos que não estão em status terminal para um objeto. A permissão `iotjobsdata:GetPendingJobExecutions` é verificada sempre que é feita uma solicitação para recuperar a lista.

`iotjobsdata:UpdateJobExecution`

Representa a permissão para atualizar uma execução de trabalho. A permissão `iotjobsdata:UpdateJobExecution` é verificada sempre que é feita uma solicitação para atualizar o estado de uma execução de trabalho.

`iotjobsdata:StartNextPendingJobExecution`

Representa a permissão para obter e iniciar a próxima execução de trabalho pendente para um objeto. (Isto é, para atualizar uma execução de tarefa com status QUEUED para IN_PROGRESS). A permissão `iotjobsdata:StartNextPendingJobExecution` é verificada sempre que é feita uma solicitação para iniciar a próxima execução de trabalho pendente.

AWS IoT Core Ação da política do provedor de credenciais

`iot:AssumeRoleWithCertificate`

Representa a permissão para chamar o provedor de AWS IoT Core credenciais para assumir uma função do IAM com autenticação baseada em certificado. A `iot:AssumeRoleWithCertificate` permissão é verificada sempre que uma solicitação é feita ao provedor de AWS IoT Core credenciais para assumir uma função.

AWS IoT Core recursos de ação

Para especificar um recurso para uma ação AWS IoT Core política, use o Amazon Resource Name (ARN) do recurso. Todos os recursos ARNs seguem o seguinte formato:

```
arn:partition:iot:region:AWS-account-ID:Resource-type/Resource-name
```

A tabela a seguir mostra o recurso a ser especificado para cada tipo de ação. Os exemplos de ARN são para o ID da conta `123456789012`, na partição `aws`, e específicos para a região `us-east-1`. Para obter mais informações sobre os formatos para ARNs, consulte Amazon Resource Names (ARNs) no Guia AWS Identity and Access Management do usuário.

Ação	Tipo de atributo	Nome do recurso	Exemplo de ARN
`iot:Connect`	`client`	O ID do cliente	`arn:aws:iot:us-east-1:123456789012:client/myClientId`
`iot:DeleteConnection`	`client`	O ID do cliente	`arn:aws:iot:us-east-1:123456789012:client/myClientId`
`iot:DeleteThingShadow`	`thing`	O nome do objeto e da sombra, se aplicável	`arn:aws:iot:us-east-1:123456789012:thing/thingOne arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne`

Ação	Tipo de atributo	Nome do recurso	Exemplo de ARN
iotjobsdata:DescribeJobExecution	thing	O nome do objeto	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iotjobsdata:GetPendingJobExecutions	thing	O nome do objeto	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:GetRetainedMessage	topic	Um tópico de mensagem retida	arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iot:GetThingShadow	thing	O nome do objeto e da sombra, se aplicável	arn:aws:iot:us-east-1:123456789012:thing/thingOne arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne
iot:ListNamedShadowsForThing	Todos	Todos	*
iot:ListRetainedMessages	Todos	Todos	*
iot:Publish	topic	Uma string de tópico	arn:aws:iot:us-east-1:123456789012:topic/myTopicName

Ação	Tipo de atributo	Nome do recurso	Exemplo de ARN
iot:Receive	topic	Uma string de tópico	arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iot:RetainPublish	topic	Um tópico para publicar com o sinalizador RETAIN definido	arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iotjobsdata:StartNextPendingJobExecution	thing	O nome do objeto	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:Subscribe	topicfilter	Uma string de filtro de tópico	arn:aws:iot:us-east-1:123456789012:topicfilter/myTopicFilter
iotjobsdata:UpdateJobExecution	thing	O nome do objeto	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:UpdateThingShadow	thing	O nome do objeto e da sombra, se aplicável	arn:aws:iot:us-east-1:123456789012:thing/thingOne arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne
iot:AssumeRoleWithCertificate	rolealias	Um alias do perfil que aponta para um ARN de perfil	arn:aws:iot:us-east-1:123456789012:rolealias/CredentialProviderRole_alias

AWS IoT Core variáveis de política

AWS IoT Core define variáveis de política que podem ser usadas em AWS IoT Core políticas no `Condition` bloco `Resource` or. Quando uma política é avaliada, as variáveis da política são substituídas por valores reais. Por exemplo, se um dispositivo estiver conectado ao agente de AWS IoT Core mensagens com uma ID de cliente de 100-234-3456, a variável de política será substituída no documento `iot:ClientId` de política por 100-234-3456.

AWS IoT Core as políticas podem usar caracteres curinga e seguir uma convenção semelhante às políticas do IAM. A inserção de um * (asterisco) na string pode ser tratada como um curinga, correspondendo a qualquer caractere. Por exemplo, você pode usar * para descrever vários nomes de tópicos do MQTT no atributo do `Resource` de uma política. Os caracteres + e # são tratados como sequências literais em uma política. Para ver um exemplo de política que mostra como usar curingas, consulte Usando caracteres curinga no MQTT e nas políticas AWS IoT Core.

Você também pode usar variáveis de políticas predefinidas com valores fixos para representar caracteres que, de outra forma, têm um significado especial. Esses caracteres especiais incluem `$(*)`, `$(?)` e `$($)`. Para obter mais informações sobre variáveis de política e os caracteres especiais, consulte Elementos de política do IAM: variáveis e tags e Criação de uma condição com várias chaves ou valores.

Tópicos

- Variáveis AWS IoT Core políticas básicas
- Variáveis de política de objetos
- Variáveis de política de certificado AWS IoT Core X.509

Variáveis AWS IoT Core políticas básicas

AWS IoT Core define as seguintes variáveis básicas de política:

- `aws:SourceIp`: o endereço IP do cliente conectado ao agente de AWS IoT Core mensagens.
- `iot:ClientId`: o ID do cliente usado para se conectar ao agente de mensagens do AWS IoT Core .
- `iot:DomainName`: o nome de domínio do cliente conectado AWS IoT Core a.

Exemplos

- Exemplos de variáveis de política ClientId e SourceIp

- Exemplos de variável de política iot:DomainName

Exemplos de variáveis de política **`ClientId`** e **`SourceIp`**

A AWS IoT Core política a seguir mostra uma política que usa variáveis de política. `aws:SourceIp`pode ser usado no elemento Condição da sua política para permitir que os diretores façam solicitações de API somente dentro de um intervalo de endereços específico. Para obter exemplos, consulte Autorizando usuários e serviços em nuvem a usar o Jobs AWS IoT.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/clientid1"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/my/topic/${iot:ClientId}"
   ],
   "Condition": {
    "IpAddress": {
     "aws:SourceIp": "123.45.167.89"
    }
   }
  }
 ]
}
```

Nesses exemplos, `${iot:ClientId}` é substituído pelo ID do cliente conectado ao agente de AWS IoT Core mensagens quando a política é avaliada. Quando você usa variáveis de políticas,

como `${iot:ClientId}`, você pode abrir acidentalmente o acesso a tópicos não intencionais. Por exemplo, se você usa uma política que usa `${iot:ClientId}` para especificar um filtro de tópico:

```
{
 "Effect": "Allow",
 "Action": [
  "iot:Subscribe"
 ],
 "Resource": [
  "arn:aws:iot:us-east-1:123456789012:topicfilter/my/${iot:ClientId}/topic"
 ]
}
```

Um cliente pode se conectar usando + como o ID do cliente. Isso permitiria que o usuário se inscrevesse em qualquer tópico que correspondesse ao filtro de tópico `my/+/topic`. Para se proteger contra essas falhas de segurança, use a ação `iot:Connect` de política para controlar qual cliente IDs pode se conectar. Por exemplo, essa política permite que apenas clientes cujo ID é `clientid1` se conectem:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/clientid"
   ]
  }
 ]
}
```

> **Note**
>
> Não é recomendável usar a variável de política `${iot:ClientId}` com `Connect`. Não há verificação do valor de `ClientId`; portanto, um anexador com um ID de cliente

diferente pode passar na validação, mas causar a desconexão. Como qualquer `ClientId` é permitido, definir um ID de cliente aleatório pode ignorar as políticas do grupo de objetos.

Exemplos de variável de política **`iot:DomainName`**

Você pode adicionar a variável de política `iot:DomainName` para restringir quais domínios podem ser usados. Adicionar a variável de política `iot:DomainName` permite que os dispositivos se conectem apenas a endpoints configurados específicos.

A política a seguir permite que os dispositivos se conectem ao domínio especificado.

```
{
 "Version":"2012-10-17",
 "Statement": {
  "Sid": "AllowConnectionsToSpecifiedDomain",
  "Effect": "Allow",
  "Action": [
   "iot:Connect"
  ],
  "Resource": "arn:aws:iot:us-east-1:123456789012:client/clientid",
  "Condition": {
   "StringEquals": {
    "iot:DomainName": "d1234567890abcdefghij-ats.iot.us-east-1.amazonaws.com"
   }
  }
 }
}
```

A política a seguir nega que os dispositivos se conectem ao domínio especificado.

```
{
 "Version":"2012-10-17",
 "Statement": {
  "Sid": "DenyConnectionsToSpecifiedDomain",
  "Effect": "Deny",
  "Action": [
   "iot:Connect"
  ],
  "Resource": "arn:aws:iot:us-east-1:123456789012:client/clientid",
```

```
  "Condition": {
   "StringEquals": {
    "iot:DomainName": "d1234567890abcdefghij-ats.iot.us-east-1.amazonaws.com"
   }
  }
 }
}
```

Para obter mais informações sobre operadores de condição de política, consulte Elementos de política JSON do IAM: operadores de condição. Para obter mais informações sobre configurações de domínio, consulte O que é uma configuração de domínio?.

Variáveis de política de objetos

As variáveis de política de coisas permitem que você escreva AWS IoT Core políticas que concedam ou neguem permissões com base em propriedades de coisas, como nomes de coisas, tipos de coisas e valores de atributos de coisas. Você pode usar variáveis de política de coisas para aplicar a mesma política para controlar vários AWS IoT Core dispositivos. Para obter mais informações sobre o provisionamento de dispositivos, consulte Provisionamento de dispositivos.

Se você usa uma associação de coisa não exclusiva, pode anexar o mesmo certificado a várias coisas. Para manter uma associação clara e evitar possíveis conflitos, você deve combinar seu ID de cliente com o nome da coisa. Nesse caso, você obtém o nome da coisa do ID do cliente na `Connect` mensagem MQTT enviada quando uma coisa se conecta a. AWS IoT Core

Lembre-se do seguinte ao usar variáveis de política de objetos em políticas do AWS IoT Core .

- Use a AttachThingPrincipalAPI para anexar certificados ou entidades principais (identidades autenticadas do Amazon Cognito) a uma coisa.
- Se houver uma associação de coisa exclusiva implementada, quando você estiver substituindo nomes de coisas por variáveis de política de coisas, o valor de `clientId` na mensagem de conexão do MQTT ou a conexão TLS deverá corresponder exatamente ao nome da coisa.

As variáveis de política de objetos a seguir estão disponíveis:

- `iot:Connection.Thing.ThingName`

  Isso se resume ao nome da coisa no AWS IoT Core registro para a qual a política está sendo avaliada. AWS IoT Core usa o certificado que o dispositivo apresenta ao se autenticar para

determinar qual coisa usar para verificar a conexão. Essa variável de política só está disponível quando um dispositivo se conecta pelo MQTT ou pelo MQTT pelo WebSocket protocolo.

- `iot:Connection.Thing.ThingTypeName`

  Isso resolve o tipo de objeto associado ao objeto para a qual a política está sendo avaliada. O ID do cliente da WebSocket conexão MQTT/ deve ser igual ao nome da coisa. Essa variável de política está disponível somente ao se conectar pelo MQTT ou pelo MQTT pelo WebSocket protocolo.

- `iot:Connection.Thing.Attributes[`*`attributeName`*`]`

  Isso resolve o valor do atributo especificado associado ao objeto para a qual a política está sendo avaliada. Um objeto pode ter até 50 atributos. Cada atributo está disponível como uma variável de política: `iot:Connection.Thing.Attributes[`*`attributeName`*`]` onde *`attributeName`* está o nome do atributo. O ID do cliente da MQTT/WebSocket conexão deve ser igual ao nome da coisa. Essa variável de política só está disponível ao se conectar pelo MQTT ou pelo MQTT pelo WebSocket protocolo.

- `iot:Connection.Thing.IsAttached`

  `iot:Connection.Thing.IsAttached: ["true"]`determina que somente os dispositivos registrados AWS IoT e conectados ao principal possam acessar as permissões dentro da política. Você pode usar essa variável para impedir que um dispositivo se conecte AWS IoT Core se ele apresentar um certificado que não esteja anexado a uma coisa de IoT no AWS IoT Core registro. Essa variável tem valores `true` ou `false` indica que a coisa conectada está anexada ao certificado ou à identidade do Amazon Cognito no registro usando a API. AttachThingPrincipal O nome do objeto é usado como ID do cliente.

Se o ID do cliente corresponder ao nome da coisa ou se você anexar seu certificado exclusivamente a uma coisa, o uso de variáveis na definição da política pode simplificar o gerenciamento de políticas. Em vez de criar políticas individuais para cada coisa da IoT, você pode definir uma única política usando as variáveis de política da coisa. É possível aplicar essa política a todos os dispositivos de forma dinâmica. Veja a seguir um exemplo de política para mostrar como funciona. Para obter mais informações, consulte ???.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
```

```
    "Condition": {
     "StringLike": {
      "iot:ClientId": "*${iot:Connection.Thing.Attributes[envType]}"
     }
    },
    "Effect": "Allow",
    "Action": "iot:Connect",
    "Resource": "arn:aws:iot:us-east-1:123456789012:client/*"
   }
  ]
}
```

Esse exemplo de política permite que as coisas se conectem AWS IoT Core se o ID do cliente terminar com o valor do `envType` atributo. Somente coisas com um padrão de ID de cliente correspondente poderão se conectar.

Variáveis de política de certificado AWS IoT Core X.509

As variáveis de política de certificado X.509 ajudam na criação AWS IoT Core de políticas. Essas políticas concedem permissões com base nos atributos do certificado X.509. As seções a seguir descrevem como usar essas variáveis de política de certificado.

> **Important**
>
> Se seu certificado X.509 não incluir um atributo de certificado específico, mas a variável de política de certificado correspondente for usada em seu documento de política, a avaliação da política poderá levar a um comportamento inesperado.

CertificateId

Na RegisterCertificateAPI, o `certificateId` aparece no corpo da resposta. Para obter informações sobre seu certificado, use o `certificateId` in DescribeCertificate.

Atributos de emissor

As variáveis AWS IoT Core de política a seguir oferecem suporte à permissão ou negação de permissões, com base nos atributos do certificado definidos pelo emissor do certificado.

- `iot:Certificate.Issuer.DistinguishedNameQualifier`

- `iot:Certificate.Issuer.Country`
- `iot:Certificate.Issuer.Organization`
- `iot:Certificate.Issuer.OrganizationalUnit`
- `iot:Certificate.Issuer.State`
- `iot:Certificate.Issuer.CommonName`
- `iot:Certificate.Issuer.SerialNumber`
- `iot:Certificate.Issuer.Title`
- `iot:Certificate.Issuer.Surname`
- `iot:Certificate.Issuer.GivenName`
- `iot:Certificate.Issuer.Initials`
- `iot:Certificate.Issuer.Pseudonym`
- `iot:Certificate.Issuer.GenerationQualifier`

Atributos de assunto

As variáveis AWS IoT Core de política a seguir oferecem suporte à concessão ou negação de permissões, com base nos atributos do titular do certificado definidos pelo emissor do certificado.

- `iot:Certificate.Subject.DistinguishedNameQualifier`
- `iot:Certificate.Subject.Country`
- `iot:Certificate.Subject.Organization`
- `iot:Certificate.Subject.OrganizationalUnit`
- `iot:Certificate.Subject.State`
- `iot:Certificate.Subject.CommonName`
- `iot:Certificate.Subject.SerialNumber`
- `iot:Certificate.Subject.Title`
- `iot:Certificate.Subject.Surname`
- `iot:Certificate.Subject.GivenName`
- `iot:Certificate.Subject.Initials`
- `iot:Certificate.Subject.Pseudonym`
- `iot:Certificate.Subject.GenerationQualifier`

Os certificados X.509 dão a esses atributos a opção de conter um ou mais valores. Por padrão, as variáveis de política para cada atributo multivalor retornam o primeiro valor. Por exemplo, o atributo `Certificate.Subject.Country` pode conter uma lista de nomes de países, mas, quando avaliado em uma política, `iot:Certificate.Subject.Country` é substituído pelo primeiro nome do país.

É possível solicitar um valor de atributo específico diferente do primeiro valor usando um índice baseado em um. Por exemplo, `iot:Certificate.Subject.Country.1` é substituído pelo segundo nome de país no atributo `Certificate.Subject.Country`. Se você especificar um valor de índice que não existe (por exemplo, se você solicitar um terceiro valor quando há apenas dois valores atribuídos ao atributo), nenhuma substituição será feita e haverá falha na autorização. Você pode usar o sufixo `.List` no nome da variável de política para especificar todos os valores do atributo.

Atributos de nome alternativo do emissor

As variáveis AWS IoT Core de política a seguir oferecem suporte à concessão ou negação de permissões, com base nos atributos de nome alternativo do emissor definidos pelo emissor do certificado.

- `iot:Certificate.Issuer.AlternativeName.RFC822Name`
- `iot:Certificate.Issuer.AlternativeName.DNSName`
- `iot:Certificate.Issuer.AlternativeName.DirectoryName`
- `iot:Certificate.Issuer.AlternativeName.UniformResourceIdentifier`
- `iot:Certificate.Issuer.AlternativeName.IPAddress`

Atributos de nome alternativo do assunto

As variáveis AWS IoT Core de política a seguir oferecem suporte à concessão ou negação de permissões, com base nos atributos de nome alternativo do sujeito definidos pelo emissor do certificado.

- `iot:Certificate.Subject.AlternativeName.RFC822Name`
- `iot:Certificate.Subject.AlternativeName.DNSName`
- `iot:Certificate.Subject.AlternativeName.DirectoryName`
- `iot:Certificate.Subject.AlternativeName.UniformResourceIdentifier`
- `iot:Certificate.Subject.AlternativeName.IPAddress`

Outros atributos

Você pode usar `iot:Certificate.SerialNumber` para permitir ou negar acesso aos AWS IoT Core recursos, com base no número de série de um certificado. A variável de política `iot:Certificate.AvailableKeys` contém o nome de todas as variáveis de política de certificado que contenham valores.

Usar variáveis de política do certificado X.509

Este tópico fornece detalhes sobre como usar variáveis de política de certificados. As variáveis de política do certificado X.509 são essenciais quando você cria políticas do AWS IoT Core que dão permissões com base em atributos do certificado X.509. Se seu certificado X.509 não incluir um atributo de certificado específico, mas a variável de política de certificado correspondente for usada em seu documento de política, a avaliação da política poderá levar a um comportamento inesperado. Isso ocorre porque a variável de política ausente não é avaliada na declaração de política.

Neste tópico:

- Exemplo de certificado X.509
- Usar atributos do emissor de certificados como variáveis de política de certificados
- Usar atributos do sujeito do certificado como variáveis de política de certificados
- Usar atributos de nome alternativo do emissor de certificados como variáveis de política de certificados
- Usar atributos de nome alternativo do sujeito do certificado como variáveis de política de certificados
- Usar outro atributo de certificado como uma variável de política de certificado
- Limitações das variáveis de política de certificado X.509
- Exemplo de políticas usando variáveis de política de certificado

Exemplo de certificado X.509

Um certificado X.509 típico pode aparecer da seguinte forma. Este exemplo de certificado inclui atributos de certificado. Durante a avaliação das políticas do AWS IoT Core , os seguintes atributos de certificado serão preenchidos como variáveis de política de certificado: `Serial Number`, `Issuer`, `Subject`, `X509v3 Issuer Alternative Name` e `X509v3 Subject Alternative Name`.

```
Certificate:
```

```
    Data:
        Version: 3 (0x2)
        Serial Number:
            92:12:85:cb:b7:a5:e0:86
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=IoT Devices, OU=SmartHome, ST=WA, CN=IoT Devices Primary CA,
    GN=Primary CA1/initials=XY/dnQualifier=Example corp,
    SN=SmartHome/ title=CA1/pseudonym=Primary_CA/generationQualifier=2/serialNumber=987

        Validity
            Not Before: Mar 26 03:25:40 2024 GMT
            Not After : Apr 28 03:25:40 2025 GMT
        Subject: C=US, O=IoT Devices, OU=LightBulb, ST=NY, CN=LightBulb Device Cert,
    GN=Bulb/initials=ZZ/dnQualifier=Bulb001,
    SN=Multi Color/title=RGB/pseudonym=RGB Device/generationQualifier=4/
serialNumber=123
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    << REDACTED >>
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Subject Alternative Name:
                DNS:example.com, IP Address:1.2.3.4, URI:ResourceIdentifier001,
 email:device1@example.com, DirName:/C=US/O=IoT/OU=SmartHome/CN=LightBulbCert
            X509v3 Issuer Alternative Name:
                DNS:issuer.com, IP Address:5.6.7.8, URI:PrimarySignerCA,
 email:primary@issuer.com, DirName:/C=US/O=Issuer/OU=IoT Devices/CN=Primary Issuer CA
    Signature Algorithm: sha256WithRSAEncryption
         << REDACTED >>
```

Usar atributos do emissor de certificados como variáveis de política de certificados

A tabela a seguir fornece detalhes de como os atributos do emissor do certificado serão preenchidos em uma AWS IoT Core política.

Atributos do emissor a serem preenchidos em uma política

Atributos do emissor do certificado	Variáveis de política do certificado
• C=US • Dispositivos O=IoT • OU = SmartHome • ST=WA • CN=IoT Devices Primary CA • GN = primário CA1 • initials=XY • dnQualifier=Example corp • SN = SmartHome • título = CA1 • pseudonym=Primary_CA • generationQualifier=2 • serialNumber=987	• `iot:Certificate.Issuer.Country = US` • `iot:Certificate.Issuer.Organization = IoT Devices` • `iot:Certificate.Issuer.OrganizationalUnit = SmartHome` • `iot:Certificate.Issuer.State = WA` • `iot:Certificate.Issuer.CommonName = IoT Devices Primary CA` • `iot:Certificate.Issuer.GivenName = Primary CA1` • `iot:Certificate.Issuer.initials = XY` • `iot:Certificate.Issuer.DistinguishedNameQualifier = Example corp` • `iot:Certificate.Issuer.Surname = SmartHome` • `iot:Certificate.Issuer.Title = CA1` • `iot:Certificate.Issuer.Pseudonym = Primary_CA` • `iot:Certificate.Issuer.GenerationQualifier = 2` • `iot:Certificate.Issuer.SerialNumber = 987`

Usar atributos do sujeito do certificado como variáveis de política de certificados

A tabela a seguir fornece detalhes de como os atributos do titular do certificado serão preenchidos em uma AWS IoT Core política.

Atributos do sujeito a serem preenchidos em uma política

Atributos do sujeito do certificado	Variáveis de política do certificado
• C=US • Dispositivos O=IoT • ST=NY • CN= Certificado de LightBulb dispositivo • GN=Bulb • initials=ZZ • dnQualifier=Bulb001 • SN=Multi Color • title=RGB • pseudonym=RGB Device • generationQualifier=4 • serialNumber=123	• `iot:Certificate.Subject.Country = US` • `iot:Certificate.Subject.Organization = IoT Devices` • `iot:Certificate.Subject.State = NY` • `iot:Certificate.Subject.CommonName = LightBulb Device Cert` • `iot:Certificate.Subject.GivenName = Bulb` • `iot:Certificate.Subject.initials = ZZ` • `iot:Certificate.Subject.DistinguishedNameQualifier = Bulb001` • `iot:Certificate.Subject.Surname = Multi Color` • `iot:Certificate.Subject.Title = RGB` • `iot:Certificate.Subject.Pseudonym = RGB Device` • `iot:Certificate.Subject.GenerationQualifier = 4` • `iot:Certificate.Subject.SerialNumber = 123`

Usar atributos de nome alternativo do emissor de certificados como variáveis de política de certificados

A tabela a seguir fornece detalhes de como os atributos de nome alternativo do emissor do certificado serão preenchidos em uma política do AWS IoT Core .

Atributos de nome alternativo do emissor a serem preenchidos em uma política

Nome alternativo do emissor X509v3	Atributo em uma política
• DNS:issuer.com • Endereço IP:5.6.7.8 • TIPO: PrimarySigner CA • email:primary@issuer.com • DirName: /C= US/O=Issuer/OU=IoT Devices/CN =Emissor primário CA	• `iot:Certificate.Issuer.AlternativeName.DNSName = issuer.com` • `iot:Certificate.Issuer.AlternativeName.IPAddress = 5.6.7.8` • `iot:Certificate.Issuer.AlternativeName.UniformResourceIdentifier = PrimarySignerCA` • `iot:Certificate.Issuer.AlternativeName.RFC822Name = primary@issuer.com` • `iot:Certificate.Issuer.AlternativeName.DirectoryName = cn=Primary Issuer CA,ou=IoT Devices,o=Issuer,c=US`

Usar atributos de nome alternativo do sujeito do certificado como variáveis de política de certificados

A tabela a seguir fornece detalhes de como os atributos de nome alternativo do sujeito do certificado serão preenchidos em uma política do AWS IoT Core .

Atributos de nome alternativo do sujeito a serem preenchidos em uma política

Nome alternativo do assunto X509v3	Atributo em uma política
• DNS:example.com • Endereço IP:1.2.3.4 • URI: ResourceIdentifier 001 • email:device1@example.com • DirName: /C= = US/O=IoT/OU=SmartHome/CN LightBulbCert	• `iot:Certificate.Subject.AlternativeName.DNSName = example.com` • `iot:Certificate.Subject.AlternativeName.IPAddress = 1.2.3.4` • `iot:Certificate.Subject.AlternativeName.UniformResourceIdentifier = ResourceIdentifier001`

Nome alternativo do assunto X509v3	Atributo em uma política
	• `iot:Certificate.Subject.AlternativeName.RFC822Name = device1@example.com` • `iot:Certificate.Subject.AlternativeName.DirectoryName = cn=LightBulbCert,ou=SmartHome,o=IoT,c=US`

Usar outro atributo de certificado como uma variável de política de certificado

A tabela a seguir fornece detalhes de como outros atributos do certificado serão preenchidos em uma AWS IoT Core política.

Outros atributos a serem preenchidos em uma política

Outro atributo do certificado	Variável de política do certificado
`Serial Number: 92:12:85:cb:b7:a5:e0:86`	`iot:Certificate.SerialNumber = 105256223 89124227206`

Limitações das variáveis de política de certificado X.509

As limitações a seguir se aplicam às variáveis de política de certificado X.509:

Variáveis de política ausentes

Se seu certificado X.509 não incluir um atributo de certificado específico, mas a variável de política de certificado correspondente for usada em seu documento de política, a avaliação da política poderá levar a um comportamento inesperado. Isso ocorre porque a variável de política ausente não é avaliada na declaração de política.

SerialNumber Formato do certificado

AWS IoT Core trata o número de série do certificado como a representação em cadeia de um número inteiro decimal. Por exemplo, se uma política só permite conexões com a ID do cliente correspondente ao número de série do certificado, o ID do cliente deve ser o número de série em formato decimal.

Curingas

Se caracteres curinga estiverem presentes em atributos de certificado, a variável de política não será substituída pelo valor do atributo do certificado. Isso deixará o texto `${policy-variable}` no documento de política. Isso pode causar falha de autorização. Os seguintes caracteres curinga podem ser usados: `*`, `$`, `+`, `?` e `#`.

Campos da matriz

Os atributos de certificado que contêm matrizes estão limitados a cinco itens. Os itens adicionais são ignorados.

Tamanho da segmento

Todos os valores de string são limitados a 1024 caracteres. Se um atributo de certificado tiver uma sequência com mais de 1024 caracteres, a variável de política não será substituída pelo valor do atributo do certificado. Isso deixará o `${policy-variable}` no documento de política. Isso pode causar falha de autorização.

Caracteres especiais

Qualquer caractere especial, como `,`, `'`, `\`, `+`, `=`, `<`, `>` e `;`, deve ser prefixado com uma barra invertida (`\`) quando usado em uma variável de política. Por exemplo, `Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington C=US` torna-se `Amazon Web Service O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US`.

Exemplo de políticas usando variáveis de política de certificado

O documento de política a seguir permite conexões com o ID do cliente que corresponde ao número de série do certificado e a publicação no tópico que corresponde ao padrão: `${iot:Certificate.Subject.Organization}/device-stats/${iot:ClientId}/*`.

> **Important**
>
> Se seu certificado X.509 não incluir um atributo de certificado específico, mas a variável de política de certificado correspondente for usada em seu documento de política, a avaliação da política poderá levar a um comportamento inesperado. Isso ocorre porque a variável de política ausente não é avaliada na declaração de política. Por exemplo, se você anexar o seguinte documento de política a um certificado que não contém o atributo `iot:Certificate.Subject.Organization`, as variáveis da política do certificado

`iot:Certificate.Subject.Organization` não serão preenchidas durante a avaliação da política.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Certificate.SerialNumber}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Certificate.Subject.Organization}/device-stats/${iot:ClientId}/*"
   ]
  }
 ]
}
```

Você também pode usar o operador de condição Null para garantir que as variáveis de política de certificado usadas em uma política sejam preenchidas durante a avaliação da política. O documento de política a seguir permite `iot:Connect` com certificados somente quando os atributos Número de Série do Certificado e Nome Comum do Assunto do Certificado estão presentes.

Todas as variáveis da política de certificado têm valores String, portanto, há suporte para todos os Operadores de condição string.

```
{
 "Version":"2012-10-17",
```

```
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/*"
   ],
   "Condition": {
    "Null": {
     "iot:Certificate.SerialNumber": "false",
     "iot:Certificate.Subject.CommonName": "false"
    }
   }
  }
 ]
}
```

Prevenção contra o ataque do “substituto confuso” em todos os serviços

O problema “confused deputy” é um problema de segurança em que uma entidade que não tem permissão para executar uma ação pode coagir uma entidade mais privilegiada a executá-la. Em AWS, a falsificação de identidade entre serviços pode resultar no problema confuso do deputado. A personificação entre serviços pode ocorrer quando um serviço (o serviço de chamada) chama outro serviço (o serviço chamado). O serviço de chamada pode ser manipulado de modo a usar suas permissões para atuar nos recursos de outro cliente de uma forma na qual ele não deveria ter permissão para acessar. Para evitar isso, a AWS fornece ferramentas que ajudam você a proteger seus dados para todos os serviços com entidades principais de serviço que receberam acesso aos recursos em sua conta.

Para limitar as permissões que AWS IoT concedem outro serviço ao recurso, recomendamos usar as chaves de contexto de condição `aws:SourceAccount`global `aws:SourceArn`e as chaves de contexto nas políticas de recursos. Se você utilizar ambas as chaves de contexto de condição global, o valor `aws:SourceAccount` e a conta `aws:SourceArn` no valor deverão utilizar o mesmo ID de conta quando utilizados na mesma instrução de política.

A maneira mais eficaz de se proteger contra o problema substituto confuso é usar a chave de contexto de condição global `aws:SourceArn` com o nome do recurso da Amazon (ARN) completo do recurso. Para AWS IoT, você `aws:SourceArn` deve estar em conformidade com o formato:

`arn:aws:iot:region:account-id:resource-type/resource-id` para permissões específicas de recursos ou`arn:aws:iot:region:account-id:*`. O resource-id pode ser o nome ou ID do recurso permitido ou uma declaração curinga do recurso permitido. IDs Certifique-se de que *region* corresponda à sua AWS IoT região e que *account-id* corresponda ao ID da sua conta de cliente.

O exemplo a seguir mostra como evitar o confuso problema adjunto usando as chaves de contexto de condição `aws:SourceAccount` global `aws:SourceArn` e as chaves de contexto na política de confiança da AWS IoT função. Para obter mais exemplos, consulte Exemplos detalhados de prevenção contra representante confuso.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"iot.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
        },
            "ArnLike":{
               "aws:SourceArn":"arn:aws:iot:us-east-1:123456789012:*"
         }
          }
      }
   ]
}
```

> **Note**
>
> Se você receber erros de negação de acesso, talvez a integração do serviço com o AWS Security Token Service (STS) não aceite as chaves de contexto `aws:SourceArn` e `aws:SourceAccount`.

Exemplos detalhados de prevenção contra representante confuso

Esta seção fornece exemplos detalhados de como evitar o confuso problema adjunto usando as chaves de contexto de condição aws:SourceAccount global aws:SourceArn e as chaves de contexto na política de confiança da AWS IoT função.

- Provisionamento de frotas
- JITP
- Provedor de credencial

Provisionamento de frotas

Você pode configurar o provisionamento da frota usando um recurso de modelo de provisionamento. Quando um modelo de provisionamento faz referência a uma função de provisionamento, a política de confiança dessa função pode incluir as chaves de condição `aws:SourceArn` e `aws:SourceAccount`. Essas chaves limitam os recursos para os quais a configuração pode invocar a solicitação `sts:AssumeRole`.

A função com a política de confiança a seguir só pode ser assumida pela entidade principal de IoT (`iot.amazonaws.com`) para o modelo de provisionamento especificado no `SourceArn`.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"iot.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
        },
            "ArnLike":{
               "aws:SourceArn":"arn:aws:iot:us-
east-1:123456789012:provisioningtemplate/example_template"
        }
         }
      }
```

```
    ]
}
```

JITP

No just-in-time provisionamento (JITP), você pode usar o modelo de provisionamento como um recurso separado da CA ou definir o corpo do modelo e a função como parte da configuração do certificado da CA. O valor da política de confiança `aws:SourceArn` na AWS IoT função depende de como você define o modelo de provisionamento.

Definir o modelo de provisionamento como um recurso separado

Se você definir seu modelo de provisionamento como um recurso separado, o valor de `aws:SourceArn` poderá ser "`arn:aws:iot:`*`region`*`:`*`account-id`*`:provisioningtemplate/`*`example_template`*"". Você pode usar o mesmo exemplo de política em Provisionamento de frotas.

Definir o modelo de provisionamento em um certificado de CA

Se você definir seu modelo de provisionamento em um recurso de certificado de CA, o valor de `aws:SourceArn` poderá ser "`arn:aws:iot:`*`region`*`:`*`account-id`*`:cacert/`*`cert_id`*"" ou "`arn:aws:iot:`*`region`*`:`*`account-id`*`:cacert/*`". Você pode usar um caractere curinga quando o identificador do recurso, como a ID de um certificado de CA, é desconhecido no momento da criação.

A função com a política de confiança a seguir só pode ser assumida pela entidade principal de IoT (`iot.amazonaws.com`) para o certificado de CA especificado no `SourceArn`.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"iot.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
```

```
                "aws:SourceAccount":"123456789012"
            },
                "ArnLike":{
                    "aws:SourceArn":"arn:aws:iot:us-
east-1:123456789012:cacert/
8ecde6884f3d87b1125ba31ac3fcb13d7016de7f57cc904fe1cb97c6ae98196e"
            }
             }
        }
    ]
}
```

Ao criar um certificado de CA, você pode referenciar uma função de provisionamento na configuração do registro. A política de confiança da função de provisionamento pode usar `aws:SourceArn` para restringir para quais recursos a função pode ser assumida. No entanto, durante a CACertificate chamada de registro inicial para registrar o certificado CA, você não teria o ARN do certificado CA para especificar na `aws:SourceArn` condição.

Para contornar isso, ou seja, especificar a política de confiança da função de aprovisionamento para o certificado de CA específico registrado AWS IoT Core, você pode fazer o seguinte:

- Primeiro, chame Register CACertificate sem fornecer o `RegistrationConfig` parâmetro.
- Depois que o certificado CA for registrado AWS IoT Core, chame Atualizar CACertificate nele.

  Na CACertificate chamada de atualização, forneça uma `RegistrationConfig` que inclua a política de confiança da função de provisionamento `aws:SourceArn` definida como o ARN do certificado CA recém-registrado.

Provedor de credencial

Para provedor de AWS IoT Core credenciais, use o mesmo Conta da AWS que você usa para criar o alias de função em `aws:SourceAccount` e especifique uma instrução que corresponda ao ARN do recurso do tipo de recurso rolealias em. `aws:SourceArn` Ao criar uma função do IAM para uso com o provedor de AWS IoT Core credenciais, você deve incluir na `aws:SourceArn` condição todos os aliases ARNs de função que possam precisar assumir a função, autorizando assim a solicitação entre serviços. `sts:AssumeRole`

A função com a seguinte política de confiança só pode ser assumida pela entidade principal do provedor de credenciais do AWS IoT Core (`credentials.iot.amazonaws.com`) para o roleAlias

especificado no `SourceArn`. Se uma entidade principal tentar recuperar as credenciais de um alias de função diferente do especificado na condição `aws:SourceArn`, a solicitação será negada, mesmo que esse outro alias de função faça referência ao mesmo perfil do IAM.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:iot:us-
east-1:123456789012:rolealias/example_rolealias"
        }
      }
    }
  ]
}
```

AWS IoT Core exemplos de políticas

Os exemplos de políticas nesta seção ilustram os documentos de política usados para concluir tarefas comuns no AWS IoT Core. Você pode usá-los como exemplos para começar a criar as políticas para suas soluções.

Os exemplos desta seção usam as seguintes etapas:

- the section called “AWS IoT Core ações políticas”
- the section called “AWS IoT Core recursos de ação”
- the section called “Exemplos de políticas baseadas em identidade”
- the section called “Variáveis AWS IoT Core políticas básicas”
- the section called “Variáveis de política de certificado AWS IoT Core X.509”

Exemplos de políticas nesta seção:

- Exemplos de política de conexão
- Exemplos de política de publicação/inscrição
- Exemplos de política para conectar e publicar
- Exemplos de políticas de mensagens retidas
- Exemplos de política de certificado
- Exemplos de política de objetos
- Exemplo básico de políticas de trabalho

Exemplos de política de conexão

A política a seguir nega permissão ao cliente IDs `client1` e à conexão AWS IoT Core, `client2` ao mesmo tempo em que permite que os dispositivos se conectem usando uma ID de cliente. O ID do cliente corresponde ao nome de uma coisa registrada no AWS IoT Core registro e anexada ao principal usado para conexão:

> **Note**
>
> Para dispositivos registrados, recomendamos usar variáveis de política de objetos para ações de `Connect` e anexar o objeto à entidade principal usada para a conexão.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Deny",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/client1",
    "arn:aws:iot:us-east-1:123456789012:client/client2"
   ]
  },
  {
   "Effect": "Allow",
```

```
    "Action": [
     "iot:Connect"
    ],
    "Resource": [
     "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
    ],
    "Condition": {
     "Bool": {
      "iot:Connection.Thing.IsAttached": "true"
     }
    }
   }
  ]
}
```

A política a seguir concede permissão para se conectar AWS IoT Core com o ID do cliente`client1`. Este exemplo de política é para dispositivos não registrados.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/client1"
   ]
  }
 ]
}
```

Exemplos de políticas de sessões persistentes do MQTT

O `connectAttributes` permite que você especifique quais atributos deseja usar na mensagem de conexão em suas políticas do IAM, como `PersistentConnect` e `LastWill`. Para obter mais informações, consulte Uso do ConnectAttributes.

A política a seguir permite se conectar ao atributo `PersistentConnect`:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect"
     ]
    }
   }
  }
 ]
}
```

A política a seguir não permite `PersistentConnect`; outros atributos são permitidos:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringNotEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect"
     ]
    }
   }
  }
 ]
```

```
}
```

A política acima também pode ser expressa usando `StringEquals`; qualquer outro atributo é permitido, incluindo um novo:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1"
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
            }
        }
        }
    ]
}
```

A política a seguir permite a conexão por `PersistentConnect` e `LastWill`; nenhum outro atributo novo é permitido:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
```

```
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect",
      "LastWill"
     ]
    }
   }
  }
 ]
}
```

A política a seguir permite a conexão limpa por clientes com ou sem `LastWill`; nenhum outro atributo será permitido:

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iot:Connect"
        ],
        "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
        "Condition": {
            "StringEquals": {
                "iot:ConnectAttributes": "LastWill"
        }
        }
    }]
}
```

A política a seguir só permite a conexão usando atributos padrão:

```
{
```

```
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": []
    }
   }
  }
 ]
}
```

A política a seguir permite a conexão somente com `PersistentConnect`; qualquer atributo novo é permitido, desde que a conexão use `PersistentConnect`:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
            }
        }
        }
    ]
}
```

A política a seguir afirma que a conexão deve usar `PersistentConnect` e `LastWill`; nenhum atributo novo é permitido:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect",
      "LastWill"
     ]
    }
   }
  },
  {
   "Effect": "Deny",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "*",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect"
     ]
    }
   }
  },
  {
   "Effect": "Deny",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "*",
   "Condition": {
    "ForAllValues:StringEquals": {
```

```
     "iot:ConnectAttributes": [
      "LastWill"
     ]
    }
   }
  },
  {
   "Effect": "Deny",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "*",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": []
    }
   }
  }
 ]
}
```

A política a seguir não pode ter `PersistentConnect`, mas pode ter `LastWill`; nenhum outro atributo novo é permitido:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
            }
        }
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "iot:ConnectAttributes": [
                        "LastWill"
                    ]
            }
        }
        }
    ]
}
```

A política a seguir permite a conexão somente por clientes que tenham um `LastWill` com o tópico `"my/lastwill/topicName"`; qualquer atributo é permitido, desde que use o tópico `LastWill`:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ArnEquals": {
                "iot:LastWillTopic": "arn:aws:iot:us-east-1:123456789012:topic/my/
lastwill/topicName"
            }
        }
        }
    ]
}
```

A política a seguir só permite a conexão limpa usando um `LastWillTopic`; qualquer atributo é permitido, desde que use o `LastWillTopic`:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ArnEquals": {
                "iot:LastWillTopic": "arn:aws:iot:us-east-1:123456789012:topic/my/
lastwill/topicName"
            }
        }
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
            }
        }
        }
    ]
}
```

Exemplos de política de publicação/inscrição

A política que você usa depende de como você está se conectando AWS IoT Core. Você pode se conectar AWS IoT Core usando um cliente MQTT, HTTP ou WebSocket. Ao se conectar a um

cliente MQTT, você está se autenticando com um certificado X.509. Ao se conectar via HTTP ou pelo WebSocket protocolo, você está se autenticando com o Signature versão 4 e o Amazon Cognito.

> **Note**
>
> Para dispositivos registrados, recomendamos usar variáveis de política de objetos para ações de `Connect` e anexar o objeto à entidade principal usada para a conexão.

Nesta seção:

- Usando caracteres curinga no MQTT e nas políticas AWS IoT Core
- Políticas para publicar, assinar e receber mensagens sobre tópicos to/from específicos
- Políticas para publicar, assinar e receber to/from tópicos de mensagens com um prefixo específico
- Políticas para publicar, assinar e receber mensagens, to/from tópicos específicos para cada dispositivo
- Políticas para publicar, assinar e receber to/from tópicos de mensagens com o atributo thing no nome do tópico
- Políticas para negar a publicação de mensagens em subtópicos do nome de um tópico
- Políticas para negar o recebimento de mensagens de subtópicos do nome de um tópico
- Políticas para assinar tópicos usando caracteres curinga do MQTT
- Políticas para HTTP e WebSocket clientes

Usando caracteres curinga no MQTT e nas políticas AWS IoT Core

O MQTT e AWS IoT Core as políticas têm caracteres curinga diferentes e você deve escolhê-los após uma análise cuidadosa. No MQTT, os caracteres + curinga # são usados nos filtros de tópicos do MQTT para assinar vários nomes de tópicos. AWS IoT Core as políticas usam * e ? como caracteres curinga e seguem as convenções das políticas do IAM. Em um documento de política, o * representa qualquer combinação caracteres, e um ponto de interrogação ? representa qualquer caractere único. Em documentos de política, os caracteres curinga do MQTT + e # são tratados como aqueles caracteres sem significado especial. Para descrever vários nomes de tópicos e filtros de tópicos no atributo `resource` de uma política, use os caracteres curinga * e ? no lugar dos caracteres curinga do MQTT.

Ao escolher caracteres curinga para usar em um documento de política, considere que o caractere * não está confinado a um único nível de tópico. O caractere + está confinado a um só nível de tópico

em um filtro de tópicos MQTT. Para ajudar a restringir uma especificação de curinga a um único nível de filtro de tópico do MQTT, considere usar vários caracteres ?. Para obter mais informações sobre o uso de caracteres curinga em um recurso de política e mais exemplos de sua correspondência, consulte Usando curingas no recurso. ARNs

A tabela abaixo mostra os diferentes caracteres curinga usados nas políticas do MQTT e do AWS IoT Core para clientes MQTT.

Caractere curinga	É um caractere curinga do MQTT	Exemplo no MQTT	A AWS IoT Core política é um caractere curinga	Exemplo de AWS IoT Core políticas para clientes MQTT
#	Sim	`some/#`	Não	N/D
+	Sim	`some/+/topic`	Não	N/D
*	Não	N/D	Sim	`topicfilter/some/*/topic` `topicfilter/some/sensor*/topic`
?	Não	N/D	Sim	`topic/some/?????/topic` `topicfilter/some/sensor???/topic`

Políticas para publicar, assinar e receber mensagens sobre tópicos to/from específicos

A seguir, são mostrados exemplos de dispositivos registrados e não registrados para publicar, assinar e receber mensagens sobre to/from o tópico chamado “some_specific_topic”. Os exemplos também destacam que `Publish` e `Receive` usam "topic" como recurso, enquanto `Subscribe` usa "topicfilter".

Registered devices

Para dispositivos registrados no AWS IoT Core registro, a política a seguir permite que os dispositivos se conectem com ClientID que corresponda ao nome de uma coisa no registro. Ele também fornece permissões de `Publish`, `Subscribe` e `Receive` para o tópico chamado “some_specific_topic”.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topicfilter/some_specific_topic"
   ]
  },
```

```
  {
   "Effect": "Allow",
   "Action": [
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
   ]
  }
 ]
}
```

Unregistered devices

Para dispositivos não registrados no AWS IoT Core registro, a política a seguir permite que os dispositivos se conectem usando ClientID1, ClientID2 ou ClientId3. Ele também fornece permissões de `Publish`, `Subscribe` e `Receive` para o tópico chamado “some_specific_topic”.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/
some_specific_topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
            ]
        }
    ]
}
```

Políticas para publicar, assinar e receber to/from tópicos de mensagens com um prefixo específico

Veja a seguir exemplos de to/from tópicos de dispositivos registrados e não registrados para publicar, assinar e receber mensagens com o prefixo “topic_prefix”.

> **Note**
>
> Observe o uso do caractere curinga * neste exemplo. Embora o * seja útil para fornecer permissões para vários nomes de tópicos em uma única instrução, ele pode levar a consequências inesperadas ao fornecer mais privilégios aos dispositivos do que o necessário. Portanto, recomendamos que você use o caractere curinga * somente após uma análise cuidadosa.

Registered devices

Para dispositivos registrados no AWS IoT Core registro, a política a seguir permite que os dispositivos se conectem com ClientID que corresponda ao nome de uma coisa no registro. Ele

também fornece permissões de `Publish`, `Subscribe` e `Receive` para tópicos com o prefixo "topic_prefix".

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish",
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/topic_prefix*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topicfilter/topic_prefix*"
   ]
  }
 ]
}
```

Unregistered devices

Para dispositivos não registrados no AWS IoT Core registro, a política a seguir permite que os dispositivos se conectem usando ClientID1, ClientID2 ou ClientId3. Ele também fornece permissões de `Publish`, `Subscribe` e `Receive` para tópicos com o prefixo “topic_prefix”.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/topic_prefix*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/topic_prefix*"
            ]
        }
    ]
}
```

Políticas para publicar, assinar e receber mensagens, to/from tópicos específicos para cada dispositivo

Veja a seguir exemplos de to/from tópicos específicos de dispositivos registrados e não registrados para publicar, assinar e receber mensagens específicas de um determinado dispositivo.

Registered devices

Para dispositivos registrados no AWS IoT Core registro, a política a seguir permite que os dispositivos se conectem com ClientID que corresponda ao nome de uma coisa no registro. Ela fornece permissão para publicar no tópico específico do objeto (`sensor/device/${iot:Connection.Thing.ThingName}`) e também assinar e receber do tópico específico do objeto (`command/device/${iot:Connection.Thing.ThingName}`). Se o nome da coisa no registro for “coisa1", o dispositivo poderá publicar no tópico" 1"sensor/device/thing1". The device will also be able to subscribe to and receive from the topic "command/device/thing.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/sensor/device/
${iot:Connection.Thing.ThingName}"
```

```
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topicfilter/command/device/
${iot:Connection.Thing.ThingName}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/command/device/
${iot:Connection.Thing.ThingName}"
   ]
  }
 ]
}
```

Unregistered devices

Para dispositivos não registrados no AWS IoT Core registro, a política a seguir permite que os dispositivos se conectem usando ClientID1, ClientID2 ou ClientId3. Ela fornece permissão para publicar no tópico específico do cliente (`sensor/device/${iot:ClientId}`) e também assinar e receber do tópico específico do cliente (`command/device/${iot:ClientId}`). Se o dispositivo se conectar ao ClientID como ClientId1, ele poderá publicar no tópico "1". sensor/device/clientId O dispositivo também poderá se inscrever e receber do tópico `device/clientId1/command`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/sensor/device/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/command/device/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/command/device/
${iot:Connection.Thing.ThingName}"
            ]
        }
    ]
}
```

Políticas para publicar, assinar e receber to/from tópicos de mensagens com o atributo thing no nome do tópico

A seguir, é mostrado um exemplo de to/from tópicos de dispositivos registrados para publicar, assinar e receber mensagens cujos nomes incluem atributos de coisas.

> **Note**
>
> Os atributos de objeto existem apenas para dispositivos registrados no Registro do AWS IoT Core . Não há um exemplo correspondente para dispositivos não registrados.

Registered devices

Para dispositivos registrados no AWS IoT Core registro, a política a seguir permite que os dispositivos se conectem com ClientID que corresponda ao nome de uma coisa no registro. Ela fornece permissão para publicar no tópico (`sensor/${iot:Connection.Thing.Attributes[version]}`) e assinar e receber do tópico (`command/${iot:Connection.Thing.Attributes[location]}`), onde o nome do tópico inclui atributos de objeto. Se o nome da coisa no registro tiver `version=v1` `elocation=Seattle`, o dispositivo poderá publicar no tópico "sensor/v1", and subscribe to and receive from the topic "command/Seattle".

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
```

```
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/sensor/
${iot:Connection.Thing.Attributes[version]}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topicfilter/command/
${iot:Connection.Thing.Attributes[location]}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/command/
${iot:Connection.Thing.Attributes[location]}"
   ]
  }
 ]
}
```

Unregistered devices

Uma vez que os atributos de objeto existem apenas para dispositivos registrados no Registro do AWS IoT Core , não há um exemplo correspondente paro objetos não registradas.

Políticas para negar a publicação de mensagens em subtópicos do nome de um tópico

Veja a seguir exemplos de dispositivos registrados e não registrados para publicar mensagens em todos os tópicos, exceto em determinados subtópicos.

Registered devices

Para dispositivos registrados no AWS IoT Core registro, a política a seguir permite que os dispositivos se conectem com ClientID que corresponda ao nome de uma coisa no registro. Ela fornece permissão para publicar em todos os tópicos com o prefixo "department/", mas não no subtópico "department/admins".

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/department/*"
   ]
  },
  {
   "Effect": "Deny",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/department/admins"
   ]
  }
```

```
 ]
}
```

Unregistered devices

Para dispositivos não registrados no AWS IoT Core registro, a política a seguir permite que os dispositivos se conectem usando ClientID1, ClientID2 ou ClientId3. Ela fornece permissão para publicar em todos os tópicos com o prefixo "department/", mas não no subtópico "department/admins".

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/department/*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/department/admins"
            ]
```

```
        }
    ]
}
```

Políticas para negar o recebimento de mensagens de subtópicos do nome de um tópico

Veja a seguir exemplos de dispositivos registrados e não registrados para publicar e receber mensagens de tópicos com prefixos específicos, exceto determinados subtópicos.

Registered devices

Para dispositivos registrados no AWS IoT Core registro, a política a seguir permite que os dispositivos se conectem com ClientID que corresponda ao nome de uma coisa no registro. A política permite que os dispositivos assinem qualquer tópico com o prefixo "topic_prefix". Ao usar `NotResource` na declaração for`iot:Receive`, permitimos que o dispositivo receba mensagens de todos os tópicos nos quais o dispositivo se inscreveu, exceto os tópicos prefixados com “prefix/restricted". For example, with this policy, devices can subscribe to "topic_prefix/topic1" and even "topic_prefix/restricted", however, they will only receive messages from the topic "topic_prefix/topic1" and no messages from the topic "topic_prefix/restrictedtópico_”.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": "iot:Subscribe",
```

```
   "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/topic_prefix/*"
  },
  {
   "Effect": "Allow",
   "Action": "iot:Receive",
   "NotResource": "arn:aws:iot:us-east-1:123456789012:topic/topic_prefix/
restricted/*"
  }
 ]
}
```

Unregistered devices

Para dispositivos não registrados no AWS IoT Core registro, a política a seguir permite que os dispositivos se conectem usando ClientID1, ClientID2 ou ClientId3. A política permite que os dispositivos assinem qualquer tópico com o prefixo "topic_prefix". Ao usar `NotResource` na declaração for`iot:Receive`, permitimos que o dispositivo receba mensagens de todos os tópicos nos quais o dispositivo se inscreveu, exceto tópicos prefixados com “prefix/restricted". For example, with this policy, devices can subscribe to "topic_prefix/topic1" and even "topic_prefix/restricted". However, they will only receive messages from the topic "topic_prefix/topic1" and no messages from the topic "topic_prefix/restrictedtópico_”.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
```

```
            "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/
topic_prefix/*"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Receive",
            "NotResource": "arn:aws:iot:us-east-1:123456789012:topic/
topic_prefix/restricted/*"
        }
    ]
}
```

Políticas para assinar tópicos usando caracteres curinga do MQTT

Os caracteres curinga MQTT + e # são tratados como cadeias de caracteres literais, mas não são tratados como curingas quando usados em políticas. AWS IoT Core No MQTT, + e # são tratados como curingas somente ao assinar um filtro de tópico, mas como uma string literal em todos os outros contextos. Recomendamos que você use esses curingas do MQTT somente como parte das AWS IoT Core políticas após uma análise cuidadosa.

A seguir, mostramos exemplos de itens registrados e não registrados usando curingas MQTT nas políticas. AWS IoT Core Esses curingas são tratados como strings literais.

Registered devices

Para dispositivos registrados no AWS IoT Core registro, a política a seguir permite que os dispositivos se conectem com ClientID que corresponda ao nome de uma coisa no registro. A política permite que os dispositivos assinem os tópicos "department/+/employees" e "location/#". Como + e # são tratados como cadeias de caracteres literais nas AWS IoT Core políticas, os dispositivos podem se inscrever no tópico “departamento/+/funcionários”, mas também não no tópico "". department/engineering/employees". Similarly, devices can subscribe to the topic "location/#" but not to the topic "location/Seattle". However, once the device subscribes to the topic "department/+/employees", the policy will allow them to receive messages from the topic "department/engineering/employees". Similarly, once the device subscribes to the topic "location/#", they will receive messages from the topic "location/Seattle

```
{
 "Version":"2012-10-17",
 "Statement": [
```

```
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": "iot:Subscribe",
   "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/department/+/
employees"
  },
  {
   "Effect": "Allow",
   "Action": "iot:Subscribe",
   "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/location/#"
  },
  {
   "Effect": "Allow",
   "Action": "iot:Receive",
   "Resource": "arn:aws:iot:us-east-1:123456789012:topic/*"
  }
 ]
}
```

Unregistered devices

Para dispositivos não registrados no AWS IoT Core registro, a política a seguir permite que os dispositivos se conectem usando ClientID1, ClientID2 ou ClientId3. A política permite que os dispositivos assinem os tópicos "department/+/employees" e "location/#". Como + e # são tratados como cadeias de caracteres literais nas AWS IoT Core políticas, os dispositivos podem se inscrever no tópico “departamento/+/funcionários”, mas também não no tópico "". department/engineering/employees". Similarly, devices can subscribe to the topic "location/

#" but not "location/Seattle". However, once the device subscribes to the topic "department/+/employees", the policy will allow them to receive messages from the topic "department/engineering/employees". Similarly, once the device subscribes to the topic "location/#", they will receive messages from the topic "location/Seattle

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/
department/+/employees"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/location/
#"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Receive",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topic/*"
        }
    ]
}
```

Políticas para HTTP e WebSocket clientes

Ao se conectar via HTTP ou pelo WebSocket protocolo, você está se autenticando com o Signature versão 4 e o Amazon Cognito. As identidades do Amazon Cognito podem ser autenticadas ou não autenticadas. As identidades autenticadas pertencem a usuários que são autenticados por qualquer provedor de identidades. As identidades não autenticadas geralmente pertencem a usuários convidados que não são autenticados com um provedor de identidade. O Amazon Cognito fornece um identificador e AWS credenciais exclusivos para oferecer suporte a identidades não autenticadas. Para obter mais informações, consulte the section called “Autorização com identidades do Amazon Cognito”.

Para as seguintes operações, AWS IoT Core usa AWS IoT Core políticas anexadas às identidades do Amazon Cognito por meio da API. `AttachPolicy` Isso define as permissões anexadas ao banco de identidades do Amazon Cognito com identidades autenticadas.

- `iot:Connect`
- `iot:Publish`
- `iot:Subscribe`
- `iot:Receive`
- `iot:GetThingShadow`
- `iot:UpdateThingShadow`
- `iot:DeleteThingShadow`

A identidade do Amazon Cognito precisa da permissão da política de perfil do IAM e da política do AWS IoT Core . Você anexa a política de função do IAM ao pool e a AWS IoT Core política à Identidade do Amazon Cognito por meio da AWS IoT Core `AttachPolicy` API.

Usuários autenticados e não autenticados são tipos de identidade diferentes. Se você não anexar uma AWS IoT política à Identidade do Amazon Cognito, um usuário autenticado falhará na autorização AWS IoT e não terá acesso aos AWS IoT recursos e ações.

> **Note**
>
> Para outras AWS IoT Core operações ou para identidades não autenticadas, AWS IoT Core não define o escopo das permissões associadas à função do grupo de identidades do

Amazon Cognito. Para as identidades autenticadas e não autenticadas, esta é a política mais permissiva que recomendamos anexar ao perfil do banco do Amazon Cognito.

HTTP

Para permitir que as identidades do Amazon Cognito não autenticadas publiquem mensagens via HTTP em um tópico específico da identidade do Amazon Cognito, anexe a seguinte política do IAM ao perfil do banco de identidades do Amazon Cognito:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/${cognito-
identity.amazonaws.com:sub}"
            ]
        }
    ]
}
```

Para permitir usuários autenticados, anexe a política anterior à função do pool de Identidade do Amazon Cognito e à Identidade do Amazon Cognito usando a API. AWS IoT Core AttachPolicy

> **Note**
>
> Ao autorizar as identidades do Amazon Cognito AWS IoT Core , considere as duas políticas e conceda os menores privilégios especificados. Uma ação será permitida somente se as duas políticas permitirem a ação solicitada. Se uma das políticas não permitir uma ação, essa ação não será autorizada.

MQTT

Para permitir que identidades não autenticadas do Amazon Cognito publiquem mensagens MQTT WebSocket sobre um tópico específico da identidade do Amazon Cognito em sua conta, anexe a seguinte política do IAM à função do pool de identidade do Amazon Cognito:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${cognito-
identity.amazonaws.com:sub}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/${cognito-
identity.amazonaws.com:sub}"]
        }
    ]
}
```

Para permitir usuários autenticados, anexe a política anterior à função do pool de Identidade do Amazon Cognito e à Identidade do Amazon Cognito usando a API. AWS IoT Core AttachPolicy

> **Note**
>
> Ao autorizar as identidades do Amazon Cognito AWS IoT Core , considere ambas e conceda os menores privilégios especificados. Uma ação será permitida somente se as duas políticas permitirem a ação solicitada. Se uma das políticas não permitir uma ação, essa ação não será autorizada.

Exemplos de política para conectar e publicar

Para dispositivos registrados como itens no AWS IoT Core registro, a política a seguir concede permissão para se conectar AWS IoT Core com um ID de cliente que corresponda ao nome do item e restringe o dispositivo à publicação em um tópico MQTT específico do ID do cliente ou do nome do item. Para que uma conexão seja bem-sucedida, o nome da coisa deve ser registrado no AWS IoT Core registro e autenticado usando uma identidade ou principal anexada à coisa:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/${iot:Connection.Thing.ThingName}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
      ]
    }
  ]
}
```

Para dispositivos não registrados como itens no AWS IoT Core registro, a política a seguir concede permissão para se conectar AWS IoT Core com o ID do cliente `client1` e restringe o dispositivo à publicação em um tópico MQTT específico do cliente:

```
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/${iot:ClientId}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/client1"
      ]
    }
  ]
}
```

Exemplos de políticas de mensagens retidas

O uso de mensagens retidas requer políticas específicas. As mensagens retidas são mensagens MQTT publicadas com o sinalizador RETAIN definido e armazenadas por. AWS IoT Core Esta seção apresenta exemplos de políticas que permitem o uso comum de mensagens retidas.

Nesta seção:

- Política para conectar e publicar mensagens retidas
- Política para conectar e publicar mensagens Will retidas
- Política para listar e receber mensagens retidas

Política para conectar e publicar mensagens retidas

Para que um dispositivo publique mensagens retidas, ele deve ser capaz de se conectar, publicar (qualquer mensagem MQTT) e publicar mensagens retidas em MQTT. A política a seguir concede essas permissões para o tópico: `device/sample/configuration` ao cliente **`device1`**. Para ver outro exemplo que concede permissão para conexão, consulte the section called “Exemplos de política para conectar e publicar”.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/device1"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish",
    "iot:RetainPublish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/device/sample/configuration"
   ]
  }
 ]
}
```

Política para conectar e publicar mensagens Will retidas

Os clientes podem configurar uma mensagem que AWS IoT Core será publicada quando o cliente se desconectar inesperadamente. O MQTT chama essa mensagem de mensagem Will. Um cliente deve ter uma condição adicional adicionada à sua permissão de conexão para incluí-la.

O documento de política a seguir concede a todos os clientes permissão para se conectar e publicar uma mensagem Will, identificada por seu tópico, `will`, que o AWS IoT Core também reterá.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
```

```
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/device1"
   ],
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "LastWill"
     ]
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish",
    "iot:RetainPublish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/will"
   ]
  }
 ]
}
```

Política para listar e receber mensagens retidas

Serviços e aplicativos podem acessar mensagens retidas sem a necessidade de oferecer suporte a um cliente MQTT chamando `ListRetainedMessages` e `GetRetainedMessage`. Os serviços e aplicativos que chamam essas ações devem ser autorizados usando uma política como o exemplo a seguir.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:ListRetainedMessages"
            ],
```

```
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/device1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:GetRetainedMessage"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/foo"
            ]
        }
    ]
}
```

Exemplos de política de certificado

Para dispositivos registrados no AWS IoT Core registro, a política a seguir concede permissão para se conectar AWS IoT Core com um ID de cliente que corresponda ao nome de um item e publicar em um tópico cujo nome seja igual ao `certificateId` do certificado que o dispositivo usou para se autenticar:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:CertificateId}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
```

```
        }
    ]
}
```

Para dispositivos não registrados no AWS IoT Core registro, a política a seguir concede permissão para se conectar AWS IoT Core com o cliente IDs `client1client2`,,, `client3` e publicar em um tópico cujo nome é igual ao `certificateId` do certificado que o dispositivo usou para se autenticar:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:CertificateId}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        }
    ]
}
```

Para dispositivos registrados no AWS IoT Core registro, a política a seguir concede permissão para se conectar AWS IoT Core com um ID de cliente que corresponda ao nome do item e publicar em um tópico cujo nome seja igual ao `CommonName` campo do assunto do certificado que o dispositivo usou para se autenticar:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:Certificate.Subject.CommonName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        }
    ]
}
```

> **Note**
>
> Neste exemplo, o nome comum do assunto do certificado é usado como o identificador do tópico, supondo-se que o nome comum do assunto seja exclusivo para cada certificado registrado. Se os certificados forem compartilhados entre vários dispositivos, o nome comum do assunto será o mesmo para todos os dispositivos que compartilham esse certificado, permitindo assim privilégios de publicação para o mesmo tópico em vários dispositivos (não recomendado).

Para dispositivos não registrados no AWS IoT Core registro, a política a seguir concede permissão para se conectar AWS IoT Core com o cliente IDs `client1client2`,,, `client3` e publicar em um tópico cujo nome é igual ao `CommonName` campo do assunto do certificado que o dispositivo usou para se autenticar:

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:Certificate.Subject.CommonName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        }
    ]
}
```

> **Note**
>
> Neste exemplo, o nome comum do assunto do certificado é usado como o identificador do tópico, supondo-se que o nome comum do assunto seja exclusivo para cada certificado registrado. Se os certificados forem compartilhados entre vários dispositivos, o nome comum do assunto será o mesmo para todos os dispositivos que compartilham esse certificado, permitindo assim privilégios de publicação para o mesmo tópico em vários dispositivos (não recomendado).

Para dispositivos registrados no AWS IoT Core registro, a política a seguir concede permissão para se conectar AWS IoT Core com um ID de cliente que corresponda ao nome do item e publicar em um tópico cujo nome é prefixado `admin/` quando o certificado usado para autenticar o dispositivo tem seu `Subject.CommonName.2` campo definido como: `Administrator`

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/*"],
            "Condition": {
                "StringEquals": {
                    "iot:Certificate.Subject.CommonName.2": "Administrator"
            }
        }
        }
    ]
}
```

Para dispositivos não registrados no AWS IoT Core registro, a política a seguir concede permissão para se conectar AWS IoT Core com o cliente IDs `client1` `client3` e publicar em um tópico cujo nome é prefixado com `admin/` quando o certificado usado para autenticar o dispositivo tem seu `Subject.CommonName.2` campo definido como: `client2` `Administrator`

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
```

```
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/*"],
            "Condition": {
                "StringEquals": {
                    "iot:Certificate.Subject.CommonName.2": "Administrator"
                }
        }
        }
    ]
}
```

Para dispositivos registrados no AWS IoT Core registro, a política a seguir permite que um dispositivo use seu nome para publicar em um tópico específico, `admin/` seguido pelo `ThingName` quando o certificado usado para autenticar o dispositivo tem qualquer um de seus `Subject.CommonName` campos definido como: `Administrator`

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
```

```
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/
${iot:Connection.Thing.ThingName}"],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:Certificate.Subject.CommonName.List": "Administrator"
            }
        }
        }
    ]
}
```

Para dispositivos não registrados no AWS IoT Core registro, a política a seguir concede permissão para AWS IoT Core se conectar ao cliente IDs `client1` `client3` e publicar no tópico `admin` quando o certificado usado para autenticar o dispositivo tem qualquer um de seus `Subject.CommonName` campos definido como: `client2` `Administrator`

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin"],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:Certificate.Subject.CommonName.List": "Administrator"
            }
        }
```

```
        }
    ]
}
```

Exemplos de política de objetos

A política a seguir permite que um dispositivo se conecte se o certificado usado para autenticação AWS IoT Core estiver anexado à coisa para a qual a política está sendo avaliada:

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Effect":"Allow",
            "Action":["iot:Connect"],
            "Resource":[ "*" ],
            "Condition": {
                "Bool": {
                    "iot:Connection.Thing.IsAttached": ["true"]
            }
        }
        }
    ]
}
```

A política a seguir permite que um dispositivo publique se o certificado estiver anexado a um objeto com um determinado tipo e se o objeto tiver um atributo de `attributeName` com valor `attributeValue`. Para obter mais informações sobre variáveis da política de objeto, consulte Variáveis da política de objeto.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:topic/device/stats",
```

```
      "Condition": {
        "StringEquals": {
          "iot:Connection.Thing.Attributes[attributeName]": "attributeValue",
          "iot:Connection.Thing.ThingTypeName": "Thing_Type_Name"
        },
        "Bool": {
          "iot:Connection.Thing.IsAttached": "true"
        }
      }
    }
  ]
}
```

A política a seguir permite que um dispositivo publique em um tópico que começa com um atributo do objeto. Se o certificado do dispositivo não estiver associado ao objeto, essa variável não será resolvida e resultará em um erro de acesso negado. Para obter mais informações sobre variáveis da política de objeto, consulte Variáveis da política de objeto.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.Attributes[attributeName]}/*"
    }
  ]
}
```

Exemplo básico de políticas de trabalho

Este exemplo mostra as instruções de política necessárias para que um destino de trabalho que é um único dispositivo receba uma solicitação de trabalho e comunique o status de execução do trabalho com AWS IoT.

*us-west-2:57EXAMPLE833*Substitua por seu Região da AWS, um caractere de dois pontos (:) e seu Conta da AWS número de 12 dígitos e, em seguida, *uniqueThingName* substitua pelo nome do recurso que representa o dispositivo em. AWS IoT

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:client/uniqueThingName"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/pubtopic",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/events/job/*",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/events/jobExecution/
*",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topicfilter/test/dc/subtopic",
                "arn:aws:iot:us-west-2:123456789012:topicfilter/$aws/events/
jobExecution/*",
                "arn:aws:iot:us-west-2:123456789012:topicfilter/$aws/
things/uniqueThingName/jobs/*"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/test/dc/subtopic",
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName/jobs/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iotjobsdata:DescribeJobExecution",
                "iotjobsdata:GetPendingJobExecutions",
                "iotjobsdata:StartNextPendingJobExecution",
                "iotjobsdata:UpdateJobExecution"
            ],
            "Resource": [
                "arn:aws:iot:us-west-2:123456789012:topic/$aws/
things/uniqueThingName"
            ]
        }
    ]
}
```

Autorização com identidades do Amazon Cognito

Existem dois tipos de identidades do Amazon Cognito: autenticadas e não autenticadas. Se o aplicativo for compatível com identidades do Amazon Cognito não autenticadas, nenhuma autenticação será executada, para que você não saiba quem é o usuário.

Identidades não autenticadas: para identidades não autenticadas do Amazon Cognito, é possível conceder permissões anexando um perfil do IAM a um banco de identidades não autenticadas. Recomendamos conceder acesso somente aos recursos que você deseja disponibilizar para usuários desconhecidos.

> **Important**
>
> Para usuários não autenticados do Amazon Cognito que se conectam AWS IoT Core ao, recomendamos que você conceda acesso a recursos muito limitados nas políticas do IAM.

Identidades autenticadas: para identidades autenticadas do Amazon Cognito, é necessário especificar as permissões em dois lugares:

- Anexar uma política do IAM ao banco de identidades autenticadas do Amazon Cognito e
- Anexe uma AWS IoT Core política à Identidade do Amazon Cognito (usuário autenticado).

Exemplos de políticas para usuários não autenticados e autenticados do Amazon Cognito que se conectam ao AWS IoT Core

O exemplo a seguir mostra as permissões na política do IAM e na política de IoT de uma identidade do Amazon Cognito. O usuário autenticado deseja publicar em um tópico específico do dispositivo (por exemplodevice/DEVICE_ID/status).

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/Client_ID"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/device/Device_ID/status"
```

```
            ]
        }
    ]
}
```

O exemplo a seguir mostra as permissões em uma política do IAM em um perfil não autenticado do Amazon Cognito. O usuário não autenticado deve publicar em tópicos não específicos do dispositivo que não exijam autenticação.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/non_device_specific_topic"
            ]
        }
    ]
}
```

GitHub exemplos

Os exemplos de aplicativos da web a seguir GitHub mostram como incorporar a anexação de políticas a usuários autenticados no processo de inscrição e autenticação do usuário.

- Uso do aplicativo web MQTT publish/subscribe React AWS Amplify e o AWS IoT Device SDK for JavaScript

- Aplicativo web MQTT publish/subscribe React usando AWS AmplifyAWS IoT Device SDK for JavaScript, a e uma função Lambda

Amplify é um conjunto de ferramentas e serviços que ajuda você a criar aplicativos web e móveis que se integram aos AWS serviços. Para obter mais informações sobre o Amplify, consulte a Documentação do Amplify Framework.

Os dois exemplos executam as seguintes etapas.

1. Quando um usuário se inscreve em uma conta, o aplicativo cria um grupo de usuários e uma identidade do Amazon Cognito.
2. Quando um usuário se autentica, o aplicativo cria e anexa uma política à identidade. Isso dá ao usuário permissões de publicação e assinatura.
3. O usuário pode usar o aplicativo para publicar e assinar tópicos do MQTT.

O primeiro exemplo usa a operação de API `AttachPolicy` diretamente dentro da operação de autenticação. O exemplo a seguir demonstra como implementar essa chamada de API em um aplicativo web React que usa Amplify e o AWS IoT Device SDK for JavaScript.

```
function attachPolicy(id, policyName) {
    var Iot = new AWS.Iot({region: AWSConfiguration.region, apiVersion:
 AWSConfiguration.apiVersion, endpoint: AWSConfiguration.endpoint});
    var params = {policyName: policyName, target: id};

    console.log("Attach IoT Policy: " + policyName + " with cognito identity id: " +
 id);
    Iot.attachPolicy(params, function(err, data) {
         if (err) {
               if (err.code !== 'ResourceAlreadyExistsException') {
                  console.log(err);
               }
          }
         else  {
            console.log("Successfully attached policy with the identity", data);
         }
     });
}
```

Esse código aparece no AuthDisplayarquivo.js.

O segundo exemplo implementa a operação da API `AttachPolicy` em uma função do Lambda. O exemplo a seguir mostra como o Lambda usa essa chamada de API.

```
iot.attachPolicy(params, function(err, data) {
     if (err) {
           if (err.code !== 'ResourceAlreadyExistsException') {
              console.log(err);
              res.json({error: err, url: req.url, body: req.body});
           }
      }
     else  {
        console.log(data);
        res.json({success: 'Create and attach policy call succeed!', url: req.url,
 body: req.body});
     }
 });
```

Esse código aparece dentro da função `iot.GetPolicy` no arquivo app.js.

> **Note**
>
> Quando você chama a função com AWS as credenciais obtidas por meio dos pools de identidade do Amazon Cognito, o objeto de contexto na sua função Lambda contém um valor para. `context.cognito_identity_id` Para obter mais informações, consulte.
>
> - AWS Lambda objeto de contexto em Node.js
> - AWS Lambda objeto de contexto em Python
> - AWS Lambda objeto de contexto em Ruby
> - AWS Lambda objeto de contexto em Java
> - AWS Lambda objeto de contexto em Go
> - AWS Lambda objeto de contexto em C#
> - AWS Lambda objeto de contexto em PowerShell

Autorizando chamadas diretas para AWS serviços usando o provedor de AWS IoT Core credenciais

Os dispositivos podem usar certificados X.509 para se conectar AWS IoT Core usando protocolos de autenticação mútua TLS. Outros AWS serviços não oferecem suporte à autenticação baseada em certificado, mas podem ser chamados usando AWS credenciais no formato AWS Signature Version 4. O algoritmo Signature Version 4 normalmente exige que o chamador tenha um ID de chave de acesso e uma chave de acesso secreta. AWS IoT Core tem um provedor de credenciais que permite que você use o certificado X.509 incorporado como a identidade exclusiva do dispositivo para autenticar solicitações. AWS Isso elimina a necessidade de armazenar um ID de chave de acesso e uma chave de acesso secreta em seu dispositivo.

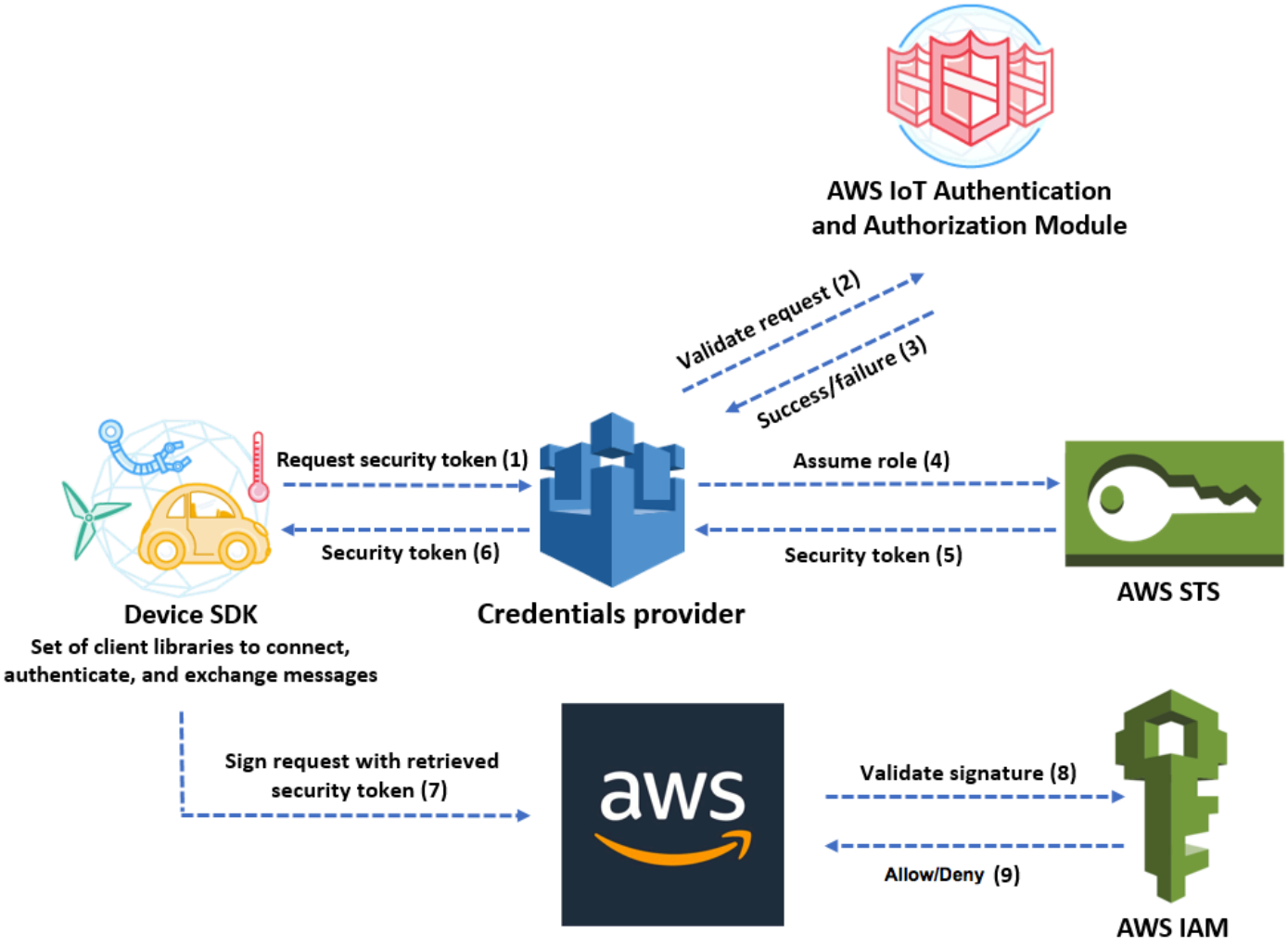
O provedor de credenciais autentica um chamador usando um certificado X.509 e emite um token de segurança temporário e de privilégio limitado. O token pode ser usado para assinar e autenticar qualquer AWS solicitação. Essa forma de autenticar suas AWS solicitações exige que você crie e configure uma função AWS Identity and Access Management (IAM) e anexe políticas apropriadas do IAM à função para que o provedor de credenciais possa assumir a função em seu nome. Para obter mais informações sobre AWS IoT Core e o IAM, consulte Gerenciamento de identidade e acesso para AWS IoT.

AWS IoT exige que os dispositivos enviem a extensão Server Name Indication (SNI) para o protocolo Transport Layer Security (TLS) e forneçam o endereço completo do endpoint no campo. `host_name` O campo `host_name` deve conter o endpoint que você está chamando e deve ser:

- O `endpointAddress` retornado por `aws iot describe-endpoint --endpoint-type iot:CredentialProvider`.

As conexões tentadas por dispositivos sem o valor `host_name` correto não funcionarão.

O diagrama a seguir mostra o fluxo de trabalho do provedor de credenciais.



1. O AWS IoT Core dispositivo faz uma solicitação HTTPS ao provedor de credenciais para obter um token de segurança. A solicitação inclui o dispositivo certificado X.509 para autenticação.
2. O provedor de credenciais encaminha a solicitação ao módulo de AWS IoT Core autenticação e autorização para validar o certificado e verificar se o dispositivo tem permissão para solicitar o token de segurança.
3. Se o certificado for válido e tiver permissão para solicitar um token de segurança, o módulo de AWS IoT Core autenticação e autorização retornará sucesso. Do contrário, envia uma exceção ao dispositivo.
4. Depois de validar o certificado, o provedor de credenciais chama o AWS Security Token Service (AWS STS) para assumir o perfil do IAM que você criou para ele.
5. AWS STS retorna um token de segurança temporário com privilégios limitados para o provedor de credenciais.
6. O provedor de credenciais retorna o token de segurança para o dispositivo.

7. O dispositivo usa o token de segurança para assinar uma AWS solicitação com o AWS Signature Version 4.
8. O serviço solicitado chama o IAM para validar a assinatura e autorizar a solicitação comparativamente às políticas de acesso anexadas à perfil do IAM que você criou para o provedor de credenciais.
9. Se o IAM validar a assinatura e autorizar a solicitação, a solicitação será concluída com êxito. Do contrário, o IAM envia uma exceção.

A seção a seguir descreve como se deve usar um certificado para obter um token de segurança. Ele é escrito com a suposição de que você já tenha registrado um dispositivo e criado e ativado seu próprio certificado para ele.

Como usar um certificado para obter um token de segurança

1. Configure o perfil do IAM que o provedor de credenciais assume em nome de seu dispositivo. Anexe à função as políticas de confiança a seguir.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": {"Service": "credentials.iot.amazonaws.com"},
        "Action": "sts:AssumeRole"
    }
}
```

   Para cada AWS serviço que você deseja chamar, anexe uma política de acesso à função. O provedor de credenciais comporta as seguintes variáveis de política:

   - `credentials-iot:ThingName`
   - `credentials-iot:ThingTypeName`
   - `credentials-iot:AwsCertificateId`

   Quando o dispositivo fornece o nome do objeto em sua solicitação a um serviço da AWS , o provedor de credenciais adiciona `credentials-iot:ThingName` e `credentials-iot:ThingTypeName` como variáveis de contexto ao token de segurança. O provedor de

credenciais fornece `credentials-iot:AwsCertificateId` como uma variável de contexto, mesmo que o dispositivo não forneça o nome do objeto na solicitação. Você passa o nome do objeto como o valor do cabeçalho da solicitação HTTP `x-amzn-iot-thingname`.

Essas três variáveis funcionam apenas para políticas do IAM, e não para políticas da AWS IoT Core .

2. O usuário que executar a próxima etapa (criar um alias de função) deve ter permissão para transmitir a função recém-criada ao AWS IoT Core. A política a seguir concede ambas `iam:GetRole` e `iam:PassRole` permissões a um AWS usuário. A permissão `iam:GetRole` permite que o usuário obtenha informações sobre a função que você acabou de criar. A `iam:PassRole` permissão permite que o usuário passe a função para outro AWS serviço.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:GetRole",
            "iam:PassRole"
        ],
        "Resource": "arn:aws:iam::123456789012:role/your role name"
    }
}
```

3. Crie um alias de AWS IoT Core função. O dispositivo que fará chamadas diretas para AWS os serviços deve saber a qual função ARN usar ao se conectar. AWS IoT Core Fixar o ARN da função no código não é uma boa solução porque isso requer que você atualize o dispositivo toda vez que o ARN da função for alterado. Uma solução mais adequada é usar a API `CreateRoleAlias` para criar um alias de função que aponte para o ARN da função. Se o ARN da função for alterado, basta atualizar o alias da função. Nenhuma alteração é necessária no dispositivo. Essa API usa os seguintes parâmetros:

   `roleAlias`

   Obrigatório. Uma string arbitrária que identifica o alias da função. Funciona como chave primária no modelo de dados do alias da função. Contém de 1 a 128 caracteres e deve incluir apenas caracteres alfanuméricos e os símbolos =, @ e -. Caracteres alfabéticos em maiúsculas e minúsculas são permitidos. Os nomes de alias de perfil diferenciam maiúsculas de minúsculas.

`roleArn`

Obrigatório. O ARN da função ao qual o alias da função se refere.

`credentialDurationSeconds`

Opcional. Por quanto tempo (em segundos) a credencial é válida. O valor mínimo é 900 segundos (15 minutos). O valor máximo é de 43.200 segundos (12 horas). O valor padrão é de 3.600 segundos (uma hora).

> **Important**
>
> O provedor de AWS IoT Core credenciais pode emitir uma credencial com uma vida útil máxima de 43.200 segundos (12 horas). Ter a credencial válida por até 12 horas pode ajudar a reduzir o número de chamadas para o provedor de credenciais, armazenando a credencial em cache por mais tempo.
> O valor `credentialDurationSeconds` deve ser menor que ou igual à duração máxima da sessão do perfil do IAM à qual o alias do perfil faz referência. Para obter mais informações, consulte Modificar a duração máxima da sessão (AWS API) de uma função no Guia do usuário do AWS Identity and Access Management.

Para obter mais informações sobre essa API, consulte CreateRoleAlias.

4. Anexe uma política a um certificado de dispositivo. A política anexada ao certificado do dispositivo deve conceder permissão ao dispositivo para assumir a função. Para isso, conceda permissão para a ação `iot:AssumeRoleWithCertificate` do alias da função, como no exemplo a seguir.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:AssumeRoleWithCertificate",
            "Resource": "arn:aws:iot:us-east-1:123456789012:rolealias/your role
 alias"
        }
    ]
}
```

5. Faça uma solicitação HTTPS ao provedor de credenciais para receber um token de segurança. Forneça as informações a seguir:

   - Certificado: como se trata de uma solicitação HTTP por meio de autenticação TLS mútua, é necessário fornecer o certificado e a chave privada para o cliente quando estiver fazendo a solicitação. Use o mesmo certificado e chave privada que você usou quando registrou seu certificado AWS IoT Core.

     Para garantir que seu dispositivo esteja se comunicando com AWS IoT Core (e não um serviço que se faça passar por ele), consulte Autenticação do servidor, siga os links para baixar os certificados CA apropriados e copie-os para o seu dispositivo.
   - RoleAlias: o nome do alias de função que você criou para o provedor de credenciais. Os nomes de alias de função diferenciam maiúsculas de minúsculas e devem corresponder ao alias de função criado em. AWS IoT Core
   - ThingName: O nome da coisa que você criou quando registrou sua AWS IoT Core coisa. Ele é passado como o valor do cabeçalho HTTP `x-amzn-iot-thingname`. Esse valor é necessário somente se você estiver usando atributos de coisas como variáveis de política AWS IoT Core ou políticas do IAM.

     > **Note**
     >
     > O ThingNameque você fornece `x-amzn-iot-thingname` deve corresponder ao nome do recurso AWS IoT Thing atribuído a um certificado. Se não corresponder, um erro 403 será retornado.

   Execute o comando a seguir no AWS CLI para obter o endpoint do provedor de credenciais para o seu. Conta da AWS Para obter mais informações sobre essa API, consulte DescribeEndpoint. Para endpoints habilitados para FIPS, consulte AWS IoT Core: endpoints do provedor de credenciais.

   ```
   aws iot describe-endpoint --endpoint-type iot:CredentialProvider
   ```

   O objeto JSON a seguir é uma saída de exemplo do comando describe-endpoint. Ele contém o `endpointAddress` que você usa para solicitar um token de segurança.

   ```
   {
   ```

```
    "endpointAddress": "your_aws_account_specific_prefix.credentials.iot.your
 region.amazonaws.com"
}
```

Use o endpoint para fazer uma solicitação HTTPS ao provedor de credenciais para que retorne um token de segurança. O comando de exemplo a seguir usa `curl`, mas é possível usar qualquer cliente HTTP.

> **Note**
>
> O nome RoleAlias diferencia maiúsculas de minúsculas e deve corresponder ao alias de função criado em. AWS IoT

```
curl --cert your certificate --key your private key -H "x-amzn-iot-thingname: your
 thing name" --cacert AmazonRootCA1.pem https://your endpoint /role-aliases/your
 role alias/credentials
```

Esse comando retorna um objeto de token de segurança que contém um `accessKeyId`, uma `secretAccessKey`, um `sessionToken` e uma expiração. O objeto JSON a seguir é uma saída de exemplo do comando `curl`.

```
   {"credentials":{"accessKeyId":"access key","secretAccessKey":"secret access
 key","sessionToken":"session token","expiration":"2018-01-18T09:18:06Z"}}
```

Em seguida, você pode usar os `sessionToken` valores `accessKeyIdsecretAccessKey`, e para assinar solicitações aos AWS serviços. Para ver uma end-to-end demonstração, consulte Como eliminar a necessidade de AWS credenciais codificadas em dispositivos usando o AWS IoT Credential Provider no blog Security Blog.AWS

Acesso entre contas com o IAM

AWS IoT Core permite que você permita que um diretor publique ou assine um tópico definido como Conta da AWS não pertencente ao diretor. É possível configurar o acesso entre contas criando uma política e perfil do IAM e, em seguida, anexando a política ao perfil.

Primeiro, crie uma política do IAM gerenciada pelo cliente, conforme descrito em Como criar políticas do IAM, assim como você faria com outros usuários e certificados em sua Conta da AWS.

Para dispositivos registrados no AWS IoT Core registro, a política a seguir concede permissão para que os dispositivos se conectem AWS IoT Core usando uma ID de cliente que corresponda ao nome do item do dispositivo e publiquem no nome do item `my/topic/`*`thing-name`* onde *`thing-name`* está o dispositivo:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/my/topic/
${iot:Connection.Thing.ThingName}"
            ]
        }
    ]
}
```

Para dispositivos não registrados no AWS IoT Core registro, a política a seguir concede permissão a um dispositivo para usar o nome do item `client1` registrado no registro da sua conta (123456789012) para se conectar AWS IoT Core e publicar em um tópico específico AWS IoT Core do ID do cliente cujo nome esteja prefixado com: `my/topic/`

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/my/topic/${iot:ClientId}"
            ]
        }
    ]
}
```

Em seguida, siga as etapas em Criação de um perfil para delegar permissões a um usuário do IAM. Insira o ID da Conta da AWS com a qual você deseja compartilhar o acesso. Em seguida, na etapa final, anexe a política recém-criada à função. Se, posteriormente, for necessário modificar o ID de conta da AWS ao qual você está concedendo acesso, você poderá usar o formato de política de confiança a seguir para fazer isso:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::567890123456:user/MyUser"
        },
            "Action": "sts:AssumeRole"
        }
```

```
    ]
}
```

Proteção de dados em AWS IoT Core

O modelo de responsabilidade AWS compartilhada modelo se aplica à proteção de dados em AWS IoT Core. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre o conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para saber mais sobre a privacidade de dados, consulte as Data Privacy FAQ. Para saber mais sobre a proteção de dados na Europa, consulte a postagem do blog AWS Shared Responsibility Model and RGPD no Blog de segurança da AWS .

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com Centro de Identidade do AWS IAM ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com AWS os recursos. Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail. Para obter informações sobre o uso de CloudTrail trilhas para capturar AWS atividades, consulte Como trabalhar com CloudTrail trilhas no Guia AWS CloudTrail do usuário.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sensíveis armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-3 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para saber mais sobre os endpoints FIPS disponíveis, consulte Federal Information Processing Standard (FIPS) 140-3.

É altamente recomendável que nunca sejam colocadas informações confidenciais ou sensíveis, como endereços de e-mail de clientes, em tags ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com AWS IoT ou Serviços da AWS usa o console, a API ou AWS SDKs. AWS CLI Quaisquer dados inseridos em tags ou em campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é fortemente recomendável que não sejam incluídas informações de credenciais no URL para validar a solicitação nesse servidor.

Consulte mais informações sobre proteção de dados na publicação do blog AWS Shared Responsibility Model and GDPR no Blog de segurança da AWS .

AWS IoT os dispositivos coletam dados, realizam alguma manipulação nesses dados e, em seguida, enviam esses dados para outro serviço da web. É possível optar por armazenar alguns dados em seu dispositivo por um curto período. Você é responsável por fornecer a proteção de dados sobre esses dados em repouso. Quando seu dispositivo envia dados para AWS IoT, ele o faz por meio de uma conexão TLS, conforme discutido posteriormente nesta seção. AWS IoT os dispositivos podem enviar dados para qualquer AWS serviço. Para obter mais informações sobre a segurança de dados de cada serviço, consulte a documentação desse serviço. AWS IoT pode ser configurado para gravar registros em CloudWatch Logs e registrar chamadas de AWS IoT API em AWS CloudTrail. Para obter mais informações sobre a segurança de dados desses serviços, consulte Autenticação e controle de acesso para a Amazon CloudWatch e Criptografia de arquivos de CloudTrail log com chaves AWS KMS gerenciadas.

Criptografia de dados em AWS IoT

Por padrão, todos os AWS IoT dados em trânsito e em repouso são criptografados. Os dados em trânsito são criptografados usando TLS e os dados em repouso são criptografados usando chaves AWS próprias. AWS IoT suporta gerenciamento de clientes AWS KMS keys (chaves KMS) a partir do AWS Key Management Service (AWS KMS). No entanto, o Device Advisor e o AWS IoT Wireless usam somente um Chave pertencente à AWS para criptografar os dados do cliente.

Segurança de transporte em AWS IoT Core

O Transport Layer Security (TLS) é um protocolo criptográfico projetado para comunicação segura em uma rede de computadores. O AWS IoT Core Device Gateway exige que os clientes criptografem toda a comunicação em trânsito usando TLS para conexões de dispositivos ao Gateway. O TLS é usado para obter a confidencialidade dos protocolos de aplicativos (MQTT, HTTP e WebSocket)

suportados pelo. AWS IoT Core O suporte ao TLS está disponível em várias de linguagens de programação e sistemas operacionais. Os dados AWS internos são criptografados pelo AWS serviço específico. Para obter mais informações sobre criptografia de dados em outros AWS serviços, consulte a documentação de segurança desse serviço.

Conteúdo

- Protocolos TLS
- Políticas de segurança
- Notas importantes para a segurança do transporte em AWS IoT Core
- Segurança de transporte para dispositivos sem fio LoRa WAN

Protocolos TLS

AWS IoT Core suporta as seguintes versões do protocolo TLS:

- TLS 1.3
- TLS 1.2

Com AWS IoT Core, você pode definir as configurações de TLS (para TLS 1.2 e TLS 1.3) nas configurações de domínio. Para obter mais informações, consulte ???.

Políticas de segurança

Uma política de segurança é uma combinação de protocolos TLS e suas cifras que determinam quais protocolos e cifras são compatíveis durante as negociações de TLS entre um cliente e um servidor. Você pode configurar seus dispositivos para usar políticas de segurança predefinidas com base em suas necessidades. Observe que isso AWS IoT Core não oferece suporte a políticas de segurança personalizadas.

Você pode escolher uma das políticas de segurança predefinidas para seus dispositivos ao conectá-los a. AWS IoT Core Os nomes das políticas de segurança predefinidas mais recentes AWS IoT Core incluem informações de versão com base no ano e mês em que foram lançadas. A política de segurança padrão predefinida é `IoTSecurityPolicy_TLS13_1_2_2022_10`. Para especificar uma política de segurança, você pode usar o AWS IoT console ou AWS CLI o. Para obter mais informações, consulte ???.

A tabela a seguir descreve as políticas de segurança predefinidas mais recentes compatíveis com o AWS IoT Core . O `IotSecurityPolicy_` foi removido dos nomes de política na linha de cabeçalho para que se ajustem ao espaço.

Política de segurança	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
Porta TCP	443/8443/8883	443/8443/8883	443/8443/8883	443	8443/8883	443	8443/8883
Protocolos TLS							
TLS 1.2		✓	✓	✓	✓	✓	✓
TLS 1.3	✓	✓					
Cifras TLS							
TLS_AES_128_GCM_SHA256	✓	✓					
TLS_AES_256_GCM_SHA384	✓	✓					
TLS_CHACHA20_POLY1305_SHA256	✓	✓					
ECDHE-RSA-AES128-GCM-SHA256		✓	✓	✓	✓	✓	✓

Política de segurança	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
ECDHE-RSA- -AES128 SHA256		✓	✓	✓	✓	✓	✓
ECDHE-RSA- -SHA AES128		✓	✓	✓	✓	✓	✓
ECDHE-RSA- -GCM-AES256 SHA384		✓	✓	✓	✓	✓	✓
ECDHE-RSA- -AES256 SHA384		✓	✓	✓	✓	✓	✓
ECDHE-RSA- -SHA AES256		✓	✓	✓	✓	✓	✓
AES128-GCM-SHA256		✓	✓	✓	✓	✓	✓
AES128-SHA256		✓	✓	✓		✓	✓

Política de segurança	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
AES128-SHAH		✓	✓	✓	✓	✓	✓
AES256-GCM-SHA384		✓	✓	✓	✓	✓	✓
AES256-SHA256		✓	✓	✓	✓	✓	✓
AES256-SHAH		✓	✓	✓	✓	✓	✓
DHE-RSA--SHA AES256						✓	✓
ECDHE-ECDSA--GCM-AES128 SHA256		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-- AES128 SHA256		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA--SHA AES128		✓	✓	✓	✓	✓	✓

Política de segurança	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
ECDHE-ECDSA--GCM-AES256 SHA384		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-- AES256 SHA384		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA--SHA AES256		✓	✓	✓	✓	✓	✓

> **Note**
>
> `TLS12_1_0_2016_01`só está disponível no seguinte Regiões da AWS: ap-east-1, ap-northeast-2, ap-south-1, ap-south-1, ap-southeast-2, ca-central-1, cn-north-1, cn-northwest-1, eu-north-1, eu-north-1, eu-west-2, eu-west-3, me-south-1, sa-lest-1, us-east-1, us-east-2, -1, -2, us-west-1. us-gov-west us-gov-west
> `TLS12_1_0_2015_01`só está disponível no seguinte Regiões da AWS: ap-northeast-1, ap-southeast-1, eu-central-1, eu-central-1, eu-west-1, us-east-1, us-west-2.

Notas importantes para a segurança do transporte em AWS IoT Core

Para dispositivos que se conectam AWS IoT Core usando o MQTT, o TLS criptografa a conexão entre os dispositivos e o agente e AWS IoT Core usa a autenticação do cliente TLS para identificar dispositivos. Para obter mais informações, consulte Autenticação do cliente. Para dispositivos que se conectam AWS IoT Core usando HTTP, o TLS criptografa a conexão entre os dispositivos e o

agente, e a autenticação é delegada à AWS Signature Version 4. Para obter mais informações, consulte Assinatura de solicitações com o SignatAre Versão 4 na Referência geral da AWS .

Quando você conecta dispositivos ao AWS IoT Core, o envio da extensão de Indicação de Nome do Servidor (SNI) não é obrigatório, mas é altamente recomendado. Para usar recursos, como registro de várias contas, domínios personalizados, endpoints da VPC e políticas de TLS configuradas, é necessário usar a extensão SNI e fornecer o endereço completo do endpoint no campo `host_name`. O campo `host_name` deve conter o endpoint que você está chamando. Esse endpoint deve ser um dos seguintes:

- O `endpointAddress` retornado por `aws iot describe-endpoint --endpoint-type iot:Data-ATS`
- O `domainName` retornado por `aws iot describe-domain-configuration --domain-configuration-name "domain_configuration_name"`

As conexões tentadas por dispositivos com o `host_name` valor incorreto ou inválido falharão. AWS IoT Core registrará falhas no tipo de autenticação personalizada. CloudWatch

AWS IoT Core não suporta a extensão SessionTicket TLS.

Segurança de transporte para dispositivos sem fio LoRa WAN

LoRaOs dispositivos WAN seguem as práticas de segurança descritas em LoRaWAN™ SECURITY: um white paper preparado para a LoRa Alliance™ pela Gemalto, Actility e Semtech.

Para obter mais informações sobre segurança de transporte com dispositivos LoRa WAN, consulte Dados de LoRa WAN e segurança de transporte.

Criptografia de dados em AWS IoT

A proteção de dados se refere à proteção de dados em trânsito (à medida que viajam de e para lá AWS IoT Core) e em repouso (enquanto são armazenados em dispositivos ou por outros AWS serviços). Todos os dados enviados AWS IoT Core são enviados por uma conexão TLS usando MQTT, HTTPS e WebSocket protocolos, tornando-os seguros por padrão durante o trânsito. AWS IoT Core coleta dados dos dispositivos e os envia para outros AWS serviços para processamento adicional. Para obter mais informações sobre criptografia de dados em outros serviços da AWS , consulte a documentação de segurança desse serviço. Para acessar mais informações, consulte Criptografia de dados em repouso.

O FreeRTOS fornece uma biblioteca do PKCS#11 que abstrai o armazenamento de chaves, acessando objetos criptográficos e gerenciando sessões. É sua responsabilidade usar essa biblioteca para criptografar dados em repouso em seus dispositivos. Para obter mais informações, consulte Biblioteca do padrão de criptografia de chave pública do FreeRTOS Nº 11 (PKCS).

Criptografia de dados em repouso em AWS IoT Core

Por padrão, todos os AWS IoT Core dados em repouso são criptografados usando chaves AWS próprias. AWS IoT Core também suporta chaves simétricas gerenciadas pelo cliente a partir de AWS Key Management Service (AWS KMS). Com as chaves gerenciadas pelo cliente, você pode criar, possuir e gerenciar as AWS KMS chaves em sua AWS conta. AWS IoT Core usará suas chaves KMS para criptografar seus dados em repouso. Você tem controle total sobre essas chaves do KMS, inclusive sobre a criação e a manutenção de suas políticas de chave. Você também pode configurar políticas do IAM para as funções que acessam AWS KMS para controlar as permissões dessas chaves.

AWS chaves de propriedade

AWS chaves próprias são uma coleção de chaves KMS que um AWS serviço possui e gerencia para uso em várias AWS contas. AWS os serviços podem usar chaves AWS próprias para proteger seus dados. Por padrão, AWS IoT Core criptografa dados em repouso usando chaves AWS próprias. Essas chaves são gerenciadas pelo serviço. Você não pode visualizar, gerenciar ou usar chaves AWS próprias. No entanto, não é necessário realizar nenhuma ação para proteger essas chaves.

Para obter mais informações sobre chaves AWS próprias, consulte chaves AWS próprias no Guia do AWS Key Management Service desenvolvedor.

Chaves gerenciadas pelo cliente

As chaves gerenciadas pelo cliente são chaves KMS em sua AWS conta que você cria, possui e gerencia. Você tem controle total sobre essas chaves do AWS KMS , inclusive sobre a criação e a manutenção de suas respectivas políticas. Você também pode configurar políticas do IAM para as funções que acessam AWS KMS para controlar as permissões dessas chaves. Você pode configurar AWS IoT Core para usar chaves KMS gerenciadas pelo cliente para criptografar seus dados.

Para obter mais informações sobre chaves gerenciadas pelo cliente, consulte Chaves gerenciadas pelo cliente no Guia do desenvolvedor do AWS Key Management Service .

Para ativar as chaves gerenciadas pelo cliente AWS IoT Core, siga estas etapas:

Tópicos

- Etapa 1: criar uma chave gerenciada pelo cliente
- Etapa 2: criar uma função do IAM para conceder AWS IoT Core permissões para usar a chave KMS
- Etapa 3: ativar as chaves gerenciadas pelo cliente no AWS IoT Core
- Etapa 4: Permissões adicionais necessárias para operações do plano de AWS IoT Core controle
- Etapa 5: gerenciar as chaves
- Etapa 6: monitorar a integridade das chaves

Etapa 1: criar uma chave gerenciada pelo cliente

Você pode criar uma chave simétrica gerenciada pelo cliente usando o AWS KMS console ou os comandos da AWS KMS CLI. A `keySpec` deve ser `SYMMETRIC_DEFAULT`, e o `keyUsage` deve ser `ENCRYPT_DECRYPT`.

> **Note**
>
> AWS IoT Core só suporta AWS KMS `SYMMETRIC_DEFAULT` chaves com especificação e uso de `ENCRYPT_DECRYPT` chaves para chaves gerenciadas pelo cliente.

Veja a seguir um exemplo de AWS CLI comando para criar uma chave KMS que pode ser usada com chaves gerenciadas AWS IoT Core pelo cliente.

```
aws kms create-key --key-spec SYMMETRIC_DEFAULT --key-usage ENCRYPT_DECRYPT --region
 us-west-2
```

O conteúdo a seguir é um exemplo de saída do comando.

```
{
    "KeyMetadata": {
        "AWSAccountId": "111122223333",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2024-09-19T11:45:23.982000-07:00",
        "Enabled": true,
        "Description": "",
        "KeyUsage": "ENCRYPT_DECRYPT",
```

```
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": false
    }
}
```

Para acessar mais informações, consulte Criar uma CMK simétrica no Guia do desenvolvedor do AWS Key Management Service .

Política de chave

Quando você cria uma chave gerenciada pelo cliente, pode especificar uma política de chave. As políticas de chaves controlam o acesso à chave gerenciada pelo cliente. Cada chave gerenciada pelo cliente deve ter exatamente uma política de chave, que contém declarações que determinam quem pode usar a chave e como pode usá-la. Para acessar mais informações, consulte Políticas de chave no Guia do desenvolvedor do AWS Key Management Service .

AWS IoT Core usa uma função do IAM em sua conta para acessar sua chave gerenciada pelo cliente. Se você estiver usando uma política de chave personalizada, verifique se o perfil do IAM criado nessa chave tem as seguintes permissões:

- `kms:DescribeKey`
- `kms:Decrypt`
- `kms:Encrypt`
- `kms:GenerateDataKeyWithoutPlaintext`
- `kms:ReEncryptTo`
- `kms:ReEncryptFrom`

Etapa 2: criar uma função do IAM para conceder AWS IoT Core permissões para usar a chave KMS

AWS IoT Core Para usar a chave KMS que você criou para criptografar seus dados em repouso, você também precisa criar uma função do IAM em sua conta, que AWS IoT Core pode assumir o acesso à chave KMS.

A função deve ter a seguinte política de confiança para AWS IoT Core permitir que você assuma a função.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": {
            "Service": "iot.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "111122223333"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:iot:us-west-2:111122223333:*"
            }
        }
    }
}
```

Verifique se as políticas do IAM associadas ao perfil do IAM têm as seguintes permissões na chave do KMS:

- `kms:DescribeKey`
- `kms:Decrypt`
- `kms:Encrypt`
- `kms:GenerateDataKeyWithoutPlaintext`
- `kms:ReEncryptTo`
- `kms:ReEncryptFrom`

Confira abaixo um exemplo de política do IAM com as permissões necessárias para as chaves gerenciadas pelo cliente.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "AllowIoTToAccessKMSResource",
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey",
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:ReEncryptTo",
                "kms:ReEncryptFrom",
                "kms:GenerateDataKeyWithoutPlaintext"
            ],
            "Resource": [
                "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:aws-crypto-ec:vendor": "iot.amazonaws.com"
                }
            }
        }
    ]
}
```

Para acessar mais informações, consulte Criar um perfil para delegar permissões a um usuário do IAM no Guia do usuário do AWS Identity and Access Management .

Etapa 3: ativar as chaves gerenciadas pelo cliente no AWS IoT Core

Depois de concluir todas as etapas anteriores, execute o comando `update-encryption-configuration` da CLI para usar as chaves gerenciadas pelo cliente no AWS IoT Core. Quando você opta por usar as chaves gerenciadas pelo cliente, todos os AWS IoT Core recursos AWS da sua conta serão criptografados usando a AWS KMS chave especificada.

1. Para ativar o AWS IoT Core uso de chaves gerenciadas pelo cliente AWS CLI, execute o comando `update-encryption-configuration` CLI.

```
aws iot update-encryption-configuration --encryption-type "CUSTOMER_MANAGED_KMS_KEY"
 \
--kms-access-role-arn "arn:aws:iam::111122223333:role/myrole" \
--kms-key-arn "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab" --region us-west-2
```

2. Para verificar se as chaves gerenciadas pelo cliente estão AWS CLI sendo usadas, execute o comando `describe-encryption-configuration` CLI: AWS IoT Core

```
aws iot describe-encryption-configuration --region us-west-2
```

Se você ativou as chaves gerenciadas pelo cliente AWS IoT Core, a saída pode ter a seguinte aparência:

```
{
    "encryptionType": "CUSTOMER_MANAGED_KMS_KEY",
    "kmsKeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "kmsAccessRoleArn": "arn:aws:iam::111122223333:role/myrole",
    "configurationDetails": {
        "configurationStatus": "HEALTHY"
    },
    "lastModifiedDate": "2024-09-26T22:01:02.365000-07:00"
}
```

O campo `lastModifiedDate` indica a data em que a configuração de criptografia foi atualizada pela última vez.

Se não tiver habilitado as chaves gerenciadas pelo cliente, a saída poderá ser parecida com o seguinte:

```
{
    "encryptionType": "AWS_OWNED_KMS_KEY",
    "lastModifiedDate": "2024-09-26T22:01:02.365000-07:00"
}
```

Etapa 4: Permissões adicionais necessárias para operações do plano de AWS IoT Core controle

Depois de optar pelas chaves gerenciadas pelo cliente, todos os AWS IoT Core recursos pertencentes à sua AWS conta são criptografados com a chave KMS fornecida. Todas as operações do plano de controle agora exigem que o chamador tenha `kms:Decrypt` permissões na chave KMS, além das permissões necessárias para a operação específica no AWS IoT Core recurso. Se o chamador não tiver a permissão `kms:Decrypt` e fizer uma chamada de API que exija criptografia ou descriptografia de dados (por exemplo, `GetPolicy`), ele receberá uma `UnauthorizedException`.

Por exemplo, ao chamar `GetPolicy`, você precisa das permissões `iot:GetPolicy` e `kms:Decrypt` e da chave do KMS gerenciada pelo cliente para que a chamada de API seja bem-sucedida.

> **Note**
>
> Ao atualizar os usuários ou funções do IAM para conceder AWS KMS permissões na chave usada para sua configuração de criptografia, certifique-se de que a política de chaves do KMS também conceda as permissões necessárias aos respectivos usuários ou funções do IAM.

AWS KMS permissões para **`UpdateEncryptionConfiguration`**

A chamada de `UpdateEncryptionConfiguration` API precisa das seguintes AWS KMS permissões na chave KMS para poder aceitar as chaves gerenciadas pelo cliente ou modificar a configuração da chave:

- `kms:DescribeKey`
- `kms:Decrypt`
- `kms:Encrypt`
- `kms:GenerateDataKeyWithoutPlaintext`
- `kms:ReEncryptTo`
- `kms:ReEncryptFrom`

AWS KMS permissões para todos os outros planos de controle APIs

A maioria dos planos de controle APIs exige `kms:Decrypt` permissões quando as chaves gerenciadas pelo cliente estão habilitadas. No entanto, alguns APIs não exigem essas permissões adicionais:

APIs que não exigem AWS KMS permissões

A `List*` e `Delete*` APIs não caia nesse balde. Os clientes sempre podem invocar qualquer API `List*` ou `Delete*` do ambiente de gerenciamento, e essas chamadas de API seriam bem-sucedidas mesmo que o chamador não tivesse a permissão `kms:Decrypt`. Essas chamadas de API serão bem-sucedidas mesmo que sua chave gerenciada pelo cliente não esteja íntegra `List*` e não `Delete*` APIs faça nenhuma decodificação.

- Lista* APIs — Todas as operações de listagem (por exemplo,`ListThings`,`ListPolicies`,`ListCertificates`)
- Excluir* APIs — Todas as operações de exclusão (por exemplo,,,) `DeleteThing` `DeletePolicy` `DeleteCertificate`

Etapa 5: gerenciar as chaves

AWS IoT Core executa verificações periódicas na configuração da chave gerenciada pelo cliente para garantir que as operações de criptografia e descriptografia não sejam afetadas. Essas verificações de saúde são executadas uma vez a cada minuto e verificam a capacidade AWS IoT Core da empresa de acessar e usar a AWS KMS chave e a função do IAM associada para operações de criptografia e descriptografia.

HEALTHY

AWS IoT Core pode acessar com êxito a AWS KMS chave por meio da função do IAM especificada e realizar encryption/decryption operações. Todos os componentes funcionam corretamente.

UNHEALTHY

AWS IoT Core não consigo acessar ou usar a AWS KMS chave. Isso evita novas operações de criptografia e pode afetar a funcionalidade do serviço. O campo `errorCode` indica se o problema está na chave ou no perfil do IAM.

Ações do cliente que podem afetar a integridade da chave

Várias ações do cliente podem fazer com que o status de integridade da chave mude de `HEALTHY` para `UNHEALTHY`:

Ações relacionadas a chaves

- Excluindo uma AWS KMS chave — Quando você agenda a exclusão de uma chave, ela está em um `Pending deletion` status e não pode ser usada
- Desabilitar uma AWS KMS chave — Quando você desativa uma chave KMS, ela não pode mais ser usada para operações de criptografar/descriptografar
- Agendar a exclusão de uma chave: a chave se torna inutilizável quando a exclusão é concluída.

- Modificando a política de chaves — Removendo as permissões necessárias para AWS IoT Core acesso
- Alterando as permissões de uso das chaves — Restringindo as ações necessárias AWS KMS

Ações relacionadas ao perfil do IAM

- Excluindo a função do IAM — não AWS IoT Core consigo assumir a função de acessar a chave
- Modificando as permissões da função — Removendo AWS KMS as permissões necessárias da política da função
- Mudando a política de confiança — impedindo que o AWS IoT Core serviço assuma a função
- Adicionar condições restritivas — Condições que AWS IoT Core impedem o uso da função

Ações em nível de conta

- Alterações de acesso das chaves entre contas: modificar permissões para chaves em contas diferentes.
- Políticas de controle de serviços (SCPs) — políticas em nível organizacional que restringem o acesso AWS KMS
- Políticas do IAM em nível de conta: políticas que substituem ou entram em conflito com o acesso da chave.

> **Important**
>
> Qualquer alteração nas AWS KMS chaves, funções do IAM ou políticas usadas pelo AWS IoT Core deve ser testada primeiro em ambientes de desenvolvimento. Monitore o status de integridade da chave de perto após fazer qualquer alteração para garantir que a AWS IoT Core funcionalidade não seja afetada.

Atualizar a configuração de criptografia

Atualize sua configuração de criptografia AWS IoT Core para mudar de uma chave gerenciada pelo cliente para outra, ou entre chaves AWS próprias e chaves gerenciadas pelo cliente.

Para alterar a configuração para uma chave gerenciada pelo cliente:

1. Crie uma chave gerenciada pelo cliente seguindo as etapas em the section called “Etapa 1: criar uma chave gerenciada pelo cliente”.

2. Atualize sua política de perfil do IAM para incluir permissões para as chaves antigas e novas durante o período de atualização.
3. Atualize sua configuração de criptografia para usar a nova chave:

```
aws iot update-encryption-configuration --encryption-type "CUSTOMER_MANAGED_KMS_KEY"
 \
--kms-access-role-arn "arn:aws:iam::111122223333:role/myrole" \
--kms-key-arn "arn:aws:kms:us-west-2:111122223333:key/new-key-id"
```

Para alterar a configuração das chaves gerenciadas pelo cliente para chaves AWS próprias:

```
aws iot update-encryption-configuration --encryption-type "AWS_OWNED_KMS_KEY"
```

> **Note**
>
> Ao atualizar a configuração de criptografia para novas chaves gerenciadas pelo cliente, garanta que as chaves antigas e novas permaneçam acessíveis para que a operação seja bem-sucedida.

Cenários e impactos comuns de falhas

A seguinte tabela descreve cenários de falha comuns quando as chaves são excluídas ou desativadas:

Cenário	Impacto imediato	Consequências no longo prazo
Chave desabilitada	Todas as novas encryption/decryption operações falham imediatamente	Interrupção do serviço até que a chave seja reabilitada ou substituída.
Chave agendada para exclusão	O status da chave é alterado para exclusão pendente e todas as encryption/decryption operações falharão	Falha automática do serviço quando a exclusão é concluída.

Cenário	Impacto imediato	Consequências no longo prazo
Chave excluída permanentemente.	Falha imediata e permanente de todas as operações.	Perda permanente de dados e incapacidade de recuperar dados criptografados.
Política de chave modificada incorretamente.	AWS IoT Core perde as permissões de acesso à chave	Falhas no serviço até que a política seja corrigida.
Perfil do IAM excluído.	AWS IoT Core não pode assumir a função de acessar a chave	Falha total no serviço de criptografia.
O perfil do IAM foi modificado incorretamente.	AWS IoT Core não pode assumir a função ou usar a função para acessar a chave	Falhas no serviço até que o perfil do IAM seja corrigido.

Prevenção e práticas recomendadas

Para evitar a exclusão ou desativação acidental da chave e minimizar o risco de falhas no serviço:

Implementar políticas de ciclo de vida de chaves

Estabeleça procedimentos claros para criação, alternância e retirada de chaves. Documente quais chaves são usadas por quais AWS IoT Core recursos e mantenha um inventário das chaves ativas.

Usar as políticas do IAM para restringir a exclusão de chaves

Crie políticas do IAM que evitem que usuários não autorizados excluam ou desabilitem chaves críticas de criptografia. Use condições para exigir aprovação adicional para operações de exclusão de chaves.

Ativar CloudTrail registro

Monitore todas as AWS KMS principais operações CloudTrail para detectar atividades de gerenciamento de chaves não autorizadas ou acidentais. Configure alertas para exclusão e desativação de chaves ou alterações na política.

Procedimentos de substituição de chaves de teste

Teste regularmente seus procedimentos de substituição de chaves em ambientes que não sejam de produção a fim de garantir que você possa se recuperar rapidamente de falhas relacionadas a chaves.

Manter backups de chaves

Embora você não possa exportar material de AWS KMS chaves, mantenha registros detalhados das chaves ARNs, das políticas e das AWS IoT Core configurações associadas para facilitar a substituição rápida das chaves, se necessário.

Monitorar a integridade das chaves

Monitore continuamente a métrica `CMK.Health` e configure alertas automatizados para alterações no status de integridade das chaves. Implemente respostas automatizadas para resolver rapidamente problemas relacionados a chaves.

> **Important**
>
> Sempre teste os procedimentos de atualização de chaves em ambientes de desenvolvimento antes de implementá-los na produção. Tenha um plano de reversão documentado e garanta que os procedimentos de substituição de chaves possam ser executados rapidamente em caso de emergências.

Etapa 6: monitorar a integridade das chaves

Como parte das verificações AWS IoT Core periódicas, CloudWatch métricas e registros são emitidos para fornecer visibilidade sobre o status de integridade da configuração de sua chave gerenciada pelo cliente

AWS IoT Core emite a `CMK.Health` métrica pelo CloudWatch menos uma vez a cada minuto. A métrica fornece informações sobre o status de integridade das chaves gerenciadas pelo cliente usadas AWS IoT Core para criptografar e descriptografar seus dados.

A métrica `CMK.Health` pode ter os seguintes valores:

- O valor é`1`: AWS IoT Core é capaz de usar as chaves de criptografia com sucesso para criptografar e descriptografar seus dados.

- O valor AWS IoT Core é `0`: não consegue usar as chaves de criptografia para criptografar e descriptografar seus dados.

AWS IoT Core também emite registros AWS IoT V2 quando o status de integridade das chaves de criptografia muda. Esses logs fornecem detalhes adicionais sobre a atualização do status de integridade. Para visualizar esses registros, você deve habilitar os registros AWS IoT V2. Os logs de `HEALTHY` são emitidos em nível de `INFO`, e os logs de `UNHEALTHY` são emitidos em nível de `ERROR`. Para acessar mais informações sobre níveis de log, consulte Níveis de log.

Os exemplos a seguir são entradas de CloudWatch registro emitidas por AWS IoT Core para indicar a atualização do status de saúde das chaves gerenciadas pelo cliente.

Para monitorar e responder de forma eficaz a mudanças no status de integridade das chaves:

1. Configure CloudWatch alarmes para a `CMK.Health` métrica:

```
aws cloudwatch put-metric-alarm --region us-west-2 \
  --alarm-name "IoTCore-CMK-Health-Alert" \
  --alarm-description "Alert when IoT Core CMK health is unhealthy" \
  --metric-name "CMK.Health" \
  --namespace "AWS/IoT" \
  --statistic "Minimum" \
  --period 300 \
  --evaluation-periods 1 \
  --threshold 1 \
  --comparison-operator "LessThanThreshold" \
  --alarm-actions "arn:aws:sns:us-west-2:111122223333:iot-alerts"
```

2. Ative o registro AWS IoT V2 para capturar eventos detalhados de alteração do status de saúde com códigos e mensagens de erro.
3. Confira o status da configuração para solucionar problemas:

```
aws iot describe-encryption-configuration --region us-west-2
```

4. Investigue o status UNHEALTHY examinando o campo `errorCode`:
   - `KMS_KEY_VALIDATION_ERROR`— Problema com a AWS KMS chave (desativado, excluído ou problemas de política)
   - `ROLE_VALIDATION_ERROR`: problema com o perfil do IAM (excluído, problemas de política ou problemas de confiança).

De UNHEALTHY para HEALTHY

Quando o status das chaves de criptografia for atualizado de UNHEALTHY paraHEALTHY, AWS IoT Core emitirá uma mensagem de log AWS IoT V2 no seguinte formato.

```
{
    "timestamp": "2017-08-10 15:37:23.476",
    "logLevel": "INFO",
    "traceId": "8421693b-f4f0-4e4a-9235-0cff8bab897d",
    "accountId": "111122223333",
    "status": "SUCCESS",
    "cmkStatus": "HEALTHY",
    "kmsKeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "kmsAccessRoleArn": "arn:aws:iam::111122223333:role/myrole",
    "eventType": "CmkHealthCheck"
}
```

De HEALTHY para UNHEALTHY

Quando o status das chaves de criptografia for atualizado de HEALTHY paraUNHEALTHY, AWS IoT Core emitirá uma mensagem de log AWS IoT V2 no seguinte formato.

```
{
    "timestamp": "2017-08-10 15:37:23.476",
    "logLevel": "ERROR",
    "traceId": "8421693b-f4f0-4e4a-9235-0cff8bab897d",
    "accountId": "111122223333",
    "status": "FAILURE",
    "cmkStatus": "UNHEALTHY",
    "errorCode": "KMS_KEY_VALIDATION_ERROR / ROLE_VALIDATION_ERROR",
    "errorMessage": "Error message on why there was a failure",
    "kmsKeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "kmsAccessRoleArn": "arn:aws:iam::111122223333:role/myrole",
    "eventType": "CmkHealthCheck"
}
```

> **Warning**
>
> Quando a integridade é fundamentalUNHEALTHY, AWS IoT Core as operações falham imediatamente. Se isso ocorrer, revise as configurações das chaves, as permissões do perfil

do IAM e as políticas. Monitore a métrica `CMK.Health` em busca de mudanças de status. Se as operações continuarem falhando após revisar suas configurações, entre em contato com o gerente de conta ou com o AWS Support Center para receber assistência adicional.

AWS CloudTrail eventos

Você também pode monitorar AWS IoT Core o uso da chave KMS para operações de criptografia e descriptografia. AWS IoT Core faráDescribeKey,, DecryptReEncrypt, e GenerateDataKeyWithoutPlaintext operações em sua chave KMS para criptografar/descriptografar dados pertencentes à sua AWS conta armazenados em repouso.

Existem CloudTrail eventos para DescribeKeyDecrypt,ReEncrypt, GenerateDataKeyWithoutPlaintext e. Esses eventos monitoram AWS KMS as operações chamadas por AWS IoT Core para acessar dados criptografados pela chave gerenciada pelo cliente.

Exemplo de **`Decrypt`**


```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "**********************",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "*****"
            },
            "attributes": {
                "creationDate": "2024-09-16T20:23:39Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "iot.amazonaws.com"
    },
    "eventTime": "2024-09-16T20:32:48Z",
```

```
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "iot.amazonaws.com",
    "userAgent": "iot.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "kms-arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "aws-crypto-ec:vendor": "iot.amazonaws.com",
            "branch-key-id": "111122223333",
            "type": "branch:ACTIVE"
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "requestID": "1afb6d98-8388-455d-8b48-e62c9e0cf7f4",
    "eventID": "b59a5f16-0d98-46d8-a590-0e040a48b39b",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Gerenciamento de identidade e acesso para AWS IoT

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) a usar AWS IoT os recursos. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

- Público
- Como autenticar com identidades do IAM
- Gerenciar o acesso usando políticas
- Como AWS IoT funciona com o IAM
- AWS IoT exemplos de políticas baseadas em identidade
- AWS políticas gerenciadas para AWS IoT
- Solução de problemas AWS IoT de identidade e acesso

Público

A forma como você usa AWS Identity and Access Management (IAM) difere com base na sua função:

- Usuário do serviço: solicite permissões ao seu administrador se você não conseguir acessar os atributos (consulte Solução de problemas AWS IoT de identidade e acesso).
- Administrador do serviço: determine o acesso do usuário e envie solicitações de permissão (consulte Como AWS IoT funciona com o IAM)
- Administrador do IAM: escreva políticas para gerenciar o acesso (consulte AWS IoT exemplos de políticas baseadas em identidade)

Como autenticar com identidades do IAM

AWS IoT As identidades internas podem ser certificados de dispositivo (X.509), identidades do Amazon Cognito ou usuários ou grupos do IAM. Este tópico aborda apenas identidades do IAM. Para obter mais informações sobre as outras identidades que oferecem AWS IoT suporte, consulteAutenticação de cliente.

A autenticação é como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado como usuário do IAM ou assumindo uma função do IAM. Usuário raiz da conta da AWS

Você pode fazer login como uma identidade federada usando credenciais de uma fonte de identidade como Centro de Identidade do AWS IAM (IAM Identity Center), autenticação de login único ou credenciais. Google/Facebook Para ter mais informações sobre como fazer login, consulte Como fazer login em sua Conta da AWS no Guia do usuário do Início de Sessão da AWS .

Para acesso programático, AWS fornece um SDK e uma CLI para assinar solicitações criptograficamente. Para ter mais informações, consulte AWS Signature Version 4 para solicitações de API no Guia do usuário do IAM.

Conta da AWS usuário root

Ao criar um Conta da AWS, você começa com uma identidade de login chamada usuário Conta da AWS raiz que tem acesso completo a todos Serviços da AWS os recursos. É altamente recomendável não usar o usuário-raiz em tarefas diárias. Consulte as tarefas que exigem credenciais de usuário-raiz em Tarefas que exigem credenciais de usuário-raiz no Guia do usuário do IAM.

Usuários e grupos do IAM

Um usuário do IAM é uma identidade com permissões específicas para uma única pessoa ou aplicação. É recomendável usar credenciais temporárias, em vez de usuários do IAM com credenciais de longo prazo. Para obter mais informações, consulte Exigir que usuários humanos usem a federação com um provedor de identidade para acessar AWS usando credenciais temporárias no Guia do usuário do IAM.

Um grupo do IAM especifica um conjunto de usuários do IAM e facilita o gerenciamento de permissões para grandes conjuntos de usuários. Para ter mais informações, consulte Casos de uso de usuários do IAM no Guia do usuário do IAM.

Perfis do IAM

Uma perfil do IAM é uma identidade com permissões específicas que oferece credenciais temporárias. Você pode assumir uma função mudando de um usuário para uma função do IAM (console) ou chamando uma operação de AWS API AWS CLI ou. Para saber mais, consulte Métodos para assumir um perfil no Manual do usuário do IAM.

Os perfis do IAM são úteis para acesso de usuário federado, permissões de usuário do IAM temporárias, acesso entre contas, acesso entre serviços e aplicações em execução no Amazon EC2. Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política define permissões quando associada a uma identidade ou recurso. AWS avalia essas políticas quando um diretor faz uma solicitação. A maioria das políticas é armazenada AWS como

documentos JSON. Para ter mais informações sobre documentos de política JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Por meio de políticas, os administradores especificam quem tem acesso a que, definindo qual entidade principal pode realizar ações em quais recursos e sob quais condições.

Por padrão, usuários e perfis não têm permissões. Um administrador do IAM cria políticas do IAM e as adiciona aos perfis, os quais os usuários podem então assumir. As políticas do IAM definem permissões, independentemente do método usado para realizar a operação.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissão JSON que você anexa a uma identidade (usuário, grupo ou perfil). Essas políticas controlam quais ações as identidades podem realizar, em quais recursos e sob quais condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser políticas em linha (incorporadas diretamente em uma única identidade) ou políticas gerenciadas (políticas autônomas anexadas a várias identidades). Para saber como escolher entre uma política gerenciada e políticas em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. Entre os exemplos estão políticas de confiança de perfil do IAM e políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. É necessário especificar uma entidade principal em uma política baseada em recursos.

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o AWS WAF Amazon VPC são exemplos de serviços que oferecem suporte. ACLs Para saber mais ACLs, consulte a visão geral da lista de controle de acesso (ACL) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais que podem definir o máximo de permissões concedidas por tipos de políticas mais comuns:

- Limites de permissões: definem o número máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM. Para saber mais sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs) — Especifique as permissões máximas para uma organização ou unidade organizacional em AWS Organizations. Para saber mais, consulte Políticas de controle de serviço no Guia do usuário do AWS Organizations .
- Políticas de controle de recursos (RCPs) — Defina o máximo de permissões disponíveis para recursos em suas contas. Para obter mais informações, consulte Políticas de controle de recursos (RCPs) no Guia AWS Organizations do usuário.
- Políticas de sessão: políticas avançadas transmitidas como um parâmetro durante a criação de uma sessão temporária para um perfil ou um usuário federado. Para saber mais, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

Como AWS IoT funciona com o IAM

Antes de usar o IAM para gerenciar o acesso AWS IoT, você deve entender quais recursos do IAM estão disponíveis para uso AWS IoT. Para ter uma visão de alto nível de como AWS IoT e outros AWS serviços funcionam com o IAM, consulte AWS Serviços que funcionam com o IAM no Guia do usuário do IAM.

Tópicos

- AWS IoT políticas baseadas em identidade

- AWS IoT políticas baseadas em recursos
- Autorização baseada em AWS IoT tags
- AWS IoT Funções do IAM

AWS IoT políticas baseadas em identidade

Com as políticas baseadas em identidade do IAM, é possível especificar ações permitidas ou negadas e recursos, bem como as condições sob as quais as ações são permitidas ou negadas. O AWS IoT oferece suporte a ações, recursos e chaves de condição específicos. Para conhecer todos os elementos usados em uma política JSON, consulte Referência de elementos de política JSON do IAM no Guia do usuário do IAM.

Ações

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que podem ser usadas para permitir ou negar acesso em uma política. Incluem ações em uma política para conceder permissões para executar a operação associada.

A tabela a seguir lista as ações do IAM IoT, a AWS IoT API associada e o recurso que a ação manipula.

Ações de políticas	AWS IoT API	Recursos
IoT: AcceptCertificateTransfer	AcceptCertificateTransfer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`* **Note** O Conta da AWS especificado no ARN deve ser a conta para a qual o certificado está sendo transferido.
IoT: AddThingToThingGroup	AddThingToThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`*

Ações de políticas	AWS IoT API	Recursos
		arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: AssociateTargetsWithJob	AssociateTargetsWithJob	nenhuma
IoT: AttachPolicy	AttachPolicy	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* or arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: AttachPrincipalPolicy	AttachPrincipalPolicy	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: AttachSecurityProfile	AttachSecurityProfile	arn:aws:iot: *region*:*account-id* :securityprofile/ *security-profile-name* arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT: AttachThingPrincipal	AttachThingPrincipal	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: CancelCertificateTransfer	CancelCertificateTransfer	arn:aws:iot: *region*:*account-id* :cert/*cert-id* **Note** O Conta da AWS especificado no ARN deve ser a conta para a qual o certificado está sendo transferido.
IoT: CancelJob	CancelJob	arn:aws:iot: *region*:*account-id* :job/*job-id*

Ações de políticas	AWS IoT API	Recursos
IoT: CancelJob Execution	CancelJob Execution	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: ClearDefaultAuthorizer	ClearDefaultAuthorizer	Nenhum
IoT: CreateAuthorizer	CreateAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-function-name*
IoT: CreateCertificateFromCsr	CreateCertificateFromCsr	*
IoT: CreateDimension	CreateDimension	arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT: CreateJob	CreateJob	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* arn:aws:iot: *region*:*account-id* :thing/*thing-name* arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT: CreateJobTemplate	CreateJobTemplate	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT: CreateKeysAndCertificate	CreateKeysAndCertificate	*

Ações de políticas	AWS IoT API	Recursos
IoT: CreatePolicy	CreatePolicy	`arn:aws:iot:` *region*`:`*account-id* `:policy/`*policy-name*
IoT: CreatePolicyVersion	CreatePolicyVersion	`arn:aws:iot:` *region*`:`*account-id* `:policy/`*policy-name* **Note** Isso deve ser uma AWS IoT política, não uma política do IAM.
IoT: CreateRoleAlias	CreateRoleAlias	(parâmetro: roleAlias) `arn:aws:iot:` *region*`:`*account-id* `:rolealias/` *role-alias-name*
IoT: CreateSecurityProfile	CreateSecurityProfile	`arn:aws:iot:` *region*`:`*account-id* `:securityprofile/` *security-profile-name* `arn:aws:iot:` *region*`:`*account-id* `:dimension/` *dimension-name*
IoT: CreateThing	CreateThing	`arn:aws:iot:` *region*`:`*account-id* `:thing/`*thing-name*
IoT: CreateThingGroup	CreateThingGroup	`arn:aws:iot:` *region*`:`*account-id* `:thinggroup/` *thing-group-name* para o grupo que está sendo criado e para o grupo pai, se usado
IoT: CreateThingType	CreateThingType	`arn:aws:iot:` *region*`:`*account-id* `:thingtype/` *thing-type-name*
IoT: CreateTopicRule	CreateTopicRule	`arn:aws:iot:` *region*`:`*account-id* `:rule/`*rule-name*

Ações de políticas	AWS IoT API	Recursos
IoT: DeleteAuthorizer	DeleteAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-name*
IoT: excluir CACertificate	Excluir CACertificate	arn:aws:iot: *region*:*account-id* :cacert/*cert-id*
IoT: DeleteCertificate	DeleteCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: DeleteDimension	DeleteDimension	arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT: DeleteJob	DeleteJob	arn:aws:iot: *region*:*account-id* :job/*job-id*
IoT: DeleteJobTemplate	DeleteJobTemplate	arn:aws:iot: *region*:*account-id* :job/*job-template-id*
IoT: DeleteJobExecution	DeleteJobExecution	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: DeletePolicy	DeletePolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: DeletePolicyVersion	DeletePolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: DeleteRegistrationCode	DeleteRegistrationCode	*
IoT: DeleteRoleAlias	DeleteRoleAlias	arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*

Ações de políticas	AWS IoT API	Recursos
IoT: DeleteSecurityProfile	DeleteSecurityProfile	arn:aws:iot: *region*:*account-id* :securityprofile/ *security-profile-name* arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT: DeleteThing	DeleteThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: DeleteThingGroup	DeleteThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DeleteThingType	DeleteThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT: DeleteTopicRule	DeleteTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: excluir V2 LoggingLevel	Excluir V2 LoggingLevel	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DeprecateThingType	DeprecateThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT: DescribeAuthorizer	DescribeAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-function-name* (parâmetro: authorizerName) nenhuma
IoT: descrever CACertificate	Descreva CACertificate	arn:aws:iot: *region*:*account-id* :cacert/*cert-id*
IoT: DescribeCertificate	DescribeCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*

Ações de políticas	AWS IoT API	Recursos
IoT: DescribeDefaultAuthorizer	DescribeDefaultAuthorizer	Nenhum
IoT: DescribeEndpoint	DescribeEndpoint	*
IoT: DescribeEventConfigurations	DescribeEventConfigurations	nenhuma
IoT: DescribeIndex	DescribeIndex	`arn:aws:iot:` *`region`*`:`*`account-id`* `:index/`*`index-name`*
IoT: DescribeJob	DescribeJob	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`*
IoT: DescribeJobExecution	DescribeJobExecution	Nenhum
IoT: DescribeJobTemplate	DescribeJobTemplate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-template-id`*
IoT: DescribeRoleAlias	DescribeRoleAlias	`arn:aws:iot:` *`region`*`:`*`account-id`* `:rolealias/` *`role-alias-name`*
IoT: DescribeThing	DescribeThing	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: DescribeThingGroup	DescribeThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`*
IoT: DescribeThingRegistrationTask	DescribeThingRegistrationTask	Nenhum
IoT: DescribeThingType	DescribeThingType	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thingtype/` *`thing-type-name`*

Ações de políticas	AWS IoT API	Recursos
IoT: DetachPolicy	DetachPolicy	arn:aws:iot: *region*:*account-id* :cert/*cert-id* or arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DetachPrincipalPolicy	DetachPrincipalPolicy	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: DetachSecurityProfile	DetachSecurityProfile	arn:aws:iot: *region*:*account-id* :securityprofile/ *security-profile-name* arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT: DetachThingPrincipal	DetachThingPrincipal	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: DisableTopicRule	DisableTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: EnableTopicRule	EnableTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: GetEffectivePolicies	GetEffectivePolicies	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: GetIndexingConfiguration	GetIndexingConfiguration	Nenhum
IoT: GetJobDocument	GetJobDocument	arn:aws:iot: *region*:*account-id* :job/*job-id*
IoT: GetLoggingOptions	GetLoggingOptions	*

Ações de políticas	AWS IoT API	Recursos
IoT: GetPolicy	GetPolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: GetPolicyVersion	GetPolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: GetRegistrationCode	GetRegistrationCode	*
IoT: GetTopicRule	GetTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: ListAttachedPolicies	ListAttachedPolicies	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* or arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: ListAuthorizers	ListAuthorizers	Nenhum
IoT: listaCACertificates	ListaCACertificates	*
IoT: ListCertificates	ListCertificates	*
IoT: CA ListCertificatesBy	ListCertificatesByCA	*
IoT: ListIndices	ListIndices	Nenhum
IoT: ListJobExecutionsForJob	ListJobExecutionsForJob	Nenhum

Ações de políticas	AWS IoT API	Recursos
IoT: ListJobExecutionsForThing	ListJobExecutionsForThing	Nenhum
IoT: ListJobs	ListJobs	`arn:aws:iot:` *region*`:`*account-id* `:thinggroup/` *thing-group-name* se o thingGroupName parâmetro for usado
IoT: ListJobTemplates	ListJobs	Nenhum
IoT: ListOutgoingCertificates	ListOutgoingCertificates	*
IoT: ListPolicies	ListPolicies	*
IoT: ListPolicyPrincipals	ListPolicyPrincipals	*
IoT: ListPolicyVersions	ListPolicyVersions	`arn:aws:iot:` *region*`:`*account-id* `:policy/`*policy-name*
IoT: ListPrincipalPolicies	ListPrincipalPolicies	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*
IoT: ListPrincipalThings	ListPrincipalThings	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*
IoT: ListRoleAliases	ListRoleAliases	Nenhum
IoT: ListTargetsForPolicy	ListTargetsForPolicy	`arn:aws:iot:` *region*`:`*account-id* `:policy/`*policy-name*
IoT: ListThingGroups	ListThingGroups	Nenhum

Ações de políticas	AWS IoT API	Recursos
IoT: ListThing GroupsForThing	ListThing GroupsForThing	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: ListThing Principals	ListThing Principals	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: ListThing RegistrationTaskReports	ListThing RegistrationTaskReports	Nenhum
IoT: ListThing RegistrationTasks	ListThing RegistrationTasks	Nenhum
IoT: ListThing Types	ListThingTypes	*
IoT: ListThings	ListThings	*
IoT: ListThingsInThingGroup	ListThingsInThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`*
IoT: ListTopicRules	ListTopicRules	*
IoT: Lista V2LoggingLevels	Lista V2LoggingLevels	Nenhum
IoT: registroCACertificate	RegistreCACertificate	*
IoT: RegisterCertificate	RegisterCertificate	*
IoT: RegisterThing	RegisterThing	Nenhum

Ações de políticas	AWS IoT API	Recursos
IoT: RejectCertificateTransfer	RejectCertificateTransfer	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: RemoveThingFromThingGroup	RemoveThingFromThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: ReplaceTopicRule	ReplaceTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: SearchIndex	SearchIndex	arn:aws:iot: *region*:*account-id* :index/*index-id*
IoT: SetDefaultAuthorizer	SetDefaultAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-function-name*
IoT: SetDefaultPolicyVersion	SetDefaultPolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: SetLoggingOptions	SetLoggingOptions	arn:aws:iot: *region*:*account-id* :role/*role-name*
IoT: SetV2LoggingLevel	Configurar V2LoggingLevel	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: SetV2LoggingOptions	Configurar V2LoggingOptions	arn:aws:iot: *region*:*account-id* :role/*role-name*
IoT: StartThingRegistrationTask	StartThingRegistrationTask	Nenhum
IoT: StopThingRegistrationTask	StopThingRegistrationTask	Nenhum

Ações de políticas	AWS IoT API	Recursos
IoT: TestAuthorization	TestAuthorization	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: TestInvokeAuthorizer	TestInvokeAuthorizer	Nenhum
IoT: TransferCertificate	TransferCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: UpdateAuthorizer	UpdateAuthorizer	arn:aws:iot: *region*:*account-id* :authorizerfunction/ *authorizer-function-name*
IoT: atualização CACertificate	Atualizar CACertificate	arn:aws:iot: *region*:*account-id* :cacert/*cert-id*
IoT: UpdateCertificate	UpdateCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: UpdateDimension	UpdateDimension	arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT: UpdateEventConfigurations	UpdateEventConfigurations	Nenhum
IoT: UpdateIndexingConfiguration	UpdateIndexingConfiguration	Nenhum
IoT: UpdateRoleAlias	UpdateRoleAlias	arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*
IoT: UpdateSecurityProfile	UpdateSecurityProfile	arn:aws:iot: *region*:*account-id* :securityprofile/ *security-profile-name* arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*

Ações de políticas	AWS IoT API	Recursos
IoT: UpdateThing	UpdateThing	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: UpdateThingGroup	UpdateThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`*
IoT: UpdateThingGroupsForThing	UpdateThingGroupsForThing	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`*

As ações políticas AWS IoT usam o seguinte prefixo antes da ação:`iot:`. Por exemplo, para conceder permissão a alguém para listar todas as coisas de IoT Conta da AWS registradas na `ListThings` API, você inclui a `iot:ListThings` ação na política dessa pessoa. As declarações de política devem incluir um `NotAction` elemento `Action` ou. AWS IoT define seu próprio conjunto de ações que descrevem as tarefas que você pode executar com esse serviço.

Para especificar várias ações em uma única instrução, separe-as com vírgulas, como segue:

```
"Action": [
      "ec2:action1",
      "ec2:action2"
```

Você também pode especificar várias ações usando caracteres curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `Describe`, inclua a seguinte ação:

```
"Action": "iot:Describe*"
```

Para ver uma lista de AWS IoT ações, consulte Ações definidas por AWS IoT no Guia do usuário do IAM.

Ações do Device Advisor

A tabela a seguir lista as ações do IoT Device Advisor do IAM, a API do AWS IoT Device Advisor associada e o recurso que a ação manipula.

Ações de políticas	AWS IoT API	Recursos
consultor de dispositivos de IoT: CreateSuiteDefinition	CreateSuiteDefinition	Nenhum
consultor de dispositivos de IoT: DeleteSuiteDefinition	DeleteSuiteDefinition	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id*
consultor de dispositivos de IoT: GetSuiteDefinition	GetSuiteDefinition	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id*
consultor de dispositivos de IoT: GetSuiteRun	GetSuiteRun	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-run-id*
consultor de dispositivos de IoT: GetSuiteRunReport	GetSuiteRunReport	arn:aws:iotdeviceadvisor: *region*:*account-id* :suiterun/ *suite-definition-id* /*suite-run-id*
consultor de dispositivos de IoT: ListSuiteDefinitions	ListSuiteDefinitions	Nenhum
consultor de dispositivos de IoT: ListSuiteRuns	ListSuiteRuns	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id*

Ações de políticas	AWS IoT API	Recursos
consultor de dispositivos de IoT: ListTagsForResource	ListTagsForResource	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id* arn:aws:iotdeviceadvisor: *region*:*account-id* :suiterun/suite-definition-id/ *suite-run-id*
consultor de dispositivos de IoT: StartSuiteRun	StartSuiteRun	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id*
consultor de dispositivos de IoT: TagResource	TagResource	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id* arn:aws:iotdeviceadvisor: *region*:*account-id* :suiterun/suite-definition-id/ *suite-run-id*
consultor de dispositivos de IoT: UntagResource	UntagResource	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id* arn:aws:iotdeviceadvisor: *region*:*account-id* :suiterun/suite-definition-id/ *suite-run-id*
consultor de dispositivos de IoT: UpdateSuiteDefinition	UpdateSuiteDefinition	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id*
consultor de dispositivos de IoT: StopSuiteRun	StopSuiteRun	arn:aws:iotdeviceadvisor: *region*:*account-id* :suiterun/suite-definition-id/ *suite-run-id*

As ações de política no AWS IoT Device Advisor usam o seguinte prefixo antes da ação:`iotdeviceadvisor:`. Por exemplo, para conceder permissão a alguém para listar todas as definições de suítes Conta da AWS registradas na ListSuiteDefinitions API, você inclui a `iotdeviceadvisor:ListSuiteDefinitions` ação na política dessa pessoa.

Recursos

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Para ações que não oferecem compatibilidade com permissões em nível de recurso, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

AWS IoT recursos

Ações de políticas	AWS IoT API	Recursos
IoT: AcceptCertificateTransfer	AcceptCertificateTransfer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`* **Note** O Conta da AWS especificado no ARN deve ser a conta para a qual o certificado está sendo transferido.
IoT: AddThingToThingGroup	AddThingToThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: AssociateTargetsWithJob	AssociateTargetsWithJob	Nenhum

Ações de políticas	AWS IoT API	Recursos
IoT: AttachPolicy	AttachPolicy	`arn:aws:iot:` *region*`:`*account-id* `:thinggroup/` *thing-group-name* or `arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*
IoT: AttachPrincipalPolicy	AttachPrincipalPolicy	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*
IoT: AttachThingPrincipal	AttachThingPrincipal	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*
IoT: CancelCertificateTransfer	CancelCertificateTransfer	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id* **Note** O Conta da AWS especificado no ARN deve ser a conta para a qual o certificado está sendo transferido.
IoT: CancelJob	CancelJob	`arn:aws:iot:` *region*`:`*account-id* `:job/`*job-id*
IoT: CancelJobExecution	CancelJobExecution	`arn:aws:iot:` *region*`:`*account-id* `:job/`*job-id* `arn:aws:iot:` *region*`:`*account-id* `:thing/`*thing-name*
IoT: ClearDefaultAuthorizer	ClearDefaultAuthorizer	Nenhum
IoT: CreateAuthorizer	CreateAuthorizer	`arn:aws:iot:` *region*`:`*account-id* `:authorizer/` *authorizer-function-name*

Ações de políticas	AWS IoT API	Recursos
IoT: CreateCertificateFromCsr	CreateCertificateFromCsr	*
IoT: CreateJob	CreateJob	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* arn:aws:iot: *region*:*account-id* :thing/*thing-name* arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT: CreateJobTemplate	CreateJobTemplate	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT: CreateKeysAndCertificate	CreateKeysAndCertificate	*
IoT: CreatePolicy	CreatePolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
CreatePolicyVersion	IoT: CreatePolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name* **Note** Isso deve ser uma AWS IoT política, não uma política do IAM.

Ações de políticas	AWS IoT API	Recursos
IoT: CreateRoleAlias	CreateRoleAlias	(parâmetro: roleAlias) arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*
IoT: CreateThing	CreateThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: CreateThingGroup	CreateThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* para o grupo que está sendo criado e para o grupo pai, se usado
IoT: CreateThingType	CreateThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT: CreateTopicRule	CreateTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: DeleteAuthorizer	DeleteAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-name*
IoT: excluir CACertificate	Excluir CACertificate	arn:aws:iot: *region*:*account-id* :cacert/*cert-id*
IoT: DeleteCertificate	DeleteCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: DeleteJob	DeleteJob	arn:aws:iot: *region*:*account-id* :job/*job-id*
IoT: DeleteJobExecution	DeleteJobExecution	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thing/*thing-name*

Ações de políticas	AWS IoT API	Recursos
IoT: DeleteJob Template	DeleteJob Template	arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT: DeletePolicy	DeletePolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: DeletePolicyVersion	DeletePolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT: DeleteRegistrationCode	DeleteRegistrationCode	*
IoT: DeleteRoleAlias	DeleteRoleAlias	arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*
IoT: DeleteThing	DeleteThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: DeleteThingGroup	DeleteThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DeleteThingType	DeleteThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT: DeleteTopicRule	DeleteTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: excluir V2 LoggingLevel	Excluir V2 LoggingLevel	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DeprecateThingType	DeprecateThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*

Ações de políticas	AWS IoT API	Recursos
IoT: DescribeAuthorizer	DescribeAuthorizer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:authorizer/` *`authorizer-function-name`* (parâmetro: authorizerName) nenhuma
IoT: descrever CACertificate	Descreva CACertificate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cacert/`*`cert-id`*
IoT: DescribeCertificate	DescribeCertificate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: DescribeDefaultAuthorizer	DescribeDefaultAuthorizer	Nenhum
IoT: DescribeEndpoint	DescribeEndpoint	*
IoT: DescribeEventConfigurations	DescribeEventConfigurations	nenhuma
IoT: DescribeIndex	DescribeIndex	`arn:aws:iot:` *`region`*`:`*`account-id`* `:index/`*`index-name`*
IoT: DescribeJob	DescribeJob	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`*
IoT: DescribeJobExecution	DescribeJobExecution	Nenhum
IoT: DescribeJobTemplate	DescribeJobTemplate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:jobtemplate/` *`job-template-id`*
IoT: DescribeRoleAlias	DescribeRoleAlias	`arn:aws:iot:` *`region`*`:`*`account-id`* `:rolealias/` *`role-alias-name`*

Ações de políticas	AWS IoT API	Recursos
IoT: DescribeThing	DescribeThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT: DescribeThingGroup	DescribeThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DescribeThingRegistrationTask	DescribeThingRegistrationTask	Nenhum
IoT: DescribeThingType	DescribeThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT: DetachPolicy	DetachPolicy	arn:aws:iot: *region*:*account-id* :cert/*cert-id* or arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT: DetachPrincipalPolicy	DetachPrincipalPolicy	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: DetachThingPrincipal	DetachThingPrincipal	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT: DisableTopicRule	DisableTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: EnableTopicRule	EnableTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT: GetEffectivePolicies	GetEffectivePolicies	arn:aws:iot: *region*:*account-id* :cert/*cert-id*

Ações de políticas	AWS IoT API	Recursos
IoT: GetIndexingConfiguration	GetIndexingConfiguration	Nenhum
IoT: GetJobDocument	GetJobDocument	`arn:aws:iot:` *region*`:`*account-id* `:job/`*job-id*
IoT: GetLoggingOptions	GetLoggingOptions	*
IoT: GetPolicy	GetPolicy	`arn:aws:iot:` *region*`:`*account-id* `:policy/`*policy-name*
IoT: GetPolicyVersion	GetPolicyVersion	`arn:aws:iot:` *region*`:`*account-id* `:policy/`*policy-name*
IoT: GetRegistrationCode	GetRegistrationCode	*
IoT: GetTopicRule	GetTopicRule	`arn:aws:iot:` *region*`:`*account-id* `:rule/`*rule-name*
IoT: ListAttachedPolicies	ListAttachedPolicies	`arn:aws:iot:` *region*`:`*account-id* `:thinggroup/` *thing-group-name* or `arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*
IoT: ListAuthorizers	ListAuthorizers	Nenhum
IoT: lista CACertificates	Lista CACertificates	*
IoT: ListCertificates	ListCertificates	*

Ações de políticas	AWS IoT API	Recursos
IoT: CA ListCertificatesBy	ListCertificatesByCA	*
IoT: ListIndices	ListIndices	Nenhum
IoT: ListJobExecutionsForJob	ListJobExecutionsForJob	Nenhum
IoT: ListJobExecutionsForThing	ListJobExecutionsForThing	Nenhum
IoT: ListJobs	ListJobs	`arn:aws:iot:` *region*`:`*account-id* `:thinggroup/` *thing-group-name* se o thingGroupName parâmetro for usado
IoT: ListJobTemplates	ListJobTemplates	Nenhum
IoT: ListOutgoingCertificates	ListOutgoingCertificates	*
IoT: ListPolicies	ListPolicies	*
IoT: ListPolicyPrincipals	ListPolicyPrincipals	`arn:aws:iot:` *region*`:`*account-id* `:policy/`*policy-name*
IoT: ListPolicyVersions	ListPolicyVersions	`arn:aws:iot:` *region*`:`*account-id* `:policy/`*policy-name*
IoT: ListPrincipalPolicies	ListPrincipalPolicies	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*
IoT: ListPrincipalThings	ListPrincipalThings	`arn:aws:iot:` *region*`:`*account-id* `:cert/`*cert-id*

Ações de políticas	AWS IoT API	Recursos
IoT: ListRoleAliases	ListRoleAliases	Nenhum
IoT: ListTargetsForPolicy	ListTargetsForPolicy	`arn:aws:iot:` *region*`:`*account-id* `:policy/`*policy-name*
IoT: ListThingGroups	ListThingGroups	Nenhum
IoT: ListThingGroupsForThing	ListThingGroupsForThing	`arn:aws:iot:` *region*`:`*account-id* `:thing/`*thing-name*
IoT: ListThingPrincipals	ListThingPrincipals	`arn:aws:iot:` *region*`:`*account-id* `:thing/`*thing-name*
IoT: ListThingRegistrationTaskReports	ListThingRegistrationTaskReports	Nenhum
IoT: ListThingRegistrationTasks	ListThingRegistrationTasks	Nenhum
IoT: ListThingTypes	ListThingTypes	*
IoT: ListThings	ListThings	*
IoT: ListThingsInThingGroup	ListThingsInThingGroup	`arn:aws:iot:` *region*`:`*account-id* `:thinggroup/` *thing-group-name*
IoT: ListTopicRules	ListTopicRules	*
IoT: Lista V2LoggingLevels	Lista V2LoggingLevels	Nenhum

Ações de políticas	AWS IoT API	Recursos
IoT: registro CACertificate	Registre-se CACertificate	*
IoT: RegisterCertificate	RegisterCertificate	*
IoT: RegisterThing	RegisterThing	Nenhum
IoT: RejectCertificateTransfer	RejectCertificateTransfer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: RemoveThingFromThingGroup	RemoveThingFromThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: ReplaceTopicRule	ReplaceTopicRule	`arn:aws:iot:` *`region`*`:`*`account-id`* `:rule/`*`rule-name`*
IoT: SearchIndex	SearchIndex	`arn:aws:iot:` *`region`*`:`*`account-id`* `:index/`*`index-id`*
IoT: SetDefaultAuthorizer	SetDefaultAuthorizer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:authorizer/` *`authorizer-function-name`*
IoT: SetDefaultPolicyVersion	SetDefaultPolicyVersion	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT: SetLoggingOptions	SetLoggingOptions	*
IoT: SetV2LoggingLevel	Configurar V2 LoggingLevel	*

Ações de políticas	AWS IoT API	Recursos
IoT: SetV2 LoggingOptions	Configurar V2 LoggingOptions	*
IoT: StartThin gRegistra tionTask	StartThin gRegistra tionTask	Nenhum
IoT: StopThing RegistrationTask	StopThing RegistrationTask	Nenhum
IoT: TestAutho rization	TestAutho rization	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: TestInvok eAuthorizer	TestInvok eAuthorizer	Nenhum
IoT: TransferC ertificate	TransferCertificat e	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: UpdateAut horizer	UpdateAut horizer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:authoriz erfunction/` *`authorizer-function-name`*
IoT: atualização CACertificate	Atualizar CACertificate	`arn:aws:iot:` *`region`*`:`*`account-i d`* `:cacert/`*`cert-id`*
IoT: UpdateCer tificate	UpdateCer tificate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT: UpdateEve ntConfigurations	UpdateEve ntConfigurations	Nenhum
IoT: UpdateInd exingConf iguration	UpdateInd exingConf iguration	Nenhum

Ações de políticas	AWS IoT API	Recursos
IoT: UpdateRoleAlias	UpdateRoleAlias	`arn:aws:iot:` *`region`*`:`*`account-id`* `:rolealias/` *`role-alias-name`*
IoT: UpdateThing	UpdateThing	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT: UpdateThingGroup	UpdateThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`*
IoT: UpdateThingGroupsForThing	UpdateThingGroupsForThing	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*

Para obter mais informações sobre o formato de ARNs, consulte Amazon Resource Names (ARNs) e AWS Service Namespaces.

Algumas AWS IoT ações, como as de criação de recursos, não podem ser executadas em um recurso específico. Nesses casos, é necessário utilizar o caractere curinga (*).

```
"Resource": "*"
```

Para ver uma lista dos tipos de AWS IoT recursos e seus ARNs, consulte Resources Defined by AWS IoT no Guia do usuário do IAM. Para saber com quais ações você pode especificar o ARN de cada recurso, consulte Ações definidas pelo AWS IoT.

Recursos do Device Advisor

Para definir restrições em nível de recurso para as políticas de IAM do AWS IoT Device Advisor, use os seguintes formatos de ARN de recursos para definições e execuções de suítes.

Formato do ARN de recurso de definição de suíte

```
arn:aws:iotdeviceadvisor:region:account-id:suitedefinition/suite-definition-id
```

Formato do ARN de recurso de execução de suíte

```
arn:aws:iotdeviceadvisor:region:account-id:suiterun/suite-definition-id/suite-run-id
```

Chaves de condição

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Condition` especifica quando as instruções são executadas com base em critérios definidos. É possível criar expressões condicionais que usem agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação. Para ver todas as chaves de condição AWS globais, consulte as chaves de contexto de condição AWS global no Guia do usuário do IAM.

AWS IoT define seu próprio conjunto de chaves de condição e também oferece suporte ao uso de algumas chaves de condição globais. Para ver todas as chaves de condição AWS globais, consulte Chaves de contexto de condição AWS global no Guia do usuário do IAM.

AWS IoT chaves de condição

AWS IoT chaves de condição	Descrição	Tipo
`aws:RequestTag/${tag-key}`	Uma chave de tag que está presente na solicitação que o usuário faz para o AWS IoT.	String
`aws:ResourceTag/${tag-key}`	O componente chave da tag de uma tag anexada a um AWS IoT recurso.	String

AWS IoT chaves de condição	Descrição	Tipo
`aws:TagKeys`	A lista de todos os nomes de chaves de etiquetas associadas ao recurso na solicitação.	String

Para ver uma lista de chaves de AWS IoT condição, consulte Chaves de condição AWS IoT no Guia do usuário do IAM. Para saber com quais ações e recursos você pode usar uma chave de condição, consulte Ações definidas por AWS IoT.

Exemplos

Para ver exemplos de políticas AWS IoT baseadas em identidade, consulte. AWS IoT exemplos de políticas baseadas em identidade

AWS IoT políticas baseadas em recursos

Políticas baseadas em recursos são documentos de política JSON que especificam quais ações um diretor específico pode realizar no AWS IoT recurso e sob quais condições.

AWS IoT não é compatível com políticas baseadas em recursos do IAM. No entanto, ele apóia políticas AWS IoT baseadas em recursos. Para obter mais informações, consulte AWS IoT Core políticas.

Autorização baseada em AWS IoT tags

Você pode anexar tags a AWS IoT recursos ou passar tags em uma solicitação para AWS IoT. Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `iot:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou chaves de condição `aws:TagKeys`. Para obter mais informações, consulte Utilização de tags com políticas do IAM. Para obter mais informações sobre a marcação de AWS IoT recursos, consulteMarcando seus recursos AWS IoT.

Para visualizar um exemplo de política baseada em identidade que visa limitar o acesso a um recurso baseado nas tags desse recurso, consulte Visualizando AWS IoT recursos com base em tags.

AWS IoT Funções do IAM

Uma função do IAM é uma entidade dentro da sua Conta da AWS que tem permissões específicas.

Usando credenciais temporárias com AWS IoT

É possível usar credenciais temporárias para fazer login com federação, assumir um perfil do IAM ou assumir um perfil entre contas. Você obtém credenciais de segurança temporárias chamando operações de AWS STS API, como AssumeRoleou GetFederationToken.

AWS IoT suporta o uso de credenciais temporárias.

Perfis vinculados ao serviço

As funções vinculadas ao serviço permitem que AWS os serviços acessem recursos em outros serviços para concluir uma ação em seu nome. Os perfis vinculados a serviço aparecem em sua conta do IAM e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para perfis vinculados a serviço.

AWS IoT não oferece suporte a funções vinculadas a serviços.

Perfis de serviço

Esse atributo permite que um serviço assuma um perfil de serviço em seu nome. O perfil permite que o serviço acesse recursos em outros serviços para concluir uma ação em seu nome. Os perfis de serviço aparecem em sua conta do IAM e são de propriedade da conta. Isso significa que um administrador do IAM pode alterar as permissões para esse perfil. Porém, fazer isso pode alterar a funcionalidade do serviço.

AWS IoT exemplos de políticas baseadas em identidade

Por padrão, os usuários e os perfis do IAM não têm permissão para criar ou modificar recursos do AWS IoT . Eles também não podem realizar tarefas usando a AWS API Console de gerenciamento da AWS AWS CLI, ou. Um administrador do IAM deve criar políticas do IAM que concedam aos usuários e perfis permissão para executarem operações de API específicas nos recursos especificados de que precisam. O administrador deve anexar essas políticas aos usuários ou grupos que exigem essas permissões.

Para saber como criar uma política baseada em identidade do IAM usando esses exemplos de documentos de política JSON, consulte Criar políticas na guia JSON no Guia do usuário do IAM.

Tópicos

- Práticas recomendadas de política
- Usando o AWS IoT console
- Permitir que os usuários visualizem suas próprias permissões
- Visualizando AWS IoT recursos com base em tags
- Visualizando recursos AWS IoT do Device Advisor com base em tags

Práticas recomendadas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir AWS IoT recursos em sua conta. Essas ações podem incorrer em custos para sua Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com políticas AWS gerenciadas e avance para permissões de privilégios mínimos — Para começar a conceder permissões para seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para saber mais, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para saber mais sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: é possível adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, é possível escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como CloudFormation. Para saber mais, consulte Elementos da política JSON do IAM: condição no Guia do usuário do IAM.

- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações práticas para ajudar a criar políticas seguras e funcionais. Para saber mais, consulte Validação de políticas do IAM Access Analyzer no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para saber mais, consulte Configuração de acesso à API protegido por MFA no Guia do Usuário do IAM.

Para saber mais sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Usando o AWS IoT console

Para acessar o AWS IoT console, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os AWS IoT recursos em seu Conta da AWS. Caso crie uma política baseada em identidade mais restritiva que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Para garantir que essas entidades ainda possam usar o AWS IoT console, anexe também a seguinte política AWS gerenciada às entidades:`AWSIoTFullAccess`. Para obter mais informações, consulte Adicionar permissões a um usuário no Guia do usuário do IAM.

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas somente para a API AWS CLI ou para a AWS API. Em vez disso, permita o acesso somente às ações que correspondem à operação da API que você está tentando executar.

Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou AWS .

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Visualizando AWS IoT recursos com base em tags

É possível utilizar condições na política baseada em identidade para controlar o acesso aos recursos do AWS IoT com base em tags. Este exemplo mostra como é possível criar uma política que permite visualizar um objeto. No entanto, a permissão é concedida somente se a tag do objeto `Owner` tiver o valor do nome de usuário desse usuário. Essa política também concede as permissões necessárias concluir essa ação no console.

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "ListBillingGroupsInConsole",
            "Effect": "Allow",
            "Action": "iot:ListBillingGroups",
            "Resource": "*"
        },
        {
            "Sid": "ViewBillingGroupsIfOwner",
            "Effect": "Allow",
           "Action": "iot:DescribeBillingGroup",
            "Resource": "arn:aws:iot:*:*:billinggroup/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
        }
        }
    ]
}
```

É possível anexar essa política aos usuários do IAM na sua conta. Se um usuário chamado `richard-roe` tentar visualizar um grupo de AWS IoT cobrança, o grupo de cobrança deverá estar marcado com `Owner=richard-roe` ou. `owner=richard-roe` Caso contrário, ele terá o acesso negado. A chave da tag de condição `Owner` corresponde a `Owner` e a `owner` porque os nomes das chaves de condição não fazem distinção entre maiúsculas e minúsculas. Para obter mais informações, consulte IAM JSON Policy Elements: Condition (Elementos da política JSON do IAM: Condição) no Guia do usuário do IAM.

Visualizando recursos AWS IoT do Device Advisor com base em tags

Você pode usar condições em sua política baseada em identidade para controlar o acesso aos recursos do AWS IoT Device Advisor com base em tags. O exemplo a seguir mostra como criar uma política que permite visualizar uma definição de suíte específica. No entanto, a permissão será concedida somente se a tag de definição de suíte tiver `SuiteType` definido como o valor de `MQTT`. Essa política também concede as permissões necessárias concluir essa ação no console.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ViewSuiteDefinition",
```

```
            "Effect": "Allow",
            "Action": "iotdeviceadvisor:GetSuiteDefinition",
            "Resource": "arn:aws:iotdeviceadvisor:*:*:suitedefinition/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/SuiteType": "MQTT"}
        }
        }
    ]
}
```

AWS políticas gerenciadas para AWS IoT

Para adicionar permissões a usuários, grupos e funções, é mais fácil usar políticas AWS gerenciadas do que escrever políticas você mesmo. É necessário tempo e experiência para criar políticas gerenciadas pelo cliente do IAM que fornecem à sua equipe apenas as permissões de que precisam. Para começar rapidamente, você pode usar nossas políticas AWS gerenciadas. Essas políticas abrangem casos de uso comuns e estão disponíveis na sua Conta da AWS. Para obter mais informações sobre políticas AWS gerenciadas, consulte políticas AWS gerenciadas no Guia do usuário do IAM.

AWS os serviços mantêm e atualizam as políticas AWS gerenciadas. Você não pode alterar as permissões nas políticas AWS gerenciadas. Os serviços ocasionalmente acrescentam permissões adicionais a uma política gerenciada pela AWS para oferecer suporte a novos recursos. Esse tipo de atualização afeta todas as identidades (usuários, grupos e funções) em que a política está anexada. É mais provável que os serviços atualizem uma política gerenciada pela AWS quando um novo recurso for iniciado ou novas operações se tornarem disponíveis. Os serviços não removem as permissões de uma política AWS gerenciada, portanto, as atualizações de políticas não violarão suas permissões existentes.

Além disso, AWS oferece suporte a políticas gerenciadas para funções de trabalho que abrangem vários serviços. Por exemplo, a política ReadOnlyAccess AWS gerenciada fornece acesso somente de leitura a todos os AWS serviços e recursos. Quando um serviço inicia um novo atributo, a AWS adiciona permissões somente leitura para novas operações e atributos. Para obter uma lista e descrições das políticas de perfis de trabalho, consulte Políticas gerenciadas pela AWS para perfis de trabalho no Guia do usuário do IAM.

> **Note**
>
> AWS IoT funciona com ambas as políticas AWS IoT e do IAM. Este tópico discute somente as políticas do IAM, que definem uma ação política para operações de API do ambiente de gerenciamento e do plano de dados. Consulte também AWS IoT Core políticas.

AWS política gerenciada: AWSIo TConfig Acesso

É possível anexar a política `AWSIoTConfigAccess` às suas identidades do IAM.

Essa política concede as permissões de identidade associadas que permitem acesso a todas as operações de configuração de AWS IoT . Essa política pode afetar o processamento de dados e armazenamento. Para ver essa política no Console de gerenciamento da AWS, consulte AWSIoTConfigAccess.

Detalhes das permissões

Esta política inclui as seguintes permissões.

- `iot`— Recupere AWS IoT dados e execute ações de configuração de IoT.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:AcceptCertificateTransfer",
                "iot:AddThingToThingGroup",
                "iot:AssociateTargetsWithJob",
                "iot:AttachPolicy",
                "iot:AttachPrincipalPolicy",
```

```
                "iot:AttachThingPrincipal",
                "iot:CancelCertificateTransfer",
                "iot:CancelJob",
                "iot:CancelJobExecution",
                "iot:ClearDefaultAuthorizer",
                "iot:CreateAuthorizer",
                "iot:CreateCertificateFromCsr",
                "iot:CreateJob",
                "iot:CreateKeysAndCertificate",
                "iot:CreateOTAUpdate",
                "iot:CreatePolicy",
                "iot:CreatePolicyVersion",
                "iot:CreateRoleAlias",
                "iot:CreateStream",
                "iot:CreateThing",
                "iot:CreateThingGroup",
                "iot:CreateThingType",
                "iot:CreateTopicRule",
                "iot:DeleteAuthorizer",
                "iot:DeleteCACertificate",
                "iot:DeleteCertificate",
                "iot:DeleteJob",
                "iot:DeleteJobExecution",
                "iot:DeleteOTAUpdate",
                "iot:DeletePolicy",
                "iot:DeletePolicyVersion",
                "iot:DeleteRegistrationCode",
                "iot:DeleteRoleAlias",
                "iot:DeleteStream",
                "iot:DeleteThing",
                "iot:DeleteThingGroup",
                "iot:DeleteThingType",
                "iot:DeleteTopicRule",
                "iot:DeleteV2LoggingLevel",
                "iot:DeprecateThingType",
                "iot:DescribeAuthorizer",
                "iot:DescribeCACertificate",
                "iot:DescribeCertificate",
                "iot:DescribeDefaultAuthorizer",
                "iot:DescribeEndpoint",
                "iot:DescribeEventConfigurations",
                "iot:DescribeIndex",
                "iot:DescribeJob",
                "iot:DescribeJobExecution",
```

```
                "iot:DescribeRoleAlias",
                "iot:DescribeStream",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:DescribeThingRegistrationTask",
                "iot:DescribeThingType",
                "iot:DetachPolicy",
                "iot:DetachPrincipalPolicy",
                "iot:DetachThingPrincipal",
                "iot:DisableTopicRule",
                "iot:EnableTopicRule",
                "iot:GetEffectivePolicies",
                "iot:GetIndexingConfiguration",
                "iot:GetJobDocument",
                "iot:GetLoggingOptions",
                "iot:GetOTAUpdate",
                "iot:GetPolicy",
                "iot:GetPolicyVersion",
                "iot:GetRegistrationCode",
                "iot:GetTopicRule",
                "iot:GetV2LoggingOptions",
                "iot:ListAttachedPolicies",
                "iot:ListAuthorizers",
                "iot:ListCACertificates",
                "iot:ListCertificates",
                "iot:ListCertificatesByCA",
                "iot:ListIndices",
                "iot:ListJobExecutionsForJob",
                "iot:ListJobExecutionsForThing",
                "iot:ListJobs",
                "iot:ListOTAUpdates",
                "iot:ListOutgoingCertificates",
                "iot:ListPolicies",
                "iot:ListPolicyPrincipals",
                "iot:ListPolicyVersions",
                "iot:ListPrincipalPolicies",
                "iot:ListPrincipalThings",
                "iot:ListRoleAliases",
                "iot:ListStreams",
                "iot:ListTargetsForPolicy",
                "iot:ListThingGroups",
                "iot:ListThingGroupsForThing",
                "iot:ListThingPrincipals",
                "iot:ListThingRegistrationTaskReports",
```

```
                "iot:ListThingRegistrationTasks",
                "iot:ListThings",
                "iot:ListThingsInThingGroup",
                "iot:ListThingTypes",
                "iot:ListTopicRules",
                "iot:ListV2LoggingLevels",
                "iot:RegisterCACertificate",
                "iot:RegisterCertificate",
                "iot:RegisterThing",
                "iot:RejectCertificateTransfer",
                "iot:RemoveThingFromThingGroup",
                "iot:ReplaceTopicRule",
                "iot:SearchIndex",
                "iot:SetDefaultAuthorizer",
                "iot:SetDefaultPolicyVersion",
                "iot:SetLoggingOptions",
                "iot:SetV2LoggingLevel",
                "iot:SetV2LoggingOptions",
                "iot:StartThingRegistrationTask",
                "iot:StopThingRegistrationTask",
                "iot:TestAuthorization",
                "iot:TestInvokeAuthorizer",
                "iot:TransferCertificate",
                "iot:UpdateAuthorizer",
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:UpdateEventConfigurations",
                "iot:UpdateIndexingConfiguration",
                "iot:UpdateRoleAlias",
                "iot:UpdateStream",
                "iot:UpdateThing",
                "iot:UpdateThingGroup",
                "iot:UpdateThingGroupsForThing",
                "iot:UpdateAccountAuditConfiguration",
                "iot:DescribeAccountAuditConfiguration",
                "iot:DeleteAccountAuditConfiguration",
                "iot:StartOnDemandAuditTask",
                "iot:CancelAuditTask",
                "iot:DescribeAuditTask",
                "iot:ListAuditTasks",
                "iot:CreateScheduledAudit",
                "iot:UpdateScheduledAudit",
                "iot:DeleteScheduledAudit",
                "iot:DescribeScheduledAudit",
```

```
                "iot:ListScheduledAudits",
                "iot:ListAuditFindings",
                "iot:CreateSecurityProfile",
                "iot:DescribeSecurityProfile",
                "iot:UpdateSecurityProfile",
                "iot:DeleteSecurityProfile",
                "iot:AttachSecurityProfile",
                "iot:DetachSecurityProfile",
                "iot:ListSecurityProfiles",
                "iot:ListSecurityProfilesForTarget",
                "iot:ListTargetsForSecurityProfile",
                "iot:ListActiveViolations",
                "iot:ListViolationEvents",
                "iot:ValidateSecurityProfileBehaviors"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS política gerenciada: AWSIo TConfig ReadOnlyAccess

É possível anexar a política `AWSIoTConfigReadOnlyAccess` às suas identidades do IAM.

Essa política concede as permissões de identidade associadas que permitem acesso somente de leitura a todas as operações de configuração de AWS IoT . Para ver essa política no Console de gerenciamento da AWS, consulte AWSIoTConfigReadOnlyAccess.

Detalhes das permissões

Esta política inclui as seguintes permissões.

- `iot` – Executar operações somente de leitura das ações de configuração de IoT.

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:DescribeAuthorizer",
                "iot:DescribeCACertificate",
                "iot:DescribeCertificate",
                "iot:DescribeDefaultAuthorizer",
                "iot:DescribeEndpoint",
                "iot:DescribeEventConfigurations",
                "iot:DescribeIndex",
                "iot:DescribeJob",
                "iot:DescribeJobExecution",
                "iot:DescribeRoleAlias",
                "iot:DescribeStream",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:DescribeThingRegistrationTask",
                "iot:DescribeThingType",
                "iot:GetEffectivePolicies",
                "iot:GetIndexingConfiguration",
                "iot:GetJobDocument",
                "iot:GetLoggingOptions",
                "iot:GetOTAUpdate",
                "iot:GetPolicy",
                "iot:GetPolicyVersion",
                "iot:GetRegistrationCode",
                "iot:GetTopicRule",
                "iot:GetV2LoggingOptions",
                "iot:ListAttachedPolicies",
                "iot:ListAuthorizers",
                "iot:ListCACertificates",
                "iot:ListCertificates",
                "iot:ListCertificatesByCA",
                "iot:ListIndices",
                "iot:ListJobExecutionsForJob",
                "iot:ListJobExecutionsForThing",
                "iot:ListJobs",
                "iot:ListOTAUpdates",
                "iot:ListOutgoingCertificates",
                "iot:ListPolicies",
                "iot:ListPolicyPrincipals",
                "iot:ListPolicyVersions",
                "iot:ListPrincipalPolicies",
```

```
                "iot:ListPrincipalThings",
                "iot:ListRoleAliases",
                "iot:ListStreams",
                "iot:ListTargetsForPolicy",
                "iot:ListThingGroups",
                "iot:ListThingGroupsForThing",
                "iot:ListThingPrincipals",
                "iot:ListThingRegistrationTaskReports",
                "iot:ListThingRegistrationTasks",
                "iot:ListThings",
                "iot:ListThingsInThingGroup",
                "iot:ListThingTypes",
                "iot:ListTopicRules",
                "iot:ListV2LoggingLevels",
                "iot:SearchIndex",
                "iot:TestAuthorization",
                "iot:TestInvokeAuthorizer",
                "iot:DescribeAccountAuditConfiguration",
                "iot:DescribeAuditTask",
                "iot:ListAuditTasks",
                "iot:DescribeScheduledAudit",
                "iot:ListScheduledAudits",
                "iot:ListAuditFindings",
                "iot:DescribeSecurityProfile",
                "iot:ListSecurityProfiles",
                "iot:ListSecurityProfilesForTarget",
                "iot:ListTargetsForSecurityProfile",
                "iot:ListActiveViolations",
                "iot:ListViolationEvents",
                "iot:ValidateSecurityProfileBehaviors"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS política gerenciada: AWSIo TData Acesso

É possível anexar a política `AWSIoTDataAccess` às suas identidades do IAM.

Essa política concede às identidades associadas permissões que permitem o acesso a todas as operações AWS IoT de dados. As operações de dados enviam dados por protocolos MQTT ou HTTP. Para visualizar essa política no Console de gerenciamento da AWS, consulte `AWSIoTDataAccess`.

Detalhes das permissões

Esta política inclui as seguintes permissões.

- `iot`— Recupere AWS IoT dados e permita acesso total às ações AWS IoT de mensagens.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect",
                "iot:Publish",
                "iot:Subscribe",
                "iot:Receive",
                "iot:GetThingShadow",
                "iot:UpdateThingShadow",
                "iot:DeleteThingShadow",
                "iot:ListNamedShadowsForThing"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS política gerenciada: AWSIo TFull Acesso

É possível anexar a política `AWSIoTFullAccess` às suas identidades do IAM.

Essa política concede as permissões de identidade associadas que permitem acesso a todas as operações de configuração e mensagens de AWS IoT . Para ver essa política no Console de gerenciamento da AWS, consulte `AWSIoTFullAccess`.

Detalhes das permissões

Esta política inclui as seguintes permissões.

- `iot`— recupere AWS IoT dados e permita acesso total às ações AWS IoT de configuração e mensagens.
- `iotjobsdata`— Recupere os dados do AWS IoT Jobs e permita acesso total às operações da API do plano de dados do AWS IoT Jobs.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:*",
                "iotjobsdata:*"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS política gerenciada: AWSIo TLogging

É possível anexar a política `AWSIoTLogging` às suas identidades do IAM.

Essa política concede às permissões de identidade associadas que permitem o acesso para criar grupos do Amazon CloudWatch Logs e transmitir registros para os grupos. Essa política está

anexada à sua função de CloudWatch registro. Para ver essa política no Console de gerenciamento da AWS, consulte `AWSIoTLogging`.

Detalhes das permissões

Esta política inclui as seguintes permissões.

- `logs`— Recuperar CloudWatch registros. Também permite a criação de grupos de CloudWatch registros e registros de streaming para os grupos.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DeleteLogStream"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWS política gerenciada: AWSIo TOTAUpdate

É possível anexar a política `AWSIoTOTAUpdate` às suas identidades do IAM.

Essa política concede às identidades associadas permissões que permitem acesso para criar AWS IoT trabalhos, trabalhos de assinatura de AWS IoT código e descrever trabalhos de assinantes de AWS código. Para ver essa política no Console de gerenciamento da AWS, consulte `AWSIoTOTAUpdate`.

Detalhes das permissões

Esta política inclui as seguintes permissões.

- `iot`— Crie AWS IoT trabalhos e trabalhos de assinatura de código.
- `signer`— Execute a criação de trabalhos de assinante de AWS código.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iot:CreateJob",
            "signer:DescribeSigningJob"
        ],
        "Resource": "*"
    }
}
```

AWS política gerenciada: AWSIo TRule ações

É possível anexar a política `AWSIoTRuleActions` às suas identidades do IAM.

Essa política concede às identidades associadas permissões que permitem acesso a tudo o que AWS service (Serviço da AWS)é suportado em ações de AWS IoT regras. Para ver essa política no Console de gerenciamento da AWS, consulte `AWSIoTRuleActions`.

Detalhes das permissões

Esta política inclui as seguintes permissões.

- `iot` - Executar ações para publicar mensagens de ação de regras.
- `dynamodb` - Inserir uma mensagem em uma tabela do DynamoDB ou divida uma mensagem em várias colunas de uma tabela do DynamoDB.
- `s3` - Armazenar um objeto em um bucket do Amazon S3.
- `kinesis` - Enviar uma mensagem para um objeto de fluxo do Amazon Kinesis.
- `firehose`: inserir um registro em um objeto de fluxo do Firehose.
- `cloudwatch`- Altere o estado do CloudWatch alarme ou envie dados da mensagem para a CloudWatch métrica.
- `sns` - Executar a operação para publicar uma notificação usando o Amazon SNS. Essa operação tem como escopo os tópicos do AWS IoT SNS.
- `sqs` - Inserir uma mensagem para adicionar à fila do SQS.
- `es`- Envie uma mensagem para o OpenSearch serviço de serviço.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "dynamodb:PutItem",
            "kinesis:PutRecord",
            "iot:Publish",
            "s3:PutObject",
            "sns:Publish",
            "sqs:SendMessage*",
            "cloudwatch:SetAlarmState",
            "cloudwatch:PutMetricData",
            "es:ESHttpPut",
            "firehose:PutRecord"
        ],
        "Resource": "*"
    }
```

```
}
```

AWS política gerenciada: AWSIo TThings Registro

É possível anexar a política `AWSIoTThingsRegistration` às suas identidades do IAM.

Essa política concede as permissões de identidade associadas que permitem o acesso para registrar objetos em massa usando a API `StartThingRegistrationTask`. Essa política pode afetar o processamento de dados e armazenamento. Para ver essa política no Console de gerenciamento da AWS, consulte `AWSIoTThingsRegistration`.

Detalhes das permissões

Esta política inclui as seguintes permissões.

- `iot` - Executar ações para criar objetos e anexar políticas e certificados ao se registrar em massa.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:AddThingToThingGroup",
                "iot:AttachPolicy",
                "iot:AttachPrincipalPolicy",
                "iot:AttachThingPrincipal",
                "iot:CreateCertificateFromCsr",
                "iot:CreatePolicy",
                "iot:CreateThing",
                "iot:DescribeCertificate",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:DescribeThingType",
                "iot:DetachPolicy",
```

```
                "iot:DetachThingPrincipal",
                "iot:GetPolicy",
                "iot:ListAttachedPolicies",
                "iot:ListPolicyPrincipals",
                "iot:ListPrincipalPolicies",
                "iot:ListPrincipalThings",
                "iot:ListTargetsForPolicy",
                "iot:ListThingGroupsForThing",
                "iot:ListThingPrincipals",
                "iot:RegisterCertificate",
                "iot:RegisterThing",
                "iot:RemoveThingFromThingGroup",
                "iot:UpdateCertificate",
                "iot:UpdateThing",
                "iot:UpdateThingGroupsForThing",
                "iot:AddThingToBillingGroup",
                "iot:DescribeBillingGroup",
                "iot:RemoveThingFromBillingGroup"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWS IoT atualizações nas políticas AWS gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas AWS IoT desde que esse serviço começou a rastrear essas alterações. Para receber alertas automáticos sobre alterações nessa página, assine o feed RSS na página Histórico do AWS IoT documento.

Alteração	Descrição	Data
AWSIoTFullAcesso — Atualização de uma política existente	AWS IoT adicionou novas permissões para permitir que os usuários acessem as operações da API do plano	11 de maio de 2022

Alteração	Descrição	Data
	de dados do AWS IoT Jobs usando o protocolo HTTP. Um novo prefixo de política do IAM,`iotjobsdata:` , fornece um controle de acesso mais refinado para acessar os endpoints do plano de dados de AWS IoT Jobs. Para operações de API do ambiente de gerenciamento, o prefixo `iot:` ainda é usado. Para obter mais informaçõ es, consulte AWS IoT Core políticas para o protocolo HTTPS.	
AWS IoT começou a rastrear as alterações	AWS IoT começou a rastrear as mudanças em suas políticas AWS gerenciadas.	11 de maio de 2022

Solução de problemas AWS IoT de identidade e acesso

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com AWS IoT um IAM.

Tópicos

- Não estou autorizado a realizar uma ação em AWS IoT
- Não estou autorizado a realizar iam: PassRole
- Quero permitir que pessoas fora da minha Conta da AWS acessem meus AWS IoT recursos

Não estou autorizado a realizar uma ação em AWS IoT

Se você receber uma mensagem de erro informando que não tem autorização para executar uma ação, suas políticas deverão ser atualizadas para permitir que você realize a ação.

O erro do exemplo a seguir ocorre quando o usuário do IAM `mateojackson` tenta usar o console para visualizar detalhes sobre um recurso de objeto, mas não tem as permissões `iot:DescribeThing`.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iot:DescribeThing on resource: MyIoTThing
```

Nesse caso, a política do usuário `mateojackson` deve ser atualizada para permitir o acesso ao recurso de objeto usando a ação `iot:DescribeThing`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Usando o AWS IoT Device Advisor

Se você estiver usando o AWS IoT Device Advisor, o exemplo de erro a seguir ocorre quando o usuário `mateojackson` tenta usar o console para visualizar detalhes sobre a definição de uma suíte, mas não tem as `iotdeviceadvisor:GetSuiteDefinition` permissões.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iotdeviceadvisor:GetSuiteDefinition on resource: MySuiteDefinition
```

Nesse caso, a política do usuário `mateojackson` deve ser atualizada para permitir o acesso ao recurso `MySuiteDefinition` usando a ação `iotdeviceadvisor:GetSuiteDefinition`.

Não estou autorizado a realizar iam: PassRole

Se você receber uma mensagem de erro informando que não está autorizado a executar a ação `iam:PassRole`, as suas políticas devem ser atualizadas para permitir que você passe uma função para o AWS IoT.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazê-lo, você deve ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta utilizar o console para executar uma ação no AWS IoT. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas fora da minha Conta da AWS acessem meus AWS IoT recursos

É possível criar um perfil que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se é AWS IoT compatível com esses recursos, consulteComo AWS IoT funciona com o IAM.
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como fornecer acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Registro em log e monitoramento

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho de AWS IoT suas AWS soluções. Você deve coletar dados de monitoramento de todas as partes da sua AWS solução para poder depurar com mais facilidade uma falha multiponto, caso

ocorra. Para obter informações sobre procedimentos de registro em log e monitoramento, consulte Monitoramento AWS IoT

Ferramentas de monitoramento

AWS fornece ferramentas que você pode usar para monitorar AWS IoT. Você pode configurar algumas dessas ferramentas para que façam o monitoramento para você. Algumas das ferramentas exigem intervenção manual. Recomendamos que as tarefas de monitoramento sejam automatizadas ao máximo possível.

Ferramentas de monitoramento automatizadas

Você pode usar as seguintes ferramentas de monitoramento automatizado para observar AWS IoT e relatar quando algo está errado:

- Amazon CloudWatch Alarms — Observe uma única métrica em um período especificado por você e execute uma ou mais ações com base no valor da métrica em relação a um determinado limite em vários períodos. A ação é uma notificação enviada para um tópico do Amazon Simple Notification Service (Amazon SNS) ou para uma política do Amazon EC2 Auto Scaling. CloudWatch os alarmes não invocam ações simplesmente porque estão em um estado específico. O estado deve ter sido alterado e mantido por uma quantidade especificada de períodos. Para obter mais informações, consulte Monitore AWS IoT alarmes e métricas usando a Amazon CloudWatch.
- Amazon CloudWatch Logs — Monitore, armazene e acesse seus arquivos de log de AWS CloudTrail ou de outras fontes. O Amazon CloudWatch Logs também permite que você veja as etapas críticas que os casos de teste do AWS IoT Device Advisor realizam, eventos gerados e mensagens MQTT enviadas de seus dispositivos ou AWS IoT Core durante a execução do teste. Esses logs possibilitam a depuração e a execução de ações corretivas em seus dispositivos. Para obter mais informações, consulte Monitore AWS IoT usando CloudWatch registros Para obter mais informações sobre o uso da Amazon CloudWatch, consulte Monitoramento de arquivos de log no Guia CloudWatch do usuário da Amazon.
- Amazon CloudWatch Events — Combine eventos e encaminhe-os para uma ou mais funções ou streams de destino para fazer alterações, capturar informações de estado e tomar medidas corretivas. Para obter mais informações, consulte O que são CloudWatch eventos da Amazon no Guia CloudWatch do usuário da Amazon.
- AWS CloudTrail Monitoramento de registros — compartilhe arquivos de log entre contas, monitore arquivos de CloudTrail log em tempo real enviando-os para o CloudWatch Logs, grave aplicativos de processamento de log em Java e valide se seus arquivos de log não foram alterados após a

entrega. CloudTrail Para obter mais informações, consulte Registrando chamadas de AWS IoT API usando AWS CloudTrail e também Como trabalhar com arquivos de CloudTrail log no Guia AWS CloudTrail do usuário.

Ferramentas de monitoramento manual

Outra parte importante do monitoramento AWS IoT envolve o monitoramento manual dos itens que os CloudWatch alarmes não cobrem. O AWS IoT, CloudWatch, e outros painéis do console de AWS serviço fornecem uma at-a-glance visão do estado do seu AWS ambiente. Recomendamos que você também verifique os arquivos de log AWS IoT.

- AWS IoT o painel mostra:
  - certificados CA
  - Certificados
  - Políticas
  - Regras
  - Objetos
- CloudWatch a página inicial mostra:
  - Alarmes e status atual.
  - Gráficos de alarmes e recursos.
  - Estado de integridade do serviço.

  Você pode usar CloudWatch para fazer o seguinte:
  - Criar painéis personalizados para monitorar os serviços com os quais você se preocupa.
  - Colocar em gráfico dados de métrica para solucionar problemas e descobrir tendências.
  - Pesquise e navegue por todas as suas métricas AWS de recursos.
  - Criar e editar alertas para ser notificado sobre problemas.

Validação de conformidade para AWS IoT Core

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte Serviços da AWS Escopo por Programa de Conformidade Serviços da AWS e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. Para obter mais informações sobre sua responsabilidade de conformidade ao usar Serviços da AWS, consulte a documentação AWS de segurança.

Resiliência no AWS IoT Core

A infraestrutura AWS global é construída em torno de Região da AWS s e zonas de disponibilidade. Região da AWS s fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e altamente redundantes. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que executam o failover automaticamente entre as zonas de disponibilidade sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre Região da AWS s e zonas de disponibilidade, consulte Infraestrutura AWS global.

AWS IoT Core armazena informações sobre seus dispositivos no registro do dispositivo. Ele também armazena certificados da CA, certificados do dispositivo e dados de sombra do dispositivo. No caso de falhas de hardware ou de rede, esses dados são replicados automaticamente entre as zonas de disponibilidade, mas não entre as regiões.

AWS IoT Core publica eventos do MQTT quando o registro do dispositivo é atualizado. É possível usar essas mensagens para fazer backup de seus dados de registro e salvá-los em outro lugar, como uma tabela do DynamoDB. Você é responsável por salvar os certificados que AWS IoT Core cria para você ou aqueles que você mesmo cria. O Device Shadow armazena dados de estado sobre seus dispositivos e pode ser reenviado quando um dispositivo fica online novamente. AWS IoT O Device Advisor armazena informações sobre a configuração da sua suíte de testes. Esses dados são replicados automaticamente no caso de falhas de hardware ou de rede.

AWS IoT Core os recursos são específicos da região e não são replicados, a Regiões da AWS menos que você o faça especificamente.

Para obter mais informações sobre as práticas recomendadas de segurança, consulte Melhores práticas de segurança em AWS IoT Core.

Usando AWS IoT Core com interface VPC endpoints

Com AWS IoT Core, você pode criar endpoints de plano de controle de IoT e endpoints de dados de IoT em sua nuvem privada virtual (VPC) usando endpoints de VPC de interface. Os endpoints VPC de interface são alimentados por AWS PrivateLink uma AWS tecnologia que você pode usar para acessar serviços em execução AWS usando endereços IP privados. Para obter mais informações, consulte o Amazon Virtual Private Cloud.

Para conectar dispositivos em campo em redes remotas, como uma rede corporativa, à sua Amazon VPC, consulte as opções listadas na matriz de conectividade da Network-to-Amazon VPC.

Conteúdo

- Criação de endpoints VPC para o plano de controle AWS IoT Core
- Criação de endpoints VPC para plano de dados AWS IoT Core
- Criação de endpoints VPC para provedor de credenciais AWS IoT Core
- Criar um endpoint de interface da Amazon VPC
- Configurar uma zona hospedada privada
- Controlando o acesso a AWS IoT Core mais de VPC endpoints
- Limitações
- Escalando endpoints de VPC com AWS IoT Core
- Usar domínios personalizados com endpoints da VPC
- Disponibilidade de VPC endpoints para AWS IoT Core
- Usando tunelamento AWS IoT Device Management seguro com endpoints de interface VPC

Criação de endpoints VPC para o plano de controle AWS IoT Core

Você pode criar um VPC endpoint para a API do plano AWS IoT Core de controle para conectar seus dispositivos a AWS IoT serviços e outros serviços. AWS Para começar a usar VPC endpoints, crie uma interface VPC endpoint e selecione como serviço. AWS IoT Core AWS Se você estiver usando a CLI, primeiro ligue describe-vpc-endpoint-servicespara garantir que está escolhendo uma zona de disponibilidade onde AWS IoT Core esteja presente em sua área específica. Região da AWS Por exemplo, na região us-east-1, esse comando ficaria da seguinte maneira:

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-east-1.iot.api
```

Veja as instruções detalhadas abaixo para criar um endpoint de interface Amazon VPC para AWS IoT Core o plano de controle.

Criação de endpoints VPC para plano de dados AWS IoT Core

Você pode criar um VPC endpoint para API de plano de AWS IoT Core dados para conectar seus dispositivos a AWS IoT serviços e outros serviços. AWS Para começar a usar VPC endpoints, crie uma interface VPC endpoint e selecione como serviço. AWS IoT Core AWS Se você estiver usando a CLI, primeiro ligue describe-vpc-endpoint-servicespara garantir que está escolhendo uma zona de disponibilidade onde AWS IoT Core esteja presente em sua área específica. Região da AWS Por exemplo, na região us-east-1, esse comando ficaria da seguinte maneira:

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-east-1.iot.data
```

> **Note**
>
> O atributo de VPC para criar automaticamente um registro DNS está desativado. Para se conectar a esses endpoints, é necessário criar manualmente um registro DNS privado. Para obter mais informações sobre registros DNS de VPC privada, consulte DNS privado para endpoints de interface. Para obter mais informações sobre as limitações da AWS IoT Core VPC, consulte. Limitações

Para conectar clientes MQTT às interfaces de endpoint da VPC:

- É necessário criar manualmente registros DNS em uma zona hospedada privada vinculada à sua VPC. Para começar, consulte Criar uma zona hospedada privada.

- Na sua zona hospedada privada, crie um registro de alias para cada IP de interface de rede elástica para o endpoint da VPC. Se você tiver várias interfaces de rede IPs para vários VPC endpoints, crie registros DNS ponderados com pesos iguais em todos os registros ponderados. Esses endereços IP estão disponíveis na chamada da DescribeNetworkInterfacesAPI quando filtrados pelo ID do VPC endpoint no campo de descrição.

Veja as instruções detalhadas abaixo para criar um endpoint de interface Amazon VPC e configurar a zona hospedada privada para AWS IoT Core o plano de dados.

Criação de endpoints VPC para provedor de credenciais AWS IoT Core

Você pode criar um VPC endpoint para que o provedor de AWS IoT Core credenciais conecte dispositivos usando a autenticação baseada em certificado do cliente e obtenha credenciais temporárias AWS no formato Signature versão 4.AWS Para começar a usar os VPC endpoints para o provedor de AWS IoT Core credenciais, execute o comando da create-vpc-endpointCLI para criar uma interface VPC endpoint e selecionar o provedor de credenciais como o serviço. AWS IoT Core AWS Para garantir que você esteja escolhendo uma zona de disponibilidade onde AWS IoT Core esteja presente em sua área específica Região da AWS, primeiro execute o describe-vpc-endpoint-servicescomando. Por exemplo, na região us-east-1, esse comando ficaria da seguinte maneira:

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-
east-1.iot.credentials
```

> **Note**
>
> O atributo de VPC para criar automaticamente um registro DNS está desativado. Para se conectar a esses endpoints, é necessário criar manualmente um registro DNS privado. Para obter mais informações sobre registros DNS de VPC privada, consulte DNS privado para endpoints de interface. Para obter mais informações sobre as limitações da AWS IoT Core VPC, consulte. Limitações

Para conectar clientes HTTP às interfaces de endpoint da VPC:

- É necessário criar manualmente registros DNS em uma zona hospedada privada vinculada à sua VPC. Para começar, consulte Criar uma zona hospedada privada.

- Na sua zona hospedada privada, crie um registro de alias para cada IP de interface de rede elástica para o endpoint da VPC. Se você tiver várias interfaces de rede IPs para vários VPC endpoints, crie registros DNS ponderados com pesos iguais em todos os registros ponderados. Esses endereços IP estão disponíveis na chamada da DescribeNetworkInterfacesAPI quando filtrados pelo ID do VPC endpoint no campo de descrição.

Veja as instruções detalhadas abaixo para criar um endpoint de interface Amazon VPC e configurar a zona hospedada privada para AWS IoT Core o provedor de credenciais.

Criar um endpoint de interface da Amazon VPC

Você pode criar uma interface VPC endpoint para se conectar aos AWS serviços fornecidos pelo. AWS PrivateLink Use o procedimento a seguir para criar uma interface VPC endpoint que se conecta ao plano de AWS IoT Core dados ou AWS IoT Core ao provedor de credenciais. Para obter mais informações, consulte Acessar um AWS serviço usando uma interface VPC endpoint.

> **Note**
>
> Os processos para criar um endpoint de interface Amazon VPC para plano de AWS IoT Core dados e provedor de AWS IoT Core credenciais são semelhantes, mas você deve fazer alterações específicas no endpoint para que a conexão funcione.

Para criar um endpoint da VPC de interface usando o console VPC Endpoints

1. Navegue até o console VPC Endpoints, em Nuvem privada virtual no menu à esquerda, escolha Endpoints e, em seguida, Criar endpoint.
2. Na página Criar endpoint, especifique as seguintes informações.
   - Escolha AWS service (Serviço da AWS) para a Categoria de serviço.
   - Para Nome do serviço, pesquise inserindo a palavra-chave `iot`. Na lista de serviços do `iot` exibida, escolha o endpoint.

     Se você criar um endpoint VPC para o plano de AWS IoT Core controle, escolha o endpoint da API do plano AWS IoT Core de controle para o seu. Região da AWS O endpoint será do formato `com.amazonaws.`*`region`*`.iot.api`.

     Se você criar um VPC endpoint para o plano de AWS IoT Core dados, escolha o endpoint da API do plano de AWS IoT Core dados para sua região. O endpoint será do formato `com.amazonaws.`*`region`*`.iot.data`.

     Se você criar um VPC endpoint para o provedor de AWS IoT Core credenciais, escolha o endpoint do provedor de AWS IoT Core credenciais para sua região. O endpoint será do formato `com.amazonaws.`*`region`*`.iot.credentials`.

     Se você criar um VPC endpoint para regiões do Federal Information Processing Standard (FIPS), escolha o endpoint da API FIPS para você. Região da AWS O endpoint será do formato `com.amazonaws.`*`region`*`.iot-fips.api`. Isso é apenas para o plano de controle.

> **Note**
>
> O nome do serviço para o plano de AWS IoT Core dados na região da China terá o formato`cn.com.amazonaws.`*`region`*`.iot.data`. O nome do serviço para AWS IoT Core o avião de controle na região da China terá o formato`com.amazonaws.`*`region`*`.iot.api`.

- Para VPC e sub-redes, escolha a VPC em que você deseja criar o endpoint e as zonas de disponibilidade (AZs) nas quais você deseja criar a rede de endpoints.
- Em Habilitar nome DNS, certifique-se de que Habilitar para este endpoint não esteja selecionado para plano de AWS IoT Core dados e provedor de AWS IoT Core credenciais. Nem o plano AWS IoT Core de dados nem o provedor de AWS IoT Core credenciais oferecem suporte a nomes DNS privados ainda.

  Para o plano AWS IoT Core de controle, Habilitar nome DNS é selecionado por padrão. Isso garante que todas as solicitações aos endpoints públicos do plano de AWS IoT Core controle sejam roteadas pelos endpoints da VPC. Quando isso está habilitado, você não precisa configurar uma zona hospedada privada.
- Em Grupo de segurança, selecione os grupos de segurança a serem associados às interfaces de rede do endpoint.
- Se quiser, adicione ou remova tags. As tags são pares de nome-valor usados para associar ao seu endpoint.

3. Para criar um endpoint da VPC, selecione Criar endpoint.

Depois de criar o AWS PrivateLink endpoint, na guia Detalhes do seu endpoint, você verá uma lista de nomes de DNS. Você pode usar um desses nomes DNS criados nesta seção para configurar a zona hospedada privada. Se você estiver usando o plano de AWS IoT Core controle, não precisará configurar uma zona hospedada privada.

Configurar uma zona hospedada privada

> **Note**
>
> Se você estiver usando o plano de AWS IoT Core controle e tiver o nome Ativar DNS selecionado, não precisará configurar uma zona hospedada privada. Se você desativá-la, deverá seguir este procedimento para configurar uma zona hospedada privada.

Você pode usar um desses nomes DNS criados na seção anterior para configurar a zona hospedada privada.

Para plano AWS IoT Core de dados

O nome DNS deve ser o nome de configuração do domínio ou o endpoint do `IoT:Data-ATS`. Um exemplo de nome DNS pode ser: *`xxx`*`-ats.data.iot.`*`region`*`.amazonaws.com`.

Para provedor de AWS IoT Core credenciais

O nome DNS deve estar sem o endpoint do `iot:CredentialProvider`. Um exemplo de nome DNS pode ser: *`xxxx`*`.credentials.iot.`*`region`*`.amazonaws.com`.

Para plano AWS IoT Core de controle

O nome DNS deve ser o ponto final do plano AWS IoT Core de controle. Um exemplo de nome DNS para o plano AWS IoT Core de controle é *`xxxx`*`.api.iot.`*`region`*`.amazonaws.com`.

> **Note**
>
> Os processos para configurar a zona hospedada privada para o plano de AWS IoT Core dados e o provedor de AWS IoT Core credenciais são semelhantes, mas você deve fazer alterações específicas no endpoint para que a conexão funcione.

Criar uma zona hospedada privada

Para criar uma zona hospedada privada usando o console do Route 53

1. Navegue até o console de zonas hospedadas do Route 53 e escolha Criar zona hospedada.
2. Na página Criar zona hospedada, especifique as seguintes informações.

- Em Nome do domínio, insira o endereço do endpoint do seu `iot:Data-ATS` ou o endpoint do `iot:CredentialProvider`. O comando da CLI da AWS a seguir mostra como obter o endpoint por meio de uma rede pública: `aws iot describe-endpoint --endpoint-type iot:Data-ATS`, ou `aws iot describe-endpoint --endpoint-type iot:CredentialProvider`.

  > **Note**
  >
  > Se você estiver usando domínios personalizados, consulte Usar domínios personalizados com endpoints da VPC. Os domínios personalizados não são compatíveis com o provedor de AWS IoT Core credenciais.

- Em Tipo, escolha Zona hospedada privada.
- Opcionalmente, você pode adicionar ou remover tags para associar à zona hospedada.

3. Para criar a zona hospedada privada, escolha Criar zona hospedada.

Para obter mais informações, consulte Criar uma zona hospedada privada.

Criar um registro

Depois de criar uma zona hospedada privada, é possível criar um registro que informe ao DNS como você deseja que o tráfego seja direcionado para esse domínio.

Para criar um registro

1. Na lista de zonas hospedadas exibida, escolha a zona hospedada privada que você criou antes e escolha Criar registro.
2. Use o método do assistente para criar o registro. Se o console apresentar o método de Criação rápida, escolha Alternar para assistente.
3. Escolha Roteamento simples em Política de roteamento e, em seguida, Próximo.
4. Na página Configurar registros, escolha Definir registro simples.
5. Na página Definir registro simples:
   - Em Nome do registro, insira endpoint do `iot:Data-ATS` ou endpoint do `iot:CredentialProvider`. Deve ser igual ao nome da zona hospedada privada.
   - Para o tipo de registro, se você quiser apenas IPv4 suporte, mantenha o valor comoA - `Routes traffic to an IPv4 address and some AWS resources`. Se você quiser

apenas IPv6 suporte, mantenha o valor comoAAAA `- Routes traffic to an IPv6 address and some AWS resources`. Se você quiser suporte de pilha dupla (ambos IPv4 e IPv6), crie dois registros (e AAAA na zona hospedada) com o mesmo nome de registro **A** e valor/roteamento de tráfego para.

- Em Valor/rotear tráfego para, escolha Alias para endpoint da VPC. Em seguida, escolha a Região e, em seguida, escolha o endpoint que você criou antes, conforme descrito em ??? na lista de endpoints exibida.

6. Escolha Definir registro simples para criar seu registro.

Controlando o acesso a AWS IoT Core mais de VPC endpoints

Você pode restringir o acesso ao dispositivo AWS IoT Core para que seja permitido somente por meio do VPC endpoint usando as chaves de contexto de condição da VPC. AWS IoT Core é compatível com as seguintes chaves de contexto relacionadas à VPC:

- SourceVpc
- SourceVpce
- VPCSourceIp

> **Note**
>
> AWS IoT Core não oferece suporte às políticas de endpoints para endpoints VPC.

Por exemplo, a política a seguir concede permissão para se conectar AWS IoT Core usando uma ID de cliente que corresponda ao nome da coisa e para publicar em qualquer tópico prefixado pelo nome da coisa, desde que o dispositivo se conecte a um VPC endpoint com uma determinada ID de VPC Endpoint. Essa política nega tentativas de conexão com o endpoint de dados de IoT público.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
```

```
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
            }
        }

        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.ThingName}/*"
            ]
        }
    ]
}
```

Limitações

Atualmente, os endpoints VPC são compatíveis com endpoints do plano de AWS IoT Core controle, endpoints de AWS IoT Core dados e endpointsdo provedor de credenciais.AWS IoT Core Os endpoints VPC são compatíveis somente com endpoints do Federal Information Processing Standard (FIPS) ao usar o plano de controle. AWS IoT Core

Limitações dos endpoints VPC do plano de controle de IoT

Esta seção aborda as limitações dos endpoints VPC do plano de controle de IoT.

- Os endpoints da VPC servirão somente certificados ATS.
- Domínios personalizados não são compatíveis com endpoints do plano de controle.
- Para obter informações sobre as políticas de segurança do FIPS, consulte as políticas de segurança do FIPS.

Limitações dos endpoints da VPC de dados de IoT

Esta seção aborda as limitações dos endpoints da VPC de dados de IoT.

- Os períodos de keep alive do MQTT são limitados a 230 segundos. Períodos de manutenção maiores do que isso serão automaticamente reduzidos para 230 segundos.
- Cada endpoint da VPC é compatível com um total de 100.000 dispositivos conectados simultaneamente. Se você precisar de mais conexões, consulteEscalando endpoints de VPC com AWS IoT Core.
- Os endpoints da VPC servirão somente certificados ATS, exceto para domínios personalizados.
- As Políticas de endpoint da VPC não são compatíveis.
- Para endpoints VPC criados para o plano de AWS IoT Core dados, AWS IoT Core não é compatível com o uso de registros DNS públicos zonais ou regionais.

Limitações dos endpoints do provedor de credenciais

Esta seção aborda as limitações dos endpoints da VPC do provedor de credenciais.

- Os endpoints da VPC servirão somente certificados ATS.
- As Políticas de endpoint da VPC não são compatíveis.
- Os domínios personalizados não são compatíveis com os endpoints do provedor de credenciais.
- Para endpoints VPC criados para o provedor de AWS IoT Core credenciais, AWS IoT Core não é compatível com o uso de registros DNS públicos zonais ou regionais.

Escalando endpoints de VPC com AWS IoT Core

AWS IoT Core Os endpoints VPC de interface são limitados a 100.000 dispositivos conectados em um único endpoint de interface. Se seu caso de uso exigir mais conexões simultâneas com o agente, recomendamos usar vários endpoints da VPC e rotear manualmente seus dispositivos pelos endpoints da interface. Ao criar registros DNS privados para rotear o tráfego para seus endpoints da VPC, certifique-se de criar o mesmo número de registros ponderados que os endpoints da VPC para distribuir o tráfego entre os vários endpoints.

Usar domínios personalizados com endpoints da VPC

Para usar domínios personalizados com endpoints da VPC, é necessário criar os registros de nome de domínio personalizados em uma zona hospedada privada e criar registros de roteamento no Route53. Para obter mais informações, consulte Criar uma zona hospedada privada.

> **Note**
>
> Domínios personalizados só são compatíveis com endpoints de AWS IoT Core dados.

Disponibilidade de VPC endpoints para AWS IoT Core

AWS IoT Core Os endpoints VPC da interface estão disponíveis em todas AWS IoT Core as regiões com suporte. AWS IoT Core Os endpoints VPC de interface para provedor de AWS IoT Core credenciais não são suportados na região da China e. AWS GovCloud (US) Regions

Usando tunelamento AWS IoT Device Management seguro com endpoints de interface VPC

AWS IoT Device Management o tunelamento seguro oferece suporte a endpoints de interface VPC. Você pode usar VPC endpoints para manter o tráfego entre sua VPC e AWS IoT Secure Tunneling dentro da AWS rede, sem precisar de um gateway de internet, dispositivo NAT, conexão VPN ou conexão. AWS Direct Connect

Os endpoints VPC da Interface são alimentados por AWS PrivateLinkuma tecnologia que permite acessar serviços de forma privada usando endereços IP privados. Para obter mais informações, consulte Acessar um AWS serviço usando uma interface VPC endpoint no Guia. AWS PrivateLink

Conteúdos

- Pré-requisitos
- Recebendo notificações de túnel por meio de VPC endpoints
- Criação de endpoints de VPC para tunelamento seguro
- Configurando políticas de VPC endpoint no servidor proxy
- Próximas etapas

Pré-requisitos

Antes de criar VPC endpoints para AWS IoT Secure Tunneling, verifique se você tem o seguinte:

- Uma AWS conta com as permissões necessárias para criar VPC endpoints.
- Uma VPC na sua conta. AWS
- Compreensão dos conceitos de tunelamento AWS IoT Device Management seguro.
- Familiaridade com as políticas AWS Identity and Access Management de VPC endpoint e (IAM)

Recebendo notificações de túnel por meio de VPC endpoints

Para receber notificações de túnel por meio de um VPC endpoint, seus dispositivos podem se conectar ao plano de AWS IoT Core dados por meio de um endpoint VPC e assinar o tópico MQTT reservado para tunelamento seguro.

Para obter instruções sobre como criar e configurar um VPC endpoint no plano de AWS IoT Core dados, consulte Como usar com AWS IoT Core interface VPC endpoints no Guia do desenvolvedor. AWS IoT

Criação de endpoints de VPC para tunelamento seguro

Você pode criar endpoints VPC tanto para o plano de controle de tunelamento seguro quanto para o servidor proxy.

Para criar um VPC endpoint para tunelamento seguro

1. Siga as etapas em Criação de um endpoint de interface no Amazon VPC Developer Guide
2. Em Nome do serviço, escolha uma das seguintes opções com base no seu tipo de endpoint:

   Ambiente de gerenciamento

   - Padrão: `com.amazonaws.<region>.iot.tunneling.api`
   - FIPS (disponível nas regiões FIPS): `com.amazonaws.<region>.iot-fips.tunneling.api`

   Servidor de proxy

   - Padrão: `com.amazonaws.<region>.iot.tunneling.data`

- FIPS (disponível nas regiões FIPS): `com.amazonaws.<region>.iot-fips.tunneling.data`

*<region>*Substitua pelo seu Região da AWS. Por exemplo, .`us-east-1`

3. Conclua as etapas restantes no processo de criação do VPC endpoint de acordo com seus requisitos de rede.

Configurando políticas de VPC endpoint no servidor proxy

Além da autorização baseada em token de acesso ao cliente que é usada para autorizar conexões com túneis, você pode usar políticas de endpoint de VPC para restringir ainda mais a forma como os dispositivos podem usar um endpoint de VPC para se conectar ao Secure Tunneling Proxy Server. As políticas de VPC endpoint seguem uma sintaxe semelhante à IAM e são configuradas no próprio VPC endpoint.

Observe que a única ação do IAM compatível com as políticas de endpoint VPC do servidor proxy é. `iot:ConnectToTunnel`

Abaixo estão exemplos de diferentes políticas de VPC endpoint.

Exemplos de políticas de endpoint VPC de servidor proxy

Os exemplos a seguir mostram as configurações da política de endpoint VPC do Proxy Server para casos de uso comuns.

Example- Política padrão

Essa política permite que dispositivos em sua VPC se conectem a qualquer túnel no mesmo Região da AWS local em que o endpoint é criado, em qualquer conta. AWS

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

Example- Restringir o acesso a AWS contas específicas

Essa política permite que o VPC endpoint se conecte somente a túneis em contas específicas. AWS

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": [
                "arn:aws:iot:us-east-1:111122223333:tunnel/*",
                "arn:aws:iot:us-east-1:444455556666:tunnel/*"
            ]
        }
    ]
}
```

Example- Restringir conexões por ponto final do túnel

Você pode restringir o acesso ao VPC endpoint para permitir que apenas os dispositivos se conectem à extremidade de origem ou destino de um túnel.

Somente fonte:

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iot:ClientMode": "source"
                }
            }
        }
    ]
}
```

Somente destino:

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iot:ClientMode": "destination"
                }
            }
        }
    ]
}
```

Example- Restrinja o acesso com base em tags de recursos

Essa política permite que o VPC endpoint se conecte somente a túneis marcados com um par de valores-chave específico.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Environment": "Production"
                }
            }
        }
    ]
}
```

Example- Condições de apólice combinadas

Essa política demonstra a combinação de vários elementos da política. Ele permite conexões com qualquer túnel em uma AWS conta específica, mas somente se o túnel estiver marcado

com `AllowConnectionsThroughPrivateLink` definido como `true` e o cliente não estiver se conectando à extremidade de destino do túnel.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": [
                "arn:aws:iot:us-east-1:111122223333:tunnel/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/AllowConnectionsThroughPrivateLink": "true"
                }
            }
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "iot:ConnectToTunnel",
            "Resource": [
                "arn:aws:iot:us-east-1:111122223333:tunnel/*"
            ],
            "Condition": {
                "StringEquals": {
                    "iot:ClientMode": "destination"
                }
            }
        }
    ]
}
```

Próximas etapas

Depois de criar e configurar seus VPC endpoints para AWS IoT Secure Tunneling, considere o seguinte:

- Teste sua configuração de VPC endpoint conectando dispositivos por meio do endpoint.
- Monitore o uso do VPC endpoint por meio de métricas. Amazon CloudWatch

- Revise e atualize suas políticas de VPC endpoint conforme necessário para seus requisitos de segurança.

Para obter mais informações sobre tunelamento AWS IoT Device Management seguro, consulte. AWS IoT Secure Tunneling

Segurança da infraestrutura em AWS IoT

Como um conjunto de serviços gerenciados, AWS IoT é protegido pelos procedimentos AWS globais de segurança de rede descritos no whitepaper Amazon Web Services: Visão geral dos processos de segurança.

Você usa chamadas de API AWS publicadas para acessar AWS IoT pela rede. Os clientes devem oferecer suporte a Transport Layer Security (TLS) 1.2 ou posterior. Os clientes também devem ter suporte a conjuntos de criptografia com perfect forward secrecy (PFS) como Ephemeral Diffie-Hellman (DHE) ou Ephemeral Elliptic Curve Diffie-Hellman (ECDHE). A maioria dos sistemas modernos, como o Java 7 e versões posteriores, oferece suporte a esses modos. Para obter mais informações, consulte Segurança de transporte em AWS IoT Core.

As solicitações devem ser assinadas usando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou você pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Monitoramento de segurança de frotas ou dispositivos de produção com Core AWS IoT

Frotas de IoT consistem em grandes quantidades de dispositivos com diversos recursos, duradouros e geograficamente distribuídos. Essas características tornam a configuração da frota complexa e propensa a erros. E como os dispositivos, quase sempre, têm restrições quanto à capacidade computacional e aos recursos de memória e armazenamento, isso limita o uso de criptografia e outras formas de segurança nos próprios dispositivos. Além disso, muitas vezes, os dispositivos usam software com vulnerabilidades conhecidas. Esses fatores tornam frotas de IoT um alvo atrativo para hackers e tornam difícil proteger a frota de dispositivos de forma contínua.

AWS IoT Device Defender aborda esses desafios fornecendo ferramentas para identificar problemas de segurança e desvios das melhores práticas. Você pode usar o AWS IoT Device Defender para analisar, auditar e monitorar dispositivos conectados para detectar comportamentos anormais e

mitigar os riscos de segurança. AWS IoT Device Defender pode auditar frotas de dispositivos para garantir que elas sigam as melhores práticas de segurança e detectem comportamentos anormais nos dispositivos. Isso possibilita aplicar políticas de segurança consistentes em toda a sua frota de AWS IoT dispositivos e responder rapidamente quando os dispositivos são comprometidos. Para obter mais informações, consulte AWS IoT Device Defender.

AWS IoT O Device Advisor envia atualizações e corrige sua frota conforme necessário. AWS IoT O Device Advisor atualiza os casos de teste automaticamente. Os casos de teste que você seleciona estão sempre com a versão mais recente. Para obter mais informações, consulte Device Advisor.

Melhores práticas de segurança em AWS IoT Core

Esta seção contém informações sobre as melhores práticas de segurança para AWS IoT Core. Para obter mais informações sobre regras de segurança para soluções de IoT industrial, consulte Dez regras de ouro de segurança para soluções de IoT industrial.

Protegendo conexões MQTT em AWS IoT

AWS IoT Coreé um serviço de nuvem gerenciado que possibilita que dispositivos conectados interajam com aplicativos em nuvem e outros dispositivos de forma fácil e segura. AWS IoT Core suporta HTTP e MQTT WebSocket, um protocolo de comunicação leve projetado especificamente para tolerar conexões intermitentes. Se você estiver se conectando AWS IoT usando o MQTT, cada uma de suas conexões deverá estar associada a um identificador conhecido como ID do cliente. O cliente MQTT identifica de IDs forma exclusiva as conexões MQTT. Se uma nova conexão for estabelecida usando um ID de cliente que já foi reivindicado para outra conexão, o agente de AWS IoT mensagens descarta a conexão antiga para permitir a nova conexão. O cliente IDs deve ser único em Conta da AWS cada um Região da AWS. Isso significa que você não precisa impor a exclusividade global do cliente IDs fora da sua Conta da AWS ou em todas as regiões da sua. Conta da AWS

O impacto e a gravidade de descartar conexões MQTT em sua frota de dispositivos dependem de vários fatores. Isso inclui:

- Seu caso de uso (por exemplo, os dados para os quais seus dispositivos enviam AWS IoT, a quantidade de dados e a frequência com que os dados são enviados).
- A configuração do cliente MQTT (por exemplo, as configurações de reconexão automática, as temporizações de retirada associadas e o uso de Sessões MQTT persistentes).
- Restrições de recursos de dispositivo.

- A causa raiz das desconexões, sua agressividade e persistência.

Para evitar conflitos de ID de cliente e seus possíveis impactos negativos, certifique-se de que cada dispositivo ou aplicativo móvel tenha uma política AWS IoT ou IAM que restrinja qual cliente IDs pode ser usado para conexões MQTT com o agente de AWS IoT mensagens. Por exemplo, você pode usar uma política do IAM para impedir que um dispositivo feche involuntariamente a conexão de outro dispositivo usando um ID de cliente que já esteja em uso. Para obter mais informações, consulte Autorização.

Todos os dispositivos da sua frota devem ter credenciais com privilégios que autorizem somente as ações pretendidas, que incluem (mas não se limitam a) ações do AWS IoT MQTT, como publicar mensagens ou assinar tópicos com escopo e contexto específicos. As políticas de permissão específicas podem variar para seus casos de uso. Identifique as políticas de permissão que melhor atendem aos seus requisitos de negócios e de segurança.

Para simplificar a criação e o gerenciamento de políticas de permissão, é possível usar AWS IoT Core variáveis de política e Variáveis de políticas do IAM. As variáveis de políticas podem ser colocadas em uma política e, quando a política for avaliada, as variáveis serão substituídas por valores fornecidos na solicitação do dispositivo. Usando variáveis de políticas, você pode criar uma única política para conceder permissões a vários dispositivos. Você pode identificar as variáveis de política relevantes para seu caso de uso com base na configuração AWS IoT da conta, no mecanismo de autenticação e no protocolo de rede usados na conexão com o agente de AWS IoT mensagens. No entanto, para escrever as melhores políticas de permissão, você precisa considerar informações específicas sobre seu caso de uso e o modelo de ameaça.

Por exemplo, se você registrou seus dispositivos no AWS IoT registro, você pode usar variáveis de política de coisas em AWS IoT políticas para conceder ou negar permissões com base em propriedades de coisas, como nomes de coisas, tipos de coisas e valores de atributos de coisas. O nome da coisa é obtido do ID do cliente na mensagem de conexão do MQTT enviada quando uma coisa se conecta a. AWS IoT As variáveis de política da coisa são substituídas quando uma coisa se conecta pelo MQTT usando a autenticação mútua TLS ou AWS IoT ao MQTT pelo WebSocket protocolo usando identidades autenticadas do Amazon Cognito. Você pode usar a AttachThingPrincipalAPI para anexar certificados e identidades autenticadas do Amazon Cognito a uma coisa. `iot:Connection.Thing.ThingName`é uma variável de política útil para impor restrições de ID do cliente. O exemplo de AWS IoT política a seguir exige que o nome de uma coisa registrada seja usado como ID do cliente para conexões MQTT com o agente de AWS IoT mensagens:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": "iot:Connect",
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ]
  }
 ]
}
```

Se você quiser identificar conflitos contínuos de ID de cliente, você pode ativar e usar o CloudWatch Logs for AWS IoT. Para cada conexão MQTT que o agente de AWS IoT mensagens desconecta devido a conflitos de ID do cliente, um registro de log semelhante ao seguinte é gerado:

```
{
    "timestamp": "2019-04-28 22:05:30.105",
    "logLevel": "ERROR",
    "traceId": "02a04a93-0b3a-b608-a27c-1ae8ebdb032a",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "Disconnect",
    "protocol": "MQTT",
    "clientId": "clientId01",
    "principalId": "1670fcf6de55adc1930169142405c4a2493d9eb5487127cd0091ca0193a3d3f6",
    "sourceIp": "203.0.113.1",
    "sourcePort": 21335,
    "reason": "DUPLICATE_CLIENT_ID",
    "details": "A new connection was established with the same client ID"
}
```

Você pode usar um filtro de CloudWatch registros `{$.reason= "DUPLICATE_CLIENT_ID" }` para pesquisar instâncias de conflitos de ID de cliente ou configurar filtros de CloudWatch métricas e CloudWatch alarmes correspondentes para monitoramento e geração de relatórios contínuos.

Você pode usar o AWS IoT Device Defender para identificar políticas excessivamente permissivas AWS IoT e do IAM. AWS IoT O Device Defender também fornece uma verificação de auditoria

que notifica você se vários dispositivos em sua frota estão se conectando ao agente de AWS IoT mensagens usando o mesmo ID de cliente.

Você pode usar o AWS IoT Device Advisor para validar se seus dispositivos podem se conectar de forma confiável AWS IoT Core e seguir as melhores práticas de segurança.

Consulte também

- AWS IoT Core
- Recursos de segurança do AWS IoT
- AWS IoT Core variáveis de política
- Variáveis de políticas do IAM
- Identidade do Amazon Cognito
- AWS IoT Device Defender
- CloudWatch Registros para AWS IoT

Manter o relógio do dispositivo sincronizado

É importante ter a hora exata no seu dispositivo. Os certificados X.509 têm data e hora de expiração. O relógio em seu dispositivo é usado para verificar se um certificado de servidor ainda é válido. Se você estiver criando dispositivos comerciais de IoT, lembre-se de que seus produtos podem ser armazenados por períodos prolongados antes de serem vendidos. Os relógios em tempo real podem ter desvios durante esse período, e as baterias podem ser descarregadas, portanto, definir a hora na definição de fábrica não é suficiente.

Para a maioria dos sistemas, isso indica que o software do dispositivo deve incluir um cliente de protocolo de tempo de rede (NTP). O dispositivo deve aguardar até ser sincronizado com um servidor NTP antes de tentar se conectar ao AWS IoT Core. Se isso não for possível, o sistema deve fornecer uma maneira de um usuário definir a hora do dispositivo para que as conexões subsequentes tenham êxito.

Depois que o dispositivo for sincronizado com um servidor NTP, ele poderá abrir uma conexão com o AWS IoT Core. A inclinação do relógio que é permitida depende do que você está tentando fazer com a conexão.

Validar o certificado do servidor

A primeira coisa que um dispositivo faz para interagir AWS IoT é abrir uma conexão segura. Ao conectar seu dispositivo a AWS IoT, verifique se você está falando com outro servidor AWS IoT e não se passando AWS IoT por outro servidor. Cada um dos AWS IoT servidores é provisionado com um certificado emitido para o `iot.amazonaws.com` domínio. Esse certificado foi emitido AWS IoT por uma autoridade de certificação confiável que verificou nossa identidade e propriedade do domínio.

Uma das primeiras coisas AWS IoT Core que fazemos quando um dispositivo se conecta é enviar ao dispositivo um certificado de servidor. Os dispositivos podem verificar se eles esperavam se conectar ao `iot.amazonaws.com` e se o servidor no destino dessa conexão possui um certificado de uma autoridade confiável para esse domínio.

Os certificados TLS estão no formato X.509 e incluem uma grande variedade de informações, como nome, localização, nome de domínio e um período de validade da organização. O período de validade é especificado como um par de valores de tempo chamados `notBefore` e `notAfter`. Serviços como AWS IoT Core usam períodos de validade limitados (por exemplo, um ano) para seus certificados de servidor e começam a oferecer novos antes que os antigos expirem.

Usar uma identidade única por dispositivo

Use uma única identidade por cliente. Os dispositivos geralmente usam certificados de cliente X.509. Aplicações da Web e móveis usam a identidade do Amazon Cognito. Isso permite aplicar permissões refinadas aos seus dispositivos.

Por exemplo, você tem um aplicativo que consiste em um dispositivo móvel que recebe atualizações de status de dois objetos domésticos inteligentes diferentes: uma lâmpada e um termostato. A lâmpada envia o status do nível de bateria e um termostato envia mensagens que relatam a temperatura.

AWS IoT autentica dispositivos individualmente e trata cada conexão individualmente. Você pode aplicar controles de acesso refinados usando políticas de autorização. É possível definir uma política para o termostato que permite que ele publique em um espaço de tópico. É possível definir uma política separada para a lâmpada que permite que ela publique em um espaço de tópico diferente. Por fim, é possível definir uma política para o aplicativo móvel que só permite que ele se conecte e se inscreva nos tópicos para o termostato e a lâmpada para receber mensagens desses dispositivos.

Aplique o princípio do privilégio mínimo e diminua o escopo das permissões por dispositivo o máximo possível. Todos os dispositivos ou usuários devem ter uma AWS IoT política AWS IoT que permita

somente a conexão com um ID de cliente conhecido e a publicação e assinatura de um conjunto fixo e identificado de tópicos.

Use um segundo Região da AWS como backup

Considere armazenar uma cópia dos seus dados em um segundo Região da AWS como backup. Observe que a AWS solução chamada Disaster Recovery AWS IoT for não está mais disponível. Embora a GitHubbiblioteca associada permaneça acessível, ela AWS foi descontinuada em julho de 2023 e não fornece mais manutenção ou suporte para ela. Para implementar suas próprias soluções ou explorar outras opções de suporte, acesse Entrar em contato com a AWS. Se houver um gerente AWS técnico de contas associado à sua conta, entre em contato com ele para obter ajuda.

Usar provisionamento just-in-time

Criar e provisionar manualmente cada dispositivo pode ser demorado. AWS IoT fornece uma maneira de definir um modelo para provisionar dispositivos quando eles se conectam pela primeira vez AWS IoT. Para obter mais informações, consulte Just-in-time provisionamento.

Permissões para executar testes AWS IoT do Device Advisor

O modelo de política a seguir mostra as permissões mínimas e a entidade IAM necessárias para executar os casos de teste AWS IoT do Device Advisor. Você precisará substituir pela função *your-device-role-arn* de dispositivo Amazon Resource Name (ARN) que você criou de acordo com os pré-requisitos.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iot:us-east-1:123456789012:thinggroup/your-thing-
group",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iotdeviceadvisor.amazonaws.com"
            }
        }
        },
```

```
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke*",
                "iam:ListRoles",
                "iot:Connect",
                "iot:CreateJob",
                "iot:DeleteJob",
                "iot:DescribeCertificate",
                "iot:DescribeEndpoint",
                "iotjobsdata:DescribeJobExecution",
                "iot:DescribeJob",
                "iot:DescribeThing",
                "iotjobsdata:GetPendingJobExecutions",
                "iot:GetPolicy",
                "iot:ListAttachedPolicies",
                "iot:ListCertificates",
                "iot:ListPrincipalPolicies",
                "iot:ListThingPrincipals",
                "iot:ListThings",
                "iot:Publish",
                "iotjobsdata:StartNextPendingJobExecution",
                "iotjobsdata:UpdateJobExecution",
                "iot:UpdateThingShadow",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents",
                "logs:PutRetentionPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": "iotdeviceadvisor:*",
            "Resource": "*"
        }
    ]
}
```

Prevenção do problema do substituto confuso entre serviços para o Device Advisor

O problema "confused deputy" é um problema de segurança em que uma entidade que não tem permissão para executar uma ação pode coagir uma entidade mais privilegiada a executar a ação. Em AWS, a falsificação de identidade entre serviços pode resultar no problema confuso do deputado. A personificação entre serviços pode ocorrer quando um serviço (o serviço de chamada) chama outro serviço (o serviço chamado). O serviço de chamada pode ser manipulado de modo a usar suas permissões para atuar nos recursos de outro cliente de uma forma na qual ele não deveria ter permissão para acessar. Para evitar isso, AWS fornece ferramentas que ajudam você a proteger seus dados para todos os serviços com diretores de serviços que receberam acesso aos recursos em sua conta.

Recomendamos o uso das chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount` em políticas de recursos para limitar as permissões que o Device Advisor concede a outro serviço para o recurso. Se você utilizar ambas as chaves de contexto de condição global, o valor `aws:SourceAccount` e a conta `aws:SourceArn` no valor deverão utilizar o mesmo ID de conta quando utilizados na mesma instrução de política.

O valor de `aws:SourceArn` deve ser o ARN do seu recurso de definição de suíte. O recurso de definição da suíte se refere à suíte de testes criada com o Device Advisor.

A maneira mais eficaz de se proteger do problema ‘confused deputy’ é usar a chave de contexto de condição global `aws:SourceArn` com o ARN completo do recurso. Se você não souber o ARN completo do recurso ou se estiver especificando vários recursos, use a chave de condição de contexto global `aws:SourceArn` com curingas (*) para as partes desconhecidas do ARN. Por exemplo, `arn:aws:iotdeviceadvisor:*:`*`account-id`*`:suitedefinition/*`.

O exemplo a seguir mostra como é possível usar as chaves de contexto de condição globais `aws:SourceArn` e `aws:SourceAccount` no Device Advisor para evitar o problema de substituto confuso.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Sid": "ConfusedDeputyPreventionExamplePolicy",
        "Effect": "Allow",
        "Principal": {
```

```
            "Service": "iotdeviceadvisor.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "ArnLike": {
                "aws:SourceArn": "arn:aws:iotdeviceadvisor:us-
east-1:123456789012:suitedefinition/ygp6rxa3tzvn"
        },
            "StringEquals": {
                "aws:SourceAccount": "123456789012"
        }
        }
    }
}
```

AWS treinamento e certificação

Faça o curso a seguir para aprender sobre os principais conceitos de AWS IoT segurança: AWS IoT Security Primer.

Monitoramento AWS IoT

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho de AWS IoT suas AWS soluções.

Recomendamos fortemente que você colete dados de monitoramento de todas as partes da sua AWS solução para facilitar a depuração de uma falha multiponto, caso ocorra. Comece criando um plano de monitoramento que responda às seguintes perguntas. Se não tiver certeza de como respondê-las, você ainda poderá continuar a habilitar o registro em log e estabelecer suas linhas de base de desempenho.

- Quais são seus objetivos de monitoramento?
- Quais recursos você vai monitorar?
- Com que frequência você vai monitorar esses recursos?
- Quais ferramentas de monitoramento você usará?
- Quem realizará o monitoramento das tarefas?
- Quem deve ser notificado quando algo der errado?

Sua próxima etapa é habilitar o registro e estabelecer uma linha de base do AWS IoT desempenho normal em seu ambiente medindo o desempenho em vários momentos e sob diferentes condições de carga. Ao monitorar AWS IoT, mantenha os dados históricos de monitoramento para poder compará-los com os dados de desempenho atuais. Isso ajuda a identificar padrões normais e anomalias de desempenho e a criar métodos para tratar os problemas.

Para estabelecer seu desempenho básico AWS IoT, você deve monitorar essas métricas para começar. Você sempre poderá monitorar mais métricas posteriormente.

- `PublishIn.Success`
- `PublishOut.Success`
- `Subscribe.Success`
- `Ping.Success`
- `Connect.Success`
- `GetThingShadow.Accepted`
- `UpdateThingShadow.Accepted`

- `DeleteThingShadow.Accepted`
- `RulesExecuted`

Os tópicos desta seção podem ajudar você a começar o registro em log e o monitoramento da AWS IoT.

Tópicos

- Configurar o AWS IoT registro
- Monitore AWS IoT alarmes e métricas usando a Amazon CloudWatch
- Monitore AWS IoT usando CloudWatch registros
- Faça upload de registros do lado do dispositivo para a Amazon CloudWatch
- Registrando chamadas de AWS IoT API usando AWS CloudTrail

Configurar o AWS IoT registro

Você deve ativar o registro usando o AWS IoT console, a CLI ou a API antes de poder monitorar e registrar AWS IoT atividades. Você pode configurar o registro AWS IoT em três níveis: nível da conta, nível do evento ou nível específico do recurso. O registro em nível de evento e específico de recursos está disponível exclusivamente com o registro V2. Os clientes que usam o registro da V1 devem realizar uma migração para a V2 para acessar esses recursos. Veja os detalhes.

Ao considerar como configurar seu AWS IoT registro, a configuração de registro em nível de conta determina como a AWS IoT atividade será registrada, a menos que especificado de outra forma. Para começar, talvez você queira obter registros detalhados com um nível de registro padrão de INFO ou DEBUG. Depois de analisar os registros iniciais, você pode alterar o nível de registro padrão para um nível menos detalhado, como WARN ou ERROR no nível da conta ou do evento, e definir um nível de registro mais detalhado específico para os recursos que talvez precisem de mais atenção. É possível alterar os níveis de log sempre que quiser.

Este tópico aborda o login na nuvem. AWS IoT Para obter informações sobre registro e monitoramento do lado do dispositivo, consulte Carregar registros do lado do dispositivo para. CloudWatch

Para obter informações sobre registro e monitoramento AWS IoT Greengrass, consulte Registro e monitoramento em AWS IoT Greengrass.

Configurando o login da V2 AWS IoT

Determinando sua versão de registro

A LoggingOptions API GetV2 retornará um NotConfiguredException se o registro V2 não estiver ativado. Esse erro ocorre quando o registro V1 está em uso ou nenhum registro foi configurado.

Entendendo os recursos de registro da V2

O registro V2 fornece dois recursos principais: registro em nível de evento e registro específico de recursos. O registro em nível de evento permite a configuração de registro direcionada com níveis de registro e CloudWatch destinos de grupos de registros personalizáveis. O registro específico de recursos permite filtrar os registros por grupo de coisas, IP de origem, ID do cliente ou ID principal. Juntos, esses recursos fornecem controle granular e visibilidade abrangente das operações de IoT, melhorando a capacidade de pesquisa de registros e reduzindo custos ao eliminar atividades de registro desnecessárias.

Migrar de V1 para V2

Você pode migrar para o registro V2 usando a LoggingOptions API SetV2 por meio da CLI AWS ou do SDK. Após a migração, encaminha AWS IoT automaticamente todos os registros para o CloudWatch grupo de registros AWSIot 'LogsV2'. Importante: se seus aplicativos ou recursos downstream consumirem informações de 'AWSIotRegistros', atualize-os para usar o caminho do grupo de registros correspondente.

Configurar a função e a política de registro em log

Antes de ativar o login AWS IoT, você deve criar uma função do IAM e uma política que dê AWS IoT permissão para gravar atividades de AWS IoT registro em grupos de CloudWatch registros em seu nome. Você também pode gerar uma função do IAM com as políticas necessárias na seção Logs do AWS IoT console.

> **Note**
>
> Antes de ativar o AWS IoT registro, certifique-se de entender as permissões de acesso aos CloudWatch registros. Os usuários com acesso aos CloudWatch registros podem ver as informações de depuração dos seus dispositivos. Para obter mais informações, consulte Autenticação e controle de acesso para Amazon CloudWatch Logs.

Se você espera altos padrões de tráfego AWS IoT Core devido ao teste de carga, considere desativar o registro de IoT para evitar a limitação. Se for detectado tráfego intenso, nosso serviço poderá desativar o registro em log em sua conta.

Veja a seguir como criar uma função de registro e uma política para AWS IoT Core recursos.

Criar uma função de registro em log

Para criar uma função de registro em log, abra o hub Roles do console do IAM e escolha a opção Criar função.

1. Em Selecionar tipo de entidade confiável, selecione a opção AWS Serviço. Em seguida, escolha a opção IoT em Caso de uso. Se você não vir a opção IoT, entre e pesquise IoT no menu suspenso Casos de uso para outros serviços da AWS :. Escolha Próximo.
2. Na página Adicionar permissões, você verá as políticas que são automaticamente anexadas ao perfil de serviço. Escolha Próximo.
3. Na página Nome, revisão e criação, insira um Nome do perfil e uma Descrição da função para a função e escolha Criar função.

Política de função de registro em log

Os documentos de política a seguir fornecem a política de função e a política de confiança que AWS IoT permitem enviar entradas de registro CloudWatch em seu nome. Se você configurar o registro em nível de evento com um grupo de CloudWatch registros personalizado, deverá atualizar a política de função para incluir o ARN do recurso personalizado.

Se você também tiver permissão AWS IoT Core for LoRaWAN para enviar entradas de registro, verá um documento de política criado para você que registra as duas atividades.

Note

Esses documentos foram criados para você quando você criou a função de registro em log. Os documentos têm variáveis, `${partition}`, `${region}`, e `${accountId}`, que você deve substituir por seus valores.

- Substitua a partição pela partição da região.

- Substitua a AWS região pela região que você usa. Certifique-se de usar a mesma AWS região usada para configurar a AWS CLI em seu dispositivo.
- Substitua o ID da conta pelo ID da sua AWS conta.

Política da função:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "iot:GetLoggingOptions",
                "iot:SetLoggingOptions",
                "iot:SetV2LoggingOptions",
                "iot:GetV2LoggingOptions",
                "iot:SetV2LoggingLevel",
                "iot:ListV2LoggingLevels",
                "iot:DeleteV2LoggingLevel"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:AWSIotLogsV2:*"
            ]
        }
    ]
}
```

Política de confiança para registrar somente AWS IoT Core atividades:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
        },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

AWS IoT O Logging pode falhar ao publicar o registro no CloudWatch Logs devido à insuficiência de permissões de função do IAM. Quando isso ocorrer, verifique as métricas de CloudWatch registro para investigar e solucionar as falhas.

Configurar o login no AWS IoT (console)

Esta seção descreve como configurar o AWS IoT registro usando o AWS IoT console. Você pode configurar o registro em nível de conta, de evento e de recursos específicos.

Para configurar o AWS IoT registro:

1. Faça login no AWS IoT console. Para obter mais informações, consulte Abra o AWS IoT console.
2. No painel de navegação esquerdo, escolha Registros (anteriormente uma seção em Configurações).
3. Configure o registro em nível de conta: o registro em nível de conta se aplica a toda a sua AWS IoT frota (dispositivos ou endpoints), a menos que seja substituído por configurações específicas de recursos ou de nível de evento.

    a. Em Registro em nível de conta, selecione Gerenciar registro em nível de conta para fazer atualizações.

    b. Marque a caixa de seleção “Ativar registro” para começar a enviar registros para CloudWatch. Quando a opção “Ativar registro” estiver desmarcada, não AWS IoT enviará nenhum CloudWatch registro para grupos de registros, independentemente das configurações de registro em nível de evento ou em nível de recurso.

    c. Em Função de log do IAM, selecione uma função existente na lista suspensa. Você pode ver os detalhes da função para inspecionar as permissões da função. Como alternativa, selecione Criar nova função para configurar uma nova função do IAM. A função de registro

fornece políticas que AWS IoT permitem enviar entradas de registro CloudWatch em seu nome. Se você configurar o registro em nível de evento com um grupo de CloudWatch registros personalizado, deverá atualizar a política de função para incluir o ARN desse grupo de registros.

d. Escolha o nível de registro padrão que corresponde ao nível de detalhe das entradas de registro que você deseja que apareçam nos CloudWatch registros. Nota: O nível de registro “DEBUG” fornece mais detalhes, mas aumenta CloudWatch os custos. CloudWatch os destinos do grupo de registros não podem ser configurados no nível da conta. No entanto, você pode especificar grupos de registros personalizados para tipos de eventos individuais, conforme descrito na seção a seguir.

e. Escolha Atualizar registro para salvar suas alterações.

4. O registro em nível de evento permite que você capture seletivamente registros de eventos relevantes e os direcione para grupos de registros dedicados CloudWatch . Isso lhe dá a flexibilidade de organizar registros por casos de uso para melhor visibilidade, compartilhá-los com diferentes públicos e reduzir CloudWatch custos ativando registros e definindo níveis de registro com base na criticidade do evento.

   Configurar o registro em nível de evento: o registro em nível de evento captura AWS IoT eventos específicos, como tentativas de autenticação do cliente. Essas configurações substituem o registro em nível de conta.

   a. Na seção Registro em nível de evento, selecione Gerenciar registro em nível de evento para fazer atualizações.

   b. Por padrão, os tipos de eventos herdam a configuração de registro em nível de conta. Observação: quando o registro específico do recurso é configurado, ele substitui as configurações em nível de conta e evento.

   c. Para modificar as configurações de eventos individuais, clique no valor na linha de eventos correspondente. Você pode ajustar o nível de registro e o destino do grupo de CloudWatch registros. Ao especificar um destino de grupo de CloudWatch registros personalizado, você deve verificar se a política de função do IAM inclui permissões para o novo grupo de registros. A falha na atualização da política de funções AWS IoT impedirá a gravação de registros no grupo de registros personalizados. Depois de fazer sua seleção, clique na marca de seleção para confirmar sua escolha. A coluna “Foi modificado” exibirá “Sim” para indicar alterações pendentes.

   d. Clique em Atualizar registro para aplicar suas alterações ou escolha Cancelar para descartar.

5. Configurar substituições específicas de recursos: as substituições específicas de recursos aplicam uma configuração de registro aos recursos selecionados. Um recurso pode ser um grupo de coisas, IP de origem, ID do cliente ou ID principal. A configuração de registro específica do recurso substitui as configurações no nível da conta e no nível do evento. Quando ativado, ele gera registros para todos os tipos de eventos no nível de registro configurado para os recursos especificados. Por exemplo, você pode definir o registro em nível de depuração para uma coisa específica enquanto mantém o registro em nível de informação para todas as outras coisas.

   a. Selecione Adicionar substituições específicas de recursos na seção Substituições específicas de recursos.

   b. Escolha um destino de registro: Thing group, Source IP, Client ID ou Principal ID.

   c. Insira o valor alvo de log correspondente para o tipo de alvo selecionado.

   d. Selecione o nível de registro desejado no menu suspenso na seção Nível de registro específico do recurso.

   e. Clique em Enviar para adicionar a substituição ou em Cancelar para descartar as alterações.

   f. Para modificar uma substituição específica de um recurso existente, marque a caixa de seleção ao lado do recurso e clique em “Remover” para excluir a substituição ou em “Editar” para modificar.

Depois que habilitar o registro em log, acesse Visualizando AWS IoT registros no CloudWatch console para saber mais sobre como visualizar as entradas de log.

Configurar login em nível de conta e evento (AWS IoT CLI)

Esta seção descreve como configurar o registro global AWS IoT usando a CLI.

Opcionalmente, você pode configurar o registro em nível de evento. O registro em nível de evento captura informações de registro no nível do evento, como eventos de autenticação e autorização ou criação de certificados. Você pode personalizar o nível do registro e os destinos do grupo de CloudWatch registros no nível do evento. O registro em nível de evento opera em um nível mais direcionado em comparação com o registro em nível de conta e, portanto, substitui as configurações de registro em nível de conta. Essa abordagem hierárquica permite que você mantenha diferentes estratégias de registro para diferentes tipos de eventos com base em sua importância operacional e considerações de custo.

> **Note**
>
> É necessário o nome do recurso da Amazon (ARN) da função que você deseja usar. Se você precisar criar uma função a ser usada para o registro em log, consulte Criar uma função de registro em log antes de continuar. Ao especificar um grupo de CloudWatch registros personalizado para qualquer tipo de evento, certifique-se de que sua função de registro tenha as permissões necessárias para o grupo de registros de destino.
> A entidade principal usada para chamar a API deve ter Transmitir as permissões de função para sua função de registro em log.

Você também pode realizar esse procedimento com a API usando os métodos na AWS API que correspondem aos comandos da CLI mostrados aqui.

Para usar a CLI para configurar o registro padrão para AWS IoT

1. Use o comando set-v2-logging-options para definir as opções de registro em log para sua conta.

   ```
   aws iot set-v2-logging-options \
       --event-configurations event-configuration-list \
       --role-arn logging-role-arn \
       --default-log-level log-level
   ```

   em que:

   --role-arn

   O ARN da função que concede AWS IoT permissão para gravar em seus registros em CloudWatch Logs. A configuração do Role-ARN é necessária para a configuração inicial.

   --default-log-level

   O nível de log a ser usado. Os valores válidos são: `ERROR`WARN,`INFO`,`DEBUG`, ou`DISABLED`. Default-log-levela configuração é necessária para a configuração inicial.

   --no-disable-all-logs

   Um parâmetro opcional que habilita todo o AWS IoT registro. Use esse parâmetro para habilitar o registro em log se ele estiver desabilitado no momento.

--disable-all-logs

Um parâmetro opcional que desativa todo o AWS IoT registro. Use esse parâmetro para desabilitar o registro em log se ele estiver habilitado no momento.

--event-configurations

Esse parâmetro é opcional e permite que você personalize as configurações de registro para tipos de eventos individuais:

- EventType: o tipo de evento que substitui a configuração no nível da conta
- LogLevel: substitua a configuração no nível da conta por DEBUG, INFO, ERROR, WARN ou DISABLED
- LogDestination: especifique um grupo de CloudWatch registros personalizado para entrega de registros

Você pode configurar o nível e o destino do registro de forma independente para cada tipo de evento. Se não forem especificados, os eventos herdarão as configurações no nível da conta

```
aws iot set-v2-logging-options \
    --event-configurations "[{\"eventType\":\"Publish-In\",\"logLevel\":\"INFO\",
\"logDestination\":\"examplePublishInLogGroup\"}]"
```

2. Use o comando get-v2-logging-options para obter as opções de registro em log atuais.

```
aws iot get-v2-logging-options \
    --verbose
```

em que:

--verbose

Um parâmetro opcional que permite recuperar todos os tipos de eventos e suas configurações.

Depois que habilitar o registro em log, acesse Visualizando AWS IoT registros no CloudWatch console para saber mais sobre como visualizar as entradas de log.

> **Note**
>
> AWS IoT continua oferecendo suporte a comandos mais antigos (set-logging-optionseget-logging-options) para definir e obter o registro global em sua conta. Lembre-se de que, quando esses comandos são usados, os logs resultantes conterão texto sem formatação, em vez de cargas úteis JSON, e a latência dos registros em logs geralmente será mais alta. Não haverá mais melhorias à implementação desses comandos mais antigos. Recomendamos que você use as versões “v2" para configurar suas opções de registro e, quando possível, alterar os aplicativos legados que usam as versões mais antigas.

Configurar substituições específicas de recursos em (CLI) AWS IoT

Esta seção descreve como configurar substituições específicas de recursos usando AWS IoT a CLI. As substituições específicas de recursos permitem que você especifique um nível de registro para um recurso específico identificado por Thing Group, ID do cliente, IP de origem ou ID principal. Quando o registro específico do recurso está ativado, ele substitui as configurações no nível da conta e no nível do evento. Todos os tipos de eventos gerarão registros para o recurso especificado no nível de registro configurado, mesmo que esses eventos estejam desativados nas configurações em nível de evento.

Grupos de objetos podem conter outros grupos de objetos para criar um relacionamento hierárquico. Este procedimento descreve como configurar o registro em log de um único grupo de objetos. É possível aplicar esse procedimento ao grupo de objetos pai em uma hierarquia para configurar o registro em log para todos os grupos de objetos na hierarquia. Também é possível aplicar esse procedimento a um grupo de objetos filho para substituir a configuração de registro em log do grupo pai.

Uma coisa pode ser membro de um grupo de coisas. Essa associação permite que a coisa herde configurações, políticas e definições aplicadas ao grupo de coisas. Grupos de coisas são usados para gerenciar e aplicar configurações a várias coisas coletivamente, em vez de lidar com cada coisa individualmente. Quando seu ID de cliente corresponder ao nome da coisa, AWS IoT Core associará automaticamente a sessão do cliente ao recurso correspondente. Isso permite que a sessão do cliente herde as configurações e as definições aplicadas aos grupos de coisas aos quais a coisa pertence, incluindo os níveis de registro em log. Se seu ID do cliente não corresponder ao nome da coisa, você poderá habilitar o anexo de coisa exclusiva para estabelecer a associação. Para obter mais informações, consulte ???.

Além dos grupos de objetos, você também pode registrar destinos como o ID do cliente, o IP de origem e o ID principal do dispositivo.

> **Note**
>
> É necessário o nome do recurso da Amazon (ARN) da função que você deseja usar. Se você precisar criar uma função a ser usada para o registro em log, consulte Criar uma função de registro em log antes de continuar.
> A entidade principal usada para chamar a API deve ter Transmitir as permissões de função para sua função de registro em log.

Você também pode realizar esse procedimento com a API usando os métodos na AWS API que correspondem aos comandos da CLI mostrados aqui.

Para usar a CLI para configurar substituições específicas de recursos para AWS IoT

1. Ative o registro em nível de conta antes de configurar o registro específico do recurso usando o seguinte comando: comando aws iot set-v2-logging-options
2. Use o set-v2-logging-levelcomando para configurar substituições específicas de recursos. Veja o exemplo a seguir para configuração de grupos de coisas:

   ```
   aws iot set-v2-logging-level \
                --log-target targetType=THING_GROUP,targetName=thing_group_name \
                --log-level log_level
   ```

   --log-target

   O tipo e o nome do recurso para o qual você está configurando o registro em log. O valor `targetType` deve ser um dos seguintes: `THING_GROUP` | `CLIENT_ID` | `SOURCE_IP` |`PRINCIPAL_ID`. O valor do parâmetro-alvo de log pode ser texto, como mostrado no exemplo de comando anterior, ou uma string JSON, como o exemplo a seguir.

   ```
   aws iot set-v2-logging-level \
                --log-target '{"targetType": "THING_GROUP","targetName":
    "thing_group_name"}' \
                --log-level log_level
   ```

--log-level

O nível de registro em log usado ao gerar logs para o recurso especificado. Os valores válidos são: DEBUG, INFO, ERROR, WARN e DISABLED.

3. Use o comando list-v2-logging-levels para listar os níveis de registro em log configurados no momento.

```
aws iot list-v2-logging-levels
```

4. Use o comando delete-v2-logging-level para excluir um nível de registro em log específico de recursos, conforme os exemplos seguintes.

```
aws iot delete-v2-logging-level \
              --target-type "THING_GROUP" \
              --target-name "thing_group_name"
```

```
aws iot delete-v2-logging-level \
              --target-type=CLIENT_ID
              --target-name=ClientId1
```

--target-type

O valor `target-type` deve ser um dos seguintes: `THING_GROUP` | `CLIENT_ID` | `SOURCE_IP` |`PRINCIPAL_ID`.

--target-name

O nome do grupo de objetos do qual remover o nível de registro em log.

Níveis de logs

Esses níveis de log determinam os eventos que são registrados em log e se aplicam aos níveis de log padrão e de recursos específicos.

ERRO

Qualquer erro que cause a falha de uma operação.

Exemplo: falha na autenticação do dispositivo devido ao certificado expirado.

Os logs incluem apenas informações do tipo ERRO.

WARN

Tudo que possa causar inconsistências no sistema, mas que não cause falha na operação.

Exemplo: aproximação do limite de taxa de mensagens.

Os logs incluem informações dos tipos ERRO e AVISO.

INFORMAÇÕES

Informações de alto nível sobre o fluxo das objetos.

Exemplo: um cliente se inscreveu com sucesso em um tópico do MQTT.

Os logs incluem informações dos tipos INFORMAÇÕES, ERRO e AVISO.

DEBUG

Informações que podem ser úteis ao depurar um problema.

Exemplo: o IoT Rules Engine detectou uma mensagem publicada no tópico de regras “regra/teste” e iniciou a execução com sucesso. A regra está configurada com RepublishAction.

Os logs incluem informações dos tipos DEBUG, INFORMAÇÕES, ERRO e AVISO.

DISABLED

Todos os registros são desativados.

Monitore AWS IoT alarmes e métricas usando a Amazon CloudWatch

Você pode monitorar AWS IoT o uso CloudWatch, que coleta e processa dados brutos AWS IoT em métricas legíveis e quase em tempo real. Essas estatísticas são registradas por um período de duas semanas, para que você possa acessar informações históricas e obter uma perspectiva melhor sobre como seu serviço ou aplicativo web estão se saindo. Por padrão, os dados AWS IoT métricos são enviados automaticamente CloudWatch em intervalos de um minuto. Para obter mais informações, consulte O que são Amazon CloudWatch, Amazon CloudWatch Events e Amazon CloudWatch Logs? no Guia do CloudWatch usuário da Amazon.

Usando AWS IoT métricas

As métricas relatadas por AWS IoT fornecem informações que você pode analisar de maneiras diferentes. Os seguintes casos de uso são baseados em um cenário com dez objetos que se conectam à internet uma vez por dia. Cada dia:

- Dez coisas se conectam aproximadamente AWS IoT ao mesmo tempo.
- Cada objeto se inscreve em um filtro do tópico e, em seguida, aguarda por uma hora antes de se desconectar. Durante esse período, os objetos se comunicam entre si e aprendem mais sobre o mundo.
- Cada objeto publica uma percepção com base nos dados recém-encontrados usando o `UpdateThingShadow`.
- Cada coisa se desconecta de AWS IoT.

Para ajudar você a começar, esses tópicos exploram algumas das dúvidas que você pode ter.

- Como posso ser notificado se a conexão das objetos não for bem-sucedida a cada dia?
- Como posso ser notificado se as objetos não estiverem publicando dados a cada dia?
- Como posso ser notificado se as atualizações da sombra do objeto estiverem sendo rejeitadas a cada dia?
- Como posso criar um CloudWatch alarme para Jobs?

Mais sobre CloudWatch alarmes e métricas

- Criação CloudWatch de alarmes para monitorar AWS IoT
- AWS IoT métricas e dimensões

Criação CloudWatch de alarmes para monitorar AWS IoT

Você pode criar um CloudWatch alarme que envia uma mensagem do Amazon SNS quando o alarme muda de estado. O alarme observa uma única métrica em um período especificado. Quando o valor da métrica exceder um determinado limite em vários períodos de tempo, uma ou mais ações serão executadas. A ação é uma notificação enviada para um tópico do Amazon SNS ou uma política de ajuste de escala automático. Os alarmes acionam ações somente para mudanças de estado sustentadas. CloudWatch os alarmes não acionam ações simplesmente porque estão em

um determinado estado; o estado deve ter sido alterado e mantido por um determinado número de períodos.

Os tópicos a seguir descrevem alguns exemplos de uso de alarmes do CloudWatch.

- Como posso ser notificado se a conexão das objetos não for bem-sucedida a cada dia?
- Como posso ser notificado se as objetos não estiverem publicando dados a cada dia?
- Como posso ser notificado se as atualizações da sombra do objeto estiverem sendo rejeitadas a cada dia?
- Como posso criar um CloudWatch alarme para trabalhos?

Você pode ver todas as métricas que CloudWatch os alarmes podem monitorar. AWS IoT métricas e dimensões

Como posso ser notificado se a conexão das objetos não for bem-sucedida a cada dia?

1. Crie um tópico do Amazon SNS chamado `things-not-connecting-successfully`, e registre o nome do recurso da Amazon (ARN). Esse procedimento se referirá ao ARN do tópico como *`sns-topic-arn`* .

   Para obter mais informações sobre como criar uma notificação do Amazon SNS, consulte Inicialização do Amazon SNS.

2. Crie o alarme.

```
aws cloudwatch put-metric-alarm \
    --alarm-name ConnectSuccessAlarm \
    --alarm-description "Alarm when my Things don't connect successfully" \
    --namespace AWS/IoT \
    --metric-name Connect.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --evaluation-periods 1 \
    --alarm-actions sns-topic-arn
```

3. Teste o alarme.

```
aws cloudwatch set-alarm-state --alarm-name ConnectSuccessAlarm --state-reason
 "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name ConnectSuccessAlarm --state-reason
 "initializing" --state-value ALARM
```

4. Verifique se o alarme aparece no console do CloudWatch .

Como posso ser notificado se as objetos não estiverem publicando dados a cada dia?

1. Crie um tópico do Amazon SNS chamado `things-not-publishing-data`, e registre o nome do recurso da Amazon (ARN). Esse procedimento se referirá ao ARN do tópico como *sns-topic-arn* .

   Para obter mais informações sobre como criar uma notificação do Amazon SNS, consulte Inicialização do Amazon SNS.

2. Crie o alarme.

```
aws cloudwatch put-metric-alarm \
    --alarm-name PublishInSuccessAlarm\
    --alarm-description "Alarm when my Things don't publish their data \
    --namespace AWS/IoT \
    --metric-name PublishIn.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --evaluation-periods 1 \
    --alarm-actions sns-topic-arn
```

3. Teste o alarme.

```
aws cloudwatch set-alarm-state --alarm-name PublishInSuccessAlarm --state-reason
 "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name PublishInSuccessAlarm --state-reason
 "initializing" --state-value ALARM
```

4. Verifique se o alarme aparece no console do CloudWatch .

Como posso ser notificado se as atualizações da sombra do objeto estiverem sendo rejeitadas a cada dia?

1. Crie um tópico do Amazon SNS chamado `things-shadow-updates-rejected`, e registre o nome do recurso da Amazon (ARN). Esse procedimento se referirá ao ARN do tópico como *`sns-topic-arn`* .

   Para obter mais informações sobre como criar uma notificação do Amazon SNS, consulte Inicialização do Amazon SNS.

2. Crie o alarme.

```
aws cloudwatch put-metric-alarm \
    --alarm-name UpdateThingShadowSuccessAlarm \
    --alarm-description "Alarm when my Things Shadow updates are getting rejected"
 \
    --namespace AWS/IoT \
    --metric-name UpdateThingShadow.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions sns-topic-arn
```

3. Teste o alarme.

```
aws cloudwatch set-alarm-state --alarm-name UpdateThingShadowSuccessAlarm --state-
reason "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name UpdateThingShadowSuccessAlarm --state-
reason "initializing" --state-value ALARM
```

4. Verifique se o alarme aparece no console do CloudWatch .

Como posso criar um CloudWatch alarme para trabalhos?

O serviço Jobs fornece CloudWatch métricas para você monitorar seus trabalhos. Você pode criar CloudWatch alarmes para monitorar qualquer Métricas de tarefas um.

O comando a seguir cria um CloudWatch alarme para monitorar o número total de execuções de tarefas malsucedidas para Job *SampleOTAJob* e notifica você quando mais de 20 execuções de tarefas falharam. O alarme monitora a métrica `FailedJobExecutionTotalCount` de Tarefas verificando o valor relatado a cada 300 segundos. Ele é ativado quando um único valor relatado é maior que 20, ou seja, que houve mais de 20 execuções de trabalho com falha desde que o trabalho foi iniciado. Quando o alarme dispara, ele envia uma notificação para o tópico do Amazon SNS fornecido.

```
aws cloudwatch put-metric-alarm \
    --alarm-name TotalFailedJobExecution-SampleOTAJob \
    --alarm-description "Alarm when total number of failed job execution exceeds the
 threshold for SampleOTAJob" \
    --namespace AWS/IoT \
    --metric-name FailedJobExecutionTotalCount \
    --dimensions Name=JobId,Value=SampleOTAJob \
    --statistic Sum \
    --threshold 20 \
    --comparison-operator GreaterThanThreshold \
    --period 300 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:<AWS_REGION>:<AWS_ACCOUNT_ID>:SampleOTAJob-has-too-
many-failed-job-ececutions
```

O comando a seguir cria um CloudWatch alarme para monitorar o número de execuções de tarefas malsucedidas para Job *SampleOTAJob* em um determinado período. Ele, então, o notifica quando há falha em mais de cinco execuções da tarefa durante esse período. O alarme monitora a métrica `FailedJobExecutionCount` de Tarefas verificando o valor relatado a cada 3.600 segundos. Ele é ativado quando um único valor relatado é maior que cinco, ou seja, que houve mais de cinco execuções de trabalho com falha na última hora. Quando o alarme dispara, ele envia uma notificação para o tópico do Amazon SNS fornecido.

```
aws cloudwatch put-metric-alarm \
    --alarm-name FailedJobExecution-SampleOTAJob \
    --alarm-description "Alarm when number of failed job execution per hour exceeds the
 threshold for SampleOTAJob" \
```

```
    --namespace AWS/IoT \
    --metric-name FailedJobExecutionCount \
    --dimensions Name=JobId,Value=SampleOTAJob \
    --statistic Sum \
    --threshold 5 \
    --comparison-operator GreaterThanThreshold \
    --period 3600 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:<AWS_REGION>:<AWS_ACCOUNT_ID>:SampleOTAJob-has-too-
many-failed-job-ececutions-per-hour
```

AWS IoT métricas e dimensões

Quando você interage com AWS IoT, o serviço envia métricas e dimensões a CloudWatch cada minuto. Você pode usar AWS IoT, usar o CloudWatch console ou AWS CLI visualizar essas métricas.

Para ver as métricas usando o CloudWatch console, abra o CloudWatch console. No painel de navegação, escolha Métricas e, em seguida, Todas as métricas. Na guia Procurar, pesquise AWS IoT para ver a lista de métricas. As métricas são agrupadas primeiro pelo namespace do serviço e, em seguida, por várias combinações de dimensão dentro de cada namespace.

Para visualizar métricas usando AWS CLI, execute o comando a seguir.

```
aws cloudwatch list-metrics --namespace "AWS/IoT"
```

CloudWatch exibe os seguintes grupos de métricas para AWS IoT:

- AWS IoT métricas
- AWS IoT Core métricas do provedor de credenciais
- Métricas de autenticação
- Métricas de grampeamento de OCSP do certificado de servidor
- Métricas de regra
- Métricas de ação da regra
- Métricas específicas da ação HTTP
- Métricas do agente de mensagens

- Métricas da sombra do dispositivo
- Métricas de registro
- Métricas de tarefas
- Métricas de auditoria do Device Defender
- Métricas do Device Defender Detect
- Métricas de provisionamento de dispositivos
- LoRaMétricas de WAN
- Métricas de indexação de frota
- Dimensões para métricas

AWS IoT métricas

Métrica	Descrição
`AddThingToDynamicThingGroupsFailed`	O número de eventos de falha associados à adição de um objeto a um grupo dinâmico. A dimensão `DynamicThingGroupName` contém o nome dos grupos dinâmicos que falharam ao adicionar objetos.
`NumLogBatchesFailedToPublishThrottled`	O lote singular de eventos de logs dentro do lote que falhou na publicação devido a erros de controle de utilização.
`NumLogEventsFailedToPublishThrottled`	O número de eventos de logs dentro do lote que falharam na publicação devido a erros de controle de utilização.

AWS IoT Core métricas do provedor de credenciais

Métrica	Description
`CredentialExchangeSuccess`	O número de solicitações `AssumeRoleWithCertificate` bem-sucedidas ao provedor AWS IoT Core de credenciais.

Métricas de autenticação

> **Note**
>
> As métricas de autenticação são exibidas no CloudWatch console em Métricas de protocolo.

Métrica	Description
`Connection.AuthNError`	O número de tentativas de conexão AWS IoT Core rejeitadas devido a falhas de autenticação. Essa métrica considera apenas conexões que enviam uma string de Indicação de Nome de Servidor (SNI) correspondente a um endpoint da sua Conta da AWS. Essa métrica inclui tentativas de conexão de fontes externas, como ferramentas de escaneamento da internet ou atividades de sondagem. A dimensão `Protocol` contém o protocolo usado para enviar a tentativa de conexão.

Métricas de grampeamento de OCSP do certificado de servidor

Métrica	Description
Recuperar dadosOCSPStaple. Sucesso	A resposta OCSP foi recebida e processada com sucesso. Essa resposta será incluída durante o handshake TLS para o domínio configurado. A dimensão `DomainConfigurationName` contém o nome do domínio configurado com o grampeamento de OCSP do certificado de servidor habilitado.

Métricas de regra

Métrica	Description
`ParseError`	O número de erros de análise JSON encontrados em mensagens publicadas em um tópico no qual uma regra está atenta. A dimensão `RuleName` contém o nome da regra.
`RuleExecutionThrottled`	O número de mensagens limitadas pelo mecanismo de regras devido a comportamentos mal-inten cionados ou porque o número de mensagens excede o limite de controle de fluxo do mecanismo de regras. A dimensão `RuleName` contém o nome da regra a ser acionada.
`RuleNotFound`	Não foi possível encontrar a regra a ser acionada. A dimensão `RuleName` contém o nome da regra.
`RulesExecuted`	O número de AWS IoT regras executadas.
`TopicMatch`	O número de mensagens de entrada publicada s em um tópico no qual uma regra está atenta. A dimensão `RuleName` contém o nome da regra.

Métricas de ação da regra

Métrica	Description
`Failure`	O número de invocações da ação de regra com falha. A dimensão `RuleName` contém o nome da regra que especifica a ação. A dimensão `ActionType` contém o tipo de ação que foi invocada.
`Success`	O número de invocações da ação de regra bem-sucedidas. A dimensão `RuleName` contém o

Métrica	Description
	nome da regra que especifica a ação. A dimensão `ActionType` contém o tipo de ação que foi invocada.
`ErrorActionFailure`	O número de ações de erro que falharam. A dimensão `RuleName` contém o nome da regra que especifica a ação. A dimensão `ActionType` contém o tipo de ação que foi invocada.
`ErrorActionSuccess`	O número de ações de erro bem-sucedidas. A dimensão `RuleName` contém o nome da regra que especifica a ação. A dimensão `ActionType` contém o tipo de ação que foi invocada.

Métricas específicas da ação HTTP

Métrica	Description
`HttpCode_Other`	Gerado se o código de status da resposta da web downstream não service/application for 2xx, 4xx ou 5xx.
`HttpCode_4XX`	Gerado se o código de status da resposta da web downstream service/application estiver entre 400 e 499.
`HttpCode_5XX`	Gerado se o código de status da resposta da web downstream service/application estiver entre 500 e 599.
`HttpInvalidUrl`	Gerado se um URL de endpoint, após a substitui ção dos modelos de substituição, não começar com `https://`.
`HttpRequestTimeout`	Gerado se a web downstream service/application não retornar a resposta dentro do limite de tempo

Métrica	Description
	limite da solicitação. Para obter mais informações, consulte Service Quotas.
`HttpUnknownHost`	Gerado se o URL for válido, mas o serviço não existe ou é inacessível.

Métricas do agente de mensagens

> **Note**
>
> As métricas do agente de mensagens são exibidas no CloudWatch console em Métricas de protocolo.

Métrica	Description
`Connect.AuthError`	O número de solicitações de conexão que não puderam ser autorizadas pelo agente de mensagens . A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `CONNECT`.
`Connect.ClientError`	O número de solicitações de conexão rejeitada s porque a mensagem MQTT não atendeu aos requisitos definidos emCotas do AWS IoT. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `CONNECT`.
`Connect.ClientIDThrottle`	O número de solicitações de conexão limitadas porque o cliente excedeu a taxa de solicitação da conexão permitida para um ID de cliente específic o. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `CONNECT`.
`Connect.ServerError`	O número de solicitações de conexão com falha porque ocorreu um erro interno. A dimensão

Métrica	Description
	`Protocol` contém o protocolo usado para enviar a mensagem `CONNECT`.
`Connect.Success`	O número de conexões bem-sucedidas com o agente de mensagens. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `CONNECT`.
`Connect.Throttle`	O número de solicitações de conexão que foram limitadas porque a conta excedeu a taxa de solicitações de conexão permitida. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `CONNECT`.
`Ping.Success`	O número de mensagens ping recebidas pelo agente de mensagens. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem ping.
`PublishIn.AuthError`	O número de solicitações de publicação que o agente de mensagens não conseguiu autorizar. A dimensão `Protocol` contém o protocolo usado para publicar a mensagem. O HTTP Publish não é compatível com essa métrica.
`PublishIn.ClientError`	O número de solicitações de publicação rejeitadas pelo agente de mensagens porque a mensagem não atendeu aos requisitos definidos em Cotas do AWS IoT. A dimensão `Protocol` contém o protocolo usado para publicar a mensagem.
`PublishIn.ServerError`	O número de solicitações de publicação que o agente de mensagens deixou de processar por causa da ocorrência de um erro interno. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `PUBLISH`.

Métrica	Description
`PublishIn.Success`	O número de solicitações de publicação processadas com êxito pelo agente de mensagens. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `PUBLISH`.
`PublishIn.Throttle`	O número de solicitações de publicação que foram limitadas porque o cliente excedeu a taxa de mensagens recebidas permitida. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `PUBLISH`. O HTTP Publish não é compatível com essa métrica.
`PublishOut.AuthError`	O número de solicitações de publicação feitas pelo agente de mensagens que não puderam ser autorizadas pelo AWS IoT. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `PUBLISH`.
`PublishOut.ClientError`	O número de solicitações de publicação feitas pelo agente de mensagens que foram rejeitadas porque a mensagem não atendia aos requisitos definidos emCotas do AWS IoT. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `PUBLISH`.
`PublishOut.Success`	O número de solicitações de publicação feitas com êxito pelo agente de mensagens. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `PUBLISH`.
`PublishOut.Throttle`	O número de solicitações de publicação que foram limitadas porque o cliente excedeu a taxa de mensagens encaminhadas permitida. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `PUBLISH`.

Métrica	Description
`PublishRetained.AuthError`	O número de solicitações de publicação com o sinalizador `RETAIN` definido que o agente de mensagens não conseguiu autorizar. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `PUBLISH`.
`PublishRetained.ServerError`	O número de solicitações de publicação retidas que o agente de mensagens deixou de processar por causa de um erro interno ocorrido. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `PUBLISH`.
`PublishRetained.Success`	O número de solicitações de publicação com o sinalizador `RETAIN` definido processadas com êxito pelo agente de mensagens. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `PUBLISH`.
`PublishRetained.Throttle`	O número de solicitações de publicação com o sinalizador `RETAIN` definido que foram limitadas porque o cliente excedeu a taxa de mensagens recebidas permitida. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `PUBLISH`.
`Queued.Success`	O número de mensagens armazenadas que foram processadas com sucesso pelo agente de mensagens para clientes que foram desconectados da sessão persistente. As mensagens com QoS de 1 são armazenadas enquanto um cliente com uma sessão permanente é desconectado.

Métrica	Description
`Queued.Throttle`	O número de mensagens que não puderam ser armazenadas e foram limitadas enquanto clientes com sessões permanentes foram desconectados. Isso ocorre quando os clientes excedem o limite de mensagens na fila por segundo por conta. As mensagens com QoS de 1 são armazenadas enquanto um cliente com uma sessão permanente é desconectado.
`Queued.ServerError`	O número de mensagens não foram armazenadas em uma sessão permanente devido a um erro interno. Quando clientes com uma sessão permanente são desconectados, as mensagens com uma Qualidade de Serviço (QoS) de 1 são armazenadas.
`Subscribe.AuthError`	O número de solicitações de assinatura feitas por um cliente não autorizadas. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `SUBSCRIBE` .
`Subscribe.ClientError`	O número de solicitações de assinatura que foram rejeitadas porque a `SUBSCRIBE` mensagem não atendia aos requisitos definidos em Cotas do AWS IoT. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `SUBSCRIBE` .
`Subscribe.ServerError`	O número de solicitações de assinatura rejeitadas em virtude da ocorrência de um erro interno. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `SUBSCRIBE` .
`Subscribe.Success`	O número de solicitações de assinatura processadas com êxito pelo agente de mensagens. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `SUBSCRIBE` .

Métrica	Description
`Subscribe.Throttle`	O número de solicitações de assinatura que foram limitadas porque os limites de taxa de solicitação de assinatura permitida foram excedidos para sua Conta da AWS. Esses limites incluem assinatur as por segundo por conta, assinaturas por conta e assinaturas por conexão descritas em cotas e limites de protocolo do agente de mensagens AWS IoT Core. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `SUBSCRIBE` .
`Throttle.Exceeded`	Essa métrica será exibida CloudWatch quando um cliente MQTT for limitado em pacotes por segundo por limite de nível de conexão. Essa métrica não se aplica às conexões HTTP.
`Unsubscribe.ClientError`	O número de solicitações de cancelamento de assinatura que foram rejeitadas porque a `UNSUBSCRIBE` mensagem não atendia aos requisitos definidos emCotas do AWS IoT. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `UNSUBSCRIBE` .
`Unsubscribe.ServerError`	O número de solicitações de cancelamento da assinatura que foram rejeitadas porque ocorreu um erro interno. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `UNSUBSCRIBE` .
`Unsubscribe.Success`	O número de solicitações de cancelamento da assinatura que foram processadas com êxito pelo agente de mensagens. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `UNSUBSCRIBE` .

Métrica	Description
`Unsubscribe.Throttle`	O número de solicitações de cancelamento da assinatura que foram rejeitadas porque o cliente excedeu a taxa de solicitação de cancelamento da assinatura permitida. A dimensão `Protocol` contém o protocolo usado para enviar a mensagem `UNSUBSCRIBE` .

Métricas da sombra do dispositivo

> **Note**
>
> As métricas de sombra do dispositivo são exibidas no CloudWatch console em Métricas de protocolo.

Métrica	Description
`DeleteThingShadow.Accepted`	O número de solicitações `DeleteThingShadow` processadas com êxito. A dimensão `Protocol` contém o protocolo usado para criar a solicitação.
`GetThingShadow.Accepted`	O número de solicitações `GetThingShadow` processadas com êxito. A dimensão `Protocol` contém o protocolo usado para criar a solicitação.
`ListThingShadow.Accepted`	O número de solicitações `ListThingShadow` processadas com êxito. A dimensão `Protocol` contém o protocolo usado para criar a solicitação.
`UpdateThingShadow.Accepted`	O número de solicitações `UpdateThingShadow` processadas com êxito. A dimensão `Protocol` contém o protocolo usado para criar a solicitação.

Métricas de registro

Métrica	Description
`CloudwatchLogs:LogGroupCreationFailed`	O número de falhas na `CreateLogGroup` solicitação. O AWS IoT Logging cria um grupo de CloudWatch logs se ele não existir. A `LogGroup` dimensão contém o grupo de registros usado para fazer a solicitação.
`CloudwatchLogs:LogStreamCreationFailed`	O número de falhas na `CreateLogStream` solicitação. O AWS IoT Logging cria um fluxo de CloudWatch log para enviar eventos de log. A `LogGroup` dimensão contém o grupo de registros usado para fazer a solicitação.
`CloudwatchLogs:PutRetentionPolicyFailed`	O número de falhas na `PutRetentionPolicy` solicitação. O AWS IoT Logging cria um grupo de CloudWatch logs se ele não existir, e a política de retenção é definida como 30 dias. A `LogGroup` dimensão contém o grupo de registros usado para fazer a solicitação.
`CloudwatchLogs:PutLogEventsFailed`	O número de falhas na `PutLogEvents` solicitação. A `LogGroup` dimensão contém o grupo de CloudWatch registros usado para fazer a solicitação.

Métricas de tarefas

Métrica	Description
`CanceledJobExecutionCount`	O número de execuções de tarefas cujo status foi alterado para `CANCELED` dentro de um período determinado por CloudWatch. (Para obter mais informações sobre CloudWatch métricas, consulte

Métrica	Description
	Amazon CloudWatch Metrics.) A dimensão `JobId` contém o ID da tarefa.
`CanceledJobExecutionTotalCount`	O número total de execuções da tarefa cujo status for `CANCELED` para a tarefa determinada. A dimensão `JobId` contém o ID da tarefa.
`ClientErrorCount`	O número de erros do cliente gerados ao executar a tarefa. A dimensão `JobId` contém o ID da tarefa.
`FailedJobExecutionCount`	O número de execuções de tarefas cujo status foi alterado para `FAILED` dentro de um período determinado por CloudWatch. (Para obter mais informações sobre CloudWatch métricas, consulte Amazon CloudWatch Metrics.) A dimensão `JobId` contém o ID da tarefa.
`FailedJobExecutionTotalCount`	O número total de execuções da tarefa cujo status for `FAILED` para a tarefa determinada. A dimensão `JobId` contém o ID da tarefa.
`InProgressJobExecutionCount`	O número de execuções de tarefas cujo status foi alterado para `IN_PROGRESS` dentro de um período determinado por CloudWatch. (Para obter mais informações sobre CloudWatch métricas, consulte Amazon CloudWatch Metrics.) A dimensão `JobId` contém o ID da tarefa.
`InProgressJobExecutionTotalCount`	O número total de execuções da tarefa cujo status for `IN_PROGRESS` para a tarefa determinada. A dimensão `JobId` contém o ID da tarefa.
`RejectedJobExecutionTotalCount`	O número total de execuções da tarefa cujo status for `REJECTED` para a tarefa determinada. A dimensão `JobId` contém o ID da tarefa.

Métrica	Description
`RemovedJobExecutionTotalCount`	O número total de execuções da tarefa cujo status for `REMOVED` para a tarefa determinada. A dimensão `JobId` contém o ID da tarefa.
`QueuedJobExecutionCount`	O número de execuções de tarefas cujo status foi alterado para `QUEUED` dentro de um período determinado por CloudWatch. (Para obter mais informações sobre CloudWatch métricas, consulte Amazon CloudWatch Metrics.) A dimensão `JobId` contém o ID da tarefa.
`QueuedJobExecutionTotalCount`	O número total de execuções da tarefa cujo status for `QUEUED` para a tarefa determinada. A dimensão `JobId` contém o ID da tarefa.
`RejectedJobExecutionCount`	O número de execuções de tarefas cujo status foi alterado para `REJECTED` dentro de um período determinado por CloudWatch. (Para obter mais informações sobre CloudWatch métricas, consulte Amazon CloudWatch Metrics.) A dimensão `JobId` contém o ID da tarefa.
`RemovedJobExecutionCount`	O número de execuções de tarefas cujo status foi alterado para `REMOVED` dentro de um período determinado por CloudWatch. (Para obter mais informações sobre CloudWatch métricas, consulte Amazon CloudWatch Metrics.) A dimensão `JobId` contém o ID da tarefa.
`ServerErrorCount`	O número de erros do servidor gerados ao executar a tarefa. A dimensão `JobId` contém o ID da tarefa.

Métrica	Description
`SucceededJobExecutionCount`	O número de execuções de tarefas cujo status foi alterado para `SUCCESS` dentro de um período determinado por CloudWatch. (Para obter mais informações sobre CloudWatch métricas, consulte Amazon CloudWatch Metrics.) A dimensão `JobId` contém o ID da tarefa.
`SucceededJobExecutionTotalCount`	O número total de execuções da tarefa cujo status for `SUCCESS` para a tarefa determinada. A dimensão `JobId` contém o ID da tarefa.

Métricas de auditoria do Device Defender

Métrica	Description
`NonCompliantResources`	O número de recursos identificados como não compatíveis com uma verificação. O sistema informará o número de recursos que estavam fora da conformidade para cada verificação de auditoria feita.
`ResourcesEvaluated`	O número de recursos avaliados para conformidade. O sistema informará o número de recursos avaliados para cada verificação de auditoria feita.
`MisconfiguredDeviceDefenderNotification`	Notifica você quando a configuração do SNS para AWS IoT Device Defender está mal configurada. Dimensões

Métricas do Device Defender Detect

Métrica	Description
`NumOfMetricsExported`	O número de métricas exportadas para uma métrica do lado da nuvem, do lado do dispositivo ou personalizada. O sistema relata o número de métricas exportadas para a conta, para uma métrica específica. Essa métrica está disponível somente para clientes que usam a exportação de métricas.
`NumOfMetricsSkipped`	O número de métricas ignoradas para uma métrica do lado da nuvem, do lado do dispositivo ou personalizada. O sistema relata o número de métricas ignoradas para a conta, para uma métrica específica devido às permissões insuficientes fornecidas ao Device Defender Detect para publicar no tópico MQTT. Essa métrica está disponível somente para clientes que usam a exportação de métricas.
`NumOfMetricsExceedingSizeLimit`	O número de métricas ignoradas para exportação para uma métrica do lado da nuvem, do lado do dispositivo ou personalizada devido ao tamanho que excede as restrições de tamanho da mensagem MQTT. O sistema relata o número de métricas ignoradas para exportação para a conta, para uma métrica específica devido ao tamanho que excede as restrições de tamanho da mensagem MQTT. Essa métrica está disponível somente para clientes que usam a exportação de métricas.
`Violations`	O número de violações novas dos comportamentos de perfil de segurança que foram localizadas desde a última vez que uma avaliação foi feita. O sistema informará o número de violações novas da conta, para um perfil de segurança específico e para um

Métrica	Description
	comportamento específico de um perfil de segurança específico.
`ViolationsCleared`	O número de violações dos comportamentos de perfil de segurança que foram resolvidos desde a última vez que uma avaliação foi feita. O sistema informará o número de violações resolvidas da conta, para um perfil de segurança específico e para um comportamento específico de um perfil de segurança específico.
`ViolationsInvalidated`	O número de violações de comportamentos de perfil de segurança cujas informações não estão mais disponíveis desde a última vez que uma avaliação foi feita (porque o dispositivo de relatório parou de gerar relatórios ou não está mais sendo monitorado por algum motivo). O sistema informará o número de violações invalidadas da conta inteira, para um perfil de segurança específico e para um comportamento específico de um perfil de segurança específico.
`MisconfiguredDeviceDefender Notification`	Notifica você quando a configuração do SNS para AWS IoT Device Defender está mal configurada. Dimensões

Métricas de provisionamento de dispositivos

AWS IoT Métricas de aprovisionamento de frotas

Métrica	Description
`ApproximateNumberOfThingsRe gistered`	A contagem de objetos que foram registrados pela opção de aprovisionamento de frota. Embora a contagem geralmente seja precisa, a arquitetura distribuída do AWS IoT Core dificulta a

Métrica	Description
	manutenção de uma contagem precisa das objetos registradas. A estatística a ser utilizada para essa métrica é: • Máximo para relatar o número total de objetos que foram registradas. Para ver a contagem dos itens registrados durante a janela de CloudWatch agregação, consulte a `RegisterThingFailed` métrica. Dimensões: ClaimCertificateId
`CreateKeysAndCertificateFailed`	O número de falhas que ocorreram ao chamar a API MQTT `CreateKeysAndCertificate` . A métrica é emitida nos casos de Sucesso (valor = 0) e Falha (valor = 1). Essa métrica pode ser usada para rastrear o número de certificados criados e registrados durante as janelas CloudWatch de agregação suportadas, como 5 minutos ou 1 hora. As estatísticas disponíveis para essa métrica são: • Soma para relatar o número de chamadas malsucedidas. • SampleCountpara relatar o número total de chamadas bem-sucedidas e malsucedidas.

Métrica	Description
`CreateCertificateFromCsrFailed`	O número de falhas que ocorreram ao chamar a API MQTT `CreateCertificateFromCsr` . A métrica é emitida nos casos de Sucesso (valor = 0) e Falha (valor = 1). Essa métrica pode ser usada para rastrear o número de itens registrados durante as janelas CloudWatch de agregação suportadas, como 5 minutos ou 1 hora. As estatísticas disponíveis para essa métrica são: • Soma para relatar o número de chamadas malsucedidas. • SampleCountpara relatar o número total de chamadas bem-sucedidas e malsucedidas.
`RegisterThingFailed`	O número de falhas que ocorreram ao chamar a API MQTT `RegisterThing` . A métrica é emitida nos casos de Sucesso (valor = 0) e Falha (valor = 1). Essa métrica pode ser usada para rastrear o número de itens registrados durante as janelas CloudWatch de agregação suportadas, como 5 minutos ou 1 hora. Para ver o número total de objetos registradas, consulte a métrica `ApproximateNumberOfThingsRegistered` . As estatísticas disponíveis para essa métrica são: • Soma para relatar o número de chamadas malsucedidas. • SampleCountpara relatar o número total de chamadas bem-sucedidas e malsucedidas. Dimensões: TemplateName

Just-in-time métricas de provisionamento

Métrica	Description
`ProvisionThing.ClientError`	O número de vezes que um dispositivo falhou no provisionamento devido a um erro do cliente. Por exemplo, a política especificada no modelo não existia.
`ProvisionThing.ServerError`	O número de vezes que um dispositivo falhou no provisionamento devido a um erro no servidor. Os clientes podem tentar provisionar o dispositivo novamente depois de esperar e entrar em contato AWS IoT se o problema continuar o mesmo.
`ProvisionThing.Success`	O número de vezes que um dispositivo foi provision ado com êxito.

LoRaMétricas de WAN

A tabela a seguir mostra as métricas para AWS IoT Core for LoRaWAN. Para acessar mais informações, consulte Métricas do AWS IoT Core for LoRaWAN.

AWS IoT Core for LoRaWAN métricas

Métrica	Description
Dispositivos/gateways ativos	O número de dispositivos LoRa WAN e gateways ativos em sua conta.
Contagem de mensagens de uplink	O número de mensagens de uplink enviadas dentro de um período especificado para todos os gateways e dispositivos ativos em sua Conta da AWS. As mensagens de uplink são mensagens enviadas do seu dispositivo para o. AWS IoT Core for LoRaWAN
Contagem de mensagens de downlink	O número de mensagens de downlink enviadas dentro de um período especificado para todos os gateways e dispositivos ativos em sua Conta da

Métrica	Description
	AWS. As mensagens de downlink são mensagens enviadas AWS IoT Core for LoRaWAN para o seu dispositivo.
Taxa de perda de mensagens	Depois de adicionar seu dispositivo e se conectar AWS IoT Core for LoRaWAN, seu dispositivo pode iniciar uma mensagem de uplink para começar a trocar mensagens com a nuvem. Você pode usar essa métrica para rastrear a taxa de mensagens de uplink que são perdidas.
Métricas de entrada	Depois de adicionar seu dispositivo e gateway, você executa um procedimento de junção para que seu dispositivo possa enviar dados de uplink e se comunicar com. AWS IoT Core for LoRaWAN Você pode usar essa métrica para obter informações sobre métricas de junção para todos os dispositivos ativos em sua Conta da AWS.
Indicador de intensidade do sinal recebido (RSSI) médio	Você pode usar essa métrica para monitorar o RSSI (indicador de intensidade do sinal recebido) médio dentro do período especificado. O RSSI é uma medida que indica se o sinal é forte o suficiente para uma boa conexão sem fio. Esse valor é negativo e deve estar mais próximo de zero para uma conexão forte.
Relação média sinal/ruído (SNR)	Você pode usar essa métrica para monitorar a média de SNR (Signal-to-noise proporção) dentro da duração de tempo especificada. A SNR é uma medida que indica se o sinal recebido é forte o suficiente em comparação ao nível de ruído para uma boa conexão sem fio. O valor da SNR é positivo e deve ser maior que zero para indicar que a potência do sinal é mais forte que a potência do ruído.

Métrica	Description
Disponibilidade do gateway	Você pode usar essa métrica para obter informações sobre a disponibilidade desse gateway dentro de um período especificado. Essa métrica exibe o tempo de conexão do websocket desse gateway por um período de tempo especificado.

Just-in-time métricas de provisionamento

Métrica	Description
`ProvisionThing.ClientError`	O número de vezes que um dispositivo falhou no provisionamento devido a um erro do cliente. Por exemplo, a política especificada no modelo não existia.
`ProvisionThing.ServerError`	O número de vezes que um dispositivo falhou no provisionamento devido a um erro no servidor. Os clientes podem tentar provisionar o dispositivo novamente depois de esperar e entrar em contato AWS IoT se o problema continuar o mesmo.
`ProvisionThing.Success`	O número de vezes que um dispositivo foi provision ado com êxito.

Métricas de indexação de frota

AWS IoT métricas de indexação de frotas

Métrica	Description
`NamedShadowCountForDynamicG roupQueryLimitExceeded`	No máximo 25 sombras nomeadas por objeto são processadas para termos de consulta que não são específicos da fonte de dados em grupos dinâmicos de objetos. Quando esse limite é violado para algo, o tipo de evento `NamedShadowCountFo`

Métrica	Description
	`rDynamicGroupQueryLimitExceeded` será emitido.

Dimensões para métricas

As métricas usam o namespace e fornecem métricas para as dimensões a seguir

Dimensão	Description
`ActionType`	O tipo de ação especificado pela regra que acionou a solicitação.
`BehaviorName`	O nome do comportamento do perfil de segurança do Device Defender Detect que está sendo monitorado.
`ClaimCertificateId`	A `certificateId` da reivindicação usada para provisionar os dispositivos.
`CheckName`	O nome da verificação de auditoria do Device Defender cujos resultados estão sendo monitorados.
`JobId`	O ID do trabalho cujo progresso ou conexão de mensagem success/failure está sendo monitorado.
`Protocol`	O protocolo usado para fazer a solicitação. Os valores válidos são: MQTT ou HTTP
`RuleName`	O nome da regra disparada pela solicitação.
`ScheduledAuditName`	O nome da auditoria programada do Device Defender cujos resultados das verificações estão sendo monitorados. Terá o valor `OnDemand` se os resultados informados forem para uma auditoria que foi feita sob demanda.

Dimensão	Description
`SecurityProfileName`	O nome do perfil de segurança do Device Defender Detect cujos comportamentos estão sendo monitorados.
`TemplateName`	O nome do modelo de provisionamento.
`SourceArn`	Refere-se ao perfil de segurança para detecção ou ao banco da conta para auditoria.
`RoleArn`	Refere-se à função que o Device Defender tentou assumir.
`TopicArn`	Refere-se ao tópico do SNS no qual o Device Defender tentou publicar.
`Error`	Fornece uma breve descrição do erro recebido ao tentar publicar no tópico do SNS. Os valores possíveis são: • “KMSKeyNotFound“: indica que a chave KMS não existe para o tópico. • “InvalidTopicName“: indica que o tópico do SNS não é válido. • “KMSAccessNegado”: indica que a função não tem permissões para a chave KMS do Tópico. • “AuthorizationError“: indica que a função fornecida não autoriza o Device Defender a publicar no tópico do SNS. • “SNSTopicNotFound“: indica que o tópico SNS fornecido não existe. • “FailureToAssumeRole“: indica que a função fornecida não autoriza o Device Defender a assumir a função. • “CrossRegionSNSTopic“: indica que o tópico SNS existe em uma região diferente.

Monitore AWS IoT usando CloudWatch registros

Quando o AWS IoT registro está ativado, AWS IoT envia eventos de progresso sobre cada mensagem à medida que ela passa de seus dispositivos por meio do agente de mensagens e do mecanismo de regras. No CloudWatch console, CloudWatch os registros aparecem em um grupo de AWSIotregistros chamado Registros.

Para obter mais informações sobre CloudWatch registros, consulte CloudWatch Registros. Para obter informações sobre AWS IoT CloudWatch os registros compatíveis, consulteCloudWatch Registra entradas AWS IoT de registro.

Visualizando AWS IoT registros no CloudWatch console

> **Note**
>
> O grupo de `AWSIotLogsV2` registros não estará visível no CloudWatch console até:
>
> - Você ativou o login AWS IoT. Para obter mais informações sobre como ativar o login AWS IoT, consulte Configurar o AWS IoT registro
> - Algumas entradas de registro foram gravadas por AWS IoT operações.

Para ver seus AWS IoT registros no CloudWatch console

1. Navegue para https://console.aws.amazon.com/cloudwatch/. No painel de navegação, escolha a opção Grupos de logs.
2. Na caixa de texto Filtro, digite **`AWSIotLogsV2`** e, depois, pressione Enter.
3. Clique duas vezes no grupo de logs `AWSIotLogsV2`.
4. Escolha Pesquisar tudo. Uma lista completa dos AWS IoT registros gerados para sua conta é exibida.
5. Escolha o ícone de expansão para ver um fluxo individual.

Você também pode inserir uma consulta na caixa de texto Filtrar eventos. Estas são algumas consultas interessantes para experimentar:

- `{ $.logLevel = "INFO" }`

  Encontre todos os logs que têm um nível de log de `INFO`.

- `{ $.status = "Success" }`

  Encontre todos os logs que têm um status de `Success`.
- `{ $.status = "Success" && $.eventType = "GetThingShadow" }`

  Encontre todos os logs que têm um status de `Success` e um tipo de evento de `GetThingShadow`.

Para obter mais informações sobre a criação de expressões de filtro, consulte Consultas de CloudWatch registros.

CloudWatch Registra entradas AWS IoT de registro

Cada componente do AWS IoT gera suas próprias entradas de registro. Cada entrada de log tem um `eventType` que especifica a operação que fez com que a entrada de log fosse gerada. Esta seção descreve as entradas de log geradas pelos seguintes componentes da AWS IoT .

Tópicos

- Entradas de log do agente de mensagens
- Entradas de log OCSP do certificado do servidor
- Entradas de log da sombra do dispositivo
- Entradas de log do mecanismo de regras
- Entradas de log de tarefas
- Entradas de log de provisionamento de dispositivos
- Entradas de log de grupo dinâmico de objetos
- Entradas de log de indexação de frota
- CloudWatch Atributos de registros comuns

Entradas de log do agente de mensagens

O agente de AWS IoT mensagens gera entradas de registro para os seguintes eventos:

Tópicos

- Entrada de log de conexão
- Entrada de log de desconexão
- DeleteConnection entrada de registro

- GetRetainedMessage entrada de registro
- ListRetainedMessage entrada de registro
- Entrada de log de publicação de entrada
- Entrada de log de publicação de saída
- Entrada de log em fila
- Entrada de log de assinatura
- Entrada de log de cancelamento de assinatura

Entrada de log de conexão

O agente de AWS IoT mensagens gera uma entrada de registro com um `eventType` of `Connect` quando um cliente MQTT se conecta.

Exemplo de entrada de log de conexão

```
{
    "timestamp": "2017-08-10 15:37:23.476",
    "logLevel": "INFO",
    "traceId": "20b23f3f-d7f1-feae-169f-82263394fbdb",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Connect",
    "protocol": "MQTT",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `Connect` registro contêm os seguintes atributos:

clientId

O ID do cliente que está fazendo a solicitação.

principalId

O ID da entidade principal que está fazendo a solicitação.

protocolo

O protocolo usado para fazer a solicitação. Os valores válidos são `MQTT` ou `HTTP`.

sourceIp

O endereço IP no qual a solicitação foi originada.

sourcePort

A porta em que a solicitação foi originada.

Entrada de log de desconexão

O agente de AWS IoT mensagens gera uma entrada de registro com um `eventType` of `Disconnect` quando um cliente MQTT se desconecta.

Exemplo de entrada de log de desconexão

```
{
    "timestamp": "2017-08-10 15:37:23.476",
    "logLevel": "INFO",
    "traceId": "20b23f3f-d7f1-feae-169f-82263394fbdb",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Disconnect",
    "protocol": "MQTT",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490,
    "reason": "DUPLICATE_CLIENT_ID",
    "details": "A new connection was established with the same client ID",
    "disconnectReason": "CLIENT_INITIATED_DISCONNECT"
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `Disconnect` registro contêm os seguintes atributos:

clientId

O ID do cliente que está fazendo a solicitação.

principalId

O ID da entidade principal que está fazendo a solicitação.

protocolo

O protocolo usado para fazer a solicitação. Os valores válidos são `MQTT` ou `HTTP`.

sourceIp

O endereço IP no qual a solicitação foi originada.

sourcePort

A porta em que a solicitação foi originada.

razão

A razão pela qual o cliente está se desconectando.

detalhes

Uma breve explicação do erro.

disconnectReason

A razão pela qual o cliente está se desconectando.

DeleteConnection entrada de registro

O agente de AWS IoT mensagens gera uma entrada de registro com um `eventType` de `DeleteConnection` quando uma conexão do cliente MQTT é excluída.

DeleteConnection exemplo de entrada de registro

```
{
    "timestamp": "2025-08-09 15:37:23.476",
    "logLevel": "INFO",
    "traceId": "20b23f3f-d7f1-feae-169f-82263394fbdb",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "DeleteConnection",
    "protocol": "HTTP",
    "clientId": "abf27092886e49a8a5c1922749736453",
```

```
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `DeleteConnection` registro contêm os seguintes atributos:

clientId

O ID do cliente que será desconectado.

principalId

O ID da entidade principal que está fazendo a solicitação. Para ter informações sobre como identificar a entidade principal usando o ID da entidade principal, consulte Comparar identidades e credenciais do IAM do Guia usuário do IAM.

protocolo

O protocolo usado para fazer a solicitação. O valor válido é `HTTP`.

sourceIp

O endereço IP no qual a solicitação foi originada.

sourcePort

A porta em que a solicitação foi originada.

GetRetainedMessage entrada de registro

O agente de AWS IoT mensagens gera uma entrada de registro com um `eventType` de `GetRetainedMessage` quando `GetRetainedMessage`é chamado.

GetRetainedMessage exemplo de entrada de registro

```
{
    "timestamp": "2017-08-07 18:47:56.664",
    "logLevel": "INFO",
    "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
    "accountId": "123456789012",
    "status": "Success",
```

```
    "eventType": "GetRetainedMessage",
    "protocol": "HTTP",
    "topicName": "a/b/c",
    "qos": "1",
    "lastModifiedDate": "2017-08-07 18:47:56.664"
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `GetRetainedMessage` registro contêm os seguintes atributos:

lastModifiedDate

A data e a hora da Epoch, em milissegundos, em que a mensagem retida foi armazenada por. AWS IoT

protocolo

O protocolo usado para fazer a solicitação. Valor válido: `HTTP`.

qos

O nível Qualidade de Serviço (QoS) usado na solicitação de publicação. Os valores válidos são `0` ou 1.

topicName

O nome do tópico que você assinou.

ListRetainedMessage entrada de registro

O agente de AWS IoT mensagens gera uma entrada de registro com um `eventType` de `ListRetainedMessage` quando `ListRetainedMessages`é chamado.

ListRetainedMessage exemplo de entrada de registro

```
{
    "timestamp": "2017-08-07 18:47:56.664",
    "logLevel": "INFO",
    "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "ListRetainedMessage",
    "protocol": "HTTP"
```

```
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `ListRetainedMessage` registro contêm o seguinte atributo:

protocolo

O protocolo usado para fazer a solicitação. Valor válido: `HTTP`.

Entrada de log de publicação de entrada

Quando o agente de AWS IoT mensagens recebe uma mensagem MQTT, ele gera uma entrada de registro com um `eventType` de`Publish-In`.

Exemplo de entrada de log de publicação de entrada

```
{
        "timestamp": "2017-08-10 15:39:30.961",
        "logLevel": "INFO",
        "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
        "accountId": "123456789012",
        "status": "Success",
        "eventType": "Publish-In",
        "protocol": "MQTT",
        "topicName": "$aws/things/MyThing/shadow/get",
        "clientId": "abf27092886e49a8a5c1922749736453",
        "principalId":
 "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
        "sourceIp": "205.251.233.181",
        "sourcePort": 13490,
        "retain": "True"
    }
```

Além doCloudWatch Atributos de registros comuns, as entradas de `Publish-In` registro contêm os seguintes atributos:

clientId

O ID do cliente que está fazendo a solicitação.

principalId

O ID da entidade principal que está fazendo a solicitação.

protocolo

O protocolo usado para fazer a solicitação. Os valores válidos são `MQTT` ou `HTTP`.

reter

O atributo usado quando uma mensagem tem o sinalizador Reter definido com um valor de `True`. Se a mensagem não tiver o sinalizador Reter definido, esse atributo não aparecerá na entrada do log. Para obter mais informações, consulte Mensagens retidas do MQTT.

sourceIp

O endereço IP no qual a solicitação foi originada.

sourcePort

A porta em que a solicitação foi originada.

topicName

O nome do tópico que você assinou.

Entrada de log de publicação de saída

Quando o agente de mensagens publica uma mensagem MQTT, ele gera um log com um `eventType` de `Publish-Out`.

Exemplo de entrada de log de publicação de saída

```
{
    "timestamp": "2017-08-10 15:39:30.961",
    "logLevel": "INFO",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Publish-Out",
    "protocol": "MQTT",
    "topicName": "$aws/things/MyThing/shadow/get",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `Publish-Out` registro contêm os seguintes atributos:

clientId

O ID do cliente assinante que recebe mensagens sobre esse tópico do MQTT.

principalId

O ID da entidade principal que está fazendo a solicitação.

protocolo

O protocolo usado para fazer a solicitação. Os valores válidos são `MQTT` ou `HTTP`.

sourceIp

O endereço IP no qual a solicitação foi originada.

sourcePort

A porta em que a solicitação foi originada.

topicName

O nome do tópico que você assinou.

Entrada de log em fila

Quando um dispositivo com uma sessão persistente é desconectado, o agente de mensagens MQTT armazena as mensagens do dispositivo e AWS IoT gera entradas de registro com um EventType de. `Queued` Para obter mais informações sobre sessões persistentes do MQTT, consulteSessões persistentes do MQTT.

Exemplo de entrada de log de erros do servidor em fila

```
{
    "timestamp": "2022-08-10 15:39:30.961",
    "logLevel": "ERROR",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "topicName": "$aws/things/MyThing/get",
    "clientId": "123123123",
    "qos": "1",
```

```
    "protocol": "MQTT",
    "eventType": "Queued",
    "status": "Failure",
    "details": "Server Error"
}
```

Além doCloudWatch Atributos de registros comuns, as entradas do registro de erros do `Queued` servidor contêm os seguintes atributos:

clientId

O ID do cliente para o qual a mensagem está na fila.

detalhes

`Server Error`

Um erro no servidor impediu que a mensagem fosse armazenada.

protocolo

O protocolo usado para fazer a solicitação. O valor sempre será `MQTT`.

qos

O nível de qualidade do serviço (QoS) da solicitação. O valor sempre será 1 porque as mensagens com QoS de 0 não são armazenadas.

topicName

O nome do tópico que você assinou.

Exemplo de entrada de log de sucesso em fila

```
{
    "timestamp": "2022-08-10 15:39:30.961",
    "logLevel": "INFO",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "topicName": "$aws/things/MyThing/get",
    "clientId": "123123123",
    "qos": "1",
    "protocol": "MQTT",
```

```
    "eventType": "Queued",
    "status": "Success"
}
```

Além doCloudWatch Atributos de registros comuns, as entradas do registro de `Queued` sucesso contêm os seguintes atributos:

clientId

O ID do cliente para o qual a mensagem está na fila.

protocolo

O protocolo usado para fazer a solicitação. O valor sempre será `MQTT`.

qos

O nível de qualidade do serviço (QoS) da solicitação. O valor sempre será 1 porque as mensagens com QoS de 0 não são armazenadas.

topicName

O nome do tópico que você assinou.

Exemplo de entrada de log mantida em fila

```
{
    "timestamp": "2022-08-10 15:39:30.961",
    "logLevel": "ERROR",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "topicName": "$aws/things/MyThing/get",
    "clientId": "123123123",
    "qos": "1",
    "protocol": "MQTT",
    "eventType": "Queued",
    "status": "Failure",
    "details": "Throttled while queueing offline message"
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de registro `Queued` limitadas contêm os seguintes atributos:

clientId

O ID do cliente para o qual a mensagem está na fila.

detalhes

`Throttled while queueing offline message`

O cliente excedeu o limite de `Queued messages per second per account` , então a mensagem não foi armazenada.

protocolo

O protocolo usado para fazer a solicitação. O valor sempre será `MQTT`.

qos

O nível de qualidade do serviço (QoS) da solicitação. O valor sempre será 1 porque as mensagens com QoS de 0 não são armazenadas.

topicName

O nome do tópico que você assinou.

Entrada de log de assinatura

O agente de AWS IoT mensagens gera uma entrada de registro com um `eventType` de `Subscribe` quando um cliente MQTT se inscreve em um tópico.

Exemplo de entrada de log de assinatura MQTT 3

```
{
    "timestamp": "2017-08-10 15:39:04.413",
    "logLevel": "INFO",
    "traceId": "7aa5c38d-1b49-3753-15dc-513ce4ab9fa6",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Subscribe",
    "protocol": "MQTT",
    "topicName": "$aws/things/MyThing/shadow/#",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
```

```
    "sourcePort": 13490
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `Subscribe` registro contêm os seguintes atributos:

clientId

O ID do cliente que está fazendo a solicitação.

principalId

O ID da entidade principal que está fazendo a solicitação.

protocolo

O protocolo usado para fazer a solicitação. O valor sempre será `MQTT`.

sourceIp

O endereço IP no qual a solicitação foi originada.

sourcePort

A porta em que a solicitação foi originada.

topicName

O nome do tópico que você assinou.

Exemplo de entrada de log de assinatura MQTT 5

```
{
 "timestamp": "2022-11-30 16:24:15.628",
 "logLevel": "INFO",
 "traceId": "7aa5c38d-1b49-3753-15dc-513ce4ab9fa6",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "Subscribe",
 "protocol": "MQTT",
 "topicName": "test/topic1,$invalid/reserved/topic",
 "subscriptions": [
  {
   "topicName": "test/topic1",
```

```
    "reasonCode": 1
   },
   {
    "topicName": "$invalid/reserved/topic",
    "reasonCode": 143
   }
  ],
  "clientId": "abf27092886e49a8a5c1922749736453",
  "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
  "sourceIp": "205.251.233.181",
  "sourcePort": 13490
}
```

Para operações de assinatura do MQTT 5, além do CloudWatch Atributos de registros comuns e dos atributos de entrada de log e de assinatura do MQTT 3, as entradas de log do MQTT 5 `Subscribe` contêm o seguinte atributo:

assinaturas

Uma lista de mapeamentos entre os tópicos solicitados na solicitação de assinatura e o código de motivo individual do MQTT 5. Para obter mais informações, consulte Códigos de motivo do MQTT.

Entrada de log de cancelamento de assinatura

O agente de AWS IoT mensagens gera uma entrada de registro com um `eventType` de `Unsubscribe` quando um cliente MQTT cancela a assinatura de um tópico do MQTT.

Exemplo de entrada de log de cancelamento de assinatura MQTT

```
{
    "timestamp": "2024-08-20 22:53:32.844",
    "logLevel": "INFO",
    "traceId": "db6bd09a-2c3f-1cd2-27cc-fd6b1ce03b58",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Unsubscribe",
    "protocol": "MQTT",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
```

```
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `Unsubscribe` registro contêm os seguintes atributos:

protocolo

O protocolo usado para fazer a solicitação. O valor sempre será `MQTT`.

clientId

O ID do cliente que está fazendo a solicitação.

principalId

O ID da entidade principal que está fazendo a solicitação.

sourceIp

O endereço IP no qual a solicitação foi originada.

sourcePort

A porta em que a solicitação foi originada.

Entradas de log OCSP do certificado do servidor

AWS IoT Core gera entradas de registro para o seguinte evento:

Tópicos

- Recuperar entrada do registro OCSPStaple de dados
- Recuperar entrada OCSPStaple de registro de dados para endpoints privados

Recuperar entrada do registro OCSPStaple de dados

AWS IoT Core gera uma entrada de registro com um `eventType` of `RetrieveOCSPStapleData` quando o servidor recupera os dados básicos do OCSP.

Exemplos de entrada OCSPStaple de registro de dados

A seguir está um exemplo de entrada de log de `Success`.

```
{
 "timestamp": "2024-01-30 15:39:30.961",
 "logLevel": "INFO",
 "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "RetrieveOCSPStapleData",
 "domainConfigName": "test-domain-config-name",
 "connectionDetails": {
  "httpStatusCode": "200",
  "ocspResponderUri": "http://ocsp.example.com",
  "sourceIp": "205.251.233.181",
  "targetIp": "250.15.5.3"
 },
 "ocspRequestDetails": {
  "requesterName": "iot.amazonaws.com",
  "requestCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
 },
 "ocspResponseDetails": {
  "responseCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
  "ocspResponseStatus": "successful",
  "certStatus": "good",
  "signature":
 "4C:6F:63:61:6C:20:52:65:73:70:6F:6E:64:65:72:20:53:69:67:6E:61:74:75:72:65",
  "thisUpdateTime": "Jan 31 01:21:02 2024 UTC",
  "nextUpdateTime": "Feb 02 00:21:02 2024 UTC",
  "producedAtTime": "Jan 31 01:37:03 2024 UTC",
  "stapledDataPayloadSize": "XXX"
 }
}
```


A seguir está um exemplo de entrada de log de `Failure`.


```
{
 "timestamp": "2024-01-30 15:39:30.961",
 "logLevel": "ERROR",
 "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
 "accountId": "123456789012",
 "status": "Failure",
 "reason": "A non 2xx HTTP response was received from the OCSP responder.",
 "eventType": "RetrieveOCSPStapleData",
```

```
 "domainConfigName": "test-domain-config-name",
 "connectionDetails": {
  "httpStatusCode": "444",
  "ocspResponderUri": "http://ocsp.example.com",
  "sourceIp": "205.251.233.181",
  "targetIp": "250.15.5.3"
 },
 "ocspRequestDetails": {
  "requesterName": "iot.amazonaws.com",
  "requestCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
 }
}
```


Para a `RetrieveOCSPStaple` operação, além doCloudWatch Atributos de registros comuns, as entradas de registro contêm os seguintes atributos:

reason

A razão pela qual a operação falha.

domainConfigName

O nome da configuração do domínio.

connectionDetails

Uma breve explicação dos detalhes da conexão.

- httpStatusCode

  Códigos de status HTTP que são retornados pelo respondente OCSP em resposta à solicitação do cliente feita ao servidor.

- ocspResponderUri

  O URI do respondente OCSP que AWS IoT Core obtém do certificado do servidor.

- sourceIp

  O endereço IP de origem do AWS IoT Core servidor.

- targetIp

  O endereço IP de destino do agente de resposta OCSP.

ocspRequestDetails

Detalhes da solicitação OCSP.

- requesterName

  O identificador do AWS IoT Core servidor que envia uma solicitação ao respondente OCSP.

- requestCertId

  O ID do certificado da solicitação. Esse é o ID do certificado para o qual a resposta OCSP está sendo solicitada.

ocspResponseDetails

Detalhes da resposta do OCSP.

- responseCertId

  O ID do certificado da resposta OCSP.

- ocspResponseStatus

  O status da resposta do OCSP.

- certStatus

  O status do certificado.

- assinatura

  A assinatura aplicada à resposta por uma entidade confiável.

- thisUpdateTime

  A hora em que se sabe que o status indicado está correto.

- nextUpdateTime

  A hora igual ou anterior àquela em que as informações mais recentes estarão disponíveis sobre o status do certificado.

- producedAtTime

  A hora em que o respondente OCSP assinou essa resposta.

- stapledDataPayloadTamanho

  O tamanho da carga útil dos dados associados.

Recuperar entrada OCSPStaple de registro de dados para endpoints privados

AWS IoT Core gera uma entrada de registro com um `eventType` of `RetrieveOCSPStapleData` quando o servidor recupera os dados básicos do OCSP.

Recupere exemplos de entrada OCSPStaple de registro de dados para endpoints privados

A seguir está um exemplo de entrada de log de `Success`.

```
{
 "timestamp": "2024-01-30 15:39:30.961",
 "logLevel": "INFO",
 "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "RetrieveOCSPStapleData",
 "domainConfigName": "test-domain-config-name",
        "lambdaDetails": {
        "lambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
        "sourceArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
testDomainConfigure/6bzfg"
      },
        "authorizedResponderArn": "arn:aws:acm:us-west-2:123456789012:certificate/
certificate_ID",
 "ocspRequestDetails": {
  "requesterName": "iot.amazonaws.com",
  "requestCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
 },
 "ocspResponseDetails": {
  "responderId": "04:C1:3F:8F:27:D6:49:13:F8:DE:B2:36:9D:85:8E:F8:31:3B:A6:D0"
              "responseCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
  "ocspResponseStatus": "successful",
  "certStatus": "good",
  "signature":
 "4C:6F:63:61:6C:20:52:65:73:70:6F:6E:64:65:72:20:53:69:67:6E:61:74:75:72:65",
  "thisUpdateTime": "Jan 31 01:21:02 2024 UTC",
  "nextUpdateTime": "Feb 02 00:21:02 2024 UTC",
  "producedAtTime": "Jan 31 01:37:03 2024 UTC",
  "stapledDataPayloadSize": "XXX"
 }
}
```

A seguir está um exemplo de entrada de log de `Failure`.


```
{
 "timestamp": "2024-01-30 15:39:30.961",
 "logLevel": "ERROR",
 "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
 "accountId": "123456789012",
 "status": "Failure",
 "reason": "The payload returned by the Lambda function exceeds the maximum response
 size of 7 kilobytes.",
 "eventType": "RetrieveOCSPStapleData",
 "domainConfigName": "test-domain-config-name",
        "lambdaDetails": {
        "lambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
        "sourceArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
testDomainConfigure/6bzfg"
      },
        "authorizedResponderArn": "arn:aws:acm:us-west-2:123456789012:certificate/
certificate_ID",
  "ocspRequestDetails": {
  "requesterName": "iot.amazonaws.com",
  "requestCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
 }
}
```


Para a `RetrieveOCSPStaple` operação, além dos atributos CloudWatch Atributos de registros comuns e na entrada de registro Recuperar OCSPStaple dados, as entradas de registro para endpoints privados contêm os seguintes atributos:

lambdaDetails

Detalhes da função do Lambda.

- lambdaArn

  O ARN da função Lambda.

- sourceArn

  O ARN da configuração do domínio.

authorizedResponderArn

O ARN do respondente autorizador, se houver um definido na configuração do domínio.

Entradas de log da sombra do dispositivo

O serviço AWS IoT Device Shadow gera entradas de registro para os seguintes eventos:

Tópicos

- DeleteThingShadow entrada de registro
- GetThingShadow entrada de registro
- UpdateThingShadow entrada de registro

DeleteThingShadow entrada de registro

O serviço sombra do dispositivo gera uma entrada de log com um `eventType` de `DeleteThingShadow` quando uma solicitação para excluir a sombra de um dispositivo é recebida.

DeleteThingShadow exemplo de entrada de registro

```
{
    "timestamp": "2017-08-07 18:47:56.664",
    "logLevel": "INFO",
    "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "DeleteThingShadow",
    "protocol": "MQTT",
    "deviceShadowName": "Jack",
    "topicName": "$aws/things/Jack/shadow/delete"
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `DeleteThingShadow` registro contêm os seguintes atributos:

deviceShadowName

O nome da sombra a ser atualizada.

protocolo

O protocolo usado para fazer a solicitação. Os valores válidos são `MQTT` ou `HTTP`.

topicName

O nome do tópico no qual a solicitação foi publicada.

GetThingShadow entrada de registro

O serviço Sombra do dispositivo gera uma entrada de log com um `eventType` de `GetThingShadow` quando uma solicitação para obter uma sombra é recebida.

GetThingShadow exemplo de entrada de registro

```
{
    "timestamp": "2017-08-09 17:56:30.941",
    "logLevel": "INFO",
    "traceId": "b575f19a-97a2-cf72-0ed0-c64a783a2504",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "GetThingShadow",
    "protocol": "MQTT",
    "deviceShadowName": "MyThing",
    "topicName": "$aws/things/MyThing/shadow/get"
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `GetThingShadow` registro contêm os seguintes atributos:

deviceShadowName

O nome da sombra solicitada.

protocolo

O protocolo usado para fazer a solicitação. Os valores válidos são `MQTT` ou `HTTP`.

topicName

O nome do tópico no qual a solicitação foi publicada.

UpdateThingShadow entrada de registro

O serviço Sombra do dispositivo gera uma entrada de log com um `eventType` de `UpdateThingShadow` quando uma solicitação para atualizar a sombra de um dispositivo é recebida.

UpdateThingShadow exemplo de entrada de registro

```
{
    "timestamp": "2017-08-07 18:43:59.436",
    "logLevel": "INFO",
    "traceId": "d0074ba8-0c4b-a400-69df-76326d414c28",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "UpdateThingShadow",
    "protocol": "MQTT",
    "deviceShadowName": "Jack",
    "topicName": "$aws/things/Jack/shadow/update"
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `UpdateThingShadow` registro contêm os seguintes atributos:

deviceShadowName

O nome da sombra a ser atualizada.

protocolo

O protocolo usado para fazer a solicitação. Os valores válidos são `MQTT` ou `HTTP`.

topicName

O nome do tópico no qual a solicitação foi publicada.

Entradas de log do mecanismo de regras

O mecanismo de AWS IoT regras gera registros para os seguintes eventos:

Tópicos

- FunctionExecution entrada de registro
- RuleExecution entrada de registro
- RuleMatch entrada de registro

- RuleExecutionThrottled entrada de registro
- RuleNotFound entrada de registro
- StartingRuleExecution entrada de registro

FunctionExecution entrada de registro

O mecanismo de regras gera uma entrada de log com um `eventType` de `FunctionExecution` quando uma consulta SQL de uma regra chama uma função externa. Uma função externa é chamada quando a ação de uma regra faz uma solicitação HTTP para AWS IoT ou outro serviço da Web (por exemplo, chamando `get_thing_shadow` ou`machinelearning_predict`).

FunctionExecution exemplo de entrada de registro

```
{
    "timestamp": "2017-07-13 18:33:51.903",
    "logLevel": "DEBUG",
    "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
    "status": "Success",
    "eventType": "FunctionExecution",
    "clientId": "N/A",
    "topicName":"rules/test",
    "ruleName": "ruleTestPredict",
    "ruleAction": "MachinelearningPredict",
    "resources": {
        "ModelId": "predict-model"
    },
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `FunctionExecution` registro contêm os seguintes atributos:

clientId

N/A para logs `FunctionExecution`.

principalId

O ID da entidade principal que está fazendo a solicitação.

recursos

Uma coleção de recursos usados pelas ações da regra.

ruleName

O nome da regra correspondente.

topicName

O nome do tópico que você assinou.

RuleExecution entrada de registro

Quando o mecanismo de AWS IoT regras aciona a ação de uma regra, ele gera uma entrada de `RuleExecution` registro.

RuleExecution exemplo de entrada de registro

```
{
    "timestamp": "2017-08-10 16:32:46.070",
    "logLevel": "INFO",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "RuleExecution",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "rules/test",
    "ruleName": "JSONLogsRule",
    "ruleAction": "RepublishAction",
    "resources": {
        "RepublishTopic": "rules/republish"
    },
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

RuleExecution com exemplo de entrada de registro em lote

```
{
    "logLevel": "INFO",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "RuleExecution",
    "ruleName": "rule_test",
    "ruleAction": "HttpAction",
    "resources": {
        "Url": "https://example.com",
```

```
        "ConfirmationUrl": "https://example.com"
    },
    "details": "HttpAction made a request to the specified endpoint",
    "batchDetails": {
        "timestamps": [
            "1234567890123",
            "1234567890123",
            "1234567890123"
        ],
        "traceIds": [
            "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
            "30aa7ccc-1d23-0b97-aa7b-76196d83537c",
            "30aa7ccc-1d23-0b97-aa7b-76196d83537d"
        ],
        "clientIds": [
            "N/A",
            "N/A",
            "N/A"
        ],
        "topicNames": [
            "topic/ruletest",
            "topic/ruletest",
            "topic/ruletest"
        ],
        "principalIds": [
            "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
            "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
            "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
        ],
        "batchSize": 3,
        "batchSizeInBytes": 114
    }
}
```


Além do CloudWatch Atributos de registros comuns, as entradas de log `RuleExecution` contêm os seguintes atributos:

clientId

O ID do cliente que está fazendo a solicitação.

principalId

O ID da entidade principal que está fazendo a solicitação.

recursos

Uma coleção de recursos usados pelas ações da regra.

ruleAction

O nome da ação disparada.

ruleName

O nome da regra correspondente.

topicName

O nome do tópico que você assinou.

RuleMatch entrada de registro

O mecanismo de AWS IoT regras gera uma entrada de registro com um `eventType` de `RuleMatch` quando o agente de mensagens recebe uma mensagem que corresponde a uma regra.

RuleMatch exemplo de entrada de registro

```
{
    "timestamp": "2017-08-10 16:32:46.002",
    "logLevel": "INFO",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "RuleMatch",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "rules/test",
    "ruleName": "JSONLogsRule",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `RuleMatch` registro contêm os seguintes atributos:

clientId

O ID do cliente que está fazendo a solicitação.

principalId

O ID da entidade principal que está fazendo a solicitação.

ruleName

O nome da regra correspondente.

topicName

O nome do tópico que você assinou.

RuleExecutionThrottled entrada de registro

Quando uma execução é limitada, o mecanismo de AWS IoT regras gera uma entrada de registro com um `eventType` de. `RuleExecutionThrottled`

RuleExecutionThrottled exemplo de entrada de registro

```
{
    "timestamp": "2017-10-04 19:25:46.070",
    "logLevel": "ERROR",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleExecutionThrottled",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "$aws/rules/example_rule",
    "ruleName": "example_rule",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "reason": "RuleExecutionThrottled",
    "details": "Exection of Rule example_rule throttled"
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `RuleExecutionThrottled` registro contêm os seguintes atributos:

clientId

O ID do cliente que está fazendo a solicitação.

detalhes

Uma breve explicação do erro.

principalId

O ID da entidade principal que está fazendo a solicitação.

reason

A string "RuleExecutionThrottled”.

ruleName

O nome da regra a ser acionada.

topicName

O nome do tópico publicado.

RuleNotFound entrada de registro

Quando o mecanismo de AWS IoT regras não consegue encontrar uma regra com um determinado nome, ele gera uma entrada de registro com um `eventType` de`RuleNotFound`.

RuleNotFound exemplo de entrada de registro

```
{
    "timestamp": "2017-10-04 19:25:46.070",
    "logLevel": "ERROR",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleNotFound",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "$aws/rules/example_rule",
    "ruleName": "example_rule",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "reason": "RuleNotFound",
    "details": "Rule example_rule not found"
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `RuleNotFound` registro contêm os seguintes atributos:

clientId

O ID do cliente que está fazendo a solicitação.

detalhes

Uma breve explicação do erro.

principalId

O ID da entidade principal que está fazendo a solicitação.

reason

A string "RuleNotFound”.

ruleName

O nome da regra que não pôde ser encontrada.

topicName

O nome do tópico publicado.

StartingRuleExecution entrada de registro

Quando o mecanismo de AWS IoT regras começa a acionar a ação de uma regra, ele gera uma entrada de registro com um `eventType` de`StartingRuleExecution`.

StartingRuleExecution exemplo de entrada de registro

```
{
    "timestamp": "2017-08-10 16:32:46.002",
    "logLevel": "DEBUG",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "StartingRuleExecution",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "rules/test",
    "ruleName": "JSONLogsRule",
    "ruleAction": "RepublishAction",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `rule-` registro contêm os seguintes atributos:

clientId

O ID do cliente que está fazendo a solicitação.

principalId

O ID da entidade principal que está fazendo a solicitação.

ruleAction

O nome da ação disparada.

ruleName

O nome da regra correspondente.

topicName

O nome do tópico que você assinou.

Entradas de log de tarefas

O serviço AWS IoT Job gera entradas de registro para os seguintes eventos. As entradas de log são geradas quando uma solicitação HTTP ou MQTT é recebida do dispositivo.

Tópicos

- DescribeJobExecution entrada de registro
- GetPendingJobExecution entrada de registro
- ReportFinalJobExecutionCount entrada de registro
- StartNextPendingJobExecution entrada de registro
- UpdateJobExecution entrada de registro

DescribeJobExecution entrada de registro

O serviço AWS IoT Jobs gera uma entrada de registro com um `eventType` de `DescribeJobExecution` quando o serviço recebe uma solicitação para descrever a execução de um trabalho.

DescribeJobExecution exemplo de entrada de registro

```
{
    "timestamp": "2017-08-10 19:13:22.841",
```

```
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "DescribeJobExecution",
    "protocol": "MQTT",
    "clientId": "thingOne",
    "jobId": "002",
    "topicName": "$aws/things/thingOne/jobs/002/get",
    "clientToken": "myToken",
    "details": "The request status is SUCCESS."
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `GetJobExecution` registro contêm os seguintes atributos:

clientId

O ID do cliente que está fazendo a solicitação.

clientToken

Um identificador exclusivo e que diferencia maiúsculas e minúsculas, para garantir a idempotência da solicitação. Para obter mais informações, consulte Como garantir a idempotência.

detalhes

Outras informações do serviço Tarefas.

jobId

O ID da tarefa para a execução do tarefa.

protocolo

O protocolo usado para fazer a solicitação. Os valores válidos são `MQTT` ou `HTTP`.

topicName

O tópico usado para fazer a solicitação.

GetPendingJobExecution entrada de registro

O serviço AWS IoT Jobs gera uma entrada de registro com um `eventType` de `GetPendingJobExecution` quando o serviço recebe uma solicitação de execução de trabalho.

GetPendingJobExecution exemplo de entrada de registro

```
{
    "timestamp": "2018-06-13 17:45:17.197",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "GetPendingJobExecution",
    "protocol": "MQTT",
    "clientId": "299966ad-54de-40b4-99d3-4fc8b52da0c5",
    "topicName": "$aws/things/299966ad-54de-40b4-99d3-4fc8b52da0c5/jobs/get",
    "clientToken": "24b9a741-15a7-44fc-bd3c-1ff2e34e5e82",
    "details": "The request status is SUCCESS."
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `GetPendingJobExecution` registro contêm os seguintes atributos:

clientId

O ID do cliente que está fazendo a solicitação.

clientToken

Um identificador exclusivo e que diferencia maiúsculas e minúsculas, para garantir a idempotência da solicitação. Para obter mais informações, consulte Como garantir a idempotência.

detalhes

Outras informações do serviço Jobs.

protocolo

O protocolo usado para fazer a solicitação. Os valores válidos são `MQTT` ou `HTTP`.

topicName

O nome do tópico que você assinou.

ReportFinalJobExecutionCount entrada de registro

O serviço AWS IoT Jobs gera uma entrada de registro com um `entryType` de `ReportFinalJobExecutionCount` quando um trabalho é concluído.

ReportFinalJobExecutionCount exemplo de entrada de registro

```
{
    "timestamp": "2017-08-10 19:44:16.776",
    "logLevel": "INFO",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "ReportFinalJobExecutionCount",
    "jobId": "002",
    "details": "Job 002 completed. QUEUED job execution count: 0 IN_PROGRESS job
 execution count: 0 FAILED job execution count: 0 SUCCEEDED job execution count: 1
 CANCELED job execution count: 0 REJECTED job execution count: 0 REMOVED job execution
 count: 0"
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `ReportFinalJobExecutionCount` registro contêm os seguintes atributos:

detalhes

Outras informações do serviço Tarefas.

jobId

O ID da tarefa para a execução do tarefa.

StartNextPendingJobExecution entrada de registro

Quando recebe uma solicitação para iniciar a próxima execução do trabalho pendente, o serviço AWS IoT Jobs gera uma entrada de registro com um `eventType` de`StartNextPendingJobExecution`.

StartNextPendingJobExecution exemplo de entrada de registro

```
{
    "timestamp": "2018-06-13 17:49:51.036",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "StartNextPendingJobExecution",
    "protocol": "MQTT",
    "clientId": "95c47808-b1ca-4794-bc68-a588d6d9216c",
    "topicName": "$aws/things/95c47808-b1ca-4794-bc68-a588d6d9216c/jobs/start-next",
```

```
    "clientToken": "bd7447c4-3a05-49f4-8517-dd89b2c68d94",
    "details": "The request status is SUCCESS."
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `StartNextPendingJobExecution` registro contêm os seguintes atributos:

clientId

O ID do cliente que está fazendo a solicitação.

clientToken

Um identificador exclusivo e que diferencia maiúsculas e minúsculas, para garantir a idempotência da solicitação. Para obter mais informações, consulte Como garantir a idempotência.

detalhes

Outras informações do serviço Jobs.

protocolo

O protocolo usado para fazer a solicitação. Os valores válidos são `MQTT` ou `HTTP`.

topicName

O tópico usado para fazer a solicitação.

UpdateJobExecution entrada de registro

O serviço AWS IoT Jobs gera uma entrada de registro com um `eventType` de `UpdateJobExecution` quando o serviço recebe uma solicitação para atualizar a execução de um trabalho.

UpdateJobExecution exemplo de entrada de registro

```
{
    "timestamp": "2017-08-10 19:25:14.758",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "UpdateJobExecution",
    "protocol": "MQTT",
```

```
    "clientId": "thingOne",
    "jobId": "002",
    "topicName": "$aws/things/thingOne/jobs/002/update",
    "clientToken": "myClientToken",
    "versionNumber": "1",
    "details": "The destination status is IN_PROGRESS. The request status is SUCCESS."
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `UpdateJobExecution` registro contêm os seguintes atributos:

clientId

O ID do cliente que está fazendo a solicitação.

clientToken

Um identificador exclusivo e que diferencia maiúsculas e minúsculas, para garantir a idempotência da solicitação. Para obter mais informações, consulte Como garantir a idempotência.

detalhes

Outras informações do serviço Tarefas.

jobId

O ID da tarefa para a execução do tarefa.

protocolo

O protocolo usado para fazer a solicitação. Os valores válidos são `MQTT` ou `HTTP`.

topicName

O tópico usado para fazer a solicitação.

versionNumber

A versão da execução da tarefa.

Entradas de log de provisionamento de dispositivos

O serviço de provisionamento de AWS IoT dispositivos gera registros para os seguintes eventos.

Tópicos

- GetDeviceCredentials entrada de registro
- ProvisionDevice entrada de registro

GetDeviceCredentials entrada de registro

O serviço de provisionamento de AWS IoT dispositivos gera uma entrada de registro com um `eventType` de `GetDeviceCredential` quando um cliente liga. `GetDeviceCredential`

GetDeviceCredentials exemplo de entrada de registro

```
{
  "timestamp" : "2019-02-20 20:31:22.932",
  "logLevel" : "INFO",
  "traceId" : "8d9c016f-6cc7-441e-8909-7ee3d5563405",
  "accountId" : "123456789101",
  "status" : "Success",
  "eventType" : "GetDeviceCredentials",
  "deviceCertificateId" :
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
  "details" : "Additional details about this log."
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `GetDeviceCredentials` registro contêm os seguintes atributos:

detalhes

Uma breve explicação do erro.

deviceCertificateId

O ID do certificado do dispositivo.

ProvisionDevice entrada de registro

O serviço de provisionamento de AWS IoT dispositivos gera uma entrada de registro com um `eventType` de `ProvisionDevice` quando um cliente liga. `ProvisionDevice`

ProvisionDevice exemplo de entrada de registro

```
{
```

```
  "timestamp" : "2019-02-20 20:31:22.932",
  "logLevel" : "INFO",
  "traceId" : "8d9c016f-6cc7-441e-8909-7ee3d5563405",
  "accountId" : "123456789101",
  "status" : "Success",
  "eventType" : "ProvisionDevice",
  "provisioningTemplateName" : "myTemplate",
  "deviceCertificateId" :
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
  "details" : "Additional details about this log."
 }
```

Além doCloudWatch Atributos de registros comuns, as entradas de `ProvisionDevice` registro contêm os seguintes atributos:

detalhes

Uma breve explicação do erro.

deviceCertificateId

O ID do certificado do dispositivo.

provisioningTemplateName

O nome do modelo de provisionamento.

Entradas de log de grupo dinâmico de objetos

AWS IoT Os Dynamic Thing Groups geram registros para o seguinte evento.

Tópicos

- AddThingToDynamicThingGroupsFailed entrada de registro

AddThingToDynamicThingGroupsFailed entrada de registro

Quando não AWS IoT foi possível adicionar algo aos grupos dinâmicos especificados, ele gera uma entrada de registro com um `eventType` de`AddThingToDynamicThingGroupsFailed`. Isso acontece quando um objeto atendia aos critérios para estar no grupo dinâmico e, no entanto, não pôde ser adicionada ao grupo dinâmico ou foi removida do grupo dinâmico. Isso pode acontecer em função de:

- A objeto já pertence ao número máximo de grupos.
- A opção --override-dynamic-groups foi usada para adicionar o objeto a um grupo de objetos estáticas. Foi removido de um grupo dinâmico de objetos para tornar isso possível.

Para obter mais informações, consulte Limitações e conflitos de grupo de objetos dinâmicas.

AddThingToDynamicThingGroupsFailed exemplo de entrada de registro

Este exemplo mostra a entrada de log de um erro `AddThingToDynamicThingGroupsFailed`. Neste exemplo, TestThingatendeu aos critérios para estar nos grupos de coisas dinâmicas listados em`dynamicThingGroupNames`, mas não pôde ser adicionado a esses grupos dinâmicos, conforme descrito em`reason`.

```
{
 "timestamp": "2020-03-16 22:24:43.804",
 "logLevel": "ERROR",
 "traceId": "70b1f2f5-d95e-f897-9dcc-31e68c3e1a30",
 "accountId": "57EXAMPLE833",
 "status": "Failure",
 "eventType": "AddThingToDynamicThingGroupsFailed",
 "thingName": "TestThing",
 "dynamicThingGroupNames": [
  "DynamicThingGroup11",
  "DynamicThingGroup12",
  "DynamicThingGroup13",
  "DynamicThingGroup14"
 ],
 "reason": "The thing failed to be added to the given dynamic thing group(s) because
 the thing already belongs to the maximum allowed number of groups."
}
```

Além doCloudWatch Atributos de registros comuns, as entradas de `AddThingToDynamicThingGroupsFailed` registro contêm os seguintes atributos:

dynamicThingGroupNomes

Uma matriz dos grupos de objetos dinâmicas aos quais o objeto não foi adicionada.

reason

A razão pela qual o objeto não foi adicionada aos grupos de objetos dinâmicas.

thingName

O nome do objeto que não foi adicionada a um grupo de objetos dinâmicas.

Entradas de log de indexação de frota

AWS IoT a indexação da frota gera entradas de registro para os seguintes eventos.

Tópicos

- NamedShadowCountForDynamicGroupQueryLimitExceeded entrada de registro

NamedShadowCountForDynamicGroupQueryLimitExceeded entrada de registro

No máximo 25 sombras nomeadas por objeto são processadas para termos de consulta que não são específicos da fonte de dados em grupos dinâmicos. Quando esse limite é violado para algo, o tipo de evento `NamedShadowCountForDynamicGroupQueryLimitExceeded` será emitido.

NamedShadowCountForDynamicGroupQueryLimitExceeded exemplo de entrada de registro

Este exemplo mostra a entrada de log de um erro `NamedShadowCountForDynamicGroupQueryLimitExceeded`. Neste exemplo, os `DynamicGroup` resultados baseados em todos os valores podem ser imprecisos, conforme descrito no campo `reason`.

```
{
"timestamp": "2020-03-16 22:24:43.804",
"logLevel": "ERROR",
"traceId": "70b1f2f5-d95e-f897-9dcc-31e68c3e1a30",
"accountId": "571032923833",
"status": "Failure",
"eventType": "NamedShadowCountForDynamicGroupQueryLimitExceeded",
"thingName": "TestThing",
"reason": "A maximum of 25 named shadows per thing are processed for non-data source
 specific query terms in dynamic groups."
}
```

CloudWatch Atributos de registros comuns

Todas as entradas de registro de CloudWatch registros incluem os seguintes atributos:

accountId

Sua Conta da AWS identidade.

eventType

O tipo de evento para o qual o log foi gerado. O valor do tipo de evento depende do evento que gerou a entrada de log. Cada descrição de entrada de log inclui o valor de `eventType` para essa entrada de log.

logLevel

O nível de log que está sendo usado. Para obter mais informações, consulte the section called “Níveis de logs”.

status

O status da solicitação.

timestamp

A data e hora do UNIX legíveis de quando o cliente se conectou ao agente de mensagens da AWS IoT .

traceId

Um identificador gerado aleatoriamente que pode ser usado para correlacionar todos os logs para uma solicitação específica.

Faça upload de registros do lado do dispositivo para a Amazon CloudWatch

Você pode fazer upload de registros históricos do lado do dispositivo na Amazon CloudWatch para monitorar e analisar a atividade de um dispositivo no campo. Os logs do lado do dispositivo podem incluir arquivos de logs do sistema, do aplicativo e do dispositivo. Esse processo usa um parâmetro de ação de regras de CloudWatch registros para publicar registros do lado do dispositivo em um grupo de registros definido pelo cliente.

Como funciona

O processo começa quando um AWS IoT dispositivo envia mensagens MQTT contendo arquivos de log formatados para um AWS IoT tópico. Uma AWS IoT regra monitora o tópico da mensagem

e envia os arquivos de log para um grupo de CloudWatch registros definido por você. Em seguida, você pode revisar e analisar as informações.

Tópicos

- Tópicos do MQTT
- Ação da regra

Tópicos do MQTT

Escolha um espaço de nome de tópico do MQTT que você usará para publicar os logs. Recomendamos usar esse formato para o espaço de tópicos comuns,`$aws/rules/things/thing_name/logs`, e esse formato para tópicos de erro, `$aws/rules/things/thing_name/logs/errors`. A estrutura de nomenclatura para logs e tópicos de erro é recomendada, mas não obrigatória. Para obter mais informações, consulte Elaboração de tópicos MQTT para AWS IoT Core.

Ao usar o espaço de tópico comum recomendado, você utiliza tópicos reservados do AWS IoT Basic Ingest. AWS IoT O Basic Ingest envia com segurança os dados do dispositivo para os AWS serviços que são suportados por AWS IoT ações de regras. Ele remove o agente de publish/subscribe mensagens do caminho de ingestão, tornando-o mais econômico. Para obter mais informações, consulte Reduzir custos do sistema de mensagens com a ingestão básica no.

Se você usa o BatchMode para carregar arquivos de log, suas mensagens devem seguir um formato específico que inclua um carimbo de data/hora e uma mensagem do UNIX. Para obter mais informações, consulte o tópico Requisitos de formato de mensagem MQTT para BatchMode em Ação de regra de CloudWatch registros.

Ação da regra

Quando AWS IoT recebe as mensagens MQTT dos dispositivos cliente, uma AWS IoT regra monitora o tópico definido pelo cliente e publica o conteúdo em um grupo de CloudWatch registros definido por você. Esse processo usa uma ação de regra de CloudWatch registros para monitorar o MQTT em lotes de arquivos de log. Para obter mais informações, consulte a ação da AWS IoT regra de CloudWatch registros.

Modo Batch

`batchMode`é um parâmetro booleano dentro da ação da regra AWS IoT CloudWatch Logs. Esse parâmetro é opcional e está desativado (`false`) por padrão. Para carregar arquivos de log do lado

do dispositivo em lotes, você deve ativar esse parâmetro (`true`) ao criar a regra. AWS IoT Para obter mais informações, consulte CloudWatch Registros na seção de ações de AWS IoT regras.

Carregar registros do lado do dispositivo usando regras AWS IoT

Você pode usar o mecanismo de AWS IoT regras para carregar registros de log de arquivos de log existentes do lado do dispositivo (registros do sistema, do aplicativo e do dispositivo-cliente) para a Amazon. CloudWatch Quando os registros do lado do dispositivo são publicados em um tópico do MQTT, a ação de regras de CloudWatch registros transfere as mensagens para os registros. CloudWatch Esse processo descreve como fazer upload de logs de dispositivos em lotes usando o `batchMode` parâmetro de ação de regras ativado (definido como `true`).

Para começar a enviar registros do lado do dispositivo para CloudWatch, preencha os pré-requisitos a seguir.

Pré-requisitos

Antes de começar, faça o seguinte:

- Crie pelo menos um dispositivo de IoT de destino que esteja registrado AWS IoT Core como uma AWS IoT coisa. Para obter mais informações, consulte Criar um objeto.
- Determine o espaço do tópico do MQTT para ingestão e erros. Para obter mais informações sobre os tópicos do MQTT e as convenções de nomenclatura recomendadas, consulte a seção de tópicos do Tópicos do MQTT MQTT em Carregar registros do lado do dispositivo para a Amazon. CloudWatch

Para obter mais informações sobre esses pré-requisitos, consulte Carregar registros do lado do dispositivo para. CloudWatch

Criação de um grupo de CloudWatch registros

Para criar um grupo de CloudWatch registros, conclua as etapas a seguir. Escolha a guia apropriada, dependendo se você prefere executar as etapas por meio do Console de gerenciamento da AWS ou do AWS Command Line Interface (AWS CLI).

Console de gerenciamento da AWS

Para criar um grupo de CloudWatch registros usando o Console de gerenciamento da AWS

1. Abra o Console de gerenciamento da AWS e navegue até CloudWatch.

2. Na barra de navegação, escolha Logs e, em seguida, escolha Grupos de logs.
3. Escolha a opção Criar grupo de logs.
4. Atualize o nome do grupo de logs e, opcionalmente, atualize os campos de de configuração de retenção.
5. Escolha Criar.

AWS CLI

Para criar um grupo de CloudWatch registros usando o AWS CLI

1. Execute os comandos a seguir para criar o grupo de logs. Para obter mais informações, consulte `create-log-group` na Referência de Comandos AWS CLI v2.

   Substitua o nome do grupo de logs no exemplo (`uploadLogsGroup`) pelo nome de sua preferência.

   ```
   aws logs create-log-group --log-group-name uploadLogsGroup
   ```

2. Para confirmar se o grupo de logs foi criado corretamente, execute o comando a seguir.

   ```
   aws logs describe-log-groups --log-group-name-prefix uploadLogsGroup
   ```

   Exemplo de resultado:

   ```
   {
       "logGroups": [
           {
               "logGroupName": "uploadLogsGroup",
               "creationTime": 1674521804657,
               "metricFilterCount": 0,
               "arn": "arn:aws:logs:us-east-1:111122223333:log-
   group:uploadLogsGroup:*",
               "storedBytes": 0
           }
       ]
   }
   ```

Criação de uma regra de tópico

Para criar uma AWS IoT regra, conclua as etapas a seguir. Escolha a guia apropriada, dependendo se você prefere executar as etapas por meio do Console de gerenciamento da AWS ou do AWS Command Line Interface (AWS CLI).

Console de gerenciamento da AWS

Para criar uma regra de tópico usando o Console de gerenciamento da AWS

1. Abra o hub de regras.
   a. Abra o Console de gerenciamento da AWS e navegue até AWS IoT.
   b. Na barra de navegação, escolha Roteamento de mensagens e, em seguida, Regras.
   c. Escolha a opção Criar regra.
2. Insira as propriedades da regra.
   a. Insira um nome de regra alfanumérico.
   b. (Opcional) Insira uma Descrição da regra e Tags.
   c. Escolha Próximo.
3. Insira uma instrução SQL.
   a. Insira uma instrução SQL usando o tópico MQTT que você definiu para ingestão.

      Por exemplo, `SELECT * FROM '$aws/rules/things/thing_name/logs'` .
   b. Escolha Próximo.
4. Insira as ações da regra.
   a. No menu Ação 1, escolha CloudWatchregistros.
   b. Escolha o nome do grupo de logs que você criou. Em seguida, escolha o grupo de logs criado.
   c. Selecione Usar modo em lote.
   d. Especifique o perfil do IAM para a regra.

      Se você tiver um perfil do IAM para a regra, faça o seguinte.

      1. No menu do perfil do IAM, escolha seu perfil do IAM.

Se você não tiver um perfil do IAM para a regra, faça o seguinte.

1. Escolha Criar nova função.
2. Em Nome do perfil, insira um nome exclusivo e escolha Criar.
3. Confirme se o nome do perfil do IAM está correto no campo do perfil do IAM.

e. Escolha Próximo.

5. Revisar a configuração do modelo.

a. Revise as configurações do modelo de Tarefa para verificar se estão corretas.

b. Depois de concluir, escolha a opção Criar.

AWS CLI

Para criar uma função do IAM e uma regra de tópico usando o AWS CLI

1. Crie uma função do IAM que conceda direitos à AWS IoT regra.

a. Crie uma política do IAM.

Execute o seguinte comando para criar uma política do IAM. Certifique-se de atualizar o valor do parâmetro `policy-name`. Para obter mais informações, consulte `create-policy`na Referência de Comandos AWS CLI v2.

> **Note**
>
> Se você estiver usando um sistema operacional Microsoft Windows, talvez seja necessário substituir o marcador de fim de linha (\) por uma marca (`) ou outro caractere.

```
aws iam create-policy \
    --policy-name uploadLogsPolicy \
    --policy-document \
'{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
```

```
        "Action": [
            "iot:CreateTopicRule",
            "iot:Publish",
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents",
            "logs:GetLogEvents"
        ],
        "Resource": "*"
    }
}'
```

b. Copie o ARN da política de sua saída em um editor de texto.

Exemplo de saída:

```
{
    "Policy": {
        "PolicyName": "uploadLogsPolicy",
        "PermissionsBoundaryUsageCount": 0,
        "CreateDate": "2023-01-23T18:30:10Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "AAABBBCCCDDDEEEFFFGGG",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::111122223333:policy/uploadLogsPolicy",
        "UpdateDate": "2023-01-23T18:30:10Z"
    }
}
```

c. Criar uma política e um perfil do IAM.

Execute o seguinte comando para criar uma política do IAM. Certifique-se de atualizar o valor do parâmetro `role-name`. Para obter mais informações, consulte create-rolena Referência de Comandos AWS CLI v2.

```
aws iam create-role \
--role-name uploadLogsRole \
--assume-role-policy-document \
'{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'
```

d. Anexe a política do IAM à função.

Execute o seguinte comando para criar uma política do IAM. Certifique-se de atualizar os valores dos parâmetros `role-name` e `policy-arn`. Para obter mais informações, consulte `attach-role-policy`na Referência de Comandos AWS CLI v2.

```
aws iam attach-role-policy \
--role-name uploadLogsRole \
--policy-arn arn:aws:iam::111122223333:policy/uploadLogsPolicy
```

e. Analise a função.

Para confirmar se o perfil do IAM foi criado corretamente, execute o comando a seguir. Certifique-se de atualizar o valor do parâmetro `role-name`. Para obter mais informações, consulte `get-role`na Referência de Comandos AWS CLI v2.

```
aws iam get-role --role-name uploadLogsRole
```

Exemplo de saída:

```
{
    "Role": {
        "Path": "/",
        "RoleName": "uploadLogsRole",
        "RoleId": "AAABBBCCCDDDEEEFFFGGG",
        "Arn": "arn:aws:iam::111122223333:role/uploadLogsRole",
        "CreateDate": "2023-01-23T19:17:15+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
```

```
                    "Sid": "Statement1",
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "iot.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Description": "",
        "MaxSessionDuration": 3600,
        "RoleLastUsed": {}
    }
}
```

2. Crie uma regra de AWS IoT tópico no AWS CLI.

   a. Para criar uma regra de AWS IoT tópico, execute o comando a seguir. Certifique-se de atualizar os valores da `sql` instrução`--rule-name`,`description`,`roleARN` , e `logGroupName` do parâmetro. Para obter mais informações, consulte create-topic-rulena Referência de Comandos AWS CLI v2.

```
aws iot create-topic-rule \
--rule-name uploadLogsRule \
--topic-rule-payload \
 '{
  "sql":"SELECT * FROM 'rules/things/thing_name/logs'",
  "description":"Upload logs test rule",
  "ruleDisabled":false,
  "awsIotSqlVersion":"2016-03-23",
  "actions":[
   {"cloudwatchLogs":
    {"roleArn":"arn:aws:iam::111122223333:role/uploadLogsRole",
    "logGroupName":"uploadLogsGroup",
    "batchMode":true}
    }
   ]
}'
```

   b. Para confirmar se a regra foi criada corretamente, execute o comando a seguir. Certifique-se de atualizar o valor do parâmetro `role-name`. Para obter mais informações, consulte get-topic-rulena Referência de Comandos AWS CLI v2.

```
aws iot get-topic-rule --rule-name uploadLogsRule
```

Exemplo de saída:

```
{
    "ruleArn": "arn:aws:iot:us-east-1:111122223333:rule/uploadLogsRule",
    "rule": {
        "ruleName": "uploadLogsRule",
        "sql": "SELECT * FROM rules/things/thing_name/logs",
        "description": "Upload logs test rule",
        "createdAt": "2023-01-24T16:28:15+00:00",
        "actions": [
            {
                "cloudwatchLogs": {
                    "roleArn": "arn:aws:iam::111122223333:role/
uploadLogsRole",
                    "logGroupName": "uploadLogsGroup",
                    "batchMode": true
                }
            }
        ],
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23"
    }
}
```

Enviando registros do lado do dispositivo para AWS IoT

Para enviar registros do lado do dispositivo para AWS IoT

1. Para enviar registros históricos para AWS IoT, comunique-se com seus dispositivos para garantir o seguinte.

   - As informações do log são enviadas para o namespace de tópico correto, conforme especificado na seção Pré-requisitos deste procedimento.

     Por exemplo, `$aws/rules/things/thing_name/logs`.

- A carga útil da mensagem MQTT está formatada corretamente. Para obter mais informações sobre o tópico MQTT e a convenção de nomenclatura recomendada, consulte a Tópicos do MQTT seção em. Faça upload de registros do lado do dispositivo para a Amazon CloudWatch

2. Confirme se as mensagens MQTT foram recebidas no cliente AWS IoT MQTT.

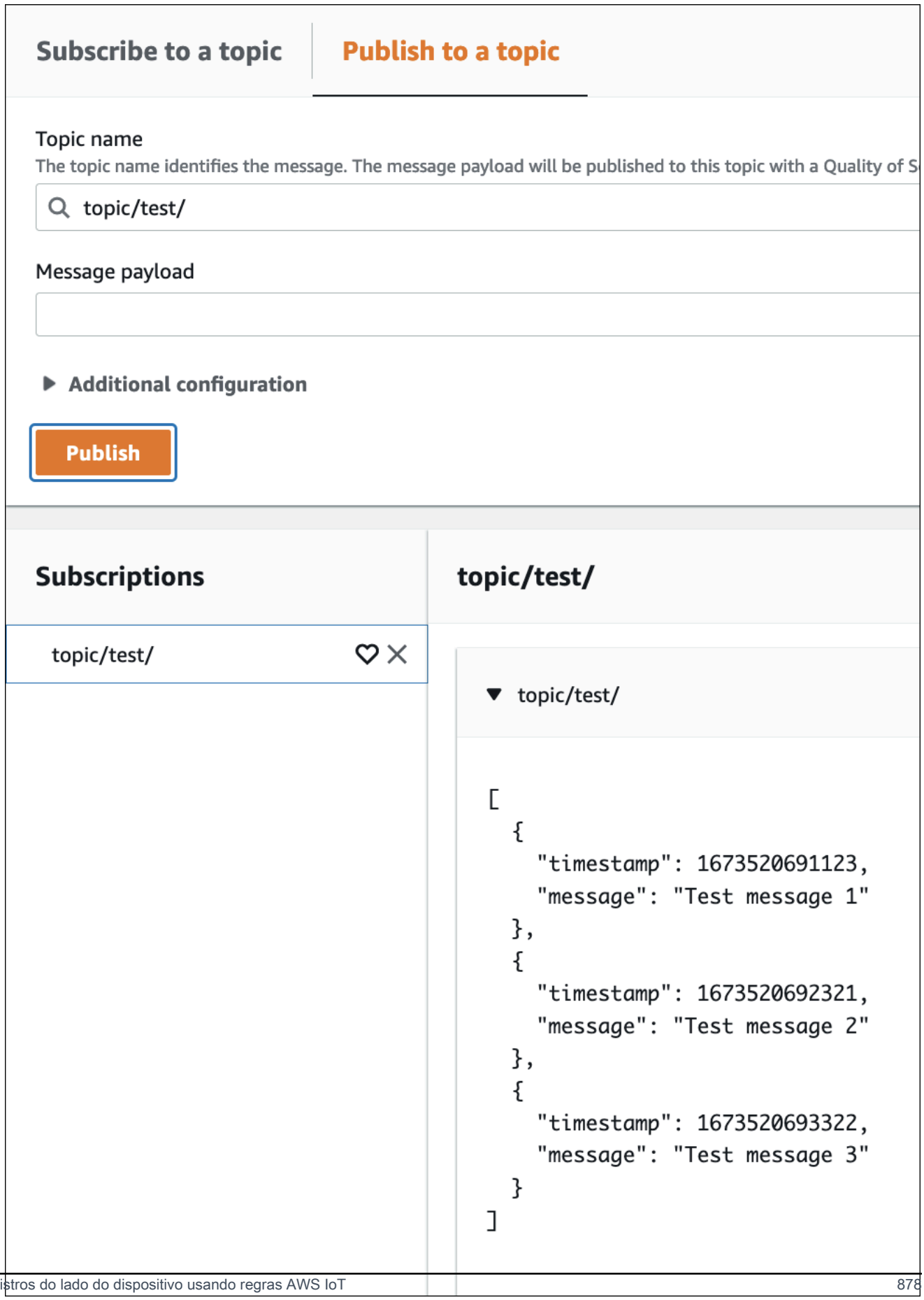
   a. Abra o Console de gerenciamento da AWS e navegue até AWS IoT.

   b. Para visualizar o cliente de teste MQTT, na barra de navegação, escolha Testar, cliente de teste MQTT.

   c. Em Inscrever-se em um tópico, Filtro de tópicos, insira o namespace do tópico.

   d. Escolha Assinar.

      As mensagens do MQTT aparecem na tabela Assinaturas e Tópicos, conforme mostrado a seguir. Essas mensagens podem levar até cinco minutos para serem exibidas.


Subscribe to a topic
Publish to a topic
Topic name
The topic name identifies the message. The message payload will be published to this topic with a Quality of S
topic/test/
Message payload
Additional configuration
Publish
Subscriptions
topic/test/
topic/test/
topic/test/
[
{
"timestamp": 1673520691123,
"message": "Test message 1"
},
{
"timestamp": 1673520692321,
"message": "Test message 2"
},
{
"timestamp": 1673520693322,
"message": "Test message 3"
}
]

Visualizando os dados do log

Para revisar seus registros de registro em CloudWatch Registros

1. Abra o Console de gerenciamento da AWS e navegue até CloudWatch.
2. Na barra de navegação, escolha Logs, Logs Insights.
3. No menu Selecionar grupo (s) de log, escolha o grupo de log que você especificou na AWS IoT regra.
4. Na página Insights de logs, escolha Executar consulta.

Registrando chamadas de AWS IoT API usando AWS CloudTrail

AWS IoT é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou AWS serviço em AWS IoT. CloudTrail captura todas as chamadas de API para eventos AWS IoT as, incluindo chamadas do AWS IoT console e de chamadas de código para o. AWS IoT APIs Se você criar uma trilha, poderá habilitar a entrega contínua de CloudTrail eventos para um bucket do Amazon S3, incluindo eventos para. AWS IoT Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no CloudTrail console no Histórico de eventos. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita AWS IoT, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e outros detalhes.

Para saber mais sobre isso CloudTrail, consulte o Guia AWS CloudTrail do usuário.

AWS IoT informações em CloudTrail

CloudTrail é ativado no seu Conta da AWS quando você cria a conta. Quando a atividade ocorre em AWS IoT, essa atividade é registrada em um CloudTrail evento junto com outros eventos AWS de serviço no histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes no seu Conta da AWS. Para obter mais informações, consulte Visualização de eventos com histórico de CloudTrail eventos.

Para obter um registro contínuo de eventos na sua Conta da AWS, incluindo eventos para o AWS IoT, crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, a trilha se aplica a todos os Região da AWS s. A trilha registra eventos de todos os Região da AWS s na AWS partição e entrega os arquivos de log ao bucket do Amazon S3 que você especificar. Você pode configurar outros AWS

serviços para analisar e agir com base nos dados de eventos coletados nos CloudTrail registros. Para obter mais informações, consulte:

- Visão geral da criação de uma trilha
- CloudTrail Serviços e integrações compatíveis
- Configurando notificações do Amazon SNS para CloudTrail
- Recebendo arquivos de CloudTrail log de várias regiões e recebendo arquivos de CloudTrail log de várias contas

> **Note**
>
> AWS IoT as ações do plano de dados (lado do dispositivo) não são registradas por CloudTrail. Use CloudWatch para monitorar essas ações.

De um modo geral, as ações do plano de AWS IoT controle que fazem alterações são registradas por CloudTrail. Chamadas como CreateThingCreateKeysAndCertificate, e UpdateCertificatedeixam CloudTrail entradas, enquanto chamadas como ListThingse ListTopicRulesnão.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário-raiz ou usuário do IAM.
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.
- Se a solicitação foi feita por outro AWS serviço.

Para obter mais informações, consulte Elemento userIdentity do CloudTrail .

AWS IoT as ações são documentadas na Referência AWS IoT da API. AWS IoT Wireless as ações são documentadas na Referência AWS IoT Wireless da API.

Entendendo as entradas do arquivo de AWS IoT log

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log para um bucket do Amazon S3 que você especificar. CloudTrail os arquivos de log contêm uma ou mais

entradas de log. Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros da solicitação e assim por diante. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das chamadas públicas de API, portanto, eles não aparecem em nenhuma ordem específica.

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a `AttachPolicy` ação.


```
{
    "timestamp":"1460159496",
    "AdditionalEventData":"",
    "Annotation":"",
    "ApiVersion":"",
    "ErrorCode":"",
    "ErrorMessage":"",
    "EventID":"8bff4fed-c229-4d2d-8264-4ab28a487505",
    "EventName":"AttachPolicy",
    "EventTime":"2016-04-08T23:51:36Z",
    "EventType":"AwsApiCall",
    "ReadOnly":"",
    "RecipientAccountList":"",
    "RequestID":"d4875df2-fde4-11e5-b829-23bf9b56cbcd",
    "RequestParamters":{
        "principal":"arn:aws:iot:us-
east-1:123456789012:cert/528ce36e8047f6a75ee51ab7beddb4eb268ad41d2ea881a10b67e8e76924d894",
        "policyName":"ExamplePolicyForIoT"
    },
    "Resources":"",
    "ResponseElements":"",
    "SourceIpAddress":"52.90.213.26",
    "UserAgent":"aws-internal/3",
    "UserIdentity":{
        "type":"AssumedRole",
        "principalId":"AKIAI44QH8DHBEXAMPLE",
        "arn":"arn:aws:sts::12345678912:assumed-role/iotmonitor-us-east-1-beta-
InstanceRole-1C5T1YCYMHPYT/i-35d0a4b6",
        "accountId":"222222222222",
        "accessKeyId":"access-key-id",
        "sessionContext":{
            "attributes":{
                "mfaAuthenticated":"false",
                "creationDate":"Fri Apr 08 23:51:10 UTC 2016"
```

```
            },
            "sessionIssuer":{
                "type":"Role",
                "principalId":"AKIAI44QH8DHBEXAMPLE",
                "arn":"arn:aws:iam::123456789012:role/executionServiceEC2Role/
iotmonitor-us-east-1-beta-InstanceRole-1C5T1YCYMHPYT",
                "accountId":"222222222222",
                "userName":"iotmonitor-us-east-1-InstanceRole-1C5T1YCYMHPYT"
            }
        },
        "invokedBy":{
            "serviceAccountId":"111111111111"
        }
    },
    "VpcEndpointId":""
}
```

Regras para AWS IoT

As regras permitem que seus dispositivos interajam com Serviços da AWS. As regras são analisadas e as ações são realizadas com base no fluxo de tópicos do MQTT. Você pode usar as regras para dar suporte a tarefas como estas:

- Aumentar ou filtrar os dados recebidos de um dispositivo.
- Gravar os dados recebidos de um dispositivo em um banco de dados do Amazon DynamoDB.
- Salvar um arquivo no Amazon S3.
- Enviar uma notificação por push para todos os usuários usando o Amazon SNS.
- Publicar dados em uma fila do Amazon SQS.
- Invocar uma função do Lambda para extrair dados.
- Processar mensagens de um grande número de dispositivos usando o Amazon Kinesis.
- Envie dados para o Amazon OpenSearch Service.
- Capture uma CloudWatch métrica.
- Troque um CloudWatch alarme.
- Envie os dados de uma mensagem MQTT para a Amazon SageMaker AI para fazer previsões com base em um modelo de aprendizado de máquina (ML).
- Enviar uma mensagem para o fluxo de entrada da Salesforce IoT.
- Iniciar a execução de uma máquina de estado do Step Functions. Iniciar o processo de uma máquina de estados do Step Functions.
- Envie dados da mensagem para uma AWS IoT Events entrada.
- Enviar dados de mensagem para uma propriedade de ativo em AWS IoT SiteWise.
- Enviar dados de mensagens para um serviço ou aplicativo da web.

Suas regras podem usar mensagens MQTT que passam pelo publish/subscribe protocolo suportado pelothe section called “Protocolos de comunicação do dispositivo”. Você também pode usar o recurso Basic Ingest para enviar com segurança os dados do dispositivo para os Serviços da AWS listados anteriormente, sem incorrer em custos de mensagens. O recurso Basic Ingest otimiza o fluxo de dados removendo o agente de publish/subscribe mensagens do caminho de ingestão. Isso o torna econômico e, ao mesmo tempo, mantém os recursos de segurança e processamento de dados do AWS IoT.

Antes de AWS IoT poder realizar essas ações, você deve conceder permissão para que ele acesse seus AWS recursos em seu nome. Quando as ações são executadas, você incorre nas cobranças padrão do Serviços da AWS que você usa.

Conteúdo

- Conceder a uma AWS IoT regra o acesso que ela exige
- Transmitir as permissões de função
- Criação de uma AWS IoT regra
- Gerenciando uma AWS IoT regra
- AWS IoT ações de regras
- Solucionar problemas de uma regra
- Acessando recursos entre contas usando regras AWS IoT
- Tratamento de erros (ação de erro)
- Reduzir custos do sistema de mensagens com Ingestão básica
- AWS IoT Referência SQL

Conceder a uma AWS IoT regra o acesso que ela exige

Use as funções do IAM para controlar os AWS recursos aos quais cada regra tem acesso. Antes de criar uma regra, você deve criar uma função do IAM com uma política que permita o acesso aos AWS recursos necessários. AWS IoT assume essa função ao implementar uma regra.

Conclua as etapas a seguir para criar a função e a AWS IoT política do IAM que concedem a uma AWS IoT regra o acesso necessário (AWS CLI).

1. Salve o seguinte documento de política de confiança, que concede AWS IoT permissão para assumir a função, em um arquivo chamado`iot-role-trust.json`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
        },
```

```
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
            },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:iot:us-east-1:123456789012:rule/
rulename"
            }
        }
        }
    ]
}
```

Use o comando criar-função para criar um perfil do IAM especificando o arquivo `iot-role-trust.json`:

```
aws iam create-role --role-name my-iot-role --assume-role-policy-document
 file://iot-role-trust.json
```

A saída desse comando é semelhante à seguinte:

```
{
 "Role": {
  "AssumeRolePolicyDocument": "url-encoded-json",
  "RoleId": "AKIAIOSFODNN7EXAMPLE",
  "CreateDate": "2015-09-30T18:43:32.821Z",
  "RoleName": "my-iot-role",
  "Path": "/",
  "Arn": "arn:aws:iam::123456789012:role/my-iot-role"
 }
}
```

2. Salve o seguinte JSON em um arquivo chamado `my-iot-policy.json`.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
```

```
    "Action": "dynamodb:*",
    "Resource": "*"
   }
  ]
}
```

Esse JSON é um exemplo de documento de política que concede acesso de AWS IoT administrador ao DynamoDB.

Use o comando create-policy para conceder AWS IoT acesso aos seus AWS recursos ao assumir a função, transmitindo o arquivo: `my-iot-policy.json`

```
aws iam create-policy --policy-name my-iot-policy --policy-document file://my-iot-policy.json
```

Para obter mais informações sobre como conceder acesso às Serviços da AWS políticas do AWS IoT, consulteCriação de uma AWS IoT regra.

A saída do comando criar-política contém o ARN da política. Anexar uma política a uma função.

```
{
 "Policy": {
  "PolicyName": "my-iot-policy",
  "CreateDate": "2015-09-30T19:31:18.620Z",
  "AttachmentCount": 0,
  "IsAttachable": true,
  "PolicyId": "ZXR6A36LTYANPAI7NJ5UV",
  "DefaultVersionId": "v1",
  "Path": "/",
  "Arn": "arn:aws:iam::123456789012:policy/my-iot-policy",
  "UpdateDate": "2015-09-30T19:31:18.620Z"
 }
}
```

3. Use o attach-role-policycomando para anexar sua política à sua função:

```
aws iam attach-role-policy --role-name my-iot-role --policy-arn
 "arn:aws:iam::123456789012:policy/my-iot-policy"
```

Revogar o acesso ao mecanismo de regras

Para revogar imediatamente o acesso ao mecanismo de regras, faça o seguinte:

1. Remova iot.amazonaws.com da política de confiança.
2. Siga as etapas para revogar as sessões do perfil de IoT.

Transmitir as permissões de função

Parte de uma definição de regra é um perfil do IAM que concede permissão para acessar os recursos especificados na ação da regra. O mecanismo de regras assume essa função quando a ação da regra é invocada. A função deve ser definida da mesma forma Conta da AWS que a regra.

Ao criar ou substituir uma regra, você está, na verdade, transferindo uma função para o mecanismo de regras. A permissão `iam:PassRole` é necessária para executar essa operação. Para garantir que você tenha essa permissão, crie uma política que conceda a `iam:PassRole` permissão e anexe-a ao seu usuário do IAM. A política a seguir mostra como conceder `iam:PassRole` a permissão para uma função.

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "Stmt1",
   "Effect": "Allow",
   "Action": [
    "iam:PassRole"
   ],
   "Resource": [
    "arn:aws:iam::123456789012:role/myRole"
   ]
  }
 ]
}
```

Nesse exemplo de política, a `iam:PassRole` permissão é concedida para a função `myRole`. A função é especificada usando o ARN da função. Anexe essa política ao usuário do IAM ou a função

a que o usuário pertence. Para obter mais informações, consulte Como trabalhar com políticas gerenciadas.

> **Note**
>
> As funções Lambda usam política baseada em recursos, em que a política é vinculada diretamente à própria função do Lambda. Ao criar uma regra que invoca uma função do Lambda, você não transmite uma função, assim, o usuário que está criando a regra não precisa da permissão do `iam:PassRole`. Para obter mais informações sobre a autorização de função do Lambda, consulte Como conceder permissões usando uma política de recurso.

Criação de uma AWS IoT regra

Você pode criar AWS IoT regras para rotear dados de suas coisas conectadas para interagir com outros AWS serviços. Uma AWS IoT regra consiste nos seguintes componentes:

Componentes de uma regra

Componente	Description	Obrigatório/opcional
Nome da regra	O nome da regra. Observe que não recomendamos o uso de informações de identificação do usuário em nomes de regras.	Obrigatório.
Descrição da regra	Um texto de descrição da regra. Observe que não recomendamos o uso de informações de identificação pessoal nas descrições de suas regras.	Opcional.
Declaração do SQL	Uma sintaxe SQL simplificada para filtrar as mensagens recebidas em um tópico do MQTT e enviar os dados para outro lugar. Para obter mais informações, consulte AWS IoT Referência SQL.	Obrigatório.
Versão do SQL	A versão do mecanismo de regras do SQL a ser usado ao avaliar a regra. Embora essa propriedade seja opcional, é recomendável especificar a versão do SQL. O AWS IoT Core console define	Obrigatório.

Componente	Description	Obrigatório/opcional
	essa `2016-03-23` propriedade como padrão. Se essa propriedade não for definida, como em um AWS CLI comando ou CloudFormation modelo, `2015-10-08` será usada. Para obter mais informações, consulte Versões do SQL.	
Uma ou mais ações	As ações são AWS IoT executadas ao promulgar a regra. Por exemplo, você pode inserir dados em uma tabela do DynamoDB, gravar dados em um bucket do Amazon S3, publicar em um tópico do Amazon SNS ou invocar uma função do Lambda.	Obrigatório.
Uma ação de erro	A ação é AWS IoT executada quando não consegue realizar a ação de uma regra.	Opcional.

Antes de criar uma AWS IoT regra, você deve criar uma função do IAM com uma política que permita o acesso aos AWS recursos necessários. AWS IoT assume essa função ao implementar uma regra. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso necessário e Transferir permissões de função.

Ao criar uma regra, veja a quantidade de dados que você está publicando nos tópicos. Se você criar regras que incluam um padrão de tópico curinga, elas poderão corresponder a uma grande porcentagem de suas mensagens. Se for esse o caso, talvez seja necessário aumentar a capacidade dos AWS recursos usados pelas ações de destino. Recomendamos evitar padrões de tópicos curinga nas regras de republicação a fim de evitar o processamento duplicado e reduzir custos.

> **Note**
>
> A criação e atualização de regras são ações no nível de administrador. Qualquer usuário com permissão para criar ou atualizar regras pode acessar os dados processados pelas regras.

Criar uma regra (console)

Para criar uma regra (Console de gerenciamento da AWS)

Use o comando Console de gerenciamento da AWS para criar uma regra:

1. Abra o console do AWS IoT.
2. No painel de navegação à esquerda, escolha Roteamento de mensagens na seção Gerenciar. Em seguida, escolha Regras.
3. Na página Regras, selecione Criar uma regra.
4. Na página Especificar propriedades da regra, insira um nome para a regra. Descrição da regra e Tags são opcionais. Escolha Próximo.
5. Na página Configurar instrução SQL, escolha uma versão SQL e insira uma instrução SQL. Um exemplo de instrução SQL pode ser `SELECT temperature FROM 'iot/topic' WHERE temperature > 50`. Para obter mais informações, consulte Versões de SQL e Referência de SQL de AWS IoT.
6. Na página Anexar ações de regras, adicione ações de regra para rotear dados para outros AWS serviços.

    1. Em Ações de regra, selecione uma ação de regra na lista suspensa. Por exemplo, você pode escolher o Kinesis Stream. Para obter mais informações sobre as ações de regra, consulte Ações de regra de AWS IoT.
    2. Dependendo da ação de regra escolhida, insira os detalhes de configuração relacionados. Por exemplo, se você escolher o Kinesis Stream, precisará escolher ou criar um recurso de fluxo de dados e, opcionalmente, inserir detalhes de configuração, como Chave de partição, que é usada para agrupar dados por fragmento em um steam.
    3. Na função do IAM, escolha ou crie uma função para conceder AWS IoT acesso ao seu endpoint. Observe que isso AWS IoT criará automaticamente uma política com o prefixo de `aws-iot-rule` abaixo da sua função do IAM selecionada. Você pode escolher Exibir para visualizar o perfil do IAM e a política no console do IAM. A ação de erro é opcional. Você pode encontrar mais informações em Tratamento de erros (ação de erro). Para obter mais informações sobre como criar um perfil do IAM para sua regra, consulte Conceder a uma regra o acesso necessário. Escolha Próximo.
7. Na página Revisar e criar, revise toda a configuração e faça edições, se necessário. Escolha Criar.

Depois de criar uma regra com sucesso, você verá a regra na página Regras. Você pode selecionar uma regra para abrir a página Detalhes, na qual pode visualizar, editar, desativar e excluir uma regra.

Criar uma regra (CLI)

Para criar uma regra (AWS CLI)

Use o comando create-topic-rule para criar uma regra:

```
aws iot create-topic-rule --rule-name myrule --topic-rule-payload file://myrule.json
```

Veja a seguir um exemplo de arquivo de carga útil com uma regra que insere todas as mensagens enviadas para o tópico `iot/test` em uma tabela do DynamoDB especificada. A instrução SQL filtra as mensagens e o ARN da função concede AWS IoT permissão para gravar na tabela do DynamoDB.

```
{
 "sql": "SELECT * FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "dynamoDB": {
    "tableName": "my-dynamodb-table",
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",
    "hashKeyField": "topic",
    "hashKeyValue": "${topic(2)}",
    "rangeKeyField": "timestamp",
    "rangeKeyValue": "${timestamp()}"
   }
  }
 ]
}
```

Veja a seguir um exemplo de arquivo de carga útil com uma regra que insere todas as mensagens enviadas para o tópico `iot/test` no bucket do S3 especificado. A instrução SQL filtra as mensagens, e o ARN da função concede AWS IoT permissão para gravar no bucket do Amazon S3.

```
{
 "awsIotSqlVersion": "2016-03-23",
 "sql": "SELECT * FROM 'iot/test'",
```

```
 "ruleDisabled": false,
 "actions": [
  {
   "s3": {
    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",
    "bucketName": "amzn-s3-demo-bucket",
    "key": "myS3Key"
   }
  }
 ]
}
```

Veja a seguir um exemplo de arquivo de carga útil com uma regra que envia dados para o Amazon OpenSearch Service:

```
{
 "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "OpenSearch": {
    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es",
    "endpoint": "https://my-endpoint",
    "index": "my-index",
    "type": "my-type",
    "id": "${newuuid()}"
   }
  }
 ]
}
```

Veja a seguir um exemplo de arquivo de carga útil com uma regra que invoca uma função do Lambda:

```
{
 "sql": "expression",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "lambda": {
    "functionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-lambda-function"
```

```
   }
  }
 ]
}
```

Veja a seguir um exemplo de arquivo de carga útil com uma regra que publica em um tópico do SNS:

```
{
 "sql": "expression",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "sns": {
    "targetArn": "arn:aws:sns:us-west-2:123456789012:my-sns-topic",
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
   }
  }
 ]
}
```

Veja a seguir um exemplo de arquivo de carga útil com uma regra que publica novamente em um tópico diferente do MQTT:

```
{
 "sql": "expression",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "republish": {
    "topic": "my-mqtt-topic",
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
   }
  }
 ]
}
```

A seguir está um exemplo de arquivo de carga útil com uma regra que envia dados para um fluxo do Amazon Data Firehose:

```
{
```

```
 "sql": "SELECT * FROM 'my-topic'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "firehose": {
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",
    "deliveryStreamName": "my-stream-name"
   }
  }
 ]
}
```

Veja a seguir um exemplo de arquivo de carga útil com uma regra que usa a `machinelearning_predict` função Amazon SageMaker AI para republicar em um tópico se os dados na carga do MQTT forem classificados como 1.

```
{
 "sql": "SELECT * FROM 'iot/test' where machinelearning_predict('my-model',
 'arn:aws:iam::123456789012:role/my-iot-aml-role', *).predictedLabel=1",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "republish": {
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",
    "topic": "my-mqtt-topic"
   }
  }
 ]
}
```

O seguinte é um exemplo de arquivo de carga útil com uma regra que publica mensagens em um fluxo de entrada do Salesforce IoT Cloud.

```
{
 "sql": "expression",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "salesforce": {
```

```
    "token": "ABCDEFGHI123456789abcdefghi123456789",
    "url": "https://ingestion-cluster-id.my-env.sfdcnow.com/streams/stream-id/
connection-id/my-event"
   }
  }
 ]
}
```

Veja a seguir um exemplo de arquivo de carga útil com uma regra que inicie a execução de uma máquina de estados Step Functions.

```
{
 "sql": "expression",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "stepFunctions": {
    "stateMachineName": "myCoolStateMachine",
    "executionNamePrefix": "coolRunning",
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
   }
  }
 ]
}
```

Gerenciando uma AWS IoT regra

Você pode usar as ações a seguir para gerenciar suas AWS IoT regras.

Neste tópico:

- Marcar uma regra
- Visualizar uma regra
- Como excluir uma regra

Marcar uma regra

Para adicionar outra camada de especificidade às suas regras novas ou existentes, você pode aplicar a marcação. A marcação aproveita os pares de valores-chave em suas regras para fornecer

maior controle sobre como e onde suas regras são aplicadas aos seus AWS IoT recursos e serviços. Por exemplo, você pode limitar o escopo da sua regra para aplicá-la somente em seu ambiente beta para testes de pré-lançamento (`Key=environment, Value=beta`) ou capturar todas as mensagens enviadas ao tópico `iot/test` de um endpoint específico e armazená-las em um bucket do Amazon S3.

Exemplo de política do IAM

Para ver um exemplo que mostra como conceder permissões de marcação para uma regra, considere um usuário que executa o comando a seguir para criar uma regra e marcá-la para aplicá-la somente ao ambiente beta.

No exemplo, substitua:

- *`MyTopicRuleName`*com o nome da regra.
- *`myrule.json`*com o nome do documento de política.

```
aws iot create-topic-rule
    --rule-name MyTopicRuleName
    --topic-rule-payload file://myrule.json
    --tags "environment=beta"
```

Para este exemplo, você deve usar a seguinte política do IAM:

```
{
    "Version":"2012-10-17",
    "Statement":
    {
        "Action": [ "iot:CreateTopicRule", "iot:TagResource" ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:rule/MyTopicRuleName"
        ]
    }
}
```

O exemplo acima mostra uma regra recém-criada chamada `MyTopicRuleName` que se aplica somente ao seu ambiente beta. A `iot:TagResource` declaração de política com a indicação

`MyTopicRuleName` específica permite a marcação ao criar ou atualizar `MyTopicRuleName`. O parâmetro `--tags "environment=beta"` usado ao criar a regra limita o escopo de `MyTopicRuleName` somente ao seu ambiente beta. Se você remover o parâmetro `--tags "environment=beta"`, então `MyTopicRuleName` será aplicado a todos os ambientes.

Para obter mais informações sobre a criação de perfis do IAM e políticas específicas para uma regra, AWS IoT consulte Conceder a uma AWS IoT regra o acesso que ela exige

Para obter informações gerais sobre a marcação de recursos, consulte Marcando seus recursos AWS IoT.

Visualizar uma regra

Use o list-topic-rulescomando para listar suas regras:

```
aws iot list-topic-rules
```

Use o get-topic-rulecomando para obter informações sobre uma regra:

```
aws iot get-topic-rule --rule-name myrule
```

Como excluir uma regra

Quando não precisar mais de uma regra, você poderá excluí-la.

Para excluir uma regra (AWS CLI)

Use o delete-topic-rulecomando para excluir uma regra:

```
aws iot delete-topic-rule --rule-name myrule
```

AWS IoT ações de regras

AWS IoT as ações de regra especificam o que fazer quando uma regra é invocada. Você pode definir ações para enviar dados para um banco de dados do Amazon DynamoDB, enviar dados para o Amazon Kinesis Data Streams, AWS Lambda invocar uma função e assim por diante. AWS IoT suporta as seguintes ações em Regiões da AWS que o serviço da ação está disponível.

Ação da regra	Descrição	Nome na API
Apache Kafka	Envia uma mensagem para um cluster Apache Kafka.	`kafka`
CloudWatch alarmes	Altera o estado de um CloudWatch alarme da Amazon.	`cloudwatchAlarm`
CloudWatch Registros	Envia uma mensagem para a Amazon CloudWatch Logs.	`cloudwatchLogs`
CloudWatch métricas	Envia uma mensagem para uma CloudWatch métrica.	`cloudwatchMetric`
DynamoDB	Envia uma mensagem para uma tabela do DynamoDB.	`dynamoDB`
Dínamo DBv2	Envia dados de mensagens para várias colunas em uma tabela do DynamoDB.	`dynamoDBv2`
Elasticsearch	Envia uma mensagem para um OpenSearch endpoint.	`OpenSearch`
HTTP	Publica uma mensagem em um endpoint HTTPS.	`http`
AWS IoT Events	Envia uma mensagem para uma AWS IoT Events entrada.	`iotEvents`
AWS IoT SiteWise	Envia dados da mensagem para as propriedades do AWS IoT SiteWise ativo.	`iotSiteWise`
Firehose	Envia uma mensagem para o fluxo de entrega do Firehose.	`firehose`

Ação da regra	Descrição	Nome na API
Kinesis Data Streams	Envia uma mensagem para o fluxo de dados do Kinesis.	`kinesis`
Lambda	Invoca uma função do Lambda com dados da mensagem como entrada.	`lambda`
Local	Envia dados de localizaç ão para o Amazon Location Service.	`location`
OpenSearch	Envia uma mensagem para um endpoint OpenSearch do Amazon Service.	`OpenSearch`
Nova publicação	Publica novamente uma mensagem em outro tópico do MQTT.	`republish`
S3	Armazena uma mensagem em um bucket do Amazon Simple Storage Service (Amazon S3).	`s3`
IoT do Salesforce	Envia uma mensagem para o fluxo de entrada da Salesforce IoT.	`salesforce`
SNS	Publica uma mensagem como uma notificação push do Amazon Simple Notification Service (Amazon SNS)	`sns`
SQS	Envia uma mensagem para uma fila do Amazon Simple Queue Service (Amazon SQS).	`sqs`

Ação da regra	Descrição	Nome na API
Step Functions	Inicia uma máquina de AWS Step Functions estado.	`stepFunctions`
the section called “Timestream”	Envia uma mensagem para uma tabela de banco de dados do Amazon Timestream.	`timestream`

Observações

- Defina a regra da mesma forma Região da AWS que o recurso de outro serviço para que a ação da regra possa interagir com esse recurso.
- O mecanismo de AWS IoT regras pode fazer várias tentativas para realizar uma ação se ocorrerem erros intermitentes. Se todas as tentativas falharem, a mensagem será descartada e o erro estará disponível nos seus CloudWatch registros. Você pode especificar uma ação de erro para cada regra que é invocada depois de ocorrer uma falha. Para obter mais informações, consulte Tratamento de erros (ação de erro).
- Algumas ações de regra acionam ações em serviços que se integram ao AWS Key Management Service (AWS KMS) para oferecer suporte à criptografia de dados em repouso. Se você usar uma chave gerenciada pelo cliente AWS KMS key (chave KMS) para criptografar dados em repouso, o serviço deverá ter permissão para usar a chave KMS em nome do chamador. Para saber como gerenciar permissões para sua chave KMS gerenciada pelo cliente, consulte os tópicos de criptografia de dados no guia de serviço apropriado. Para obter mais informações sobre chaves KMS gerenciadas pelo cliente, consulte os AWS Key Management Service conceitos no AWS Key Management Service Guia do desenvolvedor.

  Você pode usar qualquer função ou modelo de substituição na instrução SQL de uma ação de erro, incluindo as funções externas: `aws_lambda()get_dynamodb()`, `get_registry_data()`, `get_thing_shadow()`, `get_secret()`, `machinelearning_predict()`, e. `decode()` Se uma ação de erro exigir a chamada de uma função externa, a invocação da ação de erro poderá resultar em cobrança adicional para a função externa.

Apache Kafka

A ação Apache Kafka (Kafka) envia mensagens diretamente ao Amazon Managed Streaming for Apache Kafka (Amazon MSK), clusters do Apache Kafka gerenciados por provedores de terceiros, como Confluent Cloud ou clusters autogerenciados do Apache Kafka. Com a ação da regra do Kafka, você pode rotear seus dados de IoT para clusters do Kafka. Isso permite criar pipelines de dados de alto desempenho para várias finalidades, como análise de streaming, integração de dados, visualização e aplicações comerciais essenciais.

> **Note**
>
> Este tópico pressupõe familiaridade com a plataforma Apache Kafka e conceitos relacionados. Para obter mais informações sobre o Apache Kafka, consulte Apache Kafka. Não há suporte para MSK com tecnologia sem servidor. Os clusters do MSK com tecnologia sem servidor só podem ser feitos por meio da autenticação do IAM, que não é compatível com a ação de regra do Apache Kafka no momento. Para obter mais informações sobre como configurar AWS IoT Core com o Confluent, consulte Aproveitando o Confluent e AWS resolvendo desafios de gerenciamento de dados e dispositivos de IoT.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT pode ser assumida para realizar as `ec2:DescribeSecurityGroups` operações `ec2:CreateNetworkInterface` `ec2:DescribeNetworkInterfaces` `ec2:CreateNetworkInterfacePermissionec2:DeleteNetworkInterface,ec2:DescribeSubn` e. Essa função cria e gerencia interfaces de rede elásticas para o Amazon Virtual Private Cloud para alcançar seu agente Kafka. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT Core a execução dessa ação de regra.

  Para obter mais informações sobre interfaces de rede, consulte Interfaces de rede elásticas no Guia do usuário do Amazon EC2.

  A política vinculada à função que você especificar deve ser semelhante à do exemplo a seguir.

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "ec2:CreateNetworkInterface",
            "ec2:DescribeNetworkInterfaces",
            "ec2:CreateNetworkInterfacePermission",
            "ec2:DeleteNetworkInterface",
            "ec2:DescribeSubnets",
            "ec2:DescribeVpcs",
            "ec2:DescribeVpcAttribute",
            "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

- Se você costuma AWS Secrets Manager armazenar as credenciais necessárias para se conectar ao seu corretor Kafka, deve criar uma função do IAM que AWS IoT Core possa ser assumida para realizar as operações e. `secretsmanager:GetSecretValue` `secretsmanager:DescribeSecret`

  A política vinculada à função que você especificar deve ser semelhante à do exemplo a seguir.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:kafka_client_truststore-*",
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:kafka_keytab-*"
```

```
            ]
        }
    ]
}
```

- Você pode executar seus clusters do Apache Kafka dentro do Amazon Virtual Private Cloud (Amazon VPC). Você deve criar um destino de Nuvem Privada Virtual (VPC) Apache Kafka e usar um gateway NAT em suas sub-redes para encaminhar mensagens para um cluster público do Kafka. AWS IoT O mecanismo de AWS IoT regras cria uma interface de rede em cada uma das sub-redes listadas no destino para rotear o tráfego diretamente para a VPC. Quando você faz o destino, o mecanismo de AWS IoT regras cria automaticamente uma ação de regra de VPC. Para obter mais informações sobre ações de regras de VPC, consulte Destinos da Nuvem Privada Virtual (VPC) Apache Kafka.
- Se você usar uma chave gerenciada pelo cliente AWS KMS key (chave KMS) para criptografar dados em repouso, o serviço deverá ter permissão para usar a chave KMS em nome do chamador. Para obter mais informações, consulte Criptografia do Amazon MSK no Guia do desenvolvedor do Amazon Managed Streaming for Apache Kafka.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

destinationArn

O Amazon Resource Name (ARN) do destino da Nuvem Privada Virtual (VPC) Apache Kafka. Para obter informações sobre como criar um destino, consulteDestinos da Nuvem Privada Virtual (VPC) Apache Kafka.

tópico

O tópico do Kafka para mensagens enviadas ao agente do Kafka.

Você pode substituir esse campo usando um modelo de substituição. Para obter mais informações, consulte the section called “Modelos de substituição”.

chave (opcional)

A chave de mensagem do Kafka.

Você pode substituir esse campo usando um modelo de substituição. Para obter mais informações, consulte the section called “Modelos de substituição”.

cabeçalhos (opcional)

A lista de cabeçalhos Kafka que você especifica. Cada cabeçalho é um par de chave-valor que você pode especificar ao criar uma ação do Kafka. Você pode usar esses cabeçalhos para rotear dados de clientes IoT para clusters Kafka downstream sem modificar a carga útil da mensagem.

Você pode substituir esse campo usando um modelo de substituição. Para entender como passar uma função de regra integrada como um modelo de substituição no cabeçalho do Kafka Action, consulte Exemplos. Para obter mais informações, consulte the section called “Modelos de substituição”.

> **Note**
>
> Não há compatibilidade com cabeçalhos em formato binário.

partição (opcional)

A partição de mensagens do Kafka.

Você pode substituir esse campo usando um modelo de substituição. Para obter mais informações, consulte the section called “Modelos de substituição”.

clientProperties

Um objeto que define as propriedades do cliente produtor Apache Kafka.

acks (opcional)

O número de confirmações que o produtor exige que o servidor tenha recebido antes de considerar uma solicitação concluída.

Se você especificar 0 como valor, o produtor não aguardará nenhuma confirmação do servidor. Se o servidor não receber a mensagem, o produtor não tentará enviá-la novamente.

Valores válidos: `-1`, `0`, `1`, `all`. O valor padrão é `1`.

bootstrap.servers

Uma lista de pares de host e porta (por exemplo,`host1:port1`, `host2:port2`) usados para estabelecer a conexão inicial com o cluster Kafka.

compression.type (opcional)

O tipo de compressão para todos os dados gerados pelo produtor.

Valores válidos: `none`, `gzip`, `snappy`, `lz4`, `zstd`. O valor padrão é `none`.

security.protocol

O protocolo de segurança usado para se conectar ao seu agente Kafka.

Valores válidos: `SSL`, `SASL_SSL`. O valor padrão é `SSL`.

key.serializer

Especifica como transformar os principais objetos que você fornece com o`ProducerRecord` em bytes.

Valor válido: `StringSerializer`.

value.serializador

Especifica como transformar os principais objetos que você fornece com o `ProducerRecord` em bytes.

Valor válido: `ByteBufferSerializer`.

ssl.truststore

O arquivo truststore no formato base64 ou a localização do arquivo truststore em AWS Secrets Manager. Esse valor não será necessário se o seu arquivo truststore for certificado pelas autoridades de certificação (CA) da Amazon.

Este campo oferece suporte a modelos de substituição. Se você usar o Secrets Manager para armazenar as credenciais necessárias para se conectar ao seu agente Kafka, poderá usar a função `get_secret` SQL para recuperar o valor desse campo. Para obter mais informações sobre modelos de substituição, consulte the section called “Modelos de substituição”. Para obter mais informações sobre a `get_secret` função SQL, consulte the section called “get_secret (secretId, secretType, chave, roleArn)”. Se o arquivo truststore estiver na forma de um arquivo, use o parâmetro `SecretBinary`. Se o arquivo truststore estiver na forma de uma string, use o parâmetro `SecretString`.

O tamanho máximo desse valor é 65 KB.

ssl.truststore.password

A senha do arquivo truststore. Esse valor é necessário somente se você tiver criado uma senha para o arquivo truststore.

ssl.keystore

O arquivo de armazenamento de chaves. Esse valor é necessário quando você especifica `SSL` como valor para `security.protocol`.

Este campo oferece suporte a modelos de substituição. Use o Secrets Manager para armazenar as credenciais necessárias para se conectar ao seu agente Kafka. Para recuperar o valor desse campo, use a `get_secret` função SQL. Para obter mais informações sobre modelos de substituição, consulte the section called “Modelos de substituição”. Para obter mais informações sobre a `get_secret` função SQL, consulte the section called “get_secret (secretId, secretType, chave, roleArn)”. Use o parâmetro `SecretBinary`.

ssl.keystore.password

A senha de armazenamento do arquivo keystore. Esse valor será exigido se um valor para `ssl.keystore` for especificado.

O valor desse campo pode ser texto simples. Este campo oferece suporte a modelos de substituição. Use o Secrets Manager para armazenar as credenciais necessárias para se conectar ao seu agente Kafka. Para recuperar o valor desse campo, use a `get_secret` função SQL. Para obter mais informações sobre modelos de substituição, consulte the section called “Modelos de substituição”. Para obter mais informações sobre a `get_secret` função SQL, consulte the section called “get_secret (secretId, secretType, chave, roleArn)”. Use o parâmetro `SecretString`.

ssl.key.password

A senha da chave privada em seu arquivo keystore.

Este campo oferece suporte a modelos de substituição. Use o Secrets Manager para armazenar as credenciais necessárias para se conectar ao seu agente Kafka. Para recuperar o valor desse campo, use a `get_secret` função SQL. Para obter mais informações sobre modelos de substituição, consulte the section called “Modelos de substituição”. Para obter mais informações sobre a `get_secret` função SQL, consulte the section called “get_secret (secretId, secretType, chave, roleArn)”. Use o parâmetro `SecretString`.

sasl.mechanism

O mecanismo de segurança usado para se conectar ao seu agente Kafka. Esse valor é necessário quando você especifica `SASL_SSL` para `security.protocol`.

Valores válidos: `PLAIN`, `SCRAM-SHA-512`, `GSSAPI`.

> **Note**
>
> `SCRAM-SHA-512`é o único mecanismo de segurança suportado nas regiões cn-north-1, cn-northwest-1, -1 e -1. us-gov-east us-gov-west

sasl.plain.username

O nome de usuário usado para recuperar a string secreta do Secrets Manager. Esse valor é necessário quando você especifica `SASL_SSL` para `security.protocol` e `PLAIN` para `sasl.mechanism` .

sasl.plain.password

A senha usada para recuperar a string secreta do Secrets Manager. Esse valor é necessário quando você especifica `SASL_SSL` para `security.protocol` e `PLAIN` para `sasl.mechanism`.

sasl.scram.username

O nome de usuário usado para recuperar a string secreta do Secrets Manager. Esse valor é necessário quando você especifica `SASL_SSL` para `security.protocol` e `SCRAM-SHA-512` para `sasl.mechanism` .

sasl.scram.password

A senha usada para recuperar a string secreta do Secrets Manager. Esse valor é necessário quando você especifica `SASL_SSL` para `security.protocol` e `SCRAM-SHA-512` para `sasl.mechanism`.

sasl.kerberos.keytab

O arquivo keytab para autenticação Kerberos no Secrets Manager. Esse valor é necessário quando você especifica `SASL_SSL` para `security.protocol` e `GSSAPI` para `sasl.mechanism`.

Este campo oferece suporte a modelos de substituição. Use o Secrets Manager para armazenar as credenciais necessárias para se conectar ao seu agente Kafka. Para recuperar o valor desse campo, use a `get_secret` função SQL. Para obter mais informações sobre modelos de substituição, consulte the section called “Modelos de substituição”. Para obter mais informações sobre a `get_secret` função SQL, consulte the section called “get_secret (secretId, secretType, chave, roleArn)”. Use o parâmetro `SecretBinary`.

sasl.kerberos.service.name

O nome principal do Kerberos sob o qual o Apache Kafka é executado. Esse valor é necessário quando você especifica `SASL_SSL` para `security.protocol` e `GSSAPI` para `sasl.mechanism`.

sasl.kerberos.krb5.kdc

O nome do host do centro de distribuição de chaves (KDC) ao qual seu cliente produtor do Apache Kafka se conecta. Esse valor é necessário quando você especifica `SASL_SSL` para `security.protocol` e `GSSAPI` para `sasl.mechanism`.

sasl.kerberos.krb5.realm

A região à qual seu cliente produtor Apache Kafka se conecta. Esse valor é necessário quando você especifica `SASL_SSL` para `security.protocol` e `GSSAPI` para `sasl.mechanism`.

sasl.kerberos.principal

A identidade Kerberos exclusiva à qual o Kerberos pode atribuir tickets para acessar serviços compatíveis com Kerberos. Esse valor é necessário quando você especifica `SASL_SSL` para `security.protocol` e `GSSAPI` para `sasl.mechanism`.

Exemplos

O exemplo de JSON a seguir define uma ação do Apache Kafka em uma regra. AWS IoT O exemplo a seguir passa a função integrada sourceIP() como um modelo de substituição no cabeçalho Kafka Action.

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "kafka": {
     "destinationArn": "arn:aws:iot:region:123456789012:ruledestination/vpc/
VPCDestinationARN",
     "topic": "TopicName",
     "clientProperties": {
      "bootstrap.servers": "kafka.com:9092",
```

```
      "security.protocol": "SASL_SSL",
      "ssl.truststore": "${get_secret('kafka_client_truststore',
 'SecretBinary','arn:aws:iam::123456789012:role/kafka-get-secret-role-name')}",
      "ssl.truststore.password": "kafka password",
      "sasl.mechanism": "GSSAPI",
      "sasl.kerberos.service.name": "kafka",
      "sasl.kerberos.krb5.kdc": "kerberosdns.com",
      "sasl.kerberos.keytab": "${get_secret('kafka_keytab','SecretBinary',
 'arn:aws:iam::123456789012:role/kafka-get-secret-role-name')}",
      "sasl.kerberos.krb5.realm": "KERBEROSREALM",
      "sasl.kerberos.principal": "kafka-keytab/kafka-keytab.com"
     },
     "headers": [
      {
       "key": "static_header_key",
       "value": "static_header_value"
      },
      {
       "key": "substitutable_header_key",
       "value": "${value_from_payload}"
      },
      {
       "key": "source_ip",
       "value": "${sourceIp()}"
      }
     ]
    }
   }
  ]
 }
}
```

Notas importantes sobre a configuração do Kerberos

- Seu centro de distribuição de chaves (KDC) deve ser resolvido por meio de um sistema de nomes de domínio (DNS) privado em sua VPC de destino. Uma abordagem possível é adicionar a entrada DNS do KDC a uma zona hospedada privada. Para obter mais informações sobre essa abordagem, consulte Como trabalhar com zonas hospedadas privadas.
- Cada VPC deve ter a resolução DNS habilitada. Para obter mais informações, consulte Como usar o DNS com sua VPC.
- Os grupos de segurança da interface de rede e os grupos de segurança em nível de instância no destino da VPC devem permitir o tráfego de dentro da sua VPC nas seguintes portas.

  - Tráfego TCP na porta do receptor do bootstrap broker (geralmente 9092, mas deve estar na faixa de 9000—9100)
  - Tráfego TCP e UDP na porta 88 para o KDC
- `SCRAM-SHA-512`é o único mecanismo de segurança suportado nas regiões cn-north-1, cn-northwest-1, -1 e -1. us-gov-east us-gov-west

Destinos da Nuvem Privada Virtual (VPC) Apache Kafka

A ação da regra Apache Kafka roteia dados para um cluster Apache Kafka em uma Amazon Virtual Private Cloud (Amazon VPC). A configuração de VPC usada pela ação da regra do Apache Kafka é ativada automaticamente quando você especifica o destino da VPC para sua ação de regra.

Um destino da Nuvem Privada Virtual (VPC) Apache Kafka contém uma lista de sub-redes dentro da VPC. O mecanismo de regras cria uma interface de rede elástica em cada sub-rede que você especificar nessa lista. Para obter mais informações sobre interfaces de rede, consulte Interfaces de rede elásticas no Guia do usuário do Amazon EC2.

Requisitos e considerações

- Se você estiver usando um cluster Apache Kafka autogerenciado que será acessado usando um endpoint público na Internet:
  - Crie um gateway NAT para instâncias em suas sub-redes. O gateway NAT tem um endereço IP público que pode se conectar à Internet, o que permite que o mecanismo de regras encaminhe suas mensagens para o cluster público do Kafka.
  - Aloque um endereço IP elástico com as interfaces de rede elástica (ENIs) que são criadas pelo destino da Nuvem Privada Virtual (VPC) do Apache Kafka. Os grupos de segurança que você usa devem ser configurados para bloquear o tráfego de entrada.

    > **Note**
    >
    > Se o destino da Nuvem Privada Virtual (VPC) Apache Kafka for desativado e reativado, você deverá associar novamente o elástico ao novo. IPs ENIs

- Se um destino da Nuvem Privada Virtual (VPC) Apache Kafka não receber nenhum tráfego por 30 dias consecutivos, ele será desativado.
- Se algum recurso usado pelo destino da Nuvem Privada Virtual (VPC) Apache Kafka mudar, o destino será desativado e não poderá ser usado.

- Algumas mudanças que podem desativar um destino de Nuvem Privada Virtual (VPC) Apache Kafka incluem:
  - Excluindo a VPC, as sub-redes, os grupos de segurança ou a função usada.
  - Modificar a função para não ter mais as permissões necessárias.
  - Atingindo quase a capacidade da sub-rede, o que nos impede de aplicar os patches do FedRAMP.
  - Desativando o destino.

Definição de preço

Para fins de preço, uma ação de regra de VPC é medida, além da ação que envia uma mensagem a um recurso quando o recurso está em sua VPC. Para obter informações sobre preços, consulte AWS IoT Core preços.

Criação de destinos de Nuvem Privada Virtual (VPC) Apache Kafka

Você cria um destino de Nuvem Privada Virtual (VPC) Apache Kafka usando a API ou o console. CreateTopicRuleDestination AWS IoT Core

Ao criar um destino, você deve especificar as seguintes informações.

vpcId

O ID exclusivo da Amazon VPC.

subnetIds

Uma lista de sub-redes nas quais o mecanismo de regras cria interfaces de rede elásticas. O mecanismo de regras aloca uma única interface de rede para cada sub-rede na lista.

securityGroups (opcional)

Uma lista de grupos de segurança a serem aplicados às interfaces de rede.

roleArn

O nome do recurso da Amazon (ARN) de uma função que tem permissão para criar interfaces de rede em seu nome.

Esse ARN deve ter uma política anexada a ele que se pareça com o exemplo a seguir.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcs",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterfacePermission",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/VPCDestinationENI": "true"
                }
        }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateNetworkInterface",
                    "aws:RequestTag/VPCDestinationENI": "true"
                }
        }
        }
    ]
}
```

Criação de um destino de Nuvem Privada Virtual (VPC) Apache Kafka usando AWS CLI

O exemplo a seguir mostra como criar um destino usando AWS CLI o.

```
aws --region regions iot create-topic-rule-destination --destination-configuration
 'vpcConfiguration={subnetIds=["subnet-
12345678910123045 6"],securityGroups=[],vpcId="vpc-
12345678910123045 6",roleArn="arn:aws:iam::123456789012:role/role-name"}'
```

Depois de executar esse comando, o status de destino seráIN_PROGRESS. Depois de alguns minutos, seu status mudará para ERROR (se o comando não for bem-sucedido) ou ENABLED. Quando o status de destino é ENABLED, estará pronto para uso.

Você pode usar o comando a seguir para obter o status do seu destino da Nuvem Privada Virtual (VPC) Apache Kafka.

```
aws --region region iot get-topic-rule-destination --arn "VPCDestinationARN"
```

Criação de um destino de Nuvem Privada Virtual (VPC) Apache Kafka usando o console AWS IoT Core

As etapas a seguir descrevem como criar um destino usando o AWS IoT Core console.

1. Navegue até o AWS IoT Core console. No painel esquerdo, na guia Agir, escolha Destinos.
2. Insira valores para os seguintes campos.
   - ID da VPC
   - Sub-rede IDs
   - Grupo de segurança
3. Selecione uma função que tenha as permissões necessárias para criar interfaces de rede. O exemplo de política anterior contém essas permissões.

Quando o status de destino da Nuvem Privada Virtual (VPC) Apache Kafka é ATIVADO, ele está pronto para uso.

CloudWatch alarmes

A ação CloudWatch alarm (`cloudWatchAlarm`) altera o estado de um CloudWatch alarme da Amazon. Você pode especificar o motivo da alteração do estado e o valor nessa chamada.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar a `cloudwatch:SetAlarmState` operação. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT a execução dessa ação de regra.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`alarmName`

O nome do CloudWatch alarme.

Suporta modelos de substituição: API e somente AWS CLI

`stateReason`

O motivo para a alteração do alarme.

Compatível com modelos de substituição: Sim

`stateValue`

O valor do estado do alarme. Valores válidos: `OK`, `ALARM`, `INSUFFICIENT_DATA`.

Compatível com modelos de substituição: Sim

`roleArn`

A função do IAM que permite o acesso ao CloudWatch alarme. Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

Exemplos

O exemplo de JSON a seguir define uma ação CloudWatch de alarme em uma AWS IoT regra.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchAlarm": {
                    "alarmName": "IotAlarm",
                    "stateReason": "Temperature stabilized.",
                    "stateValue": "OK",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw"
                }
            }
        ]
    }
}
```

Consulte também

- O que é a Amazon CloudWatch? no Guia do CloudWatch usuário da Amazon
- Usando CloudWatch alarmes da Amazon no Guia do CloudWatch usuário da Amazon

CloudWatch Registros

A ação CloudWatch Logs (`cloudwatchLogs`) envia dados para o Amazon CloudWatch Logs. Você pode usar `batchMode` para carregar e marcar com data e hora vários registros de log do dispositivo em uma mensagem. Você pode também especificar o grupo de logs para o qual a ação envia dados.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que você AWS IoT pode assumir para realizar as `logs:PutLogEvents` operações `logs:CreateLogStreamlogs:DescribeLogStreams`, e. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT a execução dessa ação de regra.
- Se você usar uma chave gerenciada pelo cliente AWS KMS key (chave KMS) para criptografar dados de registro no CloudWatch Logs, o serviço deverá ter permissão para usar a chave KMS em nome do chamador. Para obter mais informações, consulte Criptografar dados de log em CloudWatch registros usando AWS KMS o Guia do usuário do Amazon CloudWatch Logs.

Requisitos de formato de mensagem MQTT para **`batchMode`**

Se você usar a ação de regra de CloudWatch registros com `batchMode` desativada, não há requisitos de formatação de mensagens do MQTT. (Nota: o `batchMode` valor padrão do parâmetro é `false`.) No entanto, se você usar a ação de regra de CloudWatch registros `batchMode` ativada (o valor do parâmetro é`true`), as mensagens MQTT contendo registros do lado do dispositivo deverão ser formatadas para conter um carimbo de data/hora e uma carga útil da mensagem. Nota: `timestamp` representa a hora em que o evento ocorreu e é expresso como um número de milissegundos após 1º de janeiro de 1970 00:00:00 UTC.

A seguir está um exemplo do formato de publicação:

```
[
  {"timestamp": 1673520691093, "message": "Test message 1"},
  {"timestamp": 1673520692879, "message": "Test message 2"},
  {"timestamp": 1673520693442, "message": "Test message 3"}
]
```

Dependendo de como os registros do lado do dispositivo são gerados, poderá ser necessário filtrá-los e reformatá-los antes de serem enviados para cumprir este requisito. Para obter mais informações, consulte Carga útil da mensagem MQTT.

Independentemente do `batchMode` parâmetro, o `message` conteúdo deve estar em conformidade com as limitações de tamanho da AWS IoT mensagem. Para obter mais informações, consulte AWS IoT Core Endpoints e cotas.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`logGroupName`

O grupo de CloudWatch registros para o qual a ação envia dados.

Suporta modelos de substituição: API e somente AWS CLI

`roleArn`

A função do IAM que permite acesso ao grupo de CloudWatch registros. Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

(opcional) `batchMode`

Indica se lotes de registros de log serão extraídos e enviados para. CloudWatch Os valores incluem `true` ou `false` (padrão). Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

Exemplos

O exemplo de JSON a seguir define uma ação de CloudWatch registros em uma AWS IoT regra.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchLogs": {
                    "logGroupName": "IotLogs",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw",
                    "batchMode": false
```

```
                }
            }
        ]
    }
}
```

Consulte também

- O que é o Amazon CloudWatch Logs? no Guia do usuário do Amazon CloudWatch Logs

CloudWatch métricas

A ação CloudWatch metric (`cloudwatchMetric`) captura uma CloudWatch métrica da Amazon. Você pode especificar o namespace, nome, valor, unidade e marcação de data e hora da métrica.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar a `cloudwatch:PutMetricData` operação. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT a execução dessa ação de regra.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`metricName`

O nome da CloudWatch métrica.

Compatível com modelos de substituição: Sim

`metricNamespace`

O nome do namespace CloudWatch métrico.

Compatível com modelos de substituição: Sim

`metricUnit`

A unidade métrica suportada por CloudWatch.

Compatível com modelos de substituição: Sim

`metricValue`

Uma string que contém o valor da CloudWatch métrica.

Compatível com modelos de substituição: Sim

`metricTimestamp`

(Opcional) que contém o carimbo de data/hora, expresso em segundos na época do Unix. O padrão é a época atual do Unix.

Compatível com modelos de substituição: Sim

`roleArn`

A função do IAM que permite acesso à CloudWatch métrica. Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

Exemplos

O exemplo de JSON a seguir define uma ação CloudWatch métrica em uma AWS IoT regra.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchMetric": {
                    "metricName": "IotMetric",
                    "metricNamespace": "IotNamespace",
                    "metricUnit": "Count",
                    "metricValue": "1",
                    "metricTimestamp": "1456821314",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw"
                }
```

```
            }
        ]
    }
}
```

O exemplo de JSON a seguir define uma ação CloudWatch métrica com modelos de substituição em uma AWS IoT regra.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchMetric": {
                    "metricName": "${topic()}",
                    "metricNamespace": "${namespace}",
                    "metricUnit": "${unit}",
                    "metricValue": "${value}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw"
                }
            }
        ]
    }
}
```

Consulte também

- O que é a Amazon CloudWatch? no Guia do CloudWatch usuário da Amazon
- Usando CloudWatch métricas da Amazon no Guia do CloudWatch usuário da Amazon

DynamoDB

A ação DynamoDB(dynamoDB) grava toda ou parte de uma mensagem MQTT em uma tabela do Amazon DynamoDB.

Você pode seguir um tutorial que mostra como criar e testar uma regra com uma ação do DynamoDB. Para obter mais informações, consulte Tutorial: Armazenamento de dados do dispositivo em uma tabela do DynamoDB.

> **Note**
>
> Esta regra grava dados não JSON no DynamoDB como dados binários. O console do DynamoDB exibe os dados como texto codificado em Base64.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar a `dynamodb:PutItem` operação. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT a execução dessa ação de regra.
- Se você usa uma chave gerenciada pelo cliente AWS KMS key (chave KMS) para criptografar dados em repouso no DynamoDB, o serviço deve ter permissão para usar a chave KMS em nome do chamador. Para obter mais informações, consulte a chave KMS gerenciada pelo cliente no Guia de conceitos básicos do Amazon DynamoDB.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`tableName`

O nome da tabela do DynamoDB.

Suporta modelos de substituição: API e somente AWS CLI

`hashKeyField`

O nome da chave de hash (também chamada de chave de partição).

Suporta modelos de substituição: API e somente AWS CLI

`hashKeyType`

(Opcional) O tipo de dados da chave de hash (também chamada de chave de partição). Valores válidos: `STRING`, `NUMBER`.

Suporta modelos de substituição: API e somente AWS CLI

`hashKeyValue`

O valor da chave de hash. Considere usar um modelo de substituição, como `${topic()}` ou `${timestamp()}`.

Compatível com modelos de substituição: Sim

`rangeKeyField`

(Opcional) O nome da chave de intervalo (também chamada de chave de classificação).

Suporta modelos de substituição: API e somente AWS CLI

`rangeKeyType`

(Opcional) O tipo de dados da chave de intervalo (também chamada de chave de classificação). Valores válidos: `STRING`, `NUMBER`.

Suporta modelos de substituição: API e somente AWS CLI

`rangeKeyValue`

(Opcional) O valor da chave de intervalo. Considere usar um modelo de substituição, como `${topic()}` ou `${timestamp()}`.

Compatível com modelos de substituição: Sim

`payloadField`

(Opcional) O nome da coluna em que a carga útil é gravada. Se esse valor for omitido, a carga útil será gravada na coluna nomeada `payload`.

Compatível com modelos de substituição: Sim

`operation`

(Opcional) O tipo de operação a executar. Valores válidos: `INSERT`, `UPDATE`, `DELETE`.

Compatível com modelos de substituição: Sim

`roleARN`

O perfil do IAM que permite o acesso à tabela do DynamoDB. Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

Os dados gravados na tabela do DynamoDB são o resultado da instrução SQL da regra.

Exemplos

O exemplo de JSON a seguir define uma ação do DynamoDB em uma regra. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * AS message FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "dynamoDB": {
                    "tableName": "my_ddb_table",
                    "hashKeyField": "key",
                    "hashKeyValue": "${topic()}",
                    "rangeKeyField": "timestamp",
                    "rangeKeyValue": "${timestamp()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDB"
                }
            }
        ]
    }
}
```

Consulte também

- O que é o Amazon DynamoDB? no Guia do desenvolvedor do Amazon DynamoDB
- Conceitos básicos do DynamoDB no Guia do desenvolvedor do Amazon DynamoDB
- Tutorial: Armazenamento de dados do dispositivo em uma tabela do DynamoDB

Dínamo DBv2

A ação Dynamo DBv2 (`dynamoDBv2`) grava toda ou parte de uma mensagem MQTT em uma tabela do Amazon DynamoDB. Cada atributo na carga útil é gravado em uma coluna separada no banco de dados DynamoDB.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar a `dynamodb:PutItem` operação. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT a execução dessa ação de regra.
- A carga útil da mensagem MQTT deve conter uma chave de nível raiz que corresponda à chave de partição primária da tabela e uma chave de nível raiz que corresponda à chave de classificação primária da tabela, se alguma estiver definida.
- Se você usa uma chave gerenciada pelo cliente AWS KMS key (chave KMS) para criptografar dados em repouso no DynamoDB, o serviço deve ter permissão para usar a chave KMS em nome do chamador. Para obter mais informações, consulte a chave KMS gerenciada pelo cliente no Guia de conceitos básicos do Amazon DynamoDB.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`putItem`

Um objeto que especifica a tabela do DynamoDB na qual os dados da mensagem serão gravados. Este objeto deve conter as seguintes informações:

`tableName`

O nome da tabela do DynamoDB.

Suporta modelos de substituição: API e somente AWS CLI

`roleARN`

O perfil do IAM que permite o acesso à tabela do DynamoDB. Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

Os dados gravados na tabela do DynamoDB são o resultado da instrução SQL da regra.

Exemplos

O exemplo de JSON a seguir define uma DBv2 ação do Dynamo em uma AWS IoT regra.

```
{
    "topicRulePayload": {
        "sql": "SELECT * AS message FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "dynamoDBv2": {
                    "putItem": {
                        "tableName": "my_ddb_table"
                    },
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDBv2",
                }
            }
        ]
    }
}
```

O exemplo de JSON a seguir define uma ação do DynamoDB com modelos de substituição em uma regra. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2015-10-08",
        "actions": [
            {
                "dynamoDBv2": {
                    "putItem": {
                        "tableName": "${topic()}"
                    },
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDBv2"
                }
            }
        ]
    }
}
```

Consulte também

- O que é o Amazon DynamoDB? no Guia do desenvolvedor do Amazon DynamoDB

- Conceitos básicos do DynamoDB no Guia do desenvolvedor do Amazon DynamoDB

Elasticsearch

A ação Elasticsearch (`elasticsearch`) grava dados de mensagens MQTT em um domínio do Amazon OpenSearch Service. Em seguida, você pode usar ferramentas como OpenSearch painéis para consultar e visualizar dados no OpenSearch Service.

> **Warning**
>
> A ação `Elasticsearch` só pode ser usada por ações de regras existentes. Para criar uma nova ação de regra ou atualizar uma existente, use a ação de regra `OpenSearch`. Para obter mais informações, consulte OpenSearch.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar a `es:ESHttpPut` operação. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT a execução dessa ação de regra.
- Se você usar uma chave gerenciada pelo cliente AWS KMS key (chave KMS) para criptografar dados em repouso OpenSearch, o serviço deverá ter permissão para usar a chave KMS em nome do chamador. Para obter mais informações, consulte Criptografia de dados em repouso para o Amazon OpenSearch Service no Amazon OpenSearch Service Developer Guide.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`endpoint`

O endpoint do seu domínio de serviço.

Suporta modelos de substituição: API e somente AWS CLI

`index`

O índice onde você deseja armazenar seus dados.

Compatível com modelos de substituição: Sim

`type`

O tipo de documento que você está armazenando.

Compatível com modelos de substituição: Sim

`id`

O identificador exclusivo de cada documento.

Compatível com modelos de substituição: Sim

`roleARN`

A função do IAM que permite acesso ao domínio do OpenSearch Serviço. Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

Exemplos

O exemplo de JSON a seguir define uma ação do Elasticsearch em uma AWS IoT regra e como você pode especificar os campos para a ação. `elasticsearch` Para obter mais informações, consulte ElasticsearchAction.

```
{
    "topicRulePayload": {
        "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "elasticsearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "my-index",
                    "type": "my-type",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es"
```

```
                }
            }
        ]
    }
}
```

O exemplo de JSON a seguir define uma ação do Elasticsearch com modelos de substituição em uma regra. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "elasticsearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "${topic()}",
                    "type": "${type}",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es"
                }
            }
        ]
    }
}
```

Consulte também

- OpenSearch
- O que é o Amazon OpenSearch Service?

HTTP

A ação HTTPS (`http`) envia dados de uma mensagem MQTT para um endpoint HTTPS, que pode apontar para um aplicativo ou serviço web.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Você deve confirmar e ativar os endpoints HTTPS antes que o mecanismo de regras possa usá-los. Para obter mais informações, consulte Destinos de ação HTTP.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`url`

O endpoint HTTPS para o qual a mensagem é enviada usando o método HTTP POST. Se você usar um endereço IP no lugar de um nome de host, ele deverá ser um IPv4 endereço. IPv6 endereços não são suportados.

Compatível com modelos de substituição: Sim

`confirmationUrl`

(Opcional) Se especificado, AWS IoT usa o URL de confirmação para criar um destino de regra de tópico correspondente. Você deve habilitar o destino da ação HTTP antes de usá-la em uma ação HTTP. Para obter mais informações, consulte Destinos de ação HTTP. Se você usar modelos de substituição, deverá criar manualmente um destino de ação HTTP antes que a `http` ação possa ser usada. `confirmationUrl`deve ser um prefixo de. `url`

A relação entre `url` e `confirmationUrl` é descrita pelo seguinte:

- Se `url` estiver codificado e não `confirmationUrl` for fornecido, tratamos implicitamente o `url` campo como o. `confirmationUrl` AWS IoT cria um destino de regra de tópico para`url`.
- Se `url` e `confirmationUrl` estiverem codificados, `url` deve começar com. `confirmationUrl` AWS IoT cria um destino de regra de tópico para`confirmationUrl`.
- Se `url` contiver um modelo de substituição, você deverá especificar `confirmationUrl` e `url` deverá começar com `confirmationUrl`. Se `confirmationUrl` contiver modelos de substituição, você deverá criar manualmente um destino de ação HTTP antes que a `http` ação possa ser usada. Se `confirmationUrl` não contiver modelos de substituição, AWS IoT cria um destino de regra de tópico para`confirmationUrl`.

Compatível com modelos de substituição: Sim

`headers`

(Opcional) A lista de cabeçalhos a serem incluídos nas solicitações HTTP para o endpoint. Cada cabeçalho deve conter as seguintes informações:

`key`

A chave do cabeçalho.

Compatível com modelos de substituição: Não

`value`

O valor do cabeçalho.

Compatível com modelos de substituição: Sim

> **Note**
>
> O tipo de conteúdo padrão é application/json quando a carga está no formato JSON. Caso contrário, é application/octet-stream. Você pode substituí-lo especificando o tipo de conteúdo exato no cabeçalho com o tipo de conteúdo da chave (sem distinção entre maiúsculas e minúsculas).

`auth`

(Opcional) A autenticação usada pelo mecanismo de regras para se conectar ao URL do endpoint especificado no argumento `url`. Atualmente, o Signature versão 4 é o único tipo de autenticação suportado. Para obter mais informações, consulte Autorização HTTP.

Compatível com modelos de substituição: Não

`enableBatching`

(Opcional) Se as mensagens de ação HTTP devem ser processadas em uma única solicitação para um determinado URL. O valor pode ser verdadeiro ou falso. Para obter mais informações sobre agrupamento em lotes, consulte Mensagens de ação HTTP em lotes.

Valor booleano

Compatível com modelos de substituição: Não

`batchConfig`

(Opcional) Configurações para agrupamento em lotes. Depois de ativados, `batchConfig` os parâmetros devem ser especificados. Se `batchConfig` os parâmetros não forem especificados, os valores padrão serão usados.

`maxBatchOpenMs`

O tempo máximo (em milissegundos) que uma mensagem de saída espera por outras mensagens para criar o lote. Quanto maior a configuração, maior a latência da ação HTTP em lote.

Valor mínimo: 5 ms. Valor máximo: 200 ms.

Valor padrão: 20 ms

Compatível com modelos de substituição: Não

`maxBatchSize`

O número máximo de mensagens que são agrupadas em lotes em uma única execução de ação.

Compatível com modelos de substituição: Não

Valor mínimo: 2 mensagens. Valor máximo: 10 mensagens

Valor padrão: 10 mensagens

`maxBatchSizeBytes`

Tamanho máximo de um lote de mensagens, em bytes.

Valor mínimo: 100 bytes. Valor máximo: 131.072 bytes

Valor padrão: 5.120 bytes

Compatível com modelos de substituição: Não

> **Note**
>
> O tipo de conteúdo padrão é application/json quando a carga está no formato JSON. Caso contrário, é application/octet-stream. Você pode substituí-lo especificando o tipo de conteúdo exato no cabeçalho com o tipo de conteúdo da chave (sem distinção entre maiúsculas e minúsculas).

Exemplos

O exemplo de JSON a seguir define uma AWS IoT regra com uma ação HTTP.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "http": {
                    "url": "https://www.example.com/subpath",
                    "confirmationUrl": "https://www.example.com",
                    "headers": [
                        {
                            "key": "static_header_key",
                            "value": "static_header_value"
                        },
                        {
                            "key": "substitutable_header_key",
                            "value": "${value_from_payload}"
                        }
                    ]
                }
            }
        ]
    }
}
```

```
"http": {
    "url": "https://www.example.com/subpath",
    "confirmationUrl": "https://www.example.com",
    "headers": [
        {
            "key": "Content-Type",
            "value": "application/json"
          }
    ],
    "enableBatching": true,
    "batchConfig": {
      "maxBatchOpenMs": 123,
      "maxBatchSize": 5,
      "maxBatchSizeBytes": 131072,
     }
 },
 "errorAction": {
```

```
        "http": {
            "url": "https://www.example.com/subpath",
            "confirmationUrl": "https://www.example.com"
            // batchConfig is not allowed here
        }
}
```

Lógica de repetição da ação HTTP

O mecanismo de AWS IoT regras tenta novamente a ação HTTP de acordo com estas regras:

- O mecanismo de regras tenta enviar uma mensagem pelo menos uma vez.
- O mecanismo de regras tenta novamente no máximo duas vezes. O número máximo de tentativas é três.
- O mecanismo de regras não fará uma nova tentativa se:
  - A tentativa anterior tiver fornecido uma resposta maior do que 16.384 bytes.
  - O serviço ou aplicativo web downstream fechar a conexão TCP após a tentativa.
  - O tempo total para concluir uma solicitação com repetições tiver excedido o limite de tempo limite da solicitação.
  - A solicitação retorna um código de status HTTP diferente de 429, 500–599.

> **Note**
>
> Custos padrão de transferência de dados se aplicam a novas tentativas.

Consulte também

- Agrupar mensagens de ação HTTP em lote
- Destinos de ação HTTP
- Encaminhe dados diretamente AWS IoT Core para seus serviços da Web na Internet das Coisas no AWS blog

Agrupar mensagens de ação HTTP em lote

Você pode usar o agrupamento em lotes para enviar várias mensagens de ação HTTP em uma única solicitação.

Visão geral do

O agrupamento em lotes permite que você envie mensagens do AWS IoT Core Rules Engine para seus endpoints HTTP em lotes. Essa funcionalidade pode ajudar a reduzir seus custos diminuindo o número de execuções de ações HTTP, bem como melhorar a eficiência ao reduzir a sobrecarga associada ao estabelecimento de novas conexões.

> **Note**
>
> A ação HTTP em lote é medida como uma única ação. Você é medido em incrementos de 5 KiB, com base no tamanho da carga útil em lote de saída emitida pelo Mecanismo de AWS IoT Core Regras para o serviço downstream. Para obter mais informações, consulte a página de definição de preços do AWS IoT Core.

Quando você ativa o agrupamento em lotes na definição de sua Ação de Regra de IoT, os seguintes parâmetros estarão disponíveis para configuração:

`maxBatchOpenMs`

O tempo máximo (em milissegundos) que uma mensagem de saída espera por outras mensagens para criar o lote. Quanto maior a configuração, maior a latência da ação HTTP em lote.

Valor mínimo: 5 ms. Valor máximo: 200 ms.

Valor padrão: 20 ms

Compatível com modelos de substituição: Não

`maxBatchSize`

O número máximo de mensagens que são agrupadas em lotes em uma única execução de ação de regra de IoT.

Valor mínimo: 2 mensagens. Valor máximo: 10 mensagens

Valor padrão: 10 mensagens

Compatível com modelos de substituição: Não

`maxBatchSizeBytes`

Tamanho máximo de um lote de mensagens, em bytes.

Valor mínimo: 100 bytes. Valor máximo: 131.072 bytes

Valor padrão: 5120 bytes

Compatível com modelos de substituição: Não

> **Important**
>
> Quando você especifica vários parâmetros de lote, o lote é concluído quando o primeiro limite é atingido. Por exemplo, se você especificar 100 ms como o Tempo Máximo de Abertura do Lote e 5 KiB como o Tamanho Máximo do Lote, e o Rules Engine agrupar somente 2 KiB em 100 ms, um lote de 2 KiB será criado e enviado.

Usando cabeçalhos HTTP em um lote

Quando você usa cabeçalhos em sua ação HTTP, a solicitação em lote usa o valor do cabeçalho da última mensagem que foi adicionada ao lote (não necessariamente a última mensagem que você publicou). Recomendamos usar valores de cabeçalho que sejam:

- Idêntico em todas as mensagens do lote
- Aplicável a todas as mensagens (por exemplo, credenciais de autenticação)

Os cabeçalhos são enviados com a solicitação HTTP e não fazem parte do corpo da mensagem.

> **Note**
>
> Quando o agrupamento em lotes está ativado:
>
> - A solicitação em lote inclui automaticamente o `Content-Type: application/json` cabeçalho, pois o lote é enviado como uma matriz JSON.
> - Não podemos garantir que a última mensagem do lote seja a última mensagem que você publicou. É a última mensagem que entrou no lote.

Exemplo de carga

O exemplo a seguir mostra a estrutura de uma carga de mensagens em lote enviada ao seu endpoint HTTP:

```
[
  {
    "user_id": "user1",
    "steps_today": 1000
  },
  {
    "user_id": "user2",
    "steps_today": 21000
  },
  {
    "user_id": "user8",
    "steps_today": 1500
  },
  ...
]
```

Limitações

A seguir estão as limitações do agrupamento em lotes:

- AWS IoT Core não garante a ordem geral das mensagens. O agrupamento em lotes é realizado localmente em cada host, o que pode fazer com que as mensagens dentro de um lote sejam processadas em uma ordem diferente da que foram recebidas.
- AWS IoT Core não fornece suporte ao processamento de mensagens no lado do receptor. Você é responsável por garantir que seu serviço downstream esteja configurado para aceitar e processar dados em lotes.
- O agrupamento em lotes entre contas não é suportado, mesmo que as mensagens sejam destinadas ao mesmo identificador de recurso (URL HTTP ou ARN do recurso).
- AWS IoT Core não garante que o tamanho do lote atenda à configuração especificada. Os lotes podem ser menores do que os limites configurados com base no tempo e no fluxo de mensagens.
- Quando o agrupamento em lotes está ativado, cargas binárias (dados não UTF-8) não são suportadas. Somente cargas de texto UTF-8 (como JSON) são aceitas. Para enviar dados binários, codifique-os em base64 antes de enviá-los para a ação HTTP e, em seguida, decodifique-os no ponto final de recebimento. Por exemplo, você pode usar a função de codificação nas regras

de IoT para codificar a carga binária. Como alternativa, você pode codificar a carga binária em seu dispositivo de IoT e publicá-la no. AWS IoT Core

Ações de erro para agrupamento em lotes

Você não poderá definir uma lógica de lote separada em sua definição de Ação de Erro. No entanto, sua ação de erro suportará o agrupamento em lotes se você tiver definido a lógica de lote em sua ação principal.

Quando uma solicitação em lote falha, o mecanismo de AWS IoT Core regras segue a lógica de repetição da ação HTTP. Após a última tentativa de repetição, uma ação de erro será invocada para todo o lote com falha.

Veja a seguir um exemplo de uma mensagem de ação de erro com o agrupamento em lotes ativado:


```
{
    "ruleName": "FailedTopicRule",
    "topic": "topic/rulesengine",
    "payloadsWithMetadata": [
        {
            "id": 1,
            "cloudwatchTraceId": "bebd6d93-6d4a-899e-9e40-56e82252d2be",
            "clientId": "Test",
            "sourceIp": "10.0.0.0",
            "base64OriginalPayload":
 "eyJ1c2VyX2lkIjogInVzZXI1NjQ3IiwgInN0ZXBzX3RvZGF5IjogMTMzNjUsICJ0aW1lc3RhbXAiOiAiMjAyNS0xMC0wO
        },
        {
            "id": 2,
            "cloudwatchTraceId": "af94d3b8-0b18-1dbf-2c7d-513f5cb9e2e1",
            "clientId": "Test",
            "sourceIp": "10.0.0.0",
            "base64OriginalPayload":
 "eyJ1c2VyX2lkIjogInVzZXI1NjQ3IiwgInN0ZXBzX3RvZGF5IjogMTMzNjUsICJ0aW1lc3RhbXAiOiAiMjAyNS0xMC0wO
        },
        {
            "id": 3,
            "cloudwatchTraceId": "ca441266-c2ce-c916-6aee-b9e5c7831675",
            "clientId": "Test",
            "sourceIp": "10.0.0.0",
            "base64OriginalPayload":
 "eyJ1c2VyX2lkIjogInVzZXI1NjQ3IiwgInN0ZXBzX3RvZGF5IjogMTMzNjUsICJ0aW1lc3RhbXAiOiAiMjAyNS0xMC0wO
```

```
        }
    ],
    "failures": [
        {
            "affectedIds": [
                1,
                2,
                3
            ],
            "failedAction": "HttpAction",
            "failedResource": "https://example.foobar.com/HttpAction",
            "errorMessage": "HttpAction failed to make a request to the specified
 endpoint. StatusCode: 500. Reason: Internal Server Error."
        },
        {
            "affectedIds": [
                3
            ],
            "failedAction": "S3Action",
            "failedResource": "amzn-s3-demo-bucket",
            "errorMessage": "Failed to put S3 object. The error received was The
 specified bucket does not exist"
        },
        {
            "affectedIds": [
                3
            ],
            "failedAction": "LambdaAction",
            "failedResource": "arn:aws:lambda:us-west-2:123456789012:function:dummy",
            "errorMessage": "Failed to invoke lambda function. Received Server error
 from Lambda. The error code is 403"
        }
    ]
}
```

> **Note**
>
> Falhas de ação em lote também geram cargas de ação de erro maiores, o que pode aumentar a probabilidade de falhas de ação de erro devido ao tamanho. Você pode monitorar falhas na ação de erro usando a `ErrorActionFailure` métrica. Consulte Métricas de ação da regra para obter mais informações.

Envio de mensagens de ação HTTP em lote com o AWS CLI

Criação ou atualização de uma ação de regra com o agrupamento em lotes

1. Use o AWS CLI comando apropriado para criar ou atualizar uma regra:
   - Para criar uma nova regra, use o create-topic-rulecomando:

     ```
     aws iot create-topic-rule --rule-name myrule --topic-rule-payload
      file://myrule.json
     ```

   - Para atualizar uma regra existente, use o replace-topic-rulecomando:

     ```
     aws iot replace-topic-rule --rule-name myrule --topic-rule-payload
      file://myrule.json
     ```

2. Ative os recursos de agrupamento em lotes definindo o parâmetro enableBatching como verdadeiro na carga útil da regra de tópico:

```
{
        "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "http": {
                    "url": "https://www.example.com/subpath",
                    "confirmationUrl": "https://www.example.com",
                    "headers": [
                        {
                            "key": "static_header_key",
                            "value": "static_header_value"
                        },
                        {
                            "key": "substitutable_header_key",
                            "value": "${value_from_payload}"
                         }
                    ],
                    "enableBatching": true,
                    "batchConfig": {
                       "maxBatchOpenMs": 100,
                       "maxBatchSize": 5,
                       "maxBatchSizeBytes": 1024
```

```
                    }
                }
            }
        ]
}
```

3. Configure os parâmetros de agrupamento em lotes. Você não precisa especificar todos os parâmetros do lote. Você pode optar por especificar 1, 2 ou todos os 3 parâmetros de lote. Se você não especificar um parâmetro de lote, o Mecanismo de Regras atualizará esse parâmetro com os valores padrão. Para obter mais informações sobre parâmetros de lote e seus valores padrão, consulte Parâmetros HTTP.

Destinos de ação HTTP

Um destino de ação HTTP é um serviço da Web para o qual o mecanismo de regras pode rotear dados de uma regra de tópico. Um AWS IoT Core recurso descreve o serviço web para AWS IoT. Os recursos de destino podem ser compartilhados por regras diferentes.

Antes de AWS IoT Core poder enviar dados para outro serviço da Web, ele deve confirmar que pode acessar o endpoint do serviço.

Visão geral do

Um destino de ação HTTP se refere a um serviço da Web que oferece suporte a uma URL de confirmação e a uma ou mais coletas de dados URLs. O recurso de destino contém o URL de confirmação do seu serviço web. Ao configurar uma ação HTTP, você especifica a URL real do endpoint que deve receber os dados junto com a URL de confirmação do serviço web. Após a confirmação do destino, a regra do tópico envia o resultado da instrução SQL para o endpoint HTTPS (e não para a URL de confirmação).

O destino de uma ação HTTP pode estar em um dos seguintes estados:

ENABLED

O destino foi confirmado e pode ser usado por uma ação de regra. Um destino deve estar no estado `ENABLED` para que seja usado em uma regra. Você só pode habilitar destinos com status DISABLED.

DISABLED

O destino foi confirmado mas não pode ser usado por uma ação de regra. Isso é útil se você quiser impedir temporariamente o tráfego para seu endpoint sem precisar passar pelo processo de confirmação novamente. Você só pode desabilitar destinos com status ENABLED.

IN_PROGRESS

A confirmação do destino está em andamento.

ERRO

A confirmação do destino expirou.

Depois que um destino de ação HTTP for confirmado e ativado, ele poderá ser usado com qualquer regra em sua conta.

Gerenciando destinos de ações HTTP

Você pode usar as operações a seguir para gerenciar seus destinos de ação HTTP.

Criação de destinos de ação HTTP

Você cria um destino de ação HTTP chamando a `CreateTopicRuleDestination` operação ou usando o AWS IoT console.

Depois de criar um destino, AWS IoT envia uma solicitação de confirmação para o URL de confirmação. A solicitação de confirmação tem o seguinte formato:

```
HTTP POST {confirmationUrl}/?confirmationToken={confirmationToken}
Headers:
x-amz-rules-engine-message-type: DestinationConfirmation
x-amz-rules-engine-destination-arn:"arn:aws:iot:us-east-1:123456789012:ruledestination/
http/7a280e37-b9c6-47a2-a751-0703693f46e4"
Content-Type: application/json
Body:
{
    "arn":"arn:aws:iot:us-east-1:123456789012:ruledestination/http/7a280e37-b9c6-47a2-
a751-0703693f46e4",
    "confirmationToken": "AYADeMXLrPrNY2wqJAKsFNn-…NBJndA",
    "enableUrl": "https://iot.us-east-1.amazonaws.com/confirmdestination/
AYADeMXLrPrNY2wqJAKsFNn-…NBJndA",
    "messageType": "DestinationConfirmation"
}
```

O conteúdo do pedido de confirmação inclui as seguintes informações:

arn

O Amazon Resource Name (ARN) do destino da ação HTTP a ser confirmado.

confirmationToken

O token de confirmação enviado por AWS IoT Core. O token no exemplo é truncado. Seu token será mais longo. Você precisará desse token para confirmar seu destino em AWS IoT Core.

enableUrl

O URL para o qual você navega para confirmar um destino de regra de tópico.

messageType

O tipo de mensagem.

Confirmando destinos de ações HTTP

Para concluir o processo de confirmação do endpoint, se você estiver usando a AWS CLI, deverá realizar as seguintes etapas após o URL de confirmação receber a solicitação de confirmação.

1. Confirme se o destino está pronto para receber mensagens

   Para confirmar se o destino da ação HTTP está pronto para receber mensagens de IoT, chame o `enableUrl` na solicitação de confirmação ou execute a operação da `ConfirmTopicRuleDestination` API e transmita a `confirmationToken` partir da solicitação de confirmação.
2. Defina o status da regra de tópico como ativado

   Depois de confirmar que o destino pode receber mensagens, você deve realizar a operação de API `UpdateTopicRuleDestination` para definir o status da regra de tópico como `ENABLED`.

Se você estiver usando o AWS IoT console, `confirmationToken` copie e cole na caixa de diálogo de confirmação do destino no AWS IoT console. Em seguida, você pode ativar a regra do tópico.

Enviando uma nova solicitação de confirmação

Para acionar uma nova mensagem de confirmação para um destino, faça uma chamada para `UpdateTopicRuleDestination` e defina o status do destino da regra de tópico como `IN_PROGRESS`.

Repita o processo de confirmação depois de enviar uma nova solicitação de confirmação.

Desabilitando e excluindo um destino de ação HTTP

Para desativar um destino, chame `UpdateTopicRuleDestination` e defina o status do destino da regra de tópico como `DISABLED`. Uma regra de tópico no estado DISABLED pode ser ativada novamente sem a necessidade de enviar uma nova solicitação de confirmação.

Para excluir um destino de ação HTTP, chame`DeleteTopicRuleDestination`.

Support da Autoridade Certificadora

> **Note**
>
> Não há suporte para certificados autoassinados.

Os endpoints HTTPS em um destino de ação HTTP oferecem suporte a certificados emitidos pela Autoridade de Certificação AWS Privada e pela Lets Encrypt.

AWS IoT Events

A ação AWS IoT Events (`iotEvents`) envia dados de uma mensagem MQTT para uma AWS IoT Events entrada.

> **Important**
>
> Se a carga for enviada AWS IoT Core sem `oInput attribute Key`, ou se a chave não estiver no mesmo caminho JSON especificado na chave, isso fará com que a regra de IoT falhe com o erro. `Failed to send message to Iot Events`

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar a `iotevents:BatchPutMessage` operação. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT a execução dessa ação de regra.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`batchMode`

(Opcional) Se as ações do evento devem ser processadas em lote. O valor padrão é `false`.

Quando `batchMode` é `true` e a regra a instrução SQL é avaliada como uma matriz, cada elemento da matriz é tratado como uma mensagem separada quando é enviado para AWS IoT Eventos por meio de uma chamada `BatchPutMessage`. A matriz resultante não pode ter mais de 10 mensagens.

Quando `batchMode` é `true`, não é possível especificar um `messageId`.

Compatível com modelos de substituição: Não

`inputName`

O nome da AWS IoT Events entrada.

Suporta modelos de substituição: API e somente AWS CLI

`messageId`

(Opcional) Use isso para verificar se somente uma entrada (mensagem) com uma determinada `messageId` é processada por um AWS IoT Events detector. Você pode usar o `${newuuid()}` modelo de substituição para gerar uma ID exclusiva para cada solicitação.

Quando `batchMode` estiver `true`, você não poderá especificar um `messageId`—um novo valor de UUID será atribuído.

Compatível com modelos de substituição: Sim

`roleArn`

A função do IAM que AWS IoT permite enviar uma entrada para um AWS IoT Events detector. Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

Exemplos

O exemplo JSON a seguir define uma ação do IoT Events em uma AWS IoT regra.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotEvents": {
                    "inputName": "MyIoTEventsInput",
                    "messageId": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_events"
                }
            }
        ]
    }
}
```

Consulte também

- O que AWS IoT Eventsé no Guia do AWS IoT Events desenvolvedor

AWS IoT SiteWise

A ação AWS IoT SiteWise (`iotSiteWise`) envia dados de uma mensagem MQTT para as propriedades do ativo em AWS IoT SiteWise.

Você pode seguir um tutorial que mostra como ingerir dados de AWS IoT coisas. Para obter mais informações, consulte o tutorial Como ingerir dados a AWS IoT SiteWise partir de AWS IoT coisas ou a seção Como ingerir dados usando regras AWS IoT básicas no Guia do AWS IoT SiteWise usuário.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar a `iotsitewise:BatchPutAssetPropertyValue` operação. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

Você pode anexar o exemplo de política de confiança a seguir à função.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iotsitewise:BatchPutAssetPropertyValue",
            "Resource": "*"
        }
    ]
}
```

Para melhorar a segurança, você pode especificar um caminho AWS IoT SiteWise de hierarquia de ativos na `Condition` propriedade. O exemplo a seguir é uma política de confiança que especifica um caminho de hierarquia de ativos.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iotsitewise:BatchPutAssetPropertyValue",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iotsitewise:assetHierarchyPath": [
                        "/root node asset ID",
                        "/root node asset ID/*"
                    ]
            }
        }
        }
    ]
}
```

- Quando você envia dados para AWS IoT SiteWise com essa ação, seus dados devem atender aos requisitos da `BatchPutAssetPropertyValue` operação. Para obter mais informações, consulte BatchPutAssetPropertyValue na Referência de APIs do AWS IoT SiteWise .

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`putAssetPropertyValueEntries`

Uma lista de entradas de valor de propriedade de ativo, cada uma contendo as seguintes informações:

`propertyAlias`

(Opcional) O alias da propriedade associado à propriedade do seu ativo. É necessário especificar um `propertyAlias`, ou ambos um `assetId` e um `propertyId`. Para obter mais informações sobre alias de propriedade, consulte Como realizar o mapeamento de fluxos de dados industriais para propriedades de ativos no AWS IoT SiteWise Guia do usuário .

Compatível com modelos de substituição: Sim

`assetId`

(Opcional) O ID do AWS IoT SiteWise ativo. É necessário especificar um `propertyAlias`, ou ambos um `assetId` e um `propertyId`.

Compatível com modelos de substituição: Sim

`propertyId`

(Opcional) A ID de uma propriedade de ativo. É necessário especificar um `propertyAlias`, ou ambos um `assetId` e um `propertyId`.

Compatível com modelos de substituição: Sim

`entryId`

(Opcional) Um identificador exclusivo para essa entrada. Você pode definir o `entryId` para controlar melhor qual mensagem causou um erro em caso de falha. O padrão é um novo UUID.

Compatível com modelos de substituição: Sim

`propertyValues`

Uma lista de valores de propriedade a serem inseridos, em que cada um contém carimbo de data/hora, qualidade e valor (TQV) no seguinte formato:

`timestamp`

Uma estrutura de carimbo de data/hora que contém as seguintes informações:

`timeInSeconds`

Uma string que contém o tempo em segundos no horário do Unix epoch. Se a carga de mensagem não tiver um carimbo de data/hora, você poderá usar timestamp(), que retorna a hora atual em milissegundos. Para converter essa hora em segundos, você pode usar o seguinte modelo de substituição: **`${floor(timestamp() / 1E3)}`**.

Compatível com modelos de substituição: Sim

`offsetInNanos`

(Opcional) Uma string que contém o deslocamento de tempo em nanossegundos da hora em segundos. Se a carga útil de mensagem não tiver um carimbo de data/hora, você poderá usar timestamp(), que retorna a hora atual em milissegundos. Para calcular o deslocamento em nanossegundos a partir dessa hora, é possível usar o seguinte modelo de substituição: **`${(timestamp() % 1E3) * 1E6}`**.

Compatível com modelos de substituição: Sim

Em relação ao Unix epoch time, AWS IoT SiteWise aceita somente entradas que tenham um timestamp de até 7 dias no passado e até 5 minutos no futuro.

`quality`

(Opcional) Uma string que descreve a qualidade do valor. Valores válidos: `GOOD`, `BAD`, `UNCERTAIN`.

Compatível com modelos de substituição: Sim

`value`

Uma estrutura de valor que contém um dos seguintes campos de valor, dependendo do tipo de dados da propriedade do ativo:

`booleanValue`

(Opcional) Uma string que contém o valor booleano da entrada de valor.

Compatível com modelos de substituição: Sim

`doubleValue`

(Opcional) Uma string que contém o valor duplo da entrada de valor.

Compatível com modelos de substituição: Sim

`integerValue`

(Opcional) Uma string que contém o valor inteiro da entrada de valor.

Compatível com modelos de substituição: Sim

`stringValue`

(Opcional) O valor da string da entrada do valor.

Compatível com modelos de substituição: Sim

`roleArn`

O ARN da função do IAM que concede AWS IoT permissão para enviar um valor de propriedade do ativo para. AWS IoT SiteWise Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

Exemplos

O exemplo de JSON a seguir define uma ação básica de SiteWise IoT em AWS IoT uma regra.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotSiteWise": {
                    "putAssetPropertyValueEntries": [
                        {
                            "propertyAlias": "/some/property/alias",
                            "propertyValues": [
                                {
                                    "timestamp": {
                                        "timeInSeconds": "${my.payload.timeInSeconds}"
                                    },
                                    "value": {
                                        "integerValue": "${my.payload.value}"
```

```
                                    }
                                }
                            ]
                        }
                    ],
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sitewise"
                }
            }
        ]
    }
}
```

O exemplo de JSON a seguir define uma ação de SiteWise IoT em AWS IoT uma regra. Esse exemplo usa o tópico como o alias da propriedade e a função `timestamp()`. Por exemplo, se você publicar dados em `/company/windfarm/3/turbine/7/rpm`, essa ação enviará os dados para a propriedade do ativo com um alias da propriedade igual ao tópico especificado.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM '/company/windfarm/+/turbine/+/+'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotSiteWise": {
                    "putAssetPropertyValueEntries": [
                        {
                            "propertyAlias": "${topic()}",
                            "propertyValues": [
                                {
                                    "timestamp": {
                                        "timeInSeconds": "${floor(timestamp() / 1E3)}",
                                        "offsetInNanos": "${(timestamp() % 1E3) * 1E6}"
                                    },
                                    "value": {
                                        "doubleValue": "${my.payload.value}"
                                    }
                                }
                            ]
                        }
                    ],
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sitewise"
                }
```

```
            }
        ]
    }
}
```

Consulte também

- O que é o AWS IoT SiteWise? no AWS IoT SiteWise Guia do usuário
- Ingestão de dados usando AWS IoT Core regras no Guia do AWS IoT SiteWise usuário
- Ingestão de dados AWS IoT SiteWise de AWS IoT itens no Guia do AWS IoT SiteWise usuário
- Solução de problemas de uma ação de AWS IoT SiteWise regra no Guia AWS IoT SiteWise do usuário

Firehose

A ação do Firehose (`firehose`) envia dados de uma mensagem MQTT para um stream do Amazon Data Firehose.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar a `firehose:PutRecord` operação. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT a execução dessa ação de regra.
- Se você usa o Firehose para enviar dados para um bucket do Amazon S3 e usa AWS KMS um cliente AWS KMS key gerenciado para criptografar dados em repouso no Amazon S3, o Firehose deve ter acesso ao seu bucket e permissão para usá-lo em nome do chamador. AWS KMS key Para obter mais informações, consulte Conceder ao Firehose acesso a um destino do Amazon S3 no Guia do desenvolvedor do Amazon Data Firehose.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`batchMode`

(Opcional) Especifica se o stream do Firehose deve ser entregue como um lote usando `PutRecordBatch`. O valor padrão é `false`.

Quando `batchMode` é `true` e a instrução SQL da regra é avaliada para uma matriz, cada elemento da matriz forma um registro na solicitação `PutRecordBatch`. A matriz resultante não pode ter mais de 500 registros.

Compatível com modelos de substituição: Não

`deliveryStreamName`

O fluxo do Firehose no qual gravar os dados da mensagem.

Suporta modelos de substituição: API e somente AWS CLI

`separator`

(Opcional) Um separador de caracteres usado para separar registros gravados no stream do Firehose. Se você omitir esse parâmetro, o fluxo não usará separador. Valores válidos: `,` (vírgula), `\t` (tab), `\n` (nova linha), `\r\n` (nova linha do Windows).

Compatível com modelos de substituição: Não

`roleArn`

O perfil do IAM que concede acesso ao stream do Firehose. Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

Exemplos

O exemplo de JSON a seguir define uma ação Firehose em AWS IoT uma regra.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
```

```
            {
                "firehose": {
                    "deliveryStreamName": "my_firehose_stream",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_firehose"
                }
            }
        ]
    }
}
```

O exemplo de JSON a seguir define uma ação Firehose com modelos de substituição em uma regra. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "firehose": {
                    "deliveryStreamName": "${topic()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_firehose"
                }
            }
        ]
    }
}
```

Consulte também

- O que é o Amazon Data Firehose? no Guia do desenvolvedor do Amazon Data Firehose

Kinesis Data Streams

A ação Kinesis Data Streams (`kinesis`) grava dados de uma mensagem MQTT no Amazon Kinesis Data Streams.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar a `kinesis:PutRecord` operação. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT a execução dessa ação de regra.
- Se você usa uma chave AWS KMS gerenciada pelo cliente AWS KMS key (chave KMS) para criptografar dados em repouso no Kinesis Data Streams, o serviço deve ter permissão para usá-la em nome do chamador. AWS KMS key Para obter mais informações, consulte Permissões de uso geradas pelo usuário AWS KMS keys no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`stream`

O fluxo de dados do Kinesis no qual os dados serão gravados.

Suporta modelos de substituição: API e somente AWS CLI

`partitionKey`

A chave de partição usada para determinar em qual fragmento os dados são gravados. A chave de partição geralmente é composta de uma expressão (por exemplo, `${topic()}` ou `${timestamp()}`).

Compatível com modelos de substituição: Sim

`roleArn`

O ARN da função do IAM que concede AWS IoT permissão para acessar o stream de dados do Kinesis. Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

Exemplos

O exemplo de JSON a seguir define uma ação do Kinesis Data Streams em uma regra. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "kinesis": {
                    "streamName": "my_kinesis_stream",
                    "partitionKey": "${topic()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_kinesis"
                }
            }
        ]
    }
}
```

O exemplo de JSON a seguir define uma ação do Kinesis com modelos de substituição em uma regra. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "kinesis": {
                    "streamName": "${topic()}",
                    "partitionKey": "${timestamp()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_kinesis"
                }
            }
        ]
    }
}
```

Consulte também

- O que são fluxos de dados do Amazon Kinesis? no Guia do desenvolvedor do Amazon Kinesis Data Streams

Lambda

Uma ação Lambda (`lambda`) invoca uma AWS Lambda função, transmitindo uma mensagem MQTT. AWS IoT invoca funções Lambda de forma assíncrona.

Você pode seguir um tutorial que mostra como criar e testar uma regra com uma ação do Lambda. Para obter mais informações, consulte Tutorial: Formatando uma notificação usando uma função AWS Lambda.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- AWS IoT Para invocar uma função Lambda, você deve configurar uma política que conceda `lambda:InvokeFunction` a permissão a. AWS IoT Você só pode invocar uma função do Lambda definida na Região da AWS mesma em que sua política do Lambda existe. As funções do Lambda usam políticas baseadas em recursos; por isso, você precisa anexar a política à própria função do Lambda.

  Use o AWS CLI comando a seguir para anexar uma política que conceda a `lambda:InvokeFunction` permissão. Nesse comando, substitua:
  - *`function_name`*com o nome da função Lambda. Você adiciona uma nova permissão para atualizar a política de recursos da função.
  - *`region`*com o Região da AWS da função.
  - *`account-id`*com o Conta da AWS número em que a regra está definida.
  - *`rule-name`*com o nome da AWS IoT regra para a qual você está definindo a ação Lambda.
  - *`unique_id`*com um identificador de declaração exclusivo.

  > **Important**
  >
  > Se você adicionar uma permissão para um AWS IoT principal sem fornecer o `source-arn` ou`source-account`, qualquer um Conta da AWS que crie uma regra com sua ação do Lambda poderá ativar regras para invocar sua função do Lambda. AWS IoT

  Para ter mais informações, consulte AWS Lambda permissões.

```
aws lambda add-permission \
```

```
    --function-name function_name \
    --region region \
    --principal iot.amazonaws.com \
    --source-arn arn:aws:iot:region:account-id:rule/rule_name \
    --source-account account-id
    --statement-id unique_id
    --action "lambda:InvokeFunction"
```

- Se você usar o AWS IoT console para criar uma regra para a ação da regra Lambda, a função Lambda será acionada automaticamente. Se você usar AWS CloudFormation em vez disso com o `AWS::IoT::TopicRule LambdaAction`, deverá adicionar um `AWS::lambda::Permission`recurso. O recurso então concede a você permissão para acionar a função do Lambda.

  O código a seguir mostra um exemplo de como adicionar esse recurso. Neste exemplo, substitua:

  - *`function_name`*com o nome da função Lambda.
  - *`region`*com o Região da AWS da função.
  - *`account-id`*com o Conta da AWS número em que a regra está definida.
  - *`rule-name`*com o nome da AWS IoT regra para a qual você está definindo a ação Lambda.

```
Type: AWS::Lambda::Permission
Properties:
  Action: lambda:InvokeFunction
  FunctionName: !Ref function_name
  Principal: "iot.amazonaws.com"
  SourceAccount: account-id
  SourceArn: arn:aws:iot:region:account-id:rule/rule_name
```

- Se você usa um AWS KMS cliente que conseguiu AWS KMS key criptografar dados em repouso no Lambda, o serviço deve ter permissão para usar AWS KMS key o em nome do chamador. Para obter mais informações, consulte Criptografia em repouso, no AWS Lambda Guia do Desenvolvedor.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`functionArn`

O ARN da função Lambda a ser invocada. AWS IoT deve ter permissão para invocar a função. Para obter mais informações, consulte Requisitos.

Se você não especificar uma versão ou um alias para sua função do Lambda, a versão mais recente da função será encerrada. Você pode especificar uma versão ou um alias se quiser encerrar uma versão específica da função do Lambda. Para especificar uma versão ou alias, anexe a versão ou pseudônimo ao ARN da função do Lambda.

```
arn:aws:lambda:us-east-2:123456789012:function:myLambdaFunction:someAlias
```

Para obter mais informações sobre versionamento e aliases, consulte AWS Lambda Versionamento e aliases da função.

Suporta modelos de substituição: API e somente AWS CLI

Exemplos

O exemplo de JSON a seguir define uma ação Lambda em AWS IoT uma regra.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "lambda": {
                    "functionArn": "arn:aws:lambda:us-
east-2:123456789012:function:myLambdaFunction"
                }
            }
        ]
    }
}
```

O exemplo de JSON a seguir define uma ação Lambda com modelos de substituição em uma regra. AWS IoT

```
{
```

```
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "lambda": {
                    "functionArn": "arn:aws:lambda:us-east-1:123456789012:function:
${topic()}"
                }
            }
        ]
    }
}
```

Consulte também

- O que AWS Lambdaé no Guia do AWS Lambda desenvolvedor
- Tutorial: Formatando uma notificação usando uma função AWS Lambda

Local

A ação Localização (`location`) envia seus dados de localização geográfica para o Amazon Location Service.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar a `geo:BatchUpdateDevicePosition` operação. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT a execução dessa ação de regra.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`deviceId`

O ID exclusivo do dispositivo que fornece os dados de localização. Para obter mais informações, consulte `DeviceId` da Referência de API do Amazon Location Service.

Compatível com modelos de substituição: Sim

`latitude`

Uma string que é avaliada como um valor duplo que representa a latitude da localização do dispositivo.

Compatível com modelos de substituição: Sim

`longitude`

Uma string que é avaliada como um valor duplo que representa a longitude da localização do dispositivo.

Compatível com modelos de substituição: Sim

`roleArn`

O perfil do IAM que permite acessar o domínio do Amazon Location Service. Para obter mais informações, consulte Requisitos.

`timestamp`

A hora em que os dados de localização foram amostrados. O valor padrão é a hora que a mensagem MQTT foi processada.

O valor `timestamp` consiste nos dois valores a seguir:

- `value`: Uma expressão que retorna um valor de horário epoch longo. Você pode usar a função the section called “time_to_epoch(String, String)” para criar um carimbo de data/hora válido a partir de um valor de data ou hora transmitido na carga da mensagem. Compatível com modelos de substituição: Sim.
- `unit`: (Opcional) A precisão do valor do carimbo de data/hora que resulta da expressão descrita em `value`. Valores válidos: `SECONDS` | `MILLISECONDS` | `MICROSECONDS` | `NANOSECONDS`. O padrão é `MILLISECONDS`. Suporta modelos de substituição: API e AWS CLI somente.

`trackerName`

O nome do recurso rastreador no Amazon Location no qual a localização é atualizada. Para acessar mais informações, consulte Rastreador no Guia do desenvolvedor do Amazon Location Service.

Suporta modelos de substituição: API e somente AWS CLI

Exemplos

O exemplo de JSON a seguir define uma ação de localização em uma AWS IoT regra.

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "location": {
     "roleArn": "arn:aws:iam::123454962127:role/service-role/ExampleRole",
     "trackerName": "MyTracker",
     "deviceId": "001",
     "sampleTime": {
      "value": "${timestamp()}",
      "unit": "MILLISECONDS"
     },
     "latitude": "-12.3456",
     "longitude": "65.4321"
    }
   }
  ]
 }
}
```

O exemplo de JSON a seguir define uma ação Location com modelos de substituição em uma AWS IoT regra.

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
```

```
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "location": {
     "roleArn": "arn:aws:iam::123456789012:role/service-role/ExampleRole",
     "trackerName": "${TrackerName}",
     "deviceId": "${DeviceID}",
     "timestamp": {
      "value": "${timestamp()}",
      "unit": "MILLISECONDS"
     },
     "latitude": "${get(position, 0)}",
     "longitude": "${get(position, 1)}"
    }
   }
  ]
 }
}
```

O exemplo de carga útil do MQTT a seguir mostra como os modelos de substituição no exemplo anterior acessam os dados. Você pode usar o comando get-device-position-history CLI para verificar se os dados da carga útil do MQTT são entregues no seu rastreador de localização.

```
{
 "TrackerName": "mytracker",
 "DeviceID": "001",
 "position": [
  "-12.3456",
  "65.4321"
 ]
}
```

```
aws location get-device-position-history --device-id 001 --tracker-name mytracker
```

```
{
 "DevicePositions": [
  {
   "DeviceId": "001",
   "Position": [
    -12.3456,
    65.4321
   ],
```

```
    "ReceivedTime": "2022-11-11T01:31:54.464000+00:00",
    "SampleTime": "2022-11-11T01:31:54.308000+00:00"
   }
  ]
}
```

Consulte também

- O que é o Amazon Location Service? no Guia do desenvolvedor do Amazon Location Service.

OpenSearch

A ação OpenSearch (`openSearch`) grava dados de mensagens MQTT em um domínio da Amazon OpenSearch Service. Em seguida, você pode usar ferramentas como OpenSearch painéis para consultar e visualizar dados no OpenSearch Service.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar a `es:ESHttpPut` operação. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT a execução dessa ação de regra.
- Se você usa um cliente gerenciado AWS KMS key para criptografar dados em repouso no OpenSearch Service, o serviço deve ter permissão para usar a chave KMS em nome do chamador. Para obter mais informações, consulte Criptografia de dados em repouso para o Amazon OpenSearch Service no Amazon OpenSearch Service Developer Guide.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`endpoint`

O endpoint do seu domínio do Amazon OpenSearch Service.

Suporta modelos de substituição: API e somente AWS CLI

`index`

O OpenSearch índice em que você deseja armazenar seus dados.

Compatível com modelos de substituição: Sim

`type`

O tipo de documento que você está armazenando.

> **Note**
>
> Para OpenSearch versões posteriores à 1.0, o valor do `type` parâmetro deve ser`_doc`. Para obter mais informações, consulte a documentação do OpenSearch .

Compatível com modelos de substituição: Sim

`id`

O identificador exclusivo de cada documento.

Compatível com modelos de substituição: Sim

`roleARN`

A função do IAM que permite acesso ao domínio do OpenSearch Serviço. Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

Limitações

A ação OpenSearch (`openSearch`) não pode ser usada para entregar dados aos clusters do VPC Elasticsearch.

Exemplos

O exemplo de JSON a seguir define uma OpenSearch ação em uma AWS IoT regra e como você pode especificar os campos para a `OpenSearch` ação. Para obter mais informações, consulte OpenSearchAction.

```
{
    "topicRulePayload": {
        "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "openSearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "my-index",
                    "type": "_doc",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_os"
                }
            }
        ]
    }
}
```

O exemplo de JSON a seguir define uma OpenSearch ação com modelos de substituição em uma AWS IoT regra.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "openSearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "${topic()}",
                    "type": "${type}",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_os"
                }
            }
        ]
    }
}
```

> **Note**
>
> O `type` campo substituído funciona para a OpenSearch versão 1.0. Para qualquer versão posterior à 1.0, o valor de `type` deve ser `_doc`.

Consulte também

O que é o Amazon OpenSearch Service? no Amazon OpenSearch Service Developer Guide

Nova publicação

Use a ação (`republish`) publicar novamente publica novamente uma mensagem do MQTT para outro tópico do MQTT.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar a `iot:Publish` operação. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT a execução dessa ação de regra.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`headers`

Informações de cabeçalho do MQTT versão 5.0.

Para obter mais informações, consulte RepublishAction e MqttHeaders na Referência da API do AWS .

`topic`

O tópico MQTT no qual publicar a mensagem novamente.

Para publicar novamente em um tópico reservado, que começa com $, use $$ em vez disso. Por exemplo, para publicar novamente em um tópico de sombra do dispositivo `$aws/things/`

`MyThing/shadow/update`, especifique o tópico como `$$aws/things/MyThing/shadow/update`.

> **Note**
>
> A nova publicação em tópicos de trabalho reservados não é compatível.
> AWS IoT Device Defender tópicos de reserva não oferecem suporte à publicação HTTP.

Compatível com modelos de substituição: Sim

`qos`

(Opcional) O nível Quality of Service (QoS - Qualidade de serviço) a ser usado ao republicar mensagens. Valores válidos: `0`, `1`. O valor padrão é `0`. Para obter mais informações sobre QoS MQTT, consulte MQTT.

Compatível com modelos de substituição: Não

`roleArn`

A função do IAM que permite AWS IoT publicar no tópico do MQTT. Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

Exemplos

O exemplo de JSON a seguir define uma ação de republicação em uma AWS IoT regra.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "republish": {
                    "topic": "another/topic",
                    "qos": 1,
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish"
                }
            }
```

```
        ]
    }
}
```

O exemplo de JSON a seguir define uma ação de republicação com modelos de substituição em uma regra. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "republish": {
                    "topic": "${topic()}/republish",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish"
                }
            }
        ]
    }
}
```

O exemplo de JSON a seguir define uma ação de republicação com uma `headers` AWS IoT regra.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "republish": {
                    "topic": "${topic()}/republish",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish",
                    "headers": {
                        "payloadFormatIndicator": "UTF8_DATA",
                        "contentType": "rule/contentType",
                        "correlationData": "cnVsZSBjb3JyZWxhdGlvbiBkYXRh",
                        "userProperties": [
                            {
                                "key": "ruleKey1",
                                "value": "ruleValue1"
```

```
                        },
                        {
                            "key": "ruleKey2",
                            "value": "ruleValue2"
                        }
                    ]
                }
            }
        }
    ]
  }
}
```

> **Note**
>
> O IP de origem original não será transmitido pela ação Publicar novamente.

S3

A ação S3 (`s3`) grava os dados de uma mensagem MQTT em um bucket do Amazon Simple Storage Service (Amazon S3).

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar a `s3:PutObject` operação. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT a execução dessa ação de regra.
- Se você usa um AWS KMS cliente gerenciado AWS KMS key para criptografar dados em repouso no Amazon S3, o serviço deve ter permissão para usar o AWS KMS key em nome do chamador. Para obter mais informações, consulte AWS gerenciado AWS KMS keys e gerenciado pelo cliente AWS KMS keys no Guia do desenvolvedor do Amazon Simple Storage Service.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`bucket`

O bucket do Amazon S3 no qual gravar dados.

Suporta modelos de substituição: API e somente AWS CLI

`cannedacl`

(Opcional) A ACL predefinida do Amazon S3 que controla o acesso ao objeto identificado pela chave do objeto. Para obter mais informações, inclusive os valores permitidos, consulte ACL predefinida.

Compatível com modelos de substituição: Não

`key`

O caminho para o arquivo em que os dados são gravados.

Considere um exemplo em que esse parâmetro está `${topic()}/${timestamp()}` e a regra recebe uma mensagem em que o tópico está `some/topic`. Se o carimbo de data/hora atual for `1460685389`, essa ação grava os dados em um arquivo chamado `1460685389` na `some/topic` pasta do bucket do S3.

> **Note**
>
> Se você usar uma chave estática, AWS IoT substituirá um único arquivo sempre que a regra for invocada. Recomendamos que você use o carimbo de data/hora da mensagem ou outro identificador exclusivo de mensagem para que um novo arquivo seja salvo no Amazon S3 para cada mensagem recebida.

Compatível com modelos de substituição: Sim

`roleArn`

O perfil do IAM que permite o acesso ao bucket do Amazon S3. Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

Exemplos

O exemplo de JSON a seguir define uma ação do S3 em uma AWS IoT regra.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "s3": {
                    "bucketName": "amzn-s3-demo-bucket",
                    "cannedacl": "public-read",
                    "key": "${topic()}/${timestamp()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3"
                }
            }
        ]
    }
}
```

Consulte também

- O que é o Amazon S3? no Guia do usuário do Amazon Simple Storage Service

IoT do Salesforce

Uma ação do IoT do Salesforce (`salesforce`) envia os dados da mensagem MQTT que acionou a regra para um stream de entrada do IoT do Salesforce.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`url`

O URL exposto pelo fluxo de entrada do IoT do Salesforce. O URL está disponível na plataforma do IoT do Salesforce ao criar um fluxo de entrada. Para obter mais informações, consulte a documentação do IoT do Salesforce.

Compatível com modelos de substituição: Não

`token`

O token usado para autenticar o acesso ao fluxo de entrada do IoT do Salesforce especificado. O token está disponível na plataforma do IoT do Salesforce ao criar um fluxo de entrada. Para obter mais informações, consulte a documentação do IoT do Salesforce.

Compatível com modelos de substituição: Não

Exemplos

O exemplo JSON a seguir define uma ação do IoT do Salesforce em uma regra AWS IoT .

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "salesforce": {
                    "token": "ABCDEFGHI123456789abcdefghi123456789",
                    "url": "https://ingestion-cluster-id.my-env.sfdcnow.com/streams/
stream-id/connection-id/my-event"
                }
            }
        ]
    }
}
```

SNS

A ação SNS (`sns`) envia os dados de uma mensagem MQTT como uma notificação push do Amazon Simple Notification Service (Amazon SNS)

Você pode seguir um tutorial que mostra como criar e testar uma regra com uma ação do SNS. Para obter mais informações, consulte Tutorial:r Como enviar uma notificação do Amazon SNS.

> **Note**
>
> A ação do SNS não oferece suporte a tópicos FIFO (First-In-First-Out) do Amazon SNS. Como o mecanismo de regras é um serviço totalmente distribuído, não há garantia da ordem das mensagens quando a ação do SNS é invocada.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar a `sns:Publish` operação. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT a execução dessa ação de regra.
- Se você usa um serviço gerenciado pelo AWS KMS cliente AWS KMS key para criptografar dados em repouso no Amazon SNS, o serviço deve ter permissão para usá-lo AWS KMS key em nome do chamador. Para obter mais informações, consulte Gerenciamento de chaves no Guia do desenvolvedor do Amazon Simple Notification Service.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`targetArn`

O tópico do SNS ou o dispositivo individual para o qual a notificação por push é enviada.

Suporta modelos de substituição: API e somente AWS CLI

`messageFormat`

(Opcional) O formato da mensagem. O Amazon SNS usa essa configuração para determinar se a carga deve ser analisada e se partes relevantes específicas da plataforma da carga devem ser extraídas. Valores válidos: `JSON`, `RAW`. Padronizado como `RAW`.

Compatível com modelos de substituição: Não

`roleArn`

O perfil do IAM; que permite o acesso ao SNS. Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

Exemplos

O exemplo de JSON a seguir define uma ação do SNS em uma AWS IoT regra.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-east-2:123456789012:my_sns_topic",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sns"
                }
            }
        ]
    }
}
```

O exemplo de JSON a seguir define uma ação do SNS com modelos de substituição em uma regra. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-east-1:123456789012:${topic()}",
                    "messageFormat": "JSON",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sns"
                }
            }
        ]
    }
}
```

Consulte também

- O que é o Amazon Simple Notification Service? no Guia do desenvolvedor do Amazon Simple Notification Service
- Tutorial:r Como enviar uma notificação do Amazon SNS

SQS

A ação SQS (`sqs`) envia dados de uma mensagem MQTT para uma fila do Amazon Simple Queue Service (Amazon SQS).

> **Note**
>
> A ação do SQS não é compatível com as filas do Amazon SQS FIFO (First-In-First-Out). Como o mecanismo de regras é um serviço totalmente distribuído, não há garantia da ordem das mensagens quando a ação do SQS é acionada.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar a `sqs:SendMessage` operação. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT a execução dessa ação de regra.
- Se você usa um AWS KMS cliente gerenciado AWS KMS key para criptografar dados em repouso no Amazon SQS, o serviço deve ter permissão para usar AWS KMS key o em nome do chamador. Para obter mais informações, consulte Gerenciamento de chaves no Guia do desenvolvedor do Amazon Simple Queue Service.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`queueUrl`

O URL da fila do Amazon SQS na qual os dados serão gravados. A região nesse URL não precisa ser a mesma da sua Região da AWS AWS IoT regra.

> **Note**
>
> Pode haver cobranças adicionais pela transferência de dados entre Regiões da AWS usando a ação de regra SQS. Para obter mais informações, consulte Preços do Amazon SQS.

Suporta modelos de substituição: API e somente AWS CLI

`useBase64`

Defina esse parâmetro como `true` para configurar a ação da regra para codificar em base64 os dados da mensagem antes de gravar os dados na fila do Amazon SQS. Padronizado como `false`.

Compatível com modelos de substituição: Não

`roleArn`

O perfil do IAM que permite o acesso à fila do Amazon SQS. Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

Exemplos

O exemplo de JSON a seguir define uma ação SQS em uma AWS IoT regra.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sqs": {
                    "queueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/
my_sqs_queue",
```

```
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sqs"
                }
            }
        ]
    }
}
```

O exemplo de JSON a seguir define uma ação SQS com modelos de substituição em uma regra. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sqs": {
                    "queueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/
${topic()}",
                    "useBase64": true,
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sqs"
                }
            }
        ]
    }
}
```

Consulte também

- O que é o Amazon Simple Queue Service? no Guia do desenvolvedor do Amazon Simple Queue Service

Step Functions

A ação Step Functions (`stepFunctions`) inicia uma máquina de AWS Step Functions estado.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar a `states:StartExecution` operação. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher ou criar uma função para permitir AWS IoT a execução dessa ação de regra.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`stateMachineName`

O nome da máquina de estados do Step Functions a ser iniciada.

Suporta modelos de substituição: API e somente AWS CLI

`executionNamePrefix`

(Opcional) Um nome será atribuído à execução da máquina de estados composto por esse prefixo seguido por um UUID. O Step Functions cria automaticamente um nome exclusivo para cada execução da máquina de estados, caso um nome não seja fornecido.

Compatível com modelos de substituição: Sim

`roleArn`

O ARN da função que concede AWS IoT permissão para iniciar a máquina de estado. Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

Exemplos

O exemplo de JSON a seguir define uma ação Step Functions em uma AWS IoT regra.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
```

```
                "stepFunctions": {
                    "stateMachineName": "myStateMachine",
                    "executionNamePrefix": "myExecution",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_step_functions"
                }
            }
        ]
    }
}
```

Consulte também

- O que AWS Step Functionsé no Guia do AWS Step Functions desenvolvedor

Timestream

A ação da regra Timestream grava atributos (medidas) de uma mensagem MQTT em uma tabela do Amazon Timestream. Para obter mais informações sobre o Amazon Timestream, consulte O que é o Amazon Timestream?.

> **Note**
>
> O Amazon Timestream não está disponível em todos os s. Região da AWS Se o Amazon Timestream não estiver disponível em sua região, ele não aparecerá na lista de ações de regras.

Os atributos que essa regra armazena no banco de dados do Timestream são aqueles que resultam da instrução de consulta da regra. O valor de cada atributo no resultado da instrução de consulta é analisado para inferir seu tipo de dados (como em uma ação the section called “Dínamo DBv2”). O valor de cada atributo é gravado em seu próprio registro na tabela do Timestream. Para especificar ou alterar o tipo de dados de um atributo, use a função `cast()` na instrução de consulta. Para obter mais informações sobre o conteúdo de cada registro do Timestream, consulte. the section called “Conteúdo do registro do Timestream”

> **Note**
>
> Com o SQL V2 (23-03-2016), valores numéricos que são números inteiros, como `10.0`, são convertidos em sua representação inteira (`10`). Convertê-los explicitamente em um valor

`Decimal`, como usar a função cast(), não impede esse comportamento — o resultado ainda é um valor `Integer`. Isso pode causar erros de incompatibilidade de tipos que impedem que os dados sejam registrados no banco de dados do Timestream. Para processar valores numéricos inteiros como valores `Decimal`, use o SQL V1 (08/10/2015) para a instrução de consulta de regra.

Note

O número máximo de valores que uma ação de regra do Timestream pode gravar em uma tabela do Amazon Timestream é 100. Para obter mais informações, consulte a Referência da cota do Amazon Timestream.

Requisitos

Esta ação de regra tem os seguintes requisitos:

- Uma função do IAM que AWS IoT você pode assumir para realizar as `timestream:WriteRecords` operações `timestream:DescribeEndpoints` e. Para obter mais informações, consulte Conceder a uma AWS IoT regra o acesso que ela exige.

  No AWS IoT console, você pode escolher, atualizar ou criar uma função para permitir AWS IoT a execução dessa ação de regra.
- Se você usa um cliente AWS KMS para criptografar dados em repouso no Timestream, o serviço deve ter permissão para usar o AWS KMS key em nome do chamador. Para obter mais informações, consulte Como AWS os serviços usam o AWS KMS.

Parâmetros

Ao criar uma AWS IoT regra com essa ação, você deve especificar as seguintes informações:

`databaseName`

O nome de um banco de dados do Amazon Timestream que tem a tabela para receber os registros que essa ação cria. Consulte também **`tableName`**.

Suporta modelos de substituição: API e somente AWS CLI

`dimensions`

Atributos de metadados das séries temporais que são gravados em cada registro de medida. Por exemplo, o nome e a zona de disponibilidade de uma instância do EC2 ou o nome do fabricante de uma turbina eólica são dimensões.

`name`

O nome da dimensão de metadados. Este é o nome da coluna no registro da tabela do banco de dados.

As dimensões não podem ser nomeadas:`measure_name`, `measure_value`, ou `time`. Esses nomes são reservados. Os nomes das dimensões não podem começar com `ts_` ou `measure_value` e não podem conter o caractere de dois pontos (`:`).

Compatível com modelos de substituição: Não

`value`

O valor a ser gravado nesta coluna do registro do banco de dados.

Compatível com modelos de substituição: Sim

`roleArn`

O nome do recurso da Amazon (ARN) da função que concede permissão AWS IoT para gravar na tabela do banco de dados do Timestream. Para obter mais informações, consulte Requisitos.

Compatível com modelos de substituição: Não

`tableName`

O nome da tabela de banco de dados na qual gravar os registros de medida. Consulte também **`databaseName`**.

Suporta modelos de substituição: API e somente AWS CLI

`timestamp`

O valor a ser usado para o carimbo e data/hora da entrada. Se estiver em branco, a hora em que a entrada foi processada será usada.

`unit`

A precisão do valor do carimbo de data/hora que resulta da expressão descrita em `value`.

Valores válidos: `SECONDS` | `MILLISECONDS` | `MICROSECONDS` | `NANOSECONDS`. O padrão é `MILLISECONDS`.

`value`

Uma expressão que retorna um valor de tempo epoch longo.

Você pode usar a função the section called “time_to_epoch(String, String)” para criar um carimbo de data/hora válido a partir de um valor de data ou hora transmitido na carga da mensagem.

Conteúdo do registro do Timestream

Os dados gravados na tabela do Amazon Timestream por essa ação incluem um carimbo de data e hora, metadados da ação da regra Timestream e o resultado da instrução de consulta da regra.

Para cada atributo (medida) no resultado da instrução de consulta, essa ação de regra grava um registro na tabela Timestream especificada com essas colunas.

Nome da coluna	Tipo de atributo	Valor	Comentários
`dimension-name`	DIMENSÃO	O valor especificado na entrada de ação da regra Timestream.	Cada Dimensão especificada na entrada de ação da regra cria uma coluna no banco de dados do Timestream com o nome da dimensão.
nome_medida	NOME_MEDIDA	O nome do atributo	O nome do atributo no resultado da instrução de consulta cujo valor é especificado na coluna `measure_value::` *`data-type`* .
valor_de_medida:: *`data-type`*	VALOR_MEDIDA	O nome do atributo no resultado da instrução	O valor é interpretado* e convertido

Nome da coluna	Tipo de atributo	Valor	Comentários
		de consulta. O nome do atributo está na coluna `measure_name` .	como a combinação mais adequada de:`bigint`, `boolean`, `double`, ou `varchar`. O Amazon Timestream cria uma coluna separada para cada tipo de dados. O valor na mensagem pode ser convertido em outro tipo de dados usando a função `cast()` na instrução de consulta da regra.
tempo	TIMESTAMP	A data e a hora do registro no banco de dados.	Esse valor é atribuído pelo mecanismo de regras ou pela propriedade `timestamp` , se estiver definida.

* O valor do atributo lido da carga da mensagem é interpretado da seguinte forma. Consulte a the section called “Exemplos” para obter uma ilustração de cada um desses casos.

- Um valor sem aspas de `true` ou `false` é interpretado como um tipo `boolean`.
- Um numérico decimal é interpretado como um tipo `double`.
- Um valor numérico sem ponto decimal é interpretado como um tipo `bigint`.
- Uma string entre aspas é interpretada como um tipo `varchar`.
- Objetos e valores de matriz são convertidos em strings JSON e armazenados como um tipo `varchar`.

Exemplos

O exemplo de JSON a seguir define uma ação de regra Timestream com um modelo de substituição em uma regra. AWS IoT

```
{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'iot/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "timestream": {
          "roleArn": "arn:aws:iam::123456789012:role/aws_iot_timestream",
          "tableName": "devices_metrics",
          "dimensions": [
            {
              "name": "device_id",
              "value": "${clientId()}"
            },
            {
              "name": "device_firmware_sku",
              "value": "My Static Metadata"
            }
          ],
          "databaseName": "record_devices"
        }
      }
    ]
  }
}
```

Usar a ação de regra de tópico Timestream definida no exemplo anterior com a carga útil da mensagem a seguir resulta nos registros do Amazon Timestream escritos na tabela a seguir.

```
{
  "boolean_value": true,
  "integer_value": 123456789012,
  "double_value": 123.456789012,
  "string_value": "String value",
  "boolean_value_as_string": "true",
  "integer_value_as_string": "123456789012",
```

```
  "double_value_as_string": "123.456789012",
  "array_of_integers": [23,36,56,72],
  "array of strings": ["red", "green","blue"],
  "complex_value": {
    "simple_element": 42,
    "array_of_integers": [23,36,56,72],
    "array of strings": ["red", "green","blue"]
  }
}
```

A tabela a seguir exibe as colunas e os registros do banco de dados que usam a ação de regra de tópico especificada para processar a carga útil da mensagem anterior criada. As colunas `device_firmware_sku` e `device_id` são as DIMENSÕES definidas na ação da regra de tópico. A ação de regra de tópico Timestream cria a coluna `time` e as colunas `measure_name` e `measure_value::*`, que ela preenche com os valores do resultado da instrução de consulta da ação de regra de tópico.

device_firmware_sku	id_dispositivo	nome_medida	valor_medida::bigint	valor_medida::varchar	valor_medida::duplo	valor_medida::booleano	tempo
Meus metadados estáticos	console EXAMPLE7 8 iot-159 -0	valor_complexo	-	{"elemento_simples":42,"matriz_de_inteiros":[23,36,56,72],"matriz de strings":["vermelho","verde","azul"]}	-	-	2020-08-26 22:42:16.423000000
Meus metadados estáticos	console EXAMPLE7 8 iot-159 -0	valor_inteiro_como_string	-	123456789012	-	-	2020-08-26 22:42:16.423000000

device_firmware_sku	id_dispositivo	nome_medida	valor_medida::bigint	valor_medida::varchar	valor_medida::duplo	valor_medida::booleano	tempo
Meus metadados estáticos	console EXAMPLE78 iot-159-0	valor_booleano	-	-	-	VERDADEIRO	2020-08-26 22:42:16.423000000
Meus metadados estáticos	console EXAMPLE78 iot-159-0	valor_inteiro	123456789012	-	-	-	2020-08-26 22:42:16.423000000
Meus metadados estáticos	console EXAMPLE78 iot-159-0	valor_string	-	Valor da string	-	-	2020-08-26 22:42:16.423000000
Meus metadados estáticos	console EXAMPLE78 iot-159-0	matriz_de_inteiros	-	[23,36,56,72]	-	-	2020-08-26 22:42:16.423000000
Meus metadados estáticos	console EXAMPLE78 iot-159-0	matriz de strings	-	["vermelho","verde","azul"]	-	-	2020-08-26 22:42:16.423000000
Meus metadados estáticos	console EXAMPLE78 iot-159-0	valor_booleano_como_string	-	VERDADEIRO	-	-	2020-08-26 22:42:16.423000000
Meus metadados estáticos	console EXAMPLE78 iot-159-0	valor_duplo	-	-	123.456789012	-	2020-08-26 22:42:16.423000000

device_firmware_sku	id_dispositivo	nome_medida	valor_medida::bigint	valor_medida::varchar	valor_medida::duplo	valor_medida::booleano	tempo
Meus metadados estáticos	consoleEXAMPLE78 iot-159-0	valor_duplo_como_string	-	123.45679	-	-	2020-08-26 22:42:16.423000000

Solucionar problemas de uma regra

Se você tiver algum problema com suas regras, recomendamos que você ative o CloudWatch Logs. Você pode analisar seus logs para determinar se o problema é autorização ou se, por exemplo, uma condição de cláusula WHERE não foi correspondida. Para obter mais informações, consulte Como configurar CloudWatch registros.

Acessando recursos entre contas usando regras AWS IoT

Você pode configurar AWS IoT regras para acesso entre contas para que os dados ingeridos nos tópicos do MQTT de uma conta possam ser roteados para os AWS serviços, como Amazon SQS e Lambda, de outra conta. A seguir, explicamos como configurar AWS IoT regras para a ingestão de dados entre contas, de um tópico do MQTT em uma conta até um destino em outra conta.

As regras entre contas podem ser configuradas usando permissões baseadas em recursos no recurso de destino. Portanto, somente destinos que oferecem suporte a permissões baseadas em recursos podem ser habilitados para o acesso entre contas com regras. AWS IoT Os destinos compatíveis incluem Amazon SQS, Amazon SNS, Amazon S3 e AWS Lambda.

> **Note**
>
> Para os destinos compatíveis, exceto o Amazon SQS, você deve definir a regra da mesma forma Região da AWS que o recurso de outro serviço para que a ação da regra possa interagir com esse recurso. Para obter mais informações sobre ações de AWS IoT regras, consulte ações de AWS IoT regras. Para obter mais informações sobre a ação de SQS da regra, consulte ???.

Pré-requisitos

- Familiaridade com as AWS IoT regras
- Uma compreensão dos usuários, funções e permissões baseadas em recursos do IAM
- Tendo AWS CLI instalado

Configuração entre contas para o Amazon SQS

Cenário: a conta A envia dados de uma mensagem MQTT para a fila do Amazon SQS da conta B.

Conta da AWS	Conta referida como	Description
1111-1111-1111	Conta A	Ação da regra: `sqs:SendMessage`
2222-2222-2222	Conta B	Fila do Amazon SQS • ARN: *arn:aws:sqs:region:2222-2222-2222:ExampleQueue* • URL: *https://sqs.region.amazonaws.com/2222-2222-2222/ExampleQueue*

> **Note**
>
> Sua fila de destino do Amazon SQS não precisa estar na mesma fila da sua Região da AWS regra.AWS IoT Para obter mais informações sobre a ação de SQS da regra, consulte ???.

Realizar as tarefas da Conta A

> **Observação**
>
> Para executar os comandos a seguir, seu usuário do IAM deve ter permissões para `iot:CreateTopicRule` usar o nome do recurso da Amazon (ARN) da regra como um recurso e permissões para `iam:PassRole` agir com um recurso como o ARN da função.

1. Configure AWS CLI usando o usuário do IAM da conta A.
2. Crie uma função do IAM que confie no mecanismo de AWS IoT regras e anexe uma política que permita acesso à fila do Amazon SQS da conta B. Veja exemplos de comandos e documentos de política em Concedendo AWS IoT o acesso necessário.
3. Para criar uma regra anexada a um tópico, execute o create-topic-rule comando.

```
aws iot create-topic-rule --rule-name myRule --topic-rule-payload file://./my-rule.json
```

Veja a seguir um exemplo de arquivo de carga útil com uma regra que insere todas as mensagens enviadas ao tópico `iot/test` na fila especificada do Amazon SQS. A instrução SQL filtra as mensagens e o ARN da função concede permissões AWS IoT para adicionar a mensagem à fila do Amazon SQS.

```
{
 "sql": "SELECT * FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "sqs": {
    "queueUrl": "https://sqs.region.amazonaws.com/2222-2222-2222/ExampleQueue",
    "roleArn": "arn:aws:iam::1111-1111-1111:role/my-iot-role",
    "useBase64": false
   }
  }
 ]
}
```

Para obter mais informações sobre como definir uma ação do Amazon SQS em uma AWS IoT regra, consulte ações de AWS IoT regras - Amazon SQS.

Realizar as tarefas da Conta B

1. Configure AWS CLI usando o usuário do IAM da conta B.
2. Para conceder permissões para o recurso de fila do Amazon SQS para a conta A, execute o comando adicionar-permissão.

```
aws sqs add-permission --queue-url https://sqs.region.amazonaws.com/2222-2222-2222/
ExampleQueue --label SendMessagesToMyQueue --aws-account-ids 1111-1111-1111 --
actions SendMessage
```

Configuração entre contas para o Amazon SNS

Cenário: a conta A envia dados de uma mensagem MQTT para um tópico do Amazon SNS da conta B.

Conta da AWS	Conta referida como	Description
1111-1111-1111	Conta A	Ação da regra: `sns:Publish`
2222-2222-2222	Conta B	ARN do tópico do Amazon SNS: *arn:aws:sns:region:2222-2222-2222:ExampleTopic*

Realizar as tarefas da Conta A

> **Observações**
>
> Para executar os comandos a seguir, seu usuário do IAM deve ter permissões para `iot:CreateTopicRule` usar o ARN da regra como recurso e permissões para a ação `iam:PassRole` com um recurso como ARN da função.

1. Configure AWS CLI usando o usuário do IAM da conta A.
2. Crie uma função do IAM que confie no mecanismo de AWS IoT regras e anexe uma política que permita acesso ao tópico do Amazon SNS da conta B. Por exemplo, comandos e documentos de política, consulte Concedendo AWS IoT o acesso necessário.
3. Para criar uma regra anexada a um tópico, execute o create-topic-rule comando.

```
aws iot create-topic-rule --rule-name myRule --topic-rule-payload file://./my-
rule.json
```

Veja a seguir um exemplo de arquivo de carga útil com uma regra que insere todas as mensagens enviadas ao tópico `iot/test` no tópico especificado do Amazon SNS. A instrução SQL filtra as mensagens e o ARN da função concede permissões AWS IoT para enviar a mensagem ao tópico do Amazon SNS.

```
{
 "sql": "SELECT * FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "sns": {
    "targetArn": "arn:aws:sns:region:2222-2222-2222:ExampleTopic",
    "roleArn": "arn:aws:iam::1111-1111-1111:role/my-iot-role"
   }
  }
 ]
}
```

Para obter mais informações sobre como definir uma ação do Amazon SNS em uma AWS IoT regra, consulte Ações de AWS IoT regras - Amazon SNS.

Realizar as tarefas da Conta B

1. Configure AWS CLI usando o usuário do IAM da conta B.
2. Para conceder permissão no recurso de tópico do Amazon SNS à conta A, execute o comando adicionar-permissão.

```
aws sns add-permission --topic-arn arn:aws:sns:region:2222-2222-2222:ExampleTopic
 --label Publish-Permission --aws-account-id 1111-1111-1111 --action-name Publish
```

Configuração entre contas para o Amazon S3

Cenário: a conta A envia dados de uma mensagem MQTT para um bucket do Amazon S3 da conta B.

Conta da AWS	Conta referida como	Description
1111-1111-1111	Conta A	Ação da regra: `s3:PutObject`
2222-2222-2222	Conta B	Bucket do Amazon S3: *arn:aws:s3:::amzn-s3-demo-bucket*

Realizar as tarefas da Conta A

> **Observação**
>
> Para executar os comandos a seguir, seu usuário do IAM deve ter permissões para `iot:CreateTopicRule` usar o ARN da regra como um recurso e permissões para `iam:PassRole` agir com um recurso como ARN da função.

1. Configure AWS CLI usando o usuário do IAM da conta A.
2. Crie uma função do IAM que confie no mecanismo de AWS IoT regras e anexe uma política que permita acesso ao bucket Amazon S3 da conta B. Por exemplo, comandos e documentos de política, consulte Concedendo AWS IoT o acesso necessário.
3. Para criar uma regra anexada ao seu bucket do S3 de destino, execute o create-topic-rule comando.

   ```
   aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file://./my-rule.json
   ```

   Veja a seguir um exemplo de arquivo de carga útil com uma regra que insere todas as mensagens enviadas ao tópico `iot/test` no bucket do Amazon S3 especificado. A instrução SQL filtra as mensagens e o ARN da função concede permissões AWS IoT para adicionar a mensagem ao bucket do Amazon S3.

   ```
   {
    "sql": "SELECT * FROM 'iot/test'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
   ```

```
  {
   "s3": {
    "bucketName": "amzn-s3-demo-bucket",
    "key": "${topic()}/${timestamp()}",
    "roleArn": "arn:aws:iam::1111-1111-1111:role/my-iot-role"
   }
  }
 ]
}
```

Para obter mais informações sobre como definir uma ação do Amazon S3 em uma AWS IoT regra, consulte Ações de AWS IoT regras - Amazon S3.

Realizar as tarefas da Conta B

1. Configure AWS CLI usando o usuário do IAM da conta B.
2. Crie uma política de bucket que confie na entidade principal da conta A.

   Veja a seguir um exemplo de arquivo de carga útil que define uma política de bucket que confia na entidade principal de outra conta.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AddCannedAcl",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:root"
                ]
        },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

   Para obter mais informações, consulte Exemplos de políticas de bucket.

3. Para anexar a política do bucket ao bucket especificado, execute o put-bucket-policy comando.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://./amzn-s3-demo-bucket-policy.json
```

4. Para fazer o acesso entre contas funcionar, certifique-se de ter as configurações corretas de Bloquear todo o acesso público. Para obter mais informações, consulte Práticas recomendadas de segurança para o Amazon S3.

Configuração de várias contas para AWS Lambda

Cenário: a conta A invoca uma AWS Lambda função da conta B, transmitindo uma mensagem MQTT.

Conta da AWS	Conta referida como	Description
1111-1111-1111	Conta A	Ação da regra: `lambda:InvokeFunction`
2222-2222-2222	Conta B	ARN da função do Lambda: *arn:aws:lambda:region:2222-2222-2222:function:example-function*

Realizar as tarefas da Conta A

> **Observações**
>
> Para executar os comandos a seguir, seu usuário do IAM deve ter permissões para `iot:CreateTopicRule` usar o ARN da regra como um recurso e permissões para `iam:PassRole` agir com o recurso como ARN da função.

1. Configure AWS CLI usando o usuário do IAM da conta A.
2. Execute o create-topic-rule comando para criar uma regra que defina o acesso entre contas à função Lambda da conta B.

```
aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file://./my-rule.json
```

Veja a seguir um exemplo de arquivo de carga útil com uma regra que insere todas as mensagens enviadas para o tópico `iot/test` em uma função do Lambda especificada. A instrução SQL filtra as mensagens e o ARN da função concede permissão AWS IoT para transmitir os dados para a função do Lambda.

```
{
 "sql": "SELECT * FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "lambda": {
    "functionArn": "arn:aws:lambda:region:2222-2222-2222:function:example-function"
   }
  }
 ]
}
```

Para obter mais informações sobre como definir uma AWS Lambda ação em uma AWS IoT regra, leia Ações de AWS IoT regras - Lambda.

Realizar as tarefas da Conta B

1. Configure AWS CLI usando o usuário do IAM da conta B.
2. Execute o comando add-permission do Lambda para dar permissão às AWS IoT regras para ativar a função Lambda. Para executar o comando a seguir, seu usuário do IAM deve ter permissão para `lambda:AddPermission` agir.

```
aws lambda add-permission --function-name example-function --region us-east-1 --
principal iot.amazonaws.com --source-arn arn:aws:iot:region:1111-1111-1111:rule/
example-rule --source-account 1111-1111-1111 --statement-id "unique_id" --action
 "lambda:InvokeFunction"
```

Opções:

--entidade principal

Esse campo dá permissão para AWS IoT (representado por`iot.amazonaws.com`) chamar a função Lambda.

--arn de origem

Este campo confirma que apenas `arn:aws:iot:region:1111-1111-1111:rule/example-rule` nos acionadores AWS IoT esta função do Lambda e nenhuma outra regra na mesma conta ou em conta diferente pode ativar esta função do Lambda.

--conta de origem

Esse campo confirma que AWS IoT ativa essa função Lambda somente em nome da conta. `1111-1111-1111`

> **Observações**
>
> Se você vir uma mensagem de erro "A regra não foi encontrada" no console da sua AWS Lambda função em Configuração, ignore a mensagem de erro e prossiga para testar a conexão.

Tratamento de erros (ação de erro)

Quando AWS IoT recebe uma mensagem de um dispositivo, o mecanismo de regras verifica se a mensagem corresponde a uma regra. Nesse caso, a instrução de consulta da regra é avaliada e as ações da regra são ativadas, transmitindo o resultado da instrução de consulta.

Se ocorrer um problema ao chamar uma ação, o mecanismo de regras ativará uma ação de erro, se uma ação estiver especificada para a regra. Isso pode acontecer quando:

- Uma regra não tem permissão para acessar um bucket do Amazon S3.
- Um erro do usuário faz com que o throughput provisionado do DynamoDB seja excedido.

> **Note**
>
> O tratamento de erros abordado neste tópico é para ações de regras. Para depurar problemas de SQL, incluindo funções externas, você pode configurar o AWS IoT registro em log. Para obter mais informações, consulte ???.

Formato da mensagem de ação de erro

Uma única mensagem é gerada por regra e mensagem. Por exemplo, se houver uma falha em duas ações de regra na mesma regra, a ação de erro receberá uma mensagem contendo os dois erros.

O resultado é uma mensagem que parece com o exemplo a seguir.

```
{
  "ruleName": "TestAction",
  "topic": "testme/action",
  "cloudwatchTraceId": "7e146a2c-95b5-6caf-98b9-50e3969734c7",
  "clientId": "iotconsole-1511213971966-0",
  "base64OriginalPayload":
 "ewogICJtZXNzYWdlIjogIkhlbGxvIHZyb20gQVdTIElvVCBjb25zb2xlIgp9",
  "failures": [
    {
      "failedAction": "S3Action",
      "failedResource": "us-east-1-s3-verify-user",
      "errorMessage": "Failed to put S3 object. The error received was The
 specified bucket does not exist (Service: Amazon S3; Status Code: 404; Error
 Code: NoSuchBucket; Request ID: 9DF5416B9B47B9AF; S3 Extended Request ID:
 yMah1cwPhqTH267QLPhTKeVPKJB8BO5ndBHzOmWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y=).
 Message arrived on: error/action, Action: s3, Bucket: us-
east-1-s3-verify-user, Key: \"aaa\". Value of x-amz-id-2:
 yMah1cwPhqTH267QLPhTKeVPKJB8BO5ndBHzOmWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y="
    }
  ]
}
```

ruleName

O nome da regra que acionou a ação de erro.

tópico

O tópico no qual a mensagem original foi recebida.

cloudwatchTraceId

Uma identidade exclusiva referente ao login de erro CloudWatch.

clientId

O ID do cliente do publicador da mensagem.

base 64 OriginalPayload

A carga da mensagem original codificada em Base64.

falhas

failedAction

O nome da ação que não foi concluída (por exemplo "S3Action").

failedResource

O nome do recurso (por exemplo, o nome de um bucket do S3).

errorMessage

A descrição e a explicação do erro.

Exemplo de ação de erro

Este é um exemplo de uma regra com uma ação de erro adicional. A seguinte regra tem uma ação que grava dados de mensagens em uma tabela do DynamoDB e uma ação de erro que grava dados em um bucket do Amazon S3:

```
{
    "sql" : "SELECT * FROM ..."
    "actions" : [{
        "dynamoDB" : {
            "table" : "PoorlyConfiguredTable",
            "hashKeyField" : "AConstantString",
            "hashKeyValue" : "AHashKey"}}
    ],
    "errorAction" : {
```

```
        "s3" : {
            "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",
            "bucketName" : "message-processing-errors",
            "key" : "${replace(topic(), '/', '-') + '-' + timestamp() + '-' +
 newuuid()}"
        }
    }
}
```

Você pode usar qualquer função ou modelo de substituição na instrução SQL de uma ação de erro, incluindo as funções externas: `aws_lambda()get_dynamodb()`, `get_registry_data()`, `get_thing_shadow()`, `get_secret()`, `machinelearning_predict()`, e. `decode()` Se uma ação de erro exigir a chamada de uma função externa, a invocação da ação de erro poderá resultar em cobrança adicional para a função externa.

Para obter mais informações sobre regras e como especificar uma ação de erro, consulte Criando uma AWS IoT regra.

Para obter mais informações sobre como usar CloudWatch para monitorar o sucesso ou o fracasso das regras, consulteAWS IoT métricas e dimensões.

Reduzir custos do sistema de mensagens com Ingestão básica

Você pode usar a Ingestão básica para enviar dados de dispositivos com segurança para os Serviços da AWS compatíveis com AWS IoT ações de regras sem incorrer em custos de mensagens. A Ingestão básica otimiza o fluxo de dados removendo o agente de mensagens de publicação/assinatura do caminho da ingestão.

A Ingestão básica pode enviar mensagens de seus dispositivos ou aplicativos. As mensagens têm nomes de tópicos que começam com `$aws/rules/`*`rule_name`* para os três primeiros níveis, onde *`rule_name`* é o nome da regra de AWS IoT que você deseja invocar.

Você pode continuar a usar uma regra existente com a Ingestão básica adicionando o prefixo de Ingestão básica (`$aws/rules/`*`rule_name`*) ao tópico da mensagem com o qual você normalmente invoca a regra. Por exemplo, se você tiver uma regra chamada `BuildingManager` que é invocada por mensagens com tópicos, como `Buildings/Building5/Floor2/Room201/Lights` (`"sql": "SELECT * FROM 'Buildings/#'"`), você poderá invocar a mesma regra com a Ingestão básica enviando uma mensagem com o tópico `$aws/rules/BuildingManager/Buildings/Building5/Floor2/Room201/Lights`.

> **Note**
>
> - Seus dispositivos e regras não podem se inscrever em tópicos reservados da Ingestão básica. Por exemplo, as métricas AWS IoT Device Defender e as métricas `num-messages-received` não são emitidas, pois não permitem a assinatura de tópicos. Para obter mais informações, consulte Tópicos reservados.
> - Se você precisar de um agente de publicação/assinatura para distribuir mensagens a vários assinantes (por exemplo, para entregar mensagens para outros dispositivos e para o mecanismo de regras), deverá continuar a usar o agente de mensagens do AWS IoT para lidar com a distribuição da mensagem. No entanto, certifique-se de publicar suas mensagens em tópicos que não sejam tópicos da Ingestão básica.

Usar a Ingestão básica

Antes de usar a Ingestão básica, verifique se seu dispositivo ou aplicativo está usando uma política com permissões de publicação em `$aws/rules/*`. Você também pode especificar permissões para regras individuais com `$aws/rules/rule_name/*` na política. Caso contrário, seus dispositivos e aplicativos poderão continuar a usar suas conexões existentes com o AWS IoT Core.

Quando a mensagem atinge o mecanismo de regras, não há diferença na implementação ou no tratamento de erros entre as regras invocadas pela Ingestão básica e as invocadas pelas assinaturas de agentes de mensagens.

Você pode criar regras para uso com a Ingestão básica. Lembre-se do seguinte:

- O prefixo inicial de um tópico de Ingestão básica (`$aws/rules/rule_name`) não está disponível na função topic(Decimal).
- Se você definir uma regra invocada apenas com a Ingestão básica, a cláusula `FROM` será opcional no campo `sql` da definição de `rule`. Ela ainda será necessária se a regra também for invocada por outras mensagens que devem ser enviadas por meio do agente de mensagens (por exemplo, porque essas outras mensagens devem ser distribuídas a vários assinantes). Para obter mais informações, consulte AWS IoT Referência SQL.
- Os três primeiros níveis do tópico da Ingestão básica (`$aws/rules/rule_name`) não são inclusos no cálculo do tamanho máximo de 8 segmentos ou no limite total de 256 caracteres por tópico. Caso contrário, as mesmas restrições se aplicarão conforme documentado em Limites de AWS IoT.

- Se uma mensagem for recebida com um tópico da Ingestão básica que especifica uma regra inativa ou uma regra que não existe, um log de erros será criado em um log do Amazon CloudWatch para ajudá-lo com a depuração. Para obter mais informações, consulte Entradas de log do mecanismo de regras. Uma métrica `RuleNotFound` é indicada e você pode criar alarmes nessa métrica. Para obter mais informações, consulte Métricas de regras em Métricas de regra.
- Você ainda poderá publicar com QoS 1 em tópicos de Ingestão básica. Você recebe uma PUBACK depois que a mensagem é entregue com êxito ao mecanismo de regras. O recebimento de uma PUBACK não indica que as ações da regra foram concluídas com êxito. Você pode configurar uma ação de erro para lidar com erros durante a execução de uma ação. Para obter mais informações, consulte Tratamento de erros (ação de erro).

AWS IoT Referência SQL

Em AWS IoT, as regras são definidas usando uma sintaxe semelhante a SQL. As declarações do SQL são compostas por três tipos de cláusulas:

SET

(Opcional) Define variáveis que você pode reutilizar em todas as instruções SQL e modelos de substituição. Atribua valores às variáveis usando expressões. Faça referência a essas variáveis nas cláusulas SELECT e WHERE e nos modelos de substituição de ações.

A cláusula SET suporta Tipos de dadosOperadores,Funções,Literais,Declarações de caso,,Extensões JSON, Variáveis e. Consultas de objeto aninhado

SELECT

(Obrigatório) Extrai informações da carga útil da mensagem de entrada e executa transformações nas informações. As mensagens a serem usadas são identificadas pelo filtro de tópico especificado na cláusula FROM.

A cláusula SELECT oferece suporte a Tipos de dados Operadores FunçõesLiterais,Declarações de caso,,Extensões JSON,Modelos de substituição,, VariáveisConsultas de objeto aninhado, e. Cargas binárias

FROM

O filtro de tópicos de mensagens do MQTT que identifica as mensagens das quais extrair dados. A regra é ativada para cada mensagem enviada para um tópico MQTT que corresponda ao

filtro de tópico especificado aqui. Obrigatório para regras que são ativadas por mensagens que passam pelo agente de mensagens. Opcional para regras que só são ativadas usando o atributo Ingestão básica.

WHERE

(Opcional) Adiciona lógica condicional que determina se as ações especificadas por uma regra são executadas.

A cláusula WHERE suporta Tipos de dadosOperadores,Funções,Literais,Declarações de caso,Extensões JSON,, Variáveis e. Consultas de objeto aninhado

Um exemplo de declaração do SQL é semelhante a:

```
SELECT color AS rgb FROM 'topic/subtopic' WHERE temperature > 50
```

Um exemplo de mensagem MQTT (também chamada de carga útil de entrada) é semelhante a:

```
{
    "color":"red",
    "temperature":100
}
```

Se essa mensagem for publicada no tópico `'topic/subtopic'`, a regra será acionada, e a declaração do SQL será avaliada. A declaração do SQL extrairá o valor da propriedade `color` se a propriedade `"temperature"` for superior a 50. A cláusula WHERE especifica a condição `temperature > 50`. A palavra-chave AS renomeia a propriedade `"color"` para `"rgb"`. O resultado (também chamado de carga útil de saída) é semelhante a:

```
{
    "rgb":"red"
}
```

Esses dados são encaminhados para a ação da regra, que envia os dados para realizar mais processamento. Para saber mais sobre as ações de regra, consulte AWS IoT ações de regras.

> **Note**
>
> Atualmente, não há suporte para comentários na sintaxe AWS IoT SQL.

Nomes de atributos com espaços não podem ser usados como nomes de campo na instrução SQL. Embora a carga de entrada possa ter nomes de atributos com espaços, esses nomes não podem ser usados na instrução SQL. No entanto, eles serão passados para a carga de saída se você usar uma especificação de nome de campo curinga (*).

Cláusula SELECT

A cláusula AWS IoT SELECT é essencialmente a mesma que a cláusula ANSI SQL SELECT, com algumas pequenas diferenças.

A cláusula SELECT suporta Tipos de dadosOperadores,Funções,Literais,Declarações de caso,Extensões JSON,, VariáveisConsultas de objeto aninhado, e. Cargas binárias

Você pode usar a cláusula SELECT para extrair informações das mensagens MQTT recebidas. Você também pode usar `SELECT * ` para recuperar toda a carga útil da mensagem recebida. Por exemplo:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL statement: SELECT * FROM 'topic/subtopic'
Outgoing payload: {"color":"red", "temperature":50}
```

Se a carga útil for um objeto JSON, você poderá fazer referência a chaves no objeto. A carga útil de saída contém o par de chave/valor. Por exemplo:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL statement: SELECT color FROM 'topic/subtopic'
Outgoing payload: {"color":"red"}
```

Você pode usar o teclado AS para renomear chaves. Por exemplo:

```
Incoming payload published on topic 'topic/subtopic':{"color":"red", "temperature":50}
SQL:SELECT color AS my_color FROM 'topic/subtopic'
Outgoing payload: {"my_color":"red"}
```

Você pode selecionar vários itens ao separá-los com uma vírgula. Por exemplo:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT color as my_color, temperature as fahrenheit FROM 'topic/subtopic'
```

```
Outgoing payload: {"my_color":"red","fahrenheit":50}
```

Você pode selecionar vários itens incluindo '*' para adicionar itens à carga útil de entrada. Por exemplo:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT *, 15 as speed FROM 'topic/subtopic'
Outgoing payload: {"color":"red", "temperature":50, "speed":15}
```

Você pode usar a palavra-chave "VALUE" para produzir cargas úteis de saída que não são objetos JSON. Com a versão SQL 2015-10-08, você pode selecionar apenas um item. Com a versão 2016-03-23 do SQL ou posterior, você também pode selecionar uma matriz para saída como um objeto de nível superior.

Example

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT VALUE color FROM 'topic/subtopic'
Outgoing payload: "red"
```

Você pode usar a sintaxe '.' para analisar os objetos JSON aninhados na carga útil de entrada. Por exemplo:

```
Incoming payload published on topic 'topic/subtopic': {"color":
{"red":255,"green":0,"blue":0}, "temperature":50}
SQL: SELECT color.red as red_value FROM 'topic/subtopic'
Outgoing payload: {"red_value":255}
```

Para obter informações sobre como usar nomes de objetos e propriedades JSON que incluem caracteres reservados, como números ou o caractere de hífen (menos), consulte Extensões JSON

Você pode usar funções (consulte Funções) para transformar a carga útil de entrada. Você pode usar parênteses para agrupamento. Por exemplo:

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT (temperature - 32) * 5 / 9 AS celsius, upper(color) as my_color FROM
 'topic/subtopic'
Outgoing payload: {"celsius":10,"my_color":"RED"}
```

Cláusula FROM

A cláusula FROM inscreve sua regra em um tópico ou filtro de tópicos. Coloque o filtro de tópicos ou tópico entre aspas simples ('). A regra é acionada para cada mensagem enviada para um tópico MQTT que corresponda ao filtro de tópico especificado aqui. Você pode permite se inscrever em um grupo de tópicos semelhantes usando um filtro de tópicos.

Exemplo:

Carga útil de entrada publicada no tópico `'topic/subtopic'`: `{temperature: 50}`

Carga útil de entrada publicada no tópico `'topic/subtopic-2'`: `{temperature: 50}`

SQL: `"SELECT temperature AS t FROM 'topic/subtopic'"`.

A regra é inscrita em `'topic/subtopic'`, de modo que a carga útil de entrada é passada para a regra. A carga útil de saída, passada para as ações da regra, é: `{t: 50}`. A regra não está inscrita em `'topic/subtopic-2'`; portanto, a regra não é acionada para a mensagem publicada em `'topic/subtopic-2'`.

Exemplo de curinga #:

Você pode usar o caractere curinga "#" (com vários níveis) para corresponder a um ou mais elementos de caminho específicos:

Carga útil de entrada publicada no tópico `'topic/subtopic'`: `{temperature: 50}`.

Carga útil de entrada publicada no tópico `'topic/subtopic-2'`: `{temperature: 60}`.

Carga útil de entrada publicada no tópico `'topic/subtopic-3/details'`: `{temperature: 70}`.

Carga útil de entrada publicada no tópico `'topic-2/subtopic-x'`: `{temperature: 80}`.

SQL: `"SELECT temperature AS t FROM 'topic/#'"`.

A regra está inscrita em qualquer tópico que comece com`'topic'`, então é executada três vezes, enviando cargas de saída de (para tópico/subtópico), `{t: 50}` (para tópico/subtópico-2) e `{t: 60}` (para) para suas ações. `{t: 70}` topic/subtopic-3/details Se não estiver inscrito em `'topic-2/subtopic-x'`, a regra não será acionada para a mensagem `{temperature: 80}`.

Exemplo de curinga +:

Você pode usar o caractere curinga "+" (com nível único) para corresponder a qualquer elemento de caminho específico:

Carga útil de entrada publicada no tópico `'topic/subtopic'`: `{temperature: 50}`.

Carga útil de entrada publicada no tópico `'topic/subtopic-2'`: `{temperature: 60}`.

Carga útil de entrada publicada no tópico `'topic/subtopic-3/details'`: `{temperature: 70}`.

Carga útil de entrada publicada no tópico `'topic-2/subtopic-x'`: `{temperature: 80}`.

SQL: `"SELECT temperature AS t FROM 'topic/+'"`.

A regra é inscrita em todos os tópicos com dois elementos de caminho, em que o primeiro elemento é `'topic'`. A regra é executada para as mensagens enviadas para `'topic/subtopic'` e `'topic/subtopic-2'`, mas não `'topic/subtopic-3/details'` (tem mais níveis do que o filtro de tópicos) ou `'topic-2/subtopic-x'` (não começa com `topic`).

Cláusula SET

Use a cláusula SET para definir variáveis que armazenam resultados de expressões. Você pode reutilizar essas variáveis nas cláusulas SELECT e WHERE e em modelos de substituição. Isso ajuda a evitar a duplicação de expressões complexas e a reduzir o número de chamadas de função em sua instrução SQL.

A cláusula SET suporta Tipos de dadosOperadores,Funções,Literais,Declarações de caso,,Extensões JSON, Variáveis e. Consultas de objeto aninhado

Sintaxe da cláusula SET

A cláusula SET deve aparecer antes da cláusula SELECT em sua instrução SQL. Use a seguinte sintaxe:

```
SET @variable_name = expression [, @variable_name2 = expression2]
```

Regras de sintaxe:

- Inicie os nomes das variáveis com `@`

- Os nomes das variáveis podem conter letras, números e sublinhados
- Os nomes das variáveis podem ter até 64 caracteres
- Várias variáveis podem ser definidas em uma única cláusula SET, separadas por vírgulas
- Cada variável só pode ser atribuída uma vez (as variáveis são imutáveis)
- A palavra-chave SET só pode ser usada uma vez por instrução SQL

Usando variáveis

Depois de definir as variáveis, você pode usá-las em:

- Cláusulas SELECT
- Cláusulas WHERE
- Outras atribuições de variáveis SET
- Modelos de substituição de ações
- Modelos de substituição de ações de erro
- Consultas SELECT aninhadas
- Parâmetros de função (certos parâmetros, como parâmetros do RoLearn e parâmetros que alternam o modo de uma função semelhante a `transform("enrichArray", attributes, values)` que não suportam variáveis)

As variáveis são referenciadas usando a mesma `@variable_name` sintaxe usada na cláusula SET. Você também pode usar a sintaxe da extensão JSON para acessar propriedades de variáveis que contêm objetos, como. `@variable_name.property`

Exemplos da cláusula SET

Uso básico de variáveis

O exemplo a seguir mostra uma carga publicada no tópico`device/data`: `{"temp_fahrenheit": 75, "humidity": 60}`

Declaração do SQL:

```
SET @temp_celsius = (temp_fahrenheit - 32) * 5 / 9
SELECT @temp_celsius AS celsius, humidity FROM 'device/data'
```

Carga útil de saída: `{"celsius": 23.89, "humidity": 60}`

Acesse membros em objetos JSON incorporados

O exemplo a seguir mostra uma carga publicada no tópico`device/data`: `{"device1": {"deviceId":"weather_sensor", "deviceData": {"sensors": {"temp_fahrenheit": 75, "humidity": 60}, "location": [47.606,-122.332]}}}`

Declaração do SQL:

```
SET @device_sensor_data = device1.deviceData.sensors
SELECT @device_sensor_data.temp_fahrenheit AS temp_fahrenheit,
 @device_sensor_data.humidity as humidity, device1.deviceId as deviceId FROM 'device/
data'
```

Carga útil de saída:
`{"temp_fahrenheit":75,"humidity":60,"deviceId":"weather_sensor"}`

para obter mais informações sobre como trabalhar com extensões JSON, consulte Extensões JSON

Evitando chamadas de função duplicadas

As variáveis SET ajudam a evitar a duplicação de operações complexas de decodificação:

```
SET @decoded_data = decode(encode(*, 'base64'), 'proto', 'schema', 'schema.desc',
 'message.proto', 'Message')
SELECT @decoded_data.sensor_id, @decoded_data.reading FROM 'device/protobuf'
WHERE @decoded_data.reading > 100
```

Sem as variáveis SET, você precisaria repetir a função de decodificação três vezes, o que excede os limites de chamada da função.

Várias variáveis

Você pode definir várias variáveis em uma única cláusula SET separando-as com vírgulas:

```
SET @user_data = get_user_properties(device_id), @threshold = 50
SELECT @user_data.name, temp_fahrenheit FROM 'sensors/+'
WHERE temp_fahrenheit > @threshold AND @user_data.active = true
```

Usando variáveis em modelos de substituição

As variáveis também podem ser usadas em modelos de substituição de ações, permitindo que você reutilize valores computados nas ações da instrução SQL e da regra.

Declaração do SQL:

```
SET @temp_celsius = (temp_fahrenheit - 32) * 5 / 9
SELECT @temp_celsius AS celsius, humidity FROM 'device/data'
```

Configuração da ação:

```
{
  "s3": {
    "roleArn": "arn:aws:iam::123456789012:role/testRuleRole",
    "bucketName": "bucket",
    "key": "temperature-data/${device_id}/temp-${@temp_celsius}C.json"
  }
}
```

Neste exemplo, a variável SET `@temp_celsius` é usada em um modelo de substituição para construir o campo chave da ação S3.

Uso de carga útil não JSON

As variáveis SET não oferecem suporte direto a cargas não JSON, portanto, a carga deve ser codificada ou decodificada primeiro:

```
SET @encoded_payload = encode(*, 'base64')
SELECT @encoded_payload AS raw_data FROM 'device/binary'
```

para obter mais informações sobre como trabalhar com cargas não JSON, consulte Como trabalhar com cargas úteis binárias

Limites da cláusula SET

Os limites a seguir se aplicam às variáveis SET:

- Máximo de 10 variáveis exclusivas por instrução SQL
- Tamanho máximo do valor variável de 128 KiB (string JSON UTF-8 minificada)
- Tamanho máximo do valor total de 128 KiB para todas as variáveis

- Nomes de variáveis limitados a 64 caracteres
- As variáveis podem aceitar cargas JSON diretamente como estão (cargas não JSON devem primeiro ser codificadas/decodificadas)

Cláusula WHERE

A cláusula WHERE determina se as ações especificadas por uma regra são executadas. Se a cláusula WHERE for avaliada como verdadeira, as ações da regra serão executadas. Caso contrário, as ações da regra não serão executadas.

A cláusula WHERE suporta Tipos de dadosOperadores,Funções,Literais,Declarações de caso,Extensões JSON,, Variáveis e. Consultas de objeto aninhado

Exemplo:

Carga útil de entrada publicada em `topic/subtopic`: `{"color":"red", "temperature":40}`.

SQL: `SELECT color AS my_color FROM 'topic/subtopic' WHERE temperature > 50 AND color <> 'red'`.

Nesse caso, a regra será acionada, mas as ações especificadas pela regra não serão executadas. Não haverá carga útil de saída.

Você pode usar funções e operadores na cláusula WHERE. No entanto, você não pode fazer referência a aliases criados com a palavra-chave AS no SELECT. A cláusula WHERE é avaliada primeiro para determinar se SELECT será avaliada.

Exemplo com carga útil não JSON:

Carga útil não JSON de entrada publicada em `tópico/subtópico`: `80`

SQL: `` `SELECT decode(encode(*, 'base64'), 'base64') AS value FROM 'topic/subtopic' WHERE decode(encode(*, 'base64'), 'base64') > 50 ``

Nesse caso, a regra será acionada e as ações especificadas pela regra serão executadas. A carga útil de saída será transformada pela cláusula SELECT como uma carga JSON `{"value":80}`.

Tipos de dados

O mecanismo de AWS IoT regras é compatível com todos os tipos de dados JSON.

Tipos de dados compatíveis

Tipo	Significado
`Int`	Um discreto `Int`. 34 dígitos no máximo.
`Decimal`	Um `Decimal` com uma precisão de 34 dígitos, com um mínimo de magnitude não zero de 1E-999 e um máximo de magnitude de 9,999...E999. **Note** Algumas funções geram valores `Decimal` com precisão dupla em vez de precisão de 34 dígitos. Com o SQL V2 (23/03/2016), valores numéricos que são números inteiros, como `10.0`, são processados como um valor `Int` (`10`) em vez do valor `Decimal` esperado (`10.0`). Para processar de forma confiável valores numéricos inteiros como valores `Decimal`, use o SQL V1 (08/10/2015) para a instrução de consulta de regra.
`Boolean`	`True` ou `False`.
`String`	Uma string UTF-8.
`Array`	Uma série de valores que não precisam ter o mesmo tipo.
`Object`	Um valor JSON que consiste em uma chave e um valor. As chaves devem ser strings. Os valores podem ser de qualquer tipo.
`Null`	`Null` conforme definido pelo JSON. É um valor real que representa a ausência de um valor.

Tipo	Significado
	Você pode criar explicitamente um valor `Null` usando a palavra-chave `Null` na declaração do SQL. Por exemplo: `"SELECT NULL AS n FROM 'topic/subtopic'"`

`Undefined`	Não é um valor. Não é explicitamente represent ável no JSON, exceto ao omitir o valor. Por exemplo, no objeto `{"foo": null}`, a chave "foo" gera NULL, mas a chave "bar" gera `Undefined` . Internamente, a linguagem SQL trata `Undefined` como um valor, mas não é representável no JSON; portanto, quando serializados para JSON, os resultados são `Undefined` .
	
	`{"foo":null, "bar":undefined}`
	
	é serializado para JSON como:
	
	`{"foo":null}`
	
	Da mesma forma, `Undefined` é convertid o em uma string vazia quando serializado por conta própria. Funções chamadas com argumentos inválidos (por exemplo, tipos incorretos, número incorreto de argumentos, etc.) retornam `Undefined` .

Conversões

A tabela a seguir indica os resultados quando um valor de um tipo é convertido em outro tipo (quando um valor do tipo incorreto é dado a uma função). Por exemplo, se a função de valor absoluto "abs" (que espera um `Int` ou `Decimal`) for dado a `String`, ela tentará converter `String` em um `Decimal`, seguindo essas regras. Nesse caso, “abs ("-5,123")” é tratado como “abs(-5,123)”.

> **Note**
>
> Não há tentativa de conversões para `Array`, `Object`, `Null` ou `Undefined`.

Para decimal

Tipo de argumento	Resultado
`Int`	Um `Decimal` sem ponto decimal.
`Decimal`	O valor de origem.
`Boolean`	`Undefined` . (Você pode explicitamente usar a função cast para transformar verdadeiro = 1,0, falso = 0,0.)
`String`	O mecanismo SQL tenta analisar a string como a. `Decimal` AWS IoT tenta analisar cadeias de caracteres que correspondam à expressão regular:. `^-?\d+(\.\d+)?((?i)E-?\d+)?$` "0", "-1,2" e "5E-12" são exemplos de strings que são automaticamente convertidas em `Decimals`.
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Null`.
Não definido	`Undefined` .

Para int

Tipo de argumento	Resultado
`Int`	O valor de origem.

Tipo de argumento	Resultado
`Decimal`	O valor de origem arredondado para o `Int` mais próximo.
`Boolean`	`Undefined` . (Você pode explicitamente usar a função cast para transformar verdadeiro = 1,0, falso = 0,0.)
`String`	O mecanismo SQL tenta analisar a string como a. `Decimal` AWS IoT tenta analisar cadeias de caracteres que correspondam à expressão regular:. `^-?\d+(\.\d+)?((?i)E-?\d+)?$` “0", “-1.2", “5E-12" são todos exemplos de strings que são convertid as automaticamente em `Decimal` s. AWS IoT tenta converter o em a e`Decimal`, em seguida, trunca as casas decimais `String` para formar um. `Decimal` `Int`
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Null`.
Não definido	`Undefined` .

Para Booleano

Tipo de argumento	Resultado
`Int`	`Undefined` . (Você pode explicitamente usar a função `cast` para transformar 0 = Falso, qualquer_valor_não_zero = Verdadeiro.)

Tipo de argumento	Resultado
`Decimal`	`Undefined` . (Você pode explicitamente usar a função cast para transformar 0 = Falso, qualquer_valor_não_zero = Verdadeiro.)
`Boolean`	O valor original.
`String`	"true" = Verdadeiro e "false" = Falso (não diferencia maiúsculas de minúsculas). Outros valores de string são `Undefined` .
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Undefined` .
Não definido	`Undefined` .

Para string

Tipo de argumento	Resultado
`Int`	Uma representação de string do `Int` em notação padrão.
`Decimal`	Uma string representando o valor `Decimal`, possivelmente em notação científica.
`Boolean`	"true" ou "false". Todas as letras minúsculas.
`String`	O valor original.
Array	A `Array` serializada para JSON. A string resultante é uma lista separada por vírgulas, entre colchetes. A `String` tem aspas. Um `Decimal`, `Int`, `Boolean` e `Null` não.

Tipo de argumento	Resultado
Objeto	O objeto serializado para JSON. A string resultante é uma lista separada por vírgulas de pares de chave/valor e começa e termina com chaves. A `String` tem aspas. Um `Decimal`, `Int`, `Boolean` e `Null` não.
Null	`Undefined` .
Não definido	Indefinido.

Operadores

Os operadores a seguir podem ser usados nas cláusulas SELECT e WHERE.

Operador AND

Gera um resultado `Boolean`. Realiza uma operação E lógica. Retornará verdadeiro se os operandos esquerdo e direito forem verdadeiros. Caso contrário, retorna false. Os operandos `Boolean` ou operandos de string "true" ou "false" que não diferenciam letras maiúsculas de minúsculas são necessários.

Sintaxe: *`expression`* `AND` *`expression`*.

Operador AND

Operando esquerdo	Operando direito	Output
`Boolean`	`Boolean`	`Boolean`. Verdadeiro se ambos os operandos forem verdadeiros. Caso contrário, falso.
`String`/`Boolean`	`String`/`Boolean`	Se todas as strings forem "true" ou "false" (não diferenci a letras maiúsculas de minúsculas), elas serão convertid as em `Boolean` e processadas normalmente como *`boolean`* `AND` *`boolean`*.
Outros valores	Outros valores	`Undefined` .

Operador OU

Gera um resultado `Boolean`. Executa uma operação OU lógica. Retornará verdadeiro se um dos operandos esquerdo ou direito for verdadeiro. Caso contrário, retorna false. Os operandos `Boolean` ou operandos de string "true" ou "false" que não diferenciam letras maiúsculas de minúsculas são necessários.

Sintaxe: *`expression`* `OR` *`expression`*.

Operador OU

Operando esquerdo	Operando direito	Output
`Boolean`	`Boolean`	`Boolean`. Verdadeiro se um dos operandos for verdadeiro. Caso contrário, falso.
`String/Boolean`	`String/Boolean`	Se todas as strings forem "true" ou "false" (não diferencia letras maiúsculas de minúsculas), elas serão convertidas em booleanos e processadas normalmente como *`boolean`* `OR` *`boolean`*.
Outros valores	Outros valores	`Undefined` .

Operador NOT

Gera um resultado `Boolean`. Realiza uma operação NÃO lógica. Retornará verdadeiro se o operando for falso. Caso contrário, retornará verdadeiro. Um operando `Boolean` ou operando de string "true" ou "false" que não diferencia letras maiúsculas de minúsculas é necessário.

Sintaxe: `NOT` *`expression`*.

Operador NOT

Operando	Output
`Boolean`	`Boolean`. Verdadeiro se o operando for falso. Caso contrário, verdadeiro.

Operando	Output
`String`	Se a string for "true" ou "false" (não diferenci a letras maiúsculas de minúsculas), ela será convertida no valor booliano correspondente, e o valor oposto será gerado.
Outros valores	`Undefined` .

Operador IN

Gera um resultado `Boolean`. Você pode usar o operador IN em uma cláusula WHERE para verificar se um valor corresponde a algum valor em uma matriz. Ele retornará verdadeiro se a correspondência for encontrada; caso contrário, retornará falso.

Sintaxe: *`expression`* `IN` *`expression`*.

Operador IN

Operando esquerdo	Operando direito	Output
`Int/Decimal/String/A`	`Array`	Verdadeiro se o elemento `Integer/Decimal/String/Array/Object` for encontrado na matriz. Caso contrário, falso.

Exemplo:

```
SQL: "select * from 'a/b' where 3 in arr"

JSON: {"arr":[1, 2, 3, "three", 5.7, null]}
```

Neste exemplo, a cláusula de condição `where 3 in arr` será avaliada como verdadeira porque 3 está presente na matriz chamada `arr`. Portanto, na instrução SQL, `select * from 'a/b'` será executado. Esse exemplo também mostra que a matriz pode ser heterogênea.

Operador EXISTS

Gera um resultado `Boolean`. Você poderá usar o operador EXISTS em uma cláusula condicional para testar a existência de elementos em uma subconsulta. Ele retornará verdadeiro se a subconsulta retornar um ou mais elementos e falso se a subconsulta não retornar nenhum elemento.

Sintaxe: *`expression`*.

Exemplo:

```
SQL: "select * from 'a/b' where exists (select * from arr as a where a = 3)"

JSON: {"arr":[1, 2, 3]}
```

Neste exemplo, a cláusula de condição `where exists (select * from arr as a where a = 3)` será avaliada como verdadeira porque 3 está presente na matriz chamada `arr`. Portanto, na instrução SQL, `select * from 'a/b'` será executado.

Exemplo:

```
SQL: select * from 'a/b' where exists (select * from e as e where foo = 2)

JSON: {"foo":4,"bar":5,"e":[{"foo":1},{"foo":2}]}
```

Neste exemplo, a cláusula de condição `where exists (select * from e as e where foo = 2)` será avaliada como verdadeira porque a matriz e dentro do objeto JSON contém o objeto `{"foo":2}`. Portanto, na instrução SQL, `select * from 'a/b'` será executado.

> operador

Gera um resultado `Boolean`. Verdadeiro se o operando esquerdo for superior ao operando direito. Os dois operandos são convertidos em um `Decimal` e depois comparados.

Sintaxe: *`expression`* `>` *`expression`*.

> operador

Operando esquerdo	Operando direito	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`. Verdadeiro se o operando esquerdo for superior ao operando direito. Caso contrário, falso.
`String/Int/Deci`	`String/Int/Deci`	Se todas as strings puderem ser convertidas em `Decimal`, então `Boolean`. Verdadeiro se o operando esquerdo for superior ao operando direito. Caso contrário , falso.
Outros valores	`Undefined .`	`Undefined .`

>= operador

Gera um resultado `Boolean`. Verdadeiro se o operando esquerdo for superior ou igual ao operando direito. Os dois operandos são convertidos em um `Decimal` e depois comparados.

Sintaxe: *`expression`* `>=` *`expression`*.

>= operador

Operando esquerdo	Operando direito	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`. Verdadeiro se o operando esquerdo for igual ou superior ao operando direito. Caso contrário, falso.
`String/Int/Deci`	`String/Int/Deci`	Se todas as strings puderem ser convertidas em `Decimal`, então `Boolean`. Verdadeiro se o operando esquerdo for superior ou igual ao operando direito. Caso contrário, falso.
Outros valores	`Undefined .`	`Undefined .`

< operador

Gera um resultado `Boolean`. Verdadeiro se o operando esquerdo for inferior ao operando direito. Os dois operandos são convertidos em um `Decimal` e depois comparados.

Sintaxe: *`expression`* `<` *`expression`*.

< operador

Operando esquerdo	Operando direito	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`. Verdadeiro se o operando esquerdo for inferior ao operando direito. Caso contrário, falso.
`String/Int/Deci`	`String/Int/Deci`	Se todas as strings puderem ser convertidas em `Decimal`, então `Boolean`. Verdadeiro se o operando esquerdo for inferior ao operando direito. Caso contrário, falso.
Outros valores	`Undefined`	`Undefined`

<= operador

Gera um resultado `Boolean`. Verdadeiro se o operando esquerdo for inferior ou igual ao operando direito. Os dois operandos são convertidos em um `Decimal` e depois comparados.

Sintaxe: *`expression`* `<=` *`expression`*.

<= operador

Operando esquerdo	Operando direito	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`. Verdadeiro se o operando esquerdo for igual ou inferior ao operando direito. Caso contrário, falso.
`String/Int/Deci`	`String/Int/Deci`	Se todas as strings puderem ser convertidas em `Decimal`, então `Boolean`. Verdadeiro se o operando

Operando esquerdo	Operando direito	Output
		esquerdo for inferior ou igual ao operando direito. Caso contrário, falso.
Outros valores	`Undefined`	`Undefined`

<> operador

Gera um resultado `Boolean`. Retornará verdadeiro se os operandos esquerdo e direito forem diferentes. Caso contrário, retornará falso.

Sintaxe: *`expression`* <> *`expression`*.

<> operador

Operando esquerdo	Operando direito	Output
`Int`	`Int`	Verdadeiro se o operando esquerdo não for igual ao operando direito. Caso contrário, falso.
`Decimal`	`Decimal`	Verdadeiro se o operando esquerdo não for igual ao operando direito. Caso contrário, falso. `Int` é convertido em `Decimal` antes de ser comparado.
`String`	`String`	Verdadeiro se o operando esquerdo não for igual ao operando direito. Caso contrário, falso.
Array	Array	Verdadeiro se os itens em cada operando não forem iguais e não estiverem na mesma ordem. Caso contrário, falso
Objeto	Objeto	Verdadeiro se as chaves e os valores de cada operando não forem iguais. Caso contrário, falso. A ordem de não keys/values é importante.
Null	Null	Falso.

Operando esquerdo	Operando direito	Output
Qualquer valor	`Undefined`	Indefinido.
`Undefined`	Qualquer valor	Indefinido.
Tipo não correspondente	Tipo não correspondente	Verdadeiro.

= operador

Gera um resultado `Boolean`. Retornará verdadeiro se os operandos esquerdo e direito forem iguais. Caso contrário, retornará falso.

Sintaxe: *`expression`* = *`expression`*.

= operador

Operando esquerdo	Operando direito	Output
`Int`	`Int`	Verdadeiro se o operando esquerdo for igual ao operando direito. Caso contrário, falso.
`Decimal`	`Decimal`	Verdadeiro se o operando esquerdo for igual ao operando direito. Caso contrário, falso. `Int` é convertido em `Decimal` antes de ser comparado.
`String`	`String`	Verdadeiro se o operando esquerdo for igual ao operando direito. Caso contrário, falso.
Array	Array	Verdadeiro se os itens em cada operando forem iguais e estiverem na mesma ordem. Caso contrário, falso.
Objeto	Objeto	Verdadeiro se as chaves e os valores de cada operando forem iguais. Caso contrário, falso. A ordem de não keys/values é importante.

Operando esquerdo	Operando direito	Output
Qualquer valor	`Undefined`	`Undefined` .
`Undefined`	Qualquer valor	`Undefined` .
Tipo não correspondente	Tipo não correspondente	Falso.

+ operador

" +" é um operador sobrecarregado. Ele pode ser usado para concatenação ou adição de string.

Sintaxe: *`expression`* `+` *`expression`*.

+ operador

Operando esquerdo	Operando direito	Output
`String`	Qualquer valor	Converte o operando direito em uma string e concatena-o ao final do operando esquerdo.
Qualquer valor	`String`	Converte o operando esquerdo em uma string e concatena o operando direito ao final do operando esquerdo convertido.
`Int`	`Int`	`Int` value. Adiciona operandos juntos.
`Int/Decimal`	`Int/Decimal`	`Decimal` value. Adiciona operandos juntos.
Outros valores	Outros valores	`Undefined` .

- operador

Subtrai o operando direito do operando esquerdo.

Sintaxe: *`expression`* `-` *`expression`*.

- operador

Operando esquerdo	Operando direito	Output
`Int`	`Int`	`Int` value. Subtrai o operando direito do operando esquerdo.
`Int/Decimal`	`Int/Decimal`	`Decimal` value. Subtrai o operando direito do operando esquerdo.
`String/Int/Deci`	`String/Int/Deci`	Se todas as strings forem convertidas em decimais corretamente, um valor `Decimal` será gerado. Subtrai o operando direito do operando esquerdo. Caso contrário, gera `Undefined` .
Outros valores	Outros valores	`Undefined` .
Outros valores	Outros valores	`Undefined` .

* operador

Multiplica o operando esquerdo pelo operando direito.

Sintaxe: *`expression`* * *`expression`*.

* operador

Operando esquerdo	Operando direito	Output
`Int`	`Int`	`Int` value. Multiplica o operando esquerdo pelo operando direito.
`Int/Decimal`	`Int/Decimal`	`Decimal` value. Multiplica o operando esquerdo pelo operando direito.
`String/Int/Deci`	`String/Int/Deci`	Se todas as strings forem convertidas em decimais corretamente, um valor `Decimal` será gerado. Multiplic

Operando esquerdo	Operando direito	Output
		a o operando esquerdo pelo operando direito. Caso contrário, gera `Undefined` .
Outros valores	Outros valores	`Undefined` .

/ operador

Divide o operando esquerdo pelo operando direito.

Sintaxe: *`expression`* `/` *`expression`*.

/ operador

Operando esquerdo	Operando direito	Output
`Int`	`Int`	`Int` value. Divide o operando esquerdo pelo operando direito.
`Int/Decimal`	`Int/Decimal`	`Decimal` value. Divide o operando esquerdo pelo operando direito.
`String/Int/Deci`	`String/Int/Deci`	Se todas as strings forem convertidas em decimais corretamente, um valor `Decimal` será gerado. Divide o operando esquerdo pelo operando direito. Caso contrário , gera `Undefined` .
Outros valores	Outros valores	`Undefined` .

% operador

Gera o restante da divisão do operando esquerdo pelo operando direito.

Sintaxe: *`expression`* `%` *`expression`*.

% operador

Operando esquerdo	Operando direito	Output
`Int`	`Int`	`Int` value. Gera o restante da divisão do operando esquerdo pelo operando direito.
`String/Int/Deci`	`String/Int/Deci`	Se todas as strings forem convertidas em decimais corretamente, um valor `Decimal` será gerado. Gera o restante da divisão do operando esquerdo pelo operando direito. Caso contrário, `Undefined` .
Outros valores	Outros valores	`Undefined` .

Funções

Você pode usar as seguintes funções integradas nas cláusulas SELECT ou WHERE das expressões do SQL.

As seguintes funções externas são cobradas de forma equivalente à de uma ação de regra: `aws_lambda`, `get_dynamodb()`, `get_registry_data()`, e. `get_thing_shadow()` Você também é cobrado pela função `decode()` apenas quando está decodificando uma mensagem Protobuf para JSON. Para obter mais detalhes, consulte a página de precificação do AWS IoT Core.

abs(Decimal)

Gera o valor absoluto de um número. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `abs(-5)` gera 5.

Tipo de argumento	Resultado
`Int`	`Int`, o valor absoluto do argumento.
`Decimal`	`Decimal`, o valor absoluto do argumento.
`Boolean`	`Undefined` .

Tipo de argumento	Resultado
`String`	`Decimal`. O resultado é o valor absoluto do argumento. Se a string não puder ser convertida, o resultado será `Undefined` .
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Undefined` .
Não definido	`Undefined` .

accountid()

Gera o ID da conta que possui essa regra como uma `String`. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo:

`accountid()` = "123456789012"

acos(Decimal)

Gera o cosseno inverso de um número em radianos. Argumentos `Decimal` são arredondados para dobrar a precisão antes da aplicação da função. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `acos(0)` = 1,5707963267948966

Tipo de argumento	Resultado
`Int`	`Decimal` (com precisão dupla), o cosseno inverso do argumento. Os resultados imaginários são gerados como `Undefined` .
`Decimal`	`Decimal` (com precisão dupla), o cosseno inverso do argumento. Os resultados

Tipo de argumento	Resultado
	imaginários são gerados como `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal`, o cosseno inverso do argumento . Se a string não puder ser convertida, o resultado será `Undefined` . Os resultados imaginários são gerados como `Undefined` .
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Undefined` .
Não definido	`Undefined` .

asin(Decimal)

Gera o seno inverso de um número em radianos. Argumentos `Decimal` são arredondados para dobrar a precisão antes da aplicação da função. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `asin(0)` = 0,0

Tipo de argumento	Resultado
`Int`	`Decimal` (com precisão dupla), o seno inverso do argumento. Os resultados imaginários são gerados como `Undefined` .
`Decimal`	`Decimal` (com precisão dupla), o seno inverso do argumento. Os resultados

Tipo de argumento	Resultado
	imaginários são gerados como `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal` (com precisão dupla), o seno inverso do argumento. Se a string não puder ser convertida, o resultado será `Undefined` . Os resultados imaginários são gerados como `Undefined` .
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Undefined` .
Não definido	`Undefined` .

atan(Decimal)

Gera a tangente inversa de um número em radianos. Argumentos `Decimal` são arredondados para dobrar a precisão antes da aplicação da função. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `atan(0)` = 0,0

Tipo de argumento	Resultado
`Int`	`Decimal` (com precisão dupla), a tangente inversa do argumento. Os resultados imaginários são gerados como `Undefined` .
`Decimal`	`Decimal` (com precisão dupla), a tangente inversa do argumento. Os resultados

Tipo de argumento	Resultado
	imaginários são gerados como `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal`, a tangente inversa do argumento . Se a string não puder ser convertida, o resultado será `Undefined` . Os resultados imaginários são gerados como `Undefined` .
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Undefined` .
Não definido	`Undefined` .

atan2(Decimal, Decimal)

Gera o ângulo, em radianos, entre o eixo X positivo e o ponto (x, y) definido nos dois argumentos. O ângulo é positivo para os ângulos em sentido anti-horário (metade superior, y > 0) e negativo para os ângulos em sentido horário (metade inferior, y < 0). Argumentos `Decimal` são arredondados para dobrar a precisão antes da aplicação da função. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `atan2(1, 0)` = 1,5707963267948966

Tipo de argumento	Tipo de argumento	Resultado
`Int/Decimal`	`Int/Decimal`	`Decimal` (com precisão dupla), entre o eixo x e o ponto (x, y) es
`Int/Decimal/String`	`Int/Decimal/String`	`Decimal`, a tangente inversa do descrito. Se uma string não pud convertida, o resultado será Und

Tipo de argumento	Tipo de argumento	Resultado
Outros valores	Outros valores	`Undefined` .

aws_lambda(functionArn, inputJson)

Chama a função do Lambda especificada passando `inputJson` para a função do Lambda e retorna o JSON gerado pela função do Lambda.

Argumentos

Argumento	Description
`functionArn`	O ARN da função do Lambda a ser chamada. A função do Lambda deve retornar dados JSON.
`inputJson`	A entrada JSON passada para a função do Lambda. Para passar consultas e literais de objetos aninhados, você deve usar o SQL versão 23/03/2016.

Você deve conceder AWS IoT `lambda:InvokeFunction` permissões para invocar a função Lambda especificada. O seguinte exemplo mostra como conceder a permissão `lambda:InvokeFunction` usando a AWS CLI:

```
aws lambda add-permission --function-name "function_name"
--region "region"
--principal iot.amazonaws.com
--source-arn arn:aws:iot:us-east-1:account_id:rule/rule_name
--source-account "account_id"
--statement-id "unique_id"
--action "lambda:InvokeFunction"
```

Veja os argumentos do comando add-permission:

--nome da função

Nome da função do Lambda. Você adiciona uma nova permissão para atualizar a política de recursos da função.

--region

O Região da AWS da sua conta.

--entidade principal

O principal que está recebendo a permissão. Isso deve permitir `iot.amazonaws.com` a AWS IoT permissão para chamar uma função Lambda.

--arn de origem

O ARN da regra. Você pode usar o get-topic-rule AWS CLI comando para obter o ARN de uma regra.

--conta de origem

O Conta da AWS local em que a regra é definida.

--id da declaração

Um identificador de declaração exclusivo.

--action

A ação do Lambda que você deseja permitir nesta declaração. Para permitir que o AWS IoT invoque uma função do Lambda, especifique `lambda:InvokeFunction`.

> **Important**
>
> Se você adicionar uma permissão para um AWS IoT principal sem fornecer o `source-arn` ou`source-account`, qualquer uma Conta da AWS que crie uma regra com sua ação do Lambda poderá acionar regras para invocar sua função do Lambda. AWS IoT Para obter mais informações, consulte Modelo de permissões do Lambda.

Dada uma carga da mensagem JSON como:

```
{
    "attribute1": 21,
    "attribute2": "value"
}
```

A função do `aws_lambda` pode ser usada para chamar a função do Lambda da seguinte forma.

```
SELECT
aws_lambda("arn:aws:lambda:us-east-1:account_id:function:lambda_function",
 {"payload":attribute1}) as output FROM 'topic-filter'
```

Se você deseja passar a carga da mensagem MQTT completa, especifique a carga JSON usando "*", como no exemplo a seguir.

```
SELECT
aws_lambda("arn:aws:lambda:us-east-1:account_id:function:lambda_function", *) as output
 FROM 'topic-filter'
```

`payload.inner.element` seleciona dados de mensagem publicada no tópico 'tópico/subtópico'.

`some.value` seleciona dados da saída gerada pela função do Lambda.

> **Note**
>
> O mecanismo de regras limita a duração da execução das funções do Lambda. As chamadas de função do Lambda em regras devem ser concluídas em 2.000 milissegundos.

bitand(Int, Int)

Realiza um bitwise AND nas representações do bit dos dois argumentos `Int`(-convertido). Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `bitand(13, 5)` = 5

Tipo de argumento	Tipo de argumento	Resultado
`Int`	`Int`	`Int`, um bitwise AND dos dois a
`Int/Decimal`	`Int/Decimal`	`Int`, um bitwise AND dos dois a Todos os números não `Int` são dos para o `Int` mais próximo. S um dos argumentos não puder s convertido em um `Int`, o resulta `Undefined` .

Tipo de argumento	Tipo de argumento	Resultado
`Int/Decimal/String`	`Int/Decimal/String`	`Int`, um bitwise AND dos dois a s. Todas as strings são converti decimais e são arredondadas pa mais próximo. Se não for possív , o resultado será `Undefined`
Outros valores	Outros valores	`Undefined` .

bitor(Int, Int)

Executa um bitwise OU das representações de bit dos dois argumentos. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `bitor(8, 5)` = 13

Tipo de argumento	Tipo de argumento	Resultado
`Int`	`Int`	`Int`, o bitwise OU dos dois argu
`Int/Decimal`	`Int/Decimal`	`Int`, o bitwise OU dos dois argu Todos os números não `Int` são dos para o `Int` mais próximo. S for possível converter, o resulta `Undefined` .
`Int/Decimal/String`	`Int/Decimal/String`	`Int`, o bitwise OU nos dois argu s. Todas as strings são converti decimais e são arredondadas pa mais próximo. Se não for possív , o resultado será `Undefined`
Outros valores	Outros valores	`Undefined` .

bitxor(Int, Int)

Realiza um bitwise XOR nas representações do bit dos dois argumentos `Int`(-convertido). Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo:`bitor(13, 5)` = 8

Tipo de argumento	Tipo de argumento	Resultado
`Int`	`Int`	`Int`, um bitwise XOR nos dois a
`Int/Decimal`	`Int/Decimal`	`Int`, um bitwise XOR nos dois a s. Os números não `Int` são arr para o `Int` mais próximo.
`Int/Decimal/String`	`Int/Decimal/String`	`Int`, um bitwise XOR nos dois a s. As strings são convertidas em e são arredondadas para o `Int` próximo. Se não for possível co resultado será `Undefined` .
Outros valores	Outros valores	`Undefined` .

bitnot(Int)

Realiza um bitwise NOT nas representações do bit do argumento `Int`(-convertido). Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `bitnot(13)` = 2

Tipo de argumento	Resultado
`Int`	`Int`, um bitwise NOT do argumento.
`Decimal`	`Int`, um bitwise NOT do argumento. O valor `Decimal` é arredondado para o `Int` abaixo mais próximo.

Tipo de argumento	Resultado
`String`	`Int`, um bitwise NOT do argumento. As strings são convertidas em decimais e são arredondadas para o `Int` mais próximo. Se não for possível converter, o resultado será `Undefined` .
Outros valores	Outros valores.

cast()

Converte um valor de um tipo de dados para outro. O cast é muito parecido com as conversões padrão, com a adição da capacidade de converter números em ou de boolianos. Se AWS IoT não conseguir determinar como converter um tipo em outro, o resultado é`Undefined`. Compatível com a versão de 08/10/2015 do SQL e posteriores. Formato: cast (*`value`*as*`type`*).

Exemplo:

`cast(true as Int) = 1`

As seguintes palavras-chave podem aparecer após "as" ao chamar `cast`:

Para as versões do SQL de 08/10/2015 e 23/03/2016

Palavra-chave	Resultado
`String`	Aplica valor para `String`.
Nvarchar	Aplica valor para `String`.
Texto	Aplica valor para `String`.
Ntext	Aplica valor para `String`.
varchar	Aplica valor para `String`.
`Int`	Aplica valor para `Int`.
Inteiro	Aplica valor para `Int`.

Palavra-chave	Resultado
Duplo	Converte o valor para `Decimal` (com precisão dupla).

Além disso, para as versões do SQL de 23/03/2016

Palavra-chave	Resultado
`Decimal`	Aplica valor para `Decimal`.
Bool	Aplica valor para `Boolean`.
`Boolean`	Aplica valor para `Boolean`.

Regras de cast:

Cast para decimal

Tipo de argumento	Resultado
`Int`	Um `Decimal` sem ponto decimal.
`Decimal`	O valor de origem. **Note** Com o SQL V2 (23/03/2016), valores numéricos que são números inteiros, como `10.0`, retornam um valor `Int` (`10`) em vez do valor `Decimal` esperado (`10.0`). Para converter de forma confiável valores numéricos inteiros como valores `Decimal`, use o SQL V1 (08/10/2015) para a instrução de consulta de regra.

Tipo de argumento	Resultado
`Boolean`	verdadeiro = 1,0, falso = 0,0.
`String`	Tenta analisar a string como um `Decimal`. A AWS IoT tenta analisar as strings correspondendo à regex: ^-?\d+(\.\d+)?((?i)E-?\d+)?$. "0", "-1,2" e "5E-12" são exemplos de strings que são automatic amente convertidas em decimais.
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Undefined` .
Não definido	`Undefined` .

Cast para int

Tipo de argumento	Resultado
`Int`	O valor de origem.
`Decimal`	O valor de origem arredondado para o `Int` abaixo mais próximo.
`Boolean`	verdadeiro = 1,0, falso = 0,0.
`String`	Tenta analisar a string como um `Decimal`. A AWS IoT tenta analisar as strings correspondendo à regex: ^-?\d+(\.\d+)?((?i)E-?\d+)?$. "0", "-1,2", "5E-12" são exemplos de strings que são automatic amente convertidas em decimais. A AWS IoT tenta converter a string em `Decimal` e arredondar para o `Int` mais próximo.

Tipo de argumento	Resultado
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Undefined` .
Não definido	`Undefined` .

Aplicar para **`Boolean`**

Tipo de argumento	Resultado
`Int`	0 = Falso, qualquer_valor_não_zero = Verdadeiro.
`Decimal`	0 = Falso, qualquer_valor_não_zero = Verdadeiro.
`Boolean`	O valor de origem.
`String`	"true" = Verdadeiro e "false" = Falso (não diferencia maiúsculas de minúsculas). Outros valores de string = `Undefined` .
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Undefined` .
Não definido	`Undefined` .

Cast para string

Tipo de argumento	Resultado
`Int`	Uma representação de string do `Int` em notação padrão.
`Decimal`	Uma string representando o valor `Decimal`, possivelmente em notação científica.
`Boolean`	"verdadeiro" ou "falso", todas as letras minúsculas.
`String`	O valor de origem.
Array	A matriz serializada para JSON. A string resultante é uma lista separada por vírgulas, entre colchetes. `String` tem aspas. `Decimals`, `Ints` e `Booleans` não.
Objeto	O objeto serializado para JSON. A string JSON é uma lista separada por vírgulas de pares de chave/valor e começa e termina com chaves. `String` tem aspas. `Decimals`, `Ints`, `Booleans` e `Null` não.
Null	`Undefined` .
Não definido	`Undefined` .

ceil(Decimal)

Arredonda o `Decimal` fornecido para o `Int` acima mais próximo. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplos:

`ceil(1.2)` = 2

`ceil(-1.2)` = -1

Tipo de argumento	Resultado
`Int`	`Int`, o valor do argumento.
`Decimal`	`Int`, o valor `Decimal` arredondado para o `Int` acima mais próximo.
`String`	`Int`. A string será convertida em `Decimal` e arredondada para o `Int` acima mais próximo. Se a string não puder ser convertida em um `Decimal`, o resultado será `Undefined` .
Outros valores	`Undefined` .

chr(String)

Gera o caractere ASCII que corresponde ao dado argumento `Int`. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplos:

`chr(65)` = "A".

`chr(49)` = "1".

Tipo de argumento	Resultado
`Int`	O caractere correspondente ao valor ASCII especificado. Se o argumento não for um valor ASCII válido, o resultado será `Undefined` .
`Decimal`	O caractere correspondente ao valor ASCII especificado. O argumento `Decimal` é arredondado para o `Int` abaixo mais

Tipo de argumento	Resultado
	próximo. Se o argumento não for um valor ASCII válido, o resultado será `Undefined` .
`Boolean`	`Undefined` .
`String`	Se a `String` puder ser convertida em `Decimal`, ela será arredondada para o `Int` mais próximo. Se o argumento não for um valor ASCII válido, o resultado será `Undefined` .
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Undefined` .
Outros valores	`Undefined` .

clientid()

Gera o ID do MQTT cliente que envia a mensagem ou n/a se a mensagem não foi enviada por MQTT. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo:

`clientid()` = "123456789012"

concat()

Concatena as matrizes ou strings. Essa função aceita qualquer número de argumentos e gera uma `String` ou uma `Array`. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplos:

`concat()` = `Undefined`.

`concat(1)` = "1".

`concat([1, 2, 3], 4)` = [1, 2, 3, 4].

`concat([1, 2, 3], "hello")` = [1, 2, 3, "hello"]

`concat("con", "cat")` = "concat"

`concat(1, "hello")` = "1hello"

`concat("he","is","man")` = "heisman"

`concat([1, 2, 3], "hello", [4, 5, 6])` = [1, 2, 3, "hello", 4, 5, 6]

Número de argumentos	Resultado
0	`Undefined` .
1	O argumento é gerado sem modificações.
2+	Se algum argumento for uma `Array`, o resultado será uma única matriz que contém todos os argumentos. Se não houver matrizes e pelo menos um argumento for uma `String`, o resultado será a concatenação das representações de `String` de todos os argumentos. Os argumentos são convertidos em strings usando as conversões padrão listadas anteriormente.

cos(Decimal)

Gera o cosseno de um número em radianos. Argumentos `Decimal` são arredondados para dobrar a precisão antes da aplicação da função. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo:

`cos(0)` = 1.

Tipo de argumento	Resultado
`Int`	`Decimal` (com precisão dupla), o cosseno do argumento. Os resultados imaginários são gerados como `Undefined` .
`Decimal`	`Decimal` (com precisão dupla), o cosseno do argumento. Os resultados imaginários são gerados como `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal` (com precisão dupla), o cosseno do argumento. Se a string não puder ser convertida em um `Decimal`, o resultado será `Undefined` . Os resultados imaginários são gerados como `Undefined` .
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Undefined` .
Não definido	`Undefined` .

cosh(Decimal)

Gera o cosseno hiperbólico de um número em radianos. Argumentos `Decimal` são arredondados para dobrar a precisão antes da aplicação da função. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `cosh(2.3)` = 5,037220649268761.

Tipo de argumento	Resultado
`Int`	`Decimal` (com precisão dupla), o cosseno hiperbólico do argumento. Os resultados

Tipo de argumento	Resultado
	imaginários são gerados como `Undefined` .
`Decimal`	`Decimal` (com precisão dupla), o cosseno hiperbólico do argumento. Os resultados imaginários são gerados como `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal` (com precisão dupla), o cosseno hiperbólico do argumento. Se a string não puder ser convertida em um `Decimal`, o resultado será `Undefined` . Os resultados imaginários são gerados como `Undefined` .
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Undefined` .
Não definido	`Undefined` .

decodificar (valor, esquema de decodificação)

Use a função `decode` para decodificar um valor codificado. Se a string decodificada for um documento JSON, um objeto endereçável será retornado. Caso contrário, a string decodificada será retornada como uma string. A função retornará NULL se a string não puder ser decodificada. Essa função é compatível com a decodificação de strings codificadas em base64 e o formato de mensagem Protocol Buffer (protobuf).

Compatível com a versão de 23/03/2016 do SQL e posteriores.

valor

Um valor de string ou qualquer uma das expressões válidas, conforme definido em AWS IoT Referência SQL, que retornam uma string.

decodingScheme

Uma string literal representando o esquema usado para decodificar o valor. Atualmente, apenas há suporte para `'base64'` e `'proto'`.

Decodificação de strings codificadas em base64

Neste exemplo, a carga útil da mensagem inclui um valor codificado.

```
{
    encoded_temp: "eyAidGVtcGVyYXR1cmUiOiAzMyB9Cg=="
}
```

A função `decode` nessa instrução SQL decodifica o valor na carga útil da mensagem.

```
SELECT decode(encoded_temp,"base64").temperature AS temp from 'topic/subtopic'
```

A decodificação do valor `encoded_temp` resulta no seguinte documento JSON válido, que permite que a instrução SELECT leia o valor da temperatura.

```
{ "temperature": 33 }
```

O resultado da instrução SELECT neste exemplo é mostrado aqui.

```
{ "temp": 33 }
```

Se o valor decodificado não fosse um documento JSON válido, o valor decodificado seria retornado como uma string.

Decodificação da carga útil da mensagem protobuf

Você pode usar a função decodificar SQL para configurar uma regra que pode decodificar a carga útil da mensagem protobuf. Para obter mais informações, consulte Decodificação de cargas úteis de mensagens protobuf.

> **Important**
>
> Se você omitir `source-arn` ou `source-account` ao definir permissões para um AWS IoT principal, qualquer um Conta da AWS poderá invocar sua função de decodificação por meio de outras regras. AWS IoT Para proteger sua função, consulte Políticas de bucket no Guia do usuário do Amazon Simple Storage Service.

A assinatura da função tem a seguinte aparência:

```
decode(<ENCODED DATA>, 'proto', '<S3 BUCKET NAME>', '<S3 OBJECT KEY>', '<PROTO NAME>',
 '<MESSAGE TYPE>')
```

`ENCODED DATA`

Especifica os dados codificados por protobuf a serem decodificados. Se toda a mensagem enviada para a regra for de dados codificados por protobuf, você poderá referenciar a carga útil de entrada binária bruta usando `*`. Caso contrário, esse campo deve ser uma string JSON codificada em base 64 e uma referência à string pode ser passada diretamente.

1) Para decodificar uma carga útil bruta de entrada de protobuf binário:

```
decode(*, 'proto', ...)
```

2) Para decodificar uma mensagem codificada por protobuf representada por uma string codificada em base64 'a.b':

```
decode(a.b, 'proto', ...)
```

`proto`

Especifica os dados a serem decodificados em um formato de mensagem protobuf. Se você especificar `base64` em vez de `proto`, essa função decodificará strings codificadas em base64 como JSON.

`S3 BUCKET NAME`

O nome do bucket do Amazon S3 onde você fez upload do arquivo `FileDescriptorSet`.

`S3 OBJECT KEY`

A chave de objeto que especifica o arquivo `FileDescriptorSet` dentro do bucket do Amazon S3.

`PROTO NAME`

O nome do arquivo `.proto` (excluindo a extensão) do qual o arquivo `FileDescriptorSet` foi gerado.

`MESSAGE TYPE`

O nome da estrutura da mensagem protobuf dentro do arquivo `FileDescriptorSet`, com a qual os dados a serem decodificados devem estar em conformidade.

Um exemplo de expressão SQL usando a função decodificar SQL pode ter a seguinte aparência:

```
SELECT VALUE decode(*, 'proto', 's3-bucket', 'messageformat.desc', 'myproto',
 'messagetype') FROM 'some/topic'
```

- `*`

  Representa uma carga de entrada binária, que está em conformidade com o tipo de mensagem protobuf chamado `mymessagetype`.

- `messageformat.desc`

  O arquivo `FileDescriptorSet` armazenado em um bucket do Amazon S3 chamado `s3-bucket`.

- `myproto`

  O arquivo `.proto` original usado para gerar o arquivo `FileDescriptorSet` chamado `myproto.proto`.

- `messagetype`

  O tipo de mensagem chamado `messagetype` (junto com todas as dependências importadas) conforme definido em `myproto.proto`.

encode(value, encodingScheme)

Use a função `encode` para codificar a carga útil, que provavelmente serão dados não JSON, em sua representação de string com base no esquema de codificação. Compatível com a versão de 23/03/2016 do SQL e posteriores.

valor

Qualquer uma das expressões válidas, conforme definido em AWS IoT Referência SQL. Você pode especificar * para codificar toda a carga útil, independentemente de estar no formato JSON. Se você fornecer uma expressão, o resultado da avaliação será convertido em uma string antes de ser codificado.

encodingScheme

Uma string literal que representa o esquema de codificação que você deseja usar. No momento, só há compatibilidade com `'base64'`.

endswith(String, String)

Gera um `Boolean` indicando se o primeiro argumento de `String` termina com o segundo argumento de `String`. Se um dos argumentos for `Null` ou `Undefined`, o resultado será `Undefined`. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `endswith("cat","at")` = verdadeiro.

Tipo de argumento 1	Tipo de argumento 2	Resultado
`String`	`String`	Verdadeiro se o primeiro argum terminar no segundo argumento contrário, falso.
Outros valores	Outros valores	Os dois argumentos são conver em strings usando as regras pa conversão. Verdadeiro se o prim argumento terminar no segundo . Caso contrário, falso. Se um d argumentos for `Null` ou `Undef` resultado será `Undefined` .

exp(Decimal)

Gera e elevado ao argumento `Decimal`. Argumentos `Decimal` são arredondados para dobrar a precisão antes da aplicação da função. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `exp(1)` = e.

Tipo de argumento	Resultado
`Int`	`Decimal` (com precisão dupla), argumento e ^.
`Decimal`	`Decimal` (com precisão dupla), argumento e ^.
`String`	`Decimal` (com precisão dupla), argumento e ^. Se a `String` não puder ser convertid a em um `Decimal`, o resultado será `Undefined` .
Outros valores	`Undefined` .

floor(Decimal)

Arredonda o `Decimal` fornecido para o `Int` abaixo mais próximo. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplos:

`floor(1.2)` = 1

`floor(-1.2)` = -2

Tipo de argumento	Resultado
`Int`	`Int`, o valor do argumento.

Tipo de argumento	Resultado
`Decimal`	`Int`, o valor `Decimal` é arredondado para o `Int` abaixo mais próximo.
`String`	`Int`. A string será convertida em `Decimal` e arredondada para o `Int` abaixo mais próximo. Se a string não puder ser convertida em um `Decimal`, o resultado será `Undefined` .
Outros valores	`Undefined` .

obter

Extrai um valor de um tipo de coleção (Matriz, String, Objeto). Nenhuma conversão é aplicada ao primeiro argumento. A conversão se aplica como documentado na tabela ao segundo argumento. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplos:

`get(["a", "b", "c"], 1)` = "b"

`get({"a":"b"}, "a")` = "b"

`get("abc", 0)` = "a"

Tipo de argumento 1	Tipo de argumento 2	Resultado
Array	Qualquer tipo (convertido em `Int`)	O item no índice com base 0 da fornecido pelo segundo argume (convertido em `Int`). Se não for converter, o resultado será `Und` . Se o índice estiver fora dos li `Array` (negativo ou >= matriz.c), o resultado será `Undefined`
String	Qualquer tipo (convertido em `Int`)	O caractere no índice com base string fornecido pelo segundo a

Tipo de argumento 1	Tipo de argumento 2	Resultado
		(convertido em `Int`). Se não for converter, o resultado será Und . Se o índice estiver fora dos li string (negativo ou >= string.cor o resultado será `Undefined` .
Objeto	`String` (nenhuma conversão é aplicada)	O valor armazenado no primeiro de argumento correspondente à de string fornecida como o segu argumento.
Outros valores	Qualquer valor	`Undefined` .

get_dynamodb (nome da tabela,,,,, RoLearn partitionKeyName) partitionKeyValue sortKeyName sortKeyValue

Recupera dados de uma tabela do DynamoDB. `get_dynamodb()` permite consultar uma tabela do DynamoDB enquanto uma regra é avaliada. É possível filtrar ou aumentar cargas de mensagens usando dados recuperados do DynamoDB. Compatível com a versão de 23/03/2016 do SQL e posteriores.

`get_dynamodb()` utiliza os seguintes parâmetros:

tableName

O nome da tabela do DynamoDB a ser consultada.

partitionKeyName

O nome da chave da partição. Para obter mais informações, consulte Chaves do DynamoDB.

partitionKeyValue

O valor da chave de partição usada para identificar um registro. Para obter mais informações, consulte Chaves do DynamoDB.

sortKeyName

(Opcional) O nome da chave de classificação. Esse parâmetro será necessário somente se a tabela do DynamoDB consultada usar uma chave composta. Para obter mais informações, consulte Chaves do DynamoDB.

sortKeyValue

(Opcional) O valor da chave de classificação. Esse parâmetro será necessário somente se a tabela do DynamoDB consultada usar uma chave composta. Para obter mais informações, consulte Chaves do DynamoDB.

roleArn

O ARN do perfil do IAM que concede acesso à tabela do DynamoDB. O mecanismo de regras assume essa função para acessar a tabela do DynamoDB em seu nome. Evite usar uma função excessivamente permissiva. Conceda à função apenas as permissões necessárias para a regra. O exemplo de política a seguir concede acesso a uma tabela do DynamoDB.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dynamodb:GetItem",
            "Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/table-
name"
        }
    ]
}
```

Como um exemplo de como é possível usar `get_dynamodb()`, digamos que você tenha uma tabela do DynamoDB que contém o ID do dispositivo e as informações de localização para todos os dispositivos conectados ao AWS IoT. A instrução SELECT a seguir usa a função `get_dynamodb()` para recuperar o local para o ID do dispositivo especificado:

```
SELECT *, get_dynamodb("InServiceDevices", "deviceId", id,
"arn:aws:iam::12345678910:role/getdynamo").location AS location FROM 'some/
topic'
```

> **Note**
>
> - É possível chamar `get_dynamodb()` no máximo uma vez por instrução SQL. Chamar `get_dynamodb()` várias vezes em uma única instrução SQL faz com que a regra seja encerrada sem chamar nenhuma ação.

get_mqtt_property (nome)

Faz referência a qualquer um dos seguintes MQTT5 cabeçalhos: `contentTypepayLoadFormatIndicator`,`responseTopic`, e. `correlationData` Essa função usa qualquer uma das seguintes strings literais como argumento: `content_type`, `format_indicator`, `response_topic` e `correlation_data`. Para obter mais informações, consulte a tabela de Argumentos de função a seguir.

contentType

String: Uma string codificada em UTF-8 que descreve o conteúdo da mensagem de publicação.

payLoadFormatIndicador

String: Um valor de string Enum que indica se a carga útil está formatada como UTF-8. Os valores válidos são `UNSPECIFIED_BYTES` e `UTF8_DATA`.

responseTopic

String: Uma string codificada em UTF-8 que é usada como nome de tópico para uma mensagem de resposta. O tópico da resposta é usado para descrever o tópico que o destinatário deve publicar como parte do fluxo solicitação-resposta. O tópico não deve conter caracteres curinga.

correlationData

String: Os dados binários codificados em base64 usados pelo remetente da mensagem de solicitação para identificar para qual solicitação é a mensagem de resposta.

A tabela a seguir mostra os argumentos de função aceitáveis e os tipos de retorno associados para a `get_mqtt_property` função:

Argumentos de função

SQL	Tipo de dados retornado (se presente)	Tipo de dados retornado (se não presente)
`get_mqtt_property( "format_indicator")`	Cadeia de caracteres (UNSPECIFIED_BYTES ou _DATA) UTF8	String (UNSPECIFIED_BYTES)
`get_mqtt_property( "content_type")`	String	Não definido
`get_mqtt_property( "response_topic")`	String	Não definido
`get_mqtt_property( "correlation_data")`	String codificada em base64	Não definido
`get_mqtt_property( "some_invalid_name ")`	Não definido	Não definido

O exemplo a seguir Rules SQL faz referência a qualquer um dos seguintes MQTT5 cabeçalhos: `contentTypepayLoadFormatIndicator`,`responseTopic`, e. `correlationData`

```
SELECT *, get_mqtt_property('content_type') as contentType,
          get_mqtt_property('format_indicator') as payloadFormatIndicator,
          get_mqtt_property('response_topic') as responseTopic,
          get_mqtt_property('correlation_data') as correlationData
FROM 'some/topic'
```

get_or_default (expressão, DefaultValue)

Retorna o valor padrão no segundo parâmetro, se especificado, ou então retorna indefinido, quando a expressão no primeiro parâmetro retorna nula, indefinida ou falha. Compatível com a versão de 23/03/2016 do SQL e posteriores.

> **Important**
>
> `get_or_default`não oferece suporte a cargas não JSON diretamente como estão. Se você estiver usando uma carga não JSON, use as `encode` funções ou. `decode`

`get_or_default()` utiliza os seguintes parâmetros:

expressão

Qualquer expressão válida contendoTipos de dados, FunçõesLiterais,Consultas de objeto aninhado, Variáveis ouExtensões JSON.

defaultValue

(Opcional) Qualquer expressão válida contendo Tipos de dadosFunções,Literais,Consultas de objeto aninhado, Variáveis ouExtensões JSON. Esse é o valor a ser retornado sempre que o primeiro argumento retornar nulo, indefinido ou falhar.

> **Note**
>
> Funções que buscam dados de recursos de propriedade do cliente, como get_secret, get_dynamodb, aws_lambda, get_thing_shadow, decode-protobuf e machinelearning_predict, não são permitidas no parâmetro DefaultValue.

A tabela a seguir mostra argumentos de função aceitáveis para cada argumento e suas saídas associadas:

Primeiro argumento	Segundo argumento	Output
Avaliação bem-sucedida	Qualquer valor ou não especificado	O valor do primeiro argumento.
Indefinido, nulo ou falha	Qualquer valor, incluindo indefinido ou nulo	O valor do segundo argumento.
Indefinido, nulo ou falha	não especificado	`Undefined`

Exemplos:

Exemplo 1:

O exemplo a seguir fornece um valor DefaultValue se uma tabela ou consulta do DynamoDB falhar:

```
SELECT
    device_id,
    get_or_default(
        get_dynamodb("DeviceConfig", "deviceId", nonExistentId,
 "arn:aws:iam::123456789012:role/ROLE_NAME"),
        {"mode": "standard", "timeout": 30, "enabled": true }
    ) as config
FROM 'device/telemetry'
```

Exemplo 2:

O exemplo a seguir fornece um valor padrão seguro “DESCONHECIDO” se o status for indefinido:

```
SELECT
  get_or_default( CASE status
    WHEN 'active' THEN 'GOOD'
    WHEN 'inactive' THEN 'BAD'/
    ELSE 'UNKNOWN'
  END, 'UNKNOWN') as status_category
FROM 'topic/subtopic'
```

Exemplo 3:

O exemplo a seguir mostra como você também pode usar get_or_default com um único parâmetro. Isso é útil em cenários em que você pode não ter um valor padrão claro, mas não deseja que a execução da regra falhe.

```
SELECT
  get_dynamodb("DeviceConfig", "deviceId", nonExistentId,
 "arn:aws:iam::123456789012:role/ROLE_NAME") as config
FROM 'device/telemetry'
```

Se a pesquisa do DynamoDB falhar, a execução da regra falhará e nenhuma ação será executada. Se o seguinte SQL for usado em vez disso:

```
SELECT
  get_or_default(get_dynamodb("DeviceConfig", "deviceId", nonExistentId,
 "arn:aws:iam::123456789012:role/ROLE_NAME")) as config
```

```
FROM 'device/telemetry'
```

A instrução get_or_default será avaliada como`Undefined`, portanto, neste exemplo, a instrução SELECT em geral será avaliada `{}` e qualquer ação de regra será tentada.

> **Important**
>
> Recomendamos seguir estas melhores práticas para manter a segurança ao usar essa função:
>
> - Evite usar segredos codificados em definições de regras, incluindo valores padrão
> - Use AWS Secrets Manager para gerenciar informações confidenciais

get_registry_data (RegistryAPI, ThingName, RoLearn)

Recupera dados de registro de AWS IoT coisas em uma AWS IoT regra. Você pode ler dados do registro (como atributos, tipo de coisa e grupos de coisas aos quais um dispositivo pertence) e usar essas informações para filtrar, enriquecer ou rotear mensagens dinamicamente. Compatível com a versão de 23/03/2016 do SQL e posteriores.

`get_registry_data()` utiliza os seguintes parâmetros:

API de registro

A API de registro que está sendo chamada. Os valores válidos são `DescribeThing` e `ListThingGroupsForThing`. Esses valores devem ser cadeias de caracteres constantes.

thingName

Cadeia de caracteres: o nome da coisa cujos dados de registro você deseja recuperar.

roleArn

String: um ARN de função com permissão de `iot:DescribeThing` and/or `iot:ListThingGroupsForThing` permissão com base na API que está sendo chamada.

O formato de resposta da `get_registry_data` função é o mesmo da API de registro chamada. Para obter mais informações, consulte DescribeThing e ListThingGroupsForThing APIs.

Exemplo:

Você pode recuperar informações do tipo de coisa para permitir a filtragem das mensagens de eventos do AWS IoT Core ciclo de vida das coisas (com o nome da coisa correspondendo ao ID do cliente MQTT) onde está o tipo da coisa. `testenv`

```
SELECT *
FROM '$aws/events/lifecycle/+'
WHERE
    get_registry_data("DescribeThing",clientId,[roleArn]).thingTypeName='testenv'
```

Exemplo:

Você pode recuperar atributos de um dispositivo com nome de coisa `sensor1` para todas as mensagens enviadas por seu dispositivo `gateway1` de gateway.

```
SELECT *, get_registry_data("DescribeThing","sensor1",
[roleArn]).attributes.temperature_threhold AS device1_tempthreshold
FROM home1/gateway1/sensor1/#
```

> **Note**
>
> - Você pode chamar no máximo `get_registry_data()` uma vez por instrução SQL e modelos de substituição para ações e ações de erro.

get_secret (secretId, secretType, chave, roleArn)

Recupera o valor do campo criptografado `SecretString` ou `SecretBinary` da versão atual de um segredo em AWS Secrets Manager. Para obter mais informações sobre como criar e manter segredos CreateSecretUpdateSecret, consulte, PutSecretValuee.

`get_secret()` utiliza os seguintes parâmetros:

secretId

String: O nome do recurso da Amazon (ARN) ou o nome amigável do segredo a ser recuperado.

secretType

String: O tipo de segredo. Valores válidos: `SecretString` | `SecretBinary`.

SecretString

- Para segredos que você cria como objetos JSON usando o APIs AWS CLI, o ou o AWS Secrets Manager console:
  - Se você especificar um valor para o parâmetro `key`, essa função retornará o valor da chave especificada.
  - Se você não especificar um valor para o parâmetro `key`, essa função retornará o objeto JSON inteiro.
- Para segredos que você cria como objetos não JSON usando o APIs ou o: AWS CLI
  - Se você especificar um valor para o parâmetro `key`, essa função falhará com uma exceção.
  - Se você não especificar um valor para o parâmetro `key`, essa função retornará o conteúdo do segredo.

SecretBinary

- Se você especificar um valor para o parâmetro `key`, essa função falhará com uma exceção.
- Se você não especificar um valor para o parâmetro `key`, essa função retornará o valor secreto como uma string UTF-8 codificada em base64.

key

(Opcional) String: o nome da chave dentro de um objeto JSON armazenado no campo `SecretString` de um segredo. Use esse valor quando quiser recuperar somente o valor de uma chave armazenada em um segredo em vez do objeto JSON inteiro.

Se você especificar um valor para esse parâmetro e o segredo não contiver um objeto JSON dentro do campo `SecretString`, essa função falhará com uma exceção.

roleArn

String: um ARN de função com permissões `secretsmanager:GetSecretValue` e `secretsmanager:DescribeSecret`.

> **Note**
>
> Essa função sempre retorna a versão atual do segredo (a versão com a tag `AWSCURRENT`). O mecanismo de AWS IoT regras armazena cada segredo em cache por até 15 minutos. Como resultado, o mecanismo de regras pode levar até 15 minutos para atualizar um segredo. Isso

significa que, se você recuperar um segredo até 15 minutos após uma atualização com AWS Secrets Manager, essa função poderá retornar a versão anterior.
Essa função não é medida, mas são cobradas AWS Secrets Manager taxas. Por causa do mecanismo secreto de armazenamento em cache, o mecanismo de regras chama AWS Secrets Manager ocasionalmente. Como o mecanismo de regras é um serviço totalmente distribuído, você pode ver várias chamadas de API do Secrets Manager do mecanismo de regras durante a janela de 15 minutos de armazenamento em cache.

Exemplos:

Você pode usar a função `get_secret` em um cabeçalho de autenticação em uma ação de regra HTTPS, como no exemplo de autenticação de chave de API a seguir.

```
"API_KEY": "${get_secret('API_KEY', 'SecretString', 'API_KEY_VALUE',
 'arn:aws:iam::12345678910:role/getsecret')}"
```

Para obter mais informações sobre as ações de regra HTTPS, consulte the section called “HTTP”.

get_thing_shadow (ThingName, ShadoWName, RoLearn)

Retorna a sombra do objeto especificado. Compatível com a versão de 23/03/2016 do SQL e posteriores.

thingName

String: o nome do objeto que você quer recuperar a sombra.

shadowName

(Opcional) String: o nome da sombra. Esse parâmetro é necessário somente ao fazer referência a sombras nomeadas.

roleArn

String: um ARN de função com permissão `iot:GetThingShadow`.

Exemplos:

Quando usado com uma sombra nomeada, forneça o parâmetro `shadowName`.

```
SELECT * from 'topic/subtopic'
WHERE
    get_thing_shadow("MyThing","MyThingShadow","arn:aws:iam::123456789012:role/
AllowsThingShadowAccess")
    .state.reported.alarm = 'ON'
```

Quando usado com uma sombra sem nome, omita o parâmetro `shadowName`.

```
SELECT * from 'topic/subtopic'
WHERE
    get_thing_shadow("MyThing","arn:aws:iam::123456789012:role/
AllowsThingShadowAccess")
    .state.reported.alarm = 'ON'
```

get_user_properties () userPropertyKey

Referencia as propriedades do usuário, que é um tipo de cabeçalho de propriedade suportado no MQTT5.

userProperty

String: Uma propriedade de usuário é um par de chave/valor. Essa função usa a chave como argumento e retorna uma matriz de todos os valores que correspondem à chave associada.

Argumentos de função

Para as seguintes propriedades do usuário nos cabeçalhos das mensagens:

Chave	Valor
alguma chave	algum valor
uma chave diferente	um valor diferente
alguma chave	valor com chave duplicada

A tabela a seguir mostra o comportamento esperado do SQL:

SQL	Tipo de dados de retorno	Valor de dados de retorno
get_user_properties('alguma chave')	Matriz de strings	`['some value', 'value with duplicate key']`
get_user_properties('outra chave')	Matriz de strings	`['a different value']`
get_user_properties( )	Matriz de objetos de chave-valor	`[{'"some key": "some value"'}, {"other key": "a different value"}, {"some key": "value with duplicate key"}]`
get_user_properties('chave inexistente')	Não definido	

O exemplo a seguir Rules SQL faz referência às propriedades do usuário (um tipo de cabeçalho de MQTT5 propriedade) na carga útil:

```
SELECT *, get_user_properties('user defined property key') as userProperty
FROM 'some/topic'
```

Funções de hashing

AWS IoT fornece as seguintes funções de hashing:

- md2
- md5
- sha1
- sha224
- sha256
- sha384
- sha512

Todas as funções de hash esperam um argumento de string. O resultado é o valor de hash da string em questão. Conversões de string padrão se aplicam a argumentos de não string. Todas as funções de hash são compatíveis com a versão de 08/10/2015 do SQL e posteriores.

Exemplos:

`md2("hello")` = "a9046c73e00331af68917d3804f70655"

`md5("hello")` = "5d41402abc4b2a76b9719d911017c592"

indexof(String, String)

Apresenta o primeiro índice (base em 0) do segundo argumento como uma substring no primeiro argumento. Os dois argumentos são esperados como strings. Os argumentos que não são strings estão sujeitos às regras padrão de conversão de string. Essa função não se aplica a matrizes, apenas a strings. Compatível com a versão de 23/03/2016 do SQL e posteriores.

Exemplos:

`indexof("abcd", "bc")` = 1

isNull()

Retorna verdadeiro se o argumento é o valor `Null`. Compatível com a versão de 23/03/2016 do SQL e posteriores.

Exemplos:

`isNull(5)` = false.

`isNull(Null)` = true.

Tipo de argumento	Resultado
`Int`	false
`Decimal`	false
`Boolean`	false

Tipo de argumento	Resultado
`String`	false
`Array`	false
`Object`	false
`Null`	verdadeiro
`Undefined`	false

isUndefined()

Retorna verdadeiro se o argumento for `Undefined`. Compatível com a versão de 23/03/2016 do SQL e posteriores.

Exemplos:

`isUndefined(5)` = false.

`isUndefined(floor([1,2,3]))` = true.

Tipo de argumento	Resultado
`Int`	false
`Decimal`	false
`Boolean`	false
`String`	false
`Array`	false
`Object`	false
`Null`	false
`Undefined`	verdadeiro

length(String)

Gera o número de caracteres na string fornecida. As regras padrão de conversão se aplicam a argumentos de não `String`. Compatível com a versão de 23/03/2016 do SQL e posteriores.

Exemplos:

`length("hi")` = 2

`length(false)` = 5

ln(Decimal)

Gera o logaritmo natural do argumento. Argumentos `Decimal` são arredondados para dobrar a precisão antes da aplicação da função. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `ln(e)` = 1.

Tipo de argumento	Resultado
`Int`	`Decimal` (com precisão dupla), o log natural do argumento.
`Decimal`	`Decimal` (com precisão dupla), o log natural do argumento.
`Boolean`	`Undefined` .
`String`	`Decimal` (com precisão dupla), o log natural do argumento. Se a string não puder ser convertida em um `Decimal`, o resultado será `Undefined` .
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Undefined` .
Não definido	`Undefined` .

log(Decimal)

Gera o logaritmo na base 10 do argumento. Argumentos `Decimal` são arredondados para dobrar a precisão antes da aplicação da função. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `log(100)` = 2,0.

Tipo de argumento	Resultado
`Int`	`Decimal` (com precisão dupla), o log de base 10 do argumento.
`Decimal`	`Decimal` (com precisão dupla), o log de base 10 do argumento.
`Boolean`	`Undefined .`
`String`	`Decimal` (com precisão dupla), o log de base 10 do argumento. Se a `String` não puder ser convertida em um `Decimal`, o resultado será `Undefined .`
Array	`Undefined .`
Objeto	`Undefined .`
Null	`Undefined .`
Não definido	`Undefined .`

lower(String)

Gera a versão em letra minúscula da `String` fornecida. Argumentos de não string são convertidos em strings usando as regras padrão de conversão. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplos:

`lower("HELLO")` = "hello".

`lower(["HELLO"])` = "[\"hello\"]".

lpad(String, Int)

Gera o argumento de `String`, preenchido no lado esquerdo com o número de espaços especificado pelo segundo argumento. O argumento `Int` deve estar entre 0 e 1000. Se o valor fornecido estiver fora desse intervalo válido, o argumento será definido para o valor válido mais próximo (0 ou 1.000). Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplos:

`lpad("hello", 2)` = " hello".

`lpad(1, 3)` = " 1"

Tipo de argumento 1	Tipo de argumento 2	Resultado
`String`	`Int`	`String`, a `String` preenchida no lado esquerdo com um núme espaços igual ao `Int` fornecido
`String`	`Decimal`	O argumento `Decimal` será arr do para o `Int` abaixo mais próx `String` será preenchida à esqu número especificado de espaço
`String`	`String`	O segundo argumento será con um `Decimal`, que é arredondad o `Int` abaixo mais próximo, e a será preenchida com o número especificados à esquerda. Se o argumento não puder ser conve `Int`, o resultado será `Undefin`
Outros valores	`Int`/`Decimal`/`String`	O primeiro valor será convertido `String` usando as conversões função LPAD será aplicada nes Se não for possível fazer a conv resultado será `Undefined` .

Tipo de argumento 1	Tipo de argumento 2	Resultado
Qualquer valor	Outros valores	`Undefined` .

ltrim(String)

Remove todos os espaços em branco iniciais (caracteres de tabulação e espaços) da `String` fornecida. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo:

`Ltrim(" h i ")` = "hi ".

Tipo de argumento	Resultado
`Int`	A representação de `String` do `Int` com todos os espaços em branco iniciais removidos.
`Decimal`	A representação de `String` do `Decimal` com todos os espaços em branco iniciais removidos.
`Boolean`	A representação de `String` do booliano ("verdadeiro" ou "falso") com todos os espaços em branco iniciais removidos.
`String`	O argumento com todos os espaços em branco iniciais removidos.
Array	A representação de `String` de `Array` (usando as regras padrão de conversão) com todos os espaços em branco iniciais removidos.
Objeto	A representação de `String` do objeto (usando as regras padrão de conversão)

Tipo de argumento	Resultado
	com todos os espaços em branco iniciais removidos.
Null	`Undefined` .
Não definido	`Undefined` .

machinelearning_predict(modelId, roleArn, registro)

Use a `machinelearning_predict` função para fazer previsões usando os dados de uma mensagem MQTT com base em um modelo de SageMaker IA da Amazon. Compatível com a versão de 08/10/2015 do SQL e posteriores. Os argumentos da função `machinelearning_predict` são:

modelId

O ID do modelo no qual executar a previsão. O endpoint em tempo real do modelo deve ser habilitado.

roleArn

O perfil do IAM; que tem uma política com as permissões `machinelearning:Predict` e `machinelearning:GetMLModel` e que permite acessar o modelo no qual a previsão é executada.

registro

Os dados a serem passados para a API SageMaker AI Predict. Isso deve ser representado como um único objeto JSON de camada. Se o registro for um objeto JSON de vários níveis, o registro será aplainado ao serializar seus valores. Por exemplo, o seguinte JSON:

```
{ "key1": {"innerKey1": "value1"}, "key2": 0}
```

se tornaria:

```
{ "key1": "{\"innerKey1\": \"value1\"}", "key2": 0}
```

A função gera um objeto JSON com os seguintes campos:

predictedLabel

A classificação da entrada com base no modelo.

detalhes

Contém os seguintes atributos:

PredictiveModelType

O tipo do modelo. Os valores válidos são REGRESSION, BINARY, MULTICLASS.

Algoritmo

O algoritmo usado pela SageMaker IA para fazer previsões. O valor deve ser SGD.

predictedScores

Contém a pontuação de classificação bruta correspondente a cada rótulo.

predictedValue

O valor previsto pela SageMaker IA.

mod(Decimal, Decimal)

Gera o restante da divisão do primeiro argumento pelo segundo argumento. Equivale a remainder(Decimal, Decimal). Você também pode usar "%" como um operador infixo para a mesma funcionalidade do módulo. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `mod(8, 3)` = 2.

Operando esquerdo	Operando direito	Output
`Int`	`Int`	`Int`, o primeiro argumento é mo segundo argumento.
`Int/Decimal`	`Int/Decimal`	`Decimal`, o primeiro argumento do segundo operando.
`String/Int/Decimal`	`String/Int/Decimal`	Se todas as strings forem conve em decimais, o resultado será o argumento como módulo do seg argumento. Caso contrário, Und

Operando esquerdo	Operando direito	Output
Outros valores	Outros valores	`Undefined` .

nanquim (,) AnyValue AnyValue

Gera o primeiro argumento se for um `Decimal` válido. Caso contrário, o segundo argumento é retornado. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `Nanvl(8, 3)` = 8.

Tipo de argumento 1	Tipo de argumento 2	Output
Não definido	Qualquer valor	O segundo argumento.
Null	Qualquer valor	O segundo argumento.
`Decimal` (NaN)	Qualquer valor	O segundo argumento.
`Decimal` (não NaN)	Qualquer valor	O primeiro argumento.
Outros valores	Qualquer valor	O primeiro argumento.

newuuid()

Gera uma UUID aleatória de 16 bytes. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `newuuid()` = `123a4567-b89c-12d3-e456-789012345000`

numbytes(String)

Gera o número de bytes na codificação UTF-8 da string fornecida. As regras padrão de conversão se aplicam a argumentos de não `String`. Compatível com a versão de 23/03/2016 do SQL e posteriores.

Exemplos:

`numbytes("hi")` = 2

`numbytes("€") = 3`

parse_time(String, Long, [String])

Use a `parse_time` função para formatar um carimbo de data/hora em um formato legível por humanos date/time . Compatível com a versão de 23/03/2016 do SQL e posteriores. Para converter uma string de carimbo de data/hora em milissegundos, consulte time_to_epoch(String, String).

A função `parse_time` espera os seguintes argumentos:

pattern

(String) Um date/time padrão que segue os formatos Joda-Time.

timestamp

(Long) O tempo a ser formatado em milissegundos desde o epoch do Unix. Consulte a função timestamp().

timezone

(String) O fuso horário da data/hora formatada. O padrão é "UTC". A função oferece suporte para fusos horários Joda-Time. Esse argumento é opcional.

Exemplos:

Quando essa mensagem for publicada no tópico "A/B", a carga útil `{"ts": "1970.01.01 AD at 21:46:40 CST"}` será enviada para o bucket do S3:

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT parse_time(\"yyyy.MM.dd G 'at' HH:mm:ss z\", 100000000,
 'America/Belize' ) as ts FROM 'A/B'",

        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "s3": {
                    "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                    "bucketName": "BUCKET_NAME",
                    "key": "KEY_NAME"
```

```
                }
            }
        ],
        "ruleName": "RULE_NAME"
    }
}
```

Quando essa mensagem for publicada no tópico 'A/B', uma carga útil semelhante a `{"ts": "2017.06.09 AD at 17:19:46 UTC"}` (mas com data/hora atual) será enviada para o bucket S3:

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT parse_time(\"yyyy.MM.dd G 'at' HH:mm:ss z\", timestamp() ) as ts
 FROM 'A/B'",
        "awsIotSqlVersion": "2016-03-23",
        "ruleDisabled": false,
        "actions": [
            {
                "s3": {
                    "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                    "bucketName": "BUCKET_NAME",
                    "key": "KEY_NAME"
                }
            }
        ],
        "ruleName": "RULE_NAME"
    }
}
```

`parse_time()` também pode ser usado como um modelo de substituição. Por exemplo, quando esta mensagem for publicada no tópico "A/B", a carga útil será enviada para o bucket S3 com chave = "2017":

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT * FROM 'A/B'",
        "awsIotSqlVersion": "2016-03-23",
        "ruleDisabled": false,
        "actions": [{
```

```
            "s3": {
                "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                "bucketName": "BUCKET_NAME",
                "key": "${parse_time('yyyy', timestamp(), 'UTC')}"
            }
        }],
        "ruleName": "RULE_NAME"
    }
}
```

power(Decimal, Decimal)

Gera o primeiro argumento elevado ao segundo argumento. Argumentos `Decimal` são arredondados para dobrar a precisão antes da aplicação da função. Compatível com a versão de 08/10/2015 do SQL e posteriores. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `power(2, 5)` = 32,0.

Tipo de argumento 1	Tipo de argumento 2	Output
`Int/Decimal`	`Int/Decimal`	Um `Decimal` (com precisão du primeiro argumento elevado par do segundo argumento.
`Int/Decimal/String`	`Int/Decimal/String`	Um `Decimal` (com precisão du o primeiro argumento elevado p poder do segundo argumento. C strings são convertidas em deci Se a conversão de alguma `Str` em `Decimal` falhar, o resultado `Undefined` .
Outros valores	Outros valores	`Undefined` .

principal()

Retorna a entidade principal que o dispositivo usa para autenticação, com base em como a mensagem acionadora foi publicada. A tabela a seguir descreve a entidade principal retornada para cada protocolo e método de publicação.

Como a mensagem é publicada	Protocolo	Tipo de credencial
Cliente MQTT	MQTT	Certificado de dispositivo X.509
AWS IoT cliente MQTT do console	MQTT	Usuário ou perfil do IAM
AWS CLI	HTTP	Usuário ou perfil do IAM
AWS IoT SDK do dispositivo	MQTT	Certificado de dispositivo X.509
AWS IoT SDK do dispositivo	Acabou o MQTT WebSocket	Usuário ou perfil do IAM

Os exemplos a seguir mostram os diferentes tipos de valores que `principal()` pode retornar:

- Thumbprint do certificado X.509: `ba67293af50bf2506f5f93469686da660c7c844e7b3950bfb16813e0d31e9373`
- ID do perfil do IAM e nome da sessão: `ABCD1EFG3HIJK2LMNOP5:my-session-name`
- Retorna um ID de usuário: `ABCD1EFG3HIJK2LMNOP5`

rand()

Gera um pseudoaleatório, uniformemente distribuído entre 0,0 e 1,0. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo:

`rand()` = 0,8231909191640703

regexp_matches(String, String)

Retorna verdadeiro se a string (primeiro argumento) contiver uma correspondência para a expressão regular (segundo argumento). Se você usar `|` na expressão regular, use-a com `()`.

Exemplos:

`regexp_matches("aaaa", "a{2,}")` = true.

`regexp_matches("aaaa", "b")` = false.

`regexp_matches("aaa", "(aaa|bbb)")` = true.

`regexp_matches("bbb", "(aaa|bbb)")` = true.

`regexp_matches("ccc", "(aaa|bbb)")` = false.

Primeiro argumento:

Tipo de argumento	Resultado
`Int`	A representação da `String` do `Int`.
`Decimal`	A representação da `String` do `Decimal`.
`Boolean`	A representação da `String` do booleano ("verdadeiro" ou "falso").
`String`	O `String`.
Array	A representação da `String` da `Array` (usando as regras padrão de conversão).
Objeto	A representação da `String` do Objeto (usando as regras padrão de conversão).
Null	`Undefined` .
Não definido	`Undefined` .

Segundo argumento:

Deve ser uma expressão regex válida. Tipos de não string são convertidos em `String` usando as regras padrão de conversão. Dependendo do tipo, a string resultante pode não ser uma expressão regular válida. Se o argumento (convertido) não for uma regex válida, o resultado será `Undefined`.

regexp_replace(String, String, String)

Substitui todas as ocorrências do segundo argumento (expressão regular) no primeiro argumento com o terceiro argumento. Faz referência a grupos de captura com "$". Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo:

`regexp_replace("abcd", "bc", "x")` = "axd".

`regexp_replace("abcd", "b(.*)d", "$1")` = "ac".

Primeiro argumento:

Tipo de argumento	Resultado
`Int`	A representação da `String` do `Int`.
`Decimal`	A representação da `String` do `Decimal`.
`Boolean`	A representação da `String` do booleano ("verdadeiro" ou "falso").
`String`	O valor de origem.
Array	A representação da `String` da `Array` (usando as regras padrão de conversão).
Objeto	A representação da `String` do Objeto (usando as regras padrão de conversão).
Null	`Undefined` .
Não definido	`Undefined` .

Segundo argumento:

Deve ser uma expressão regex válida. Tipos de não string são convertidos em `String` usando as regras padrão de conversão. Dependendo do tipo, a string resultante pode não ser uma expressão regular válida. Se o argumento (convertido) não for uma expressão regex válida, o resultado será `Undefined`.

Terceiro argumento:

Deve ser uma string de substituição regex válida. (Pode fazer referência a grupos de captura.) Tipos de não string são convertidos em `String` usando as regras padrão de conversão. Se o argumento (convertido) não for uma string de substituição regex válida, o resultado será `Undefined`.

regexp_substr(String, String)

Localiza a primeira correspondência do segundo parâmetro (regex) no primeiro parâmetro. Faz referência a grupos de captura com "$". Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo:

`regexp_substr("hihihello", "hi")` = "hi"

`regexp_substr("hihihello", "(hi)*")` = "hihi"

Primeiro argumento:

Tipo de argumento	Resultado
`Int`	A representação da `String` do `Int`.
`Decimal`	A representação da `String` do `Decimal`.
`Boolean`	A representação da `String` do booleano ("verdadeiro" ou "falso").
`String`	O argumento da `String`.
Array	A representação da `String` da `Array` (usando as regras padrão de conversão).
Objeto	A representação da `String` do Objeto (usando as regras padrão de conversão).
Null	`Undefined` .
Não definido	`Undefined` .

Segundo argumento:

Deve ser uma expressão regex válida. Tipos de não string são convertidos em `String` usando as regras padrão de conversão. Dependendo do tipo, a string resultante pode não ser uma expressão regular válida. Se o argumento (convertido) não for uma expressão regex válida, o resultado será `Undefined`.

remainder(Decimal, Decimal)

Gera o restante da divisão do primeiro argumento pelo segundo argumento. Equivale a mod(Decimal, Decimal). Você também pode usar "%" como um operador infixo para a mesma funcionalidade do módulo. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `remainder(8, 3)` = 2.

Operando esquerdo	Operando direito	Output
`Int`	`Int`	`Int`, o primeiro argumento é mo segundo argumento.
`Int/Decimal`	`Int/Decimal`	`Decimal`, o primeiro argumento do segundo operando.
`String/Int/Decimal`	`String/Int/Decimal`	Se todas as strings forem conve em decimais, o resultado será o argumento como módulo do seg argumento. Caso contrário, Und
Outros valores	Outros valores	`Undefined` .

replace(String, String, String)

Substitui todas as ocorrências do segundo argumento no primeiro argumento com o terceiro argumento. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo:

`replace("abcd", "bc", "x")` = `"axd"`.

`replace("abcdabcd", "b", "x")` = `"axcdaxcd"`.

Todos os argumentos

Tipo de argumento	Resultado
`Int`	A representação da `String` do `Int`.
`Decimal`	A representação da `String` do `Decimal`.
`Boolean`	A representação da `String` do booleano ("verdadeiro" ou "falso").
`String`	O valor de origem.
Array	A representação da `String` da `Array` (usando as regras padrão de conversão).
Objeto	A representação da `String` do Objeto (usando as regras padrão de conversão).
Null	`Undefined` .
Não definido	`Undefined` .

rpad(String, Int)

Gera o argumento da string, preenchido no lado direito com o número de espaços especificado no segundo argumento. O argumento `Int` deve estar entre 0 e 1000. Se o valor fornecido estiver fora desse intervalo válido, o argumento será definido para o valor válido mais próximo (0 ou 1.000). Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplos:

`rpad("hello", 2)` = `"hello  "`.

`rpad(1, 3)` = `"1  "`.

Tipo de argumento 1	Tipo de argumento 2	Resultado
`String`	`Int`	A `String` é preenchid a no lado direito

Tipo de argumento 1	Tipo de argumento 2	Resultado
		com um número de espaços igual ao `Int` fornecido.
`String`	`Decimal`	O argumento `Decimal` será arredondado para o `Int` abaixo mais próximo, e a string será preenchida no lado direito com um número de espaços igual ao `Int` fornecido.
`String`	`String`	O segundo argumento é convertido em um `Decimal`, que é arredondado para o `Int` mais próximo. A `String` é preenchida no lado direito com um número de espaços igual ao valor `Int`.
Outros valores	`Int`/`Decimal`/`String`	O primeiro valor será convertido em uma `String` usando as conversões padrão, e a função rpad será aplicada nessa `String`. Se não for possível fazer a conversão, o resultado será `Undefined` .
Qualquer valor	Outros valores	`Undefined` .

round(Decimal)

Arredonda o `Decimal` fornecido para o `Int` mais próximo. Se o `Decimal` for equidistante de dois valores `Int` (por exemplo, 0,5), o `Decimal` será arredondado. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `Round(1.2)` = 1.

`Round(1.5)` = 2.

`Round(1.7)` = 2.

`Round(-1.1)` = -1.

`Round(-1.5)` = -2.

Tipo de argumento	Resultado
`Int`	O argumento.
`Decimal`	`Decimal` é arredondado para o `Int` abaixo mais próximo.
`String`	`Decimal` é arredondado para o `Int` abaixo mais próximo. Se a string não puder ser convertida em um `Decimal`, o resultado será `Undefined` .
Outros valores	`Undefined` .

rtrim(String)

Remove todos os espaços em branco finais (caracteres de tabulação e espaços) da `String` fornecida. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplos:

`rtrim(" h i ")` = "h i"

Tipo de argumento	Resultado
`Int`	A representação da `String` do `Int`.
`Decimal`	A representação da `String` do `Decimal`.
`Boolean`	A representação da `String` do booleano ("verdadeiro" ou "falso").
Array	A representação da `String` da `Array` (usando as regras padrão de conversão).
Objeto	A representação da `String` do Objeto (usando as regras padrão de conversão).
Null	`Undefined` .
Não definido	`Undefined`

sign(Decimal)

Gera o sinal do número fornecido. Quando o sinal do argumento for positivo, 1 será gerado. Quando o sinal do argumento for negativo, -1 será gerado. Se o argumento for 0, 0 será gerado. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplos:

`sign(-7)` = -1.

`sign(0)` = 0.

`sign(13)` = 1.

Tipo de argumento	Resultado
`Int`	`Int`, o sinal do valor `Int`.
`Decimal`	`Int`, o sinal do valor `Decimal`.

Tipo de argumento	Resultado
`String`	`Int`, o sinal do valor `Decimal`. A string é convertida em um valor `Decimal`, e o sinal do valor `Decimal` é gerado. Se a `String` não puder ser convertida em um `Decimal`, o resultado será `Undefined` . Compatível com a versão de 08/10/2015 do SQL e posteriores.
Outros valores	`Undefined` .

sin(Decimal)

Gera o seno de um número em radianos. Argumentos `Decimal` são arredondados para dobrar a precisão antes da aplicação da função. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `sin(0)` = 0,0

Tipo de argumento	Resultado
`Int`	`Decimal` (com precisão dupla), o seno do argumento.
`Decimal`	`Decimal` (com precisão dupla), o seno do argumento.
`Boolean`	`Undefined` .
`String`	`Decimal` (com precisão dupla), o seno do argumento. Se a string não puder ser convertida em um `Decimal`, o resultado será `Undefined` .
Array	`Undefined` .
Objeto	`Undefined` .

Tipo de argumento	Resultado
Null	`Undefined` .
`Undefined`	`Undefined` .

sinh(Decimal)

Gera o seno hiperbólico de um número em radianos. Valores `Decimal` são arredondados para dobrar a precisão antes da aplicação da função. O resultado é um valor `Decimal` de precisão dupla. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `sinh(2.3)` = 4,936961805545957

Tipo de argumento	Resultado
`Int`	`Decimal` (com precisão dupla), o seno hiperbólico do argumento.
`Decimal`	`Decimal` (com precisão dupla), o seno hiperbólico do argumento.
`Boolean`	`Undefined` .
`String`	`Decimal` (com precisão dupla), o seno hiperbólico do argumento. Se a string não puder ser convertida em um `Decimal`, o resultado será `Undefined` .
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Undefined` .
Não definido	`Undefined` .

sourceip()

Recupera o endereço IP de um dispositivo ou do roteador que se conecta a ele. Se o seu dispositivo estiver conectado diretamente à Internet, a função retornará o endereço IP de origem do dispositivo. Se o seu dispositivo estiver conectado a um roteador conectado à Internet, a função retornará o endereço IP de origem do dispositivo. Compatível com a versão SQL 23/03/2016. `sourceip()` não usa nenhum parâmetro.

> **Important**
>
> O endereço IP de origem pública de um dispositivo geralmente é o endereço IP do último gateway de conversão de endereços de rede (NAT), como o roteador ou modem a cabo do seu provedor de serviços de Internet.

Exemplos:

```
sourceip()="192.158.1.38"
```

```
sourceip()="1.102.103.104"
```

```
sourceip()="2001:db8:ff00::12ab:34cd"
```

Exemplo de SQL:

```
SELECT *, sourceip() as deviceIp FROM 'some/topic'
```

Exemplos de como usar a função sourceip () em ações de AWS IoT Core regras:

Exemplo 1

O exemplo a seguir mostra como chamar a função () como modelo de substituição em uma ação do DynamoDB.

```
{
 "topicRulePayload": {
  "sql": "SELECT * AS message FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
```

```
   {
    "dynamoDB": {
     "tableName": "my_ddb_table",
     "hashKeyField": "key",
     "hashKeyValue": "${sourceip()}",
     "rangeKeyField": "timestamp",
     "rangeKeyValue": "${timestamp()}",
     "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDB"
    }
   }
  ]
 }
}
```

Exemplo 2

O exemplo a seguir mostra como adicionar a função sourceip() como uma propriedade de usuário do MQTT usando modelos de substituição.

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "republish": {
     "topic": "${topic()}/republish",
     "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish",
     "headers": {
      "payloadFormatIndicator": "UTF8_DATA",
      "contentType": "rule/contentType",
      "correlationData": "cnVsZSBjb3JyZWxhdGlvbiBkYXRh",
      "userProperties": [
       {
        "key": "ruleKey1",
        "value": "ruleValue1"
       },
       {
        "key": "sourceip",
        "value": "${sourceip()}"
       }
      ]
```

```
      }
     }
    }
   ]
  }
}
```

Você pode recuperar o endereço IP de origem das mensagens que passam para AWS IoT Core as regras dos caminhos do Message Broker e do Basic Ingest. Você também pode recuperar o IP de origem de ambos IPv4 e IPv6 das mensagens. O IP de origem será exibido da seguinte forma:

IPv6: `yyyy:yyyy:yyyy::yyyy:yyyy`

IPv4: `xxx.xxx.xxx.xxx`

> **Note**
>
> O IP de origem original não será transmitido pela ação Republicar.

substring(String, Int[, Int])

Espera uma `String` seguida de um ou dois valores `Int`. No caso de uma `String` e um único argumento `Int`, essa função gera a substring da `String` fornecida do índice `Int` fornecido (baseado em 0, inclusive) ao final da `String`. No caso de uma `String` e dois argumentos `Int`, essa função gera a substring da `String` fornecida no primeiro argumento de índice`Int` (baseado em 0, inclusive) para o segundo argumento de índice `Int` (baseado em 0, inclusive). Índices inferiores a zero são definidos como zero. Índices superiores ao comprimento da `String` são definidos para o comprimento da `String`. No caso da versão de argumento três, se o primeiro índice for superior ou igual ao segundo índice, o resultado será a `String` vazia.

Se os argumentos fornecidos não forem (*`String`*,*`Int`*) ou (*`String`*,*`Int`*,*`Int`*), as conversões padrão serão aplicadas aos argumentos para tentar convertê-los nos tipos corretos. Se os tipos não puderem ser convertidos, o resultado da função será `Undefined`. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplos:

`substring("012345", 0)` = "012345".

`substring("012345", 2)` = "2345".

`substring("012345", 2.745)` = "2345".

`substring(123, 2)` = "3".

`substring("012345", -1)` = "012345".

`substring(true, 1.2)` = "rue".

`substring(false, -2.411E247)` = "false".

`substring("012345", 1, 3)` = "12".

`substring("012345", -50, 50)` = "012345".

`substring("012345", 3, 1)` = "".

sql_version()

Retorna a versão SQL especificada nesta regra. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo:

`sql_version()` = "23/03/2016"

sqrt(Decimal)

Gera a raiz quadrada de um número. Argumentos `Decimal` são arredondados para dobrar a precisão antes da aplicação da função. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `sqrt(9)` = 3,0.

Tipo de argumento	Resultado
`Int`	A raiz quadrada do argumento.
`Decimal`	A raiz quadrada do argumento.

Tipo de argumento	Resultado
`Boolean`	`Undefined` .
`String`	A raiz quadrada do argumento. Se a string não puder ser convertida em um `Decimal`, o resultado será `Undefined` .
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Undefined` .
Não definido	`Undefined` .

startswith(String, String)

Gera `Boolean`, independentemente de o primeiro argumento da string começar com o segundo argumento da string. Se um dos argumentos for `Null` ou `Undefined`, o resultado será `Undefined`. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo:

`startswith("ranger","ran")` = true

Tipo de argumento 1	Tipo de argumento 2	Resultado
`String`	`String`	Independentemente de a primei começar com a segunda string.
Outros valores	Outros valores	Os dois argumentos são conver em strings usando as regras pa de conversão. Retorna verdade independentemente de a primei começar com a segunda string. argumentos for `Null` ou `Undef` resultado será `Undefined` .

tan(Decimal)

Gera a tangente de um número em radianos. Valores `Decimal` são arredondados para dobrar a precisão antes da aplicação da função. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `tan(3)` = -0,1425465430742778

Tipo de argumento	Resultado
`Int`	`Decimal` (com precisão dupla), a tangente do argumento.
`Decimal`	`Decimal` (com precisão dupla), a tangente do argumento.
`Boolean`	`Undefined` .
`String`	`Decimal` (com precisão dupla), a tangente do argumento. Se a string não puder ser convertida em um `Decimal`, o resultado será `Undefined` .
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Undefined` .
Não definido	`Undefined` .

tanh(Decimal)

Gera a tangente hiperbólica de um número em radianos. Valores `Decimal` são arredondados para dobrar a precisão antes da aplicação da função. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `tanh(2.3)` = 0,9800963962661914

Tipo de argumento	Resultado
`Int`	`Decimal` (com precisão dupla), a tangente hiperbólica do argumento.
`Decimal`	`Decimal` (com precisão dupla), a tangente hiperbólica do argumento.
`Boolean`	`Undefined` .
`String`	`Decimal` (com precisão dupla), a tangente hiperbólica do argumento. Se a string não puder ser convertida em um `Decimal`, o resultado será `Undefined` .
Array	`Undefined` .
Objeto	`Undefined` .
Null	`Undefined` .
Não definido	`Undefined` .

time_to_epoch(String, String)

Use a função `time_to_epoch` para converter uma string de timestamp em um número de milissegundos tempo de epoch do Unix. Compatível com a versão de 23/03/2016 do SQL e posteriores. Para converter milissegundos em uma string de carimbo de data/hora, consulte parse_time(String, Long, [String]).

A função `time_to_epoch` espera os seguintes argumentos:

timestamp

(String) A string de carimbo de data/hora a ser convertida em milissegundos desde o epoch do Unix. Se a string do carimbo de data/hora não especificar um fuso horário, a função usará o fuso horário UTC.

pattern

(String) Um date/time padrão que segue os formatos de JDK11 hora.

Exemplos:

`time_to_epoch("2020-04-03 09:45:18 UTC+01:00", "yyyy-MM-dd HH:mm:ss VV")` = 1585903518000

`time_to_epoch("18 December 2015", "dd MMMM yyyy")` = 1450396800000

`time_to_epoch("2007-12-03 10:15:30.592 America/Los_Angeles", "yyyy-MM-dd HH:mm:ss.SSS z")` = 1196705730592

timestamp()

Retorna o timestamp atual em milissegundos a partir das 00:00:00 Horário Universal Coordenado (UTC), quinta-feira, 1º de janeiro de 1970, conforme observado pelo mecanismo de regras. AWS IoT Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo: `timestamp()` = `1481825251155`

topic(Decimal)

Gera o tópico para o qual a mensagem que acionou a regra foi enviada. Se nenhum parâmetro for especificado, todo o tópico será gerado. O parâmetro `Decimal` é usado para especificar um segmento de tópico específico, com 1 designando o primeiro segmento. Para o tópico `foo/bar/baz`, topic(1) retornará `foo`, topic(2) retornará `bar`, e assim por diante. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplos:

`topic()` = "things/myThings/thingOne"

`topic(1)` = "objetos"

Quando a Ingestão básica for usada, o prefixo inicial do tópico (`$aws/rules/`*`rule-name`*) não estará disponível na função topic(). Por exemplo, considerando o tópico:

`$aws/rules/BuildingManager/Buildings/Building5/Floor2/Room201/Lights`

`topic()` = "Buildings/Building5/Floor2/Room201/Lights"

`topic(3)` = "Floor2"

traceid()

Gera o ID de rastreamento (UUID) da mensagem MQTT ou `Undefined` caso a mensagem não tenha sido enviada por MQTT. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo:

`traceid()` = "12345678-1234-1234-1234-123456789012"

transform(String, Objeto, Matriz)

Retorna uma matriz de objetos que contém o resultado da transformação especificada do parâmetro `Object` no parâmetro `Array`.

Compatível com a versão de 23/03/2016 do SQL e posteriores.

String

O modo de transformação a ser usado. Consulte a tabela a seguir para ver os modos de transformação compatíveis e como eles criam os `Result` a partir dos parâmetros `Object` e `Array`.

Objeto

Um objeto que contém os atributos a serem aplicados a cada elemento do `Array`.

Array

Uma matriz de objetos nos quais os atributos de `Object` são aplicados.

Cada objeto nessa matriz corresponde a um objeto na resposta da função. Cada objeto na resposta da função contém os atributos presentes no objeto original e os atributos fornecidos por `Object`, conforme determinado pelo modo de transformação especificado na `String`.

Parâmetro **`String`**	Parâmetro **`Object`**	Parâmetro **`Array`**	Resultado
`enrichArray`	Objeto	Matriz de objetos	Uma matriz de objetos na qual

Parâmetro **`String`**	Parâmetro **`Object`**	Parâmetro **`Array`**	Resultado
			cada objeto contém os atributos de um elemento do parâmetro `Array` e os atributos do parâmetro `Object`.
Qualquer outro valor	Qualquer valor	Qualquer valor	Não definido

> **Note**
>
> A matriz retornada por essa função está limitada a 128 KiB.

Exemplo 1 da função de transformação

Este exemplo mostra como a função transform() produz uma única matriz de objetos a partir de um objeto de dados e uma matriz.

Neste exemplo, a mensagem a seguir é publicada no tópico `A/B` do MQTT.

```
{
    "attributes": {
        "data1": 1,
        "data2": 2
    },
    "values": [
        {
            "a": 3
        },
        {
            "b": 4
        },
        {
            "c": 5
        }
    ]
}
```

Essa instrução SQL para uma ação de regra de tópico usa a função transform() com um valor de `String` de `enrichArray`. Neste exemplo, `Object` é a propriedade de `attributes` da carga da mensagem e `Array` é a matriz de `values`, que contém três objetos.

```
select value transform("enrichArray", attributes, values) from 'A/B'
```

Ao receber a carga útil da mensagem, a instrução SQL avalia a seguinte resposta.

```
[
  {
    "a": 3,
    "data1": 1,
    "data2": 2
  },
  {
    "b": 4,
    "data1": 1,
    "data2": 2
  },
  {
    "c": 5,
    "data1": 1,
    "data2": 2
  }
]
```

Exemplo 2 da função de transformação

Este exemplo mostra como a função transform() pode usar valores literais para incluir e renomear atributos individuais da carga útil da mensagem.

Neste exemplo, a mensagem a seguir é publicada no tópico `A/B` do MQTT. Essa é a mesma mensagem que foi usada em the section called “Exemplo 1 da função de transformação”.

```
{
    "attributes": {
        "data1": 1,
        "data2": 2
    },
    "values": [
        {
            "a": 3
```

```
        },
        {
            "b": 4
        },
        {
            "c": 5
        }
    ]
}
```

Essa instrução SQL para uma ação de regra de tópico usa a função transform() com um valor de `String` de `enrichArray`. O `0bject` na função transform() tem um único atributo chamado `key` com o valor de `attributes.data1` na carga útil da mensagem e `Array` é a matriz de `values`, que contém os mesmos três objetos usados no exemplo anterior.

```
select value transform("enrichArray", {"key": attributes.data1}, values) from 'A/B'
```

Ao receber a carga útil da mensagem, esta instrução SQL avalia a seguinte resposta. Observe como a propriedade `data1` é nomeada `key` na resposta.

```
[
  {
    "a": 3,
    "key": 1
  },
  {
    "b": 4,
    "key": 1
  },
  {
    "c": 5,
    "key": 1
  }
]
```

Exemplo 3 da função de transformação

Este exemplo mostra como a função transform() pode ser usada em cláusulas SELECT aninhadas para selecionar vários atributos e criar novos objetos para processamento posterior.

Neste exemplo, a mensagem a seguir é publicada no tópico `A/B` do MQTT.

```
{
  "data1": "example",
  "data2": {
    "a": "first attribute",
    "b": "second attribute",
    "c": [
      {
        "x": {
          "someInt": 5,
          "someString": "hello"
        },
        "y": true
      },
      {
        "x": {
          "someInt": 10,
          "someString": "world"
        },
        "y": false
      }
    ]
  }
}
```

O `0bject` para esta função de transformação é o objeto retornado pela instrução SELECT, que contém os elementos a e b do objeto `data2` da mensagem. O parâmetro `Array` consiste nos dois objetos da matriz `data2.c` na mensagem original.

```
select value transform('enrichArray', (select a, b from data2), (select value c from
 data2)) from 'A/B'
```

Com a mensagem anterior, a instrução SQL avalia a seguinte resposta.

```
[
  {
    "x": {
      "someInt": 5,
      "someString": "hello"
    },
    "y": true,
    "a": "first attribute",
    "b": "second attribute"
```

```
  },
  {
    "x": {
      "someInt": 10,
      "someString": "world"
    },
    "y": false,
    "a": "first attribute",
    "b": "second attribute"
  }
]
```

A matriz retornada nessa resposta pode ser usada com ações de regras de tópico compatíveis com `batchMode`.

trim(String)

Remove todos os espaços em branco iniciais e finais da `String` fornecida. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplo:

`Trim(" hi ")` = "hi"

Tipo de argumento	Resultado
`Int`	A representação de `String` de `Int` com todos os espaços em branco iniciais e finais removidos.
`Decimal`	A representação de `String` de `Decimal` com todos os espaços em branco iniciais e finais removidos.
`Boolean`	A representação de `String` de `Boolean` ("verdadeiro" ou "falso") com todos os espaços em branco iniciais e finais removidos.
`String`	A `String` com todos os espaços em branco iniciais e finais removidos.

Tipo de argumento	Resultado
Array	A representação da `String` da `Array` usando as regras padrão de conversão.
Objeto	A representação da `String` do Objeto usando as regras padrão de conversão.
Null	`Undefined` .
Não definido	`Undefined` .

trunc(Decimal, Int)

Trunca o primeiro argumento para o número de lugares `Decimal` especificado pelo segundo argumento. Se o segundo argumento for inferior a zero, ele será definido como zero. Se o segundo argumento for superior a 34, ele será definido como 34. Os zeros finais são removidos do resultado. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplos:

`trunc(2.3, 0)` = 2.

`trunc(2.3123, 2)` = 2,31.

`trunc(2.888, 2)` = 2,88.

`trunc(2.00, 5)` = 2.

Tipo de argumento 1	Tipo de argumento 2	Resultado
`Int`	`Int`	O valor de origem.
`Int`/`Decimal`	`Int`/`Decimal`	O primeiro argumento é truncad o comprimento descrito pelo se argumento. O segundo argume for um `Int`, será arredondado p abaixo mais próximo.

Tipo de argumento 1	Tipo de argumento 2	Resultado
`Int/Decimal/String`	`Int/Decimal`	O primeiro argumento é truncad o comprimento descrito pelo se argumento. O segundo argume não for um `Int`, será arredonda o `Int` abaixo mais próximo. A S convertida em um valor `Decima` for possível converter a string, o será `Undefined` .
Outros valores		`Undefined` .

upper(String)

Gera a versão em letra maiúscula da `String` fornecida. Os argumentos de não`String` são convertidos em `String` usando as regras de conversão padrão. Compatível com a versão de 08/10/2015 do SQL e posteriores.

Exemplos:

`upper("hello")` = "HELLO"

`upper(["hello"])` = "[\"HELLO\"]"

Literais

Você pode especificar diretamente objetos literais nas cláusulas WHERE e SELECT da regra SQL, que pode ser útil para transmitir informações.

> **Note**
>
> Literais estão disponíveis somente ao usar a versão de 23/03/2016 do SQL ou posteriores.

A sintaxe de objeto JSON é usada (pares de chave-valor, separado por vírgula, em que as chaves são strings, e os valores são valores JSON, entre chaves {}). Por exemplo:

Carga útil de entrada publicada no tópico `topic/subtopic`: `{"lat_long": [47.606,-122.332]}`

Declaração do SQL: `SELECT {'latitude': get(lat_long, 0),'longitude':get(lat_long, 1)} as lat_long FROM 'topic/subtopic'`

A carga útil de saída resultante seria: `{"lat_long": {"latitude":47.606,"longitude":-122.332}}`.

Você também pode especificar diretamente matrizes nas cláusulas WHERE e SELECT da regra SQL, o que permite agrupar informações. A sintaxe JSON é usada (encapsular itens separados por vírgula entre colchetes [] para criar uma matriz literal). Por exemplo:

Carga útil de entrada publicada no tópico `topic/subtopic`: `{"lat": 47.696, "long": -122.332}`

Declaração do SQL: `SELECT [lat,long] as lat_long FROM 'topic/subtopic'`

A carga útil de saída resultante seria: `{"lat_long": [47.606,-122.332]}`.

Declarações de caso

As declarações de caso podem ser usadas para execução de ramificação, como uma declaração de troca.

Sintaxe:

```
CASE v WHEN t[1] THEN r[1]
  WHEN t[2] THEN r[2] ...
  WHEN t[n] THEN r[n]
  ELSE r[e] END
```

A expressão *v* é avaliada e comparada quanto à igualdade com o valor *t[i]* de cada cláusula `WHEN`. Se uma correspondência for encontrada, a expressão *r[i]* correspondente se tornará o resultado da declaração do `CASE`. As cláusulas `WHEN` são avaliadas em ordem para que, se houver mais de uma cláusula correspondente, o resultado da primeira cláusula correspondente se torne o resultado da declaração do `CASE`. Se não houver correspondências, *r[e]* da cláusula `ELSE` é o resultado. Se não houver correspondências e não houver cláusula `ELSE`, o resultado é `Undefined`.

As declarações de `CASE` exigem, pelo menos, uma cláusula `WHEN`. Qualquer cláusula `ELSE` é opcional.

Por exemplo:

Carga útil de entrada publicada no tópico `topic/subtopic`:

```
{
    "color":"yellow"
}
```

Declaração do SQL:

```
SELECT CASE color
        WHEN 'green' THEN 'go'
        WHEN 'yellow' THEN 'caution'
        WHEN 'red' THEN 'stop'
        ELSE 'you are not at a stop light' END as instructions
    FROM 'topic/subtopic'
```

A carga útil de saída resultante seria:

```
{
    "instructions":"caution"
}
```

> **Note**
>
> Se *v* for `Undefined`, o resultado da declaração do caso será `Undefined`.

Extensões JSON

Você pode usar as seguintes extensões para a sintaxe SQL padrão ANSI para facilitar o trabalho com objetos JSON aninhados.

Operador "."

Esse operador acessa membros em objetos JSON incorporados e funciona de forma idêntica ao ANSI SQL e. JavaScript Por exemplo:

```
SELECT foo.bar AS bar.baz FROM 'topic/subtopic'
```

seleciona o valor da propriedade `bar` no objeto `foo` a partir da seguinte carga útil de mensagem enviada ao tópico `topic/subtopic`.

```
{
  "foo": {
    "bar": "RED",
    "bar1": "GREEN",
    "bar2": "BLUE"
  }
}
```

Se o nome de uma propriedade JSON incluir um caractere de hífen ou caracteres numéricos, a notação 'ponto' não funcionará. Em vez disso, você deve usar a função get para extrair o valor da propriedade.

Neste exemplo, a mensagem a seguir é enviada ao tópico `iot/rules`.

```
{
  "mydata": {
    "item2": {
      "0": {
        "my-key": "myValue"
      }
    }
  }
}
```

Normalmente, o valor de `my-key` seria identificado como nessa consulta.

```
SELECT * from iot/rules WHERE mydata.item2.0.my-key= "myValue"
```

No entanto, como o nome da propriedade `my-key` contém um hífen e `item2` contém um caractere numérico, a função get deve ser usada conforme mostra a consulta a seguir.

```
SELECT * from 'iot/rules' WHERE get(get(get(mydata,"item2"),"0"),"my-key") = "myValue"
```

*Operador

Funciona da mesma maneira que o caractere curinga * em SQL padrão ANSI. É usado somente na cláusula SELECT e cria outro objeto JSON contendo os dados da mensagem. Se a carga útil

da mensagem não estiver no formato JSON, * gerará toda a carga útil da mensagem como bytes brutos. Por exemplo:

```
SELECT * FROM 'topic/subtopic'
```

Como aplicar uma função a um valor de atributo

Veja a seguir um exemplo de carga útil JSON que pode ser publicada por um dispositivo:

```
{
    "deviceid" : "iot123",
    "temp" : 54.98,
    "humidity" : 32.43,
    "coords" : {
        "latitude" : 47.615694,
        "longitude" : -122.3359976
    }
}
```

O exemplo a seguir aplica uma função a um valor de atributo em uma carga útil JSON:

```
SELECT temp, md5(deviceid) AS hashed_id FROM topic/#
```

O resultado dessa consulta é o seguinte objeto JSON:

```
{
   "temp": 54.98,
   "hashed_id": "e37f81fb397e595c4aeb5645b8cbbbd1"
}
```

Modelos de substituição

Você pode usar um modelo de substituição para aumentar os dados JSON retornados quando uma regra é acionada e AWS IoT executa uma ação. A sintaxe de um modelo de substituição é ${ expressão*}*, em que expressão pode ser qualquer expressão suportada AWS IoT nas cláusulas SELECT, nas cláusulas WHERE e. AWS IoT ações de regras Essa expressão pode ser conectada a um campo de ação em uma regra, permitindo que você configure dinamicamente uma ação. Na verdade, esse recurso substitui uma informação em uma ação. Isso inclui funções, operadores e informações presentes na carga da mensagem original.

> **Important**
>
> Como uma expressão em um modelo de substituição é avaliada separadamente da instrução "SELECT...", você não pode fazer referência a um alias criado com a cláusula AS. Você pode fazer referência somente às informações presentes na carga original, além das funções e dos operadores compatíveis.

Para obter mais informações sobre as expressões compatíveis, consulte AWS IoT Referência SQL.

As ações de regra a seguir oferecem suporte a modelos de substituição. Cada ação oferece suporte a campos diferentes que podem ser substituídos.

- Apache Kafka
- CloudWatch alarmes
- CloudWatch Registros
- CloudWatch métricas
- DynamoDB
- Dínamo DBv2
- Elasticsearch
- HTTP
- AWS IoT Events
- AWS IoT SiteWise
- Kinesis Data Streams
- Firehose
- Lambda
- Local
- OpenSearch
- Nova publicação
- S3
- SNS
- SQS

- Step Functions
- Timestream

Os modelos de substituição aparecem nos parâmetros de ação dentro de uma regra:

```
{
    "sql": "SELECT *, timestamp() AS timestamp FROM 'my/iot/topic'",
    "ruleDisabled": false,
    "actions": [{
        "republish": {
            "topic": "${topic()}/republish",
            "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
        }
    }]
}
```

Se essa regra for acionada pelo JSON a seguir publicado em `my/iot/topic`:

```
{
    "deviceid": "iot123",
    "temp": 54.98,
    "humidity": 32.43,
    "coords": {
        "latitude": 47.615694,
        "longitude": -122.3359976
    }
}
```

Em seguida, essa regra publica o seguinte JSON em`my/iot/topic/republish`, que AWS IoT substitui de: `${topic()}/republish`

```
{
    "deviceid": "iot123",
    "temp": 54.98,
    "humidity": 32.43,
    "coords": {
        "latitude": 47.615694,
        "longitude": -122.3359976
    },
    "timestamp": 1579637878451
}
```

Consultas de objeto aninhado

Você pode usar cláusulas SELECT aninhadas para consultar atributos dentro de matrizes e objetos JSON internos. Compatível com a versão de 23/03/2016 do SQL e posteriores.

Considere a seguinte mensagem MQTT:

```
{
    "e": [
        { "n": "temperature", "u": "Cel", "t": 1234, "v": 22.5 },
        { "n": "light", "u": "lm", "t": 1235, "v": 135 },
        { "n": "acidity", "u": "pH", "t": 1235, "v": 7 }
    ]
}
```

Example

Você pode converter valores em uma nova matriz com a seguinte regra.

```
SELECT (SELECT VALUE n FROM e) as sensors FROM 'my/topic'
```

A regra gera a seguinte saída:

```
{
    "sensors": [
        "temperature",
        "light",
        "acidity"
    ]
}
```

Example

Usando a mesma mensagem MQTT, você também pode consultar um valor específico dentro de um objeto aninhado com a seguinte regra.

```
SELECT (SELECT v FROM e WHERE n = 'temperature') as temperature FROM 'my/topic'
```

A regra gera a seguinte saída:

```
{
    "temperature": [
        {
            "v": 22.5
        }
    ]
}
```

Example

Você também pode nivelar a saída com uma regra mais complicada.

```
SELECT get((SELECT v FROM e WHERE n = 'temperature'), 0).v as temperature FROM 'topic'
```

A regra gera a seguinte saída:

```
{
    "temperature": 22.5
}
```

Como trabalhar com cargas úteis binárias

Para tratar a carga da mensagem como dados binários brutos (em vez de um objeto JSON), você pode usar o operador * para fazer referência a ela em uma cláusula SELECT.

Neste tópico:

- Exemplos de cargas úteis binárias
- Decodificar as cargas úteis da mensagem protobuf

Exemplos de cargas úteis binárias

Ao usar * para se referir à carga da mensagem como dados binários brutos, você pode adicionar dados à regra. Se você tiver uma carga vazia ou JSON, a carga resultante poderá ter dados adicionados usando a regra. Veja a seguir exemplos de cláusulas `SELECT` compatíveis.

- Você pode usar as cláusulas `SELECT` a seguir com apenas um * para cargas binárias.
  - ```
    SELECT * FROM 'topic/subtopic'
    ```
    ```

- ```
  SELECT * FROM 'topic/subtopic' WHERE timestamp() % 12 = 0
  ```

- Você também pode adicionar dados e usar as cláusulas SELECT a seguir.
  - ```
    SELECT *, principal() as principal, timestamp() as time FROM 'topic/subtopic'
    ```
  - ```
    SELECT encode(*, 'base64') AS data, timestamp() AS ts FROM 'topic/subtopic'
    ```
- Você também pode usar essas cláusulas SELECT com cargas binárias.
  - O seguinte se refere ao `device_type` na cláusula WHERE.

    ```
    SELECT * FROM 'topic/subtopic' WHERE device_type = 'thermostat'
    ```

  - Também há suporte para:

    ```
    {
     "sql": "SELECT * FROM 'topic/subtopic'",
     "actions": [
      {
       "republish": {
        "topic": "device/${device_id}"
       }
      }
     ]
    }
    ```

As ações de regra a seguir não oferecem suporte a cargas binárias, portanto, você deve decodificá-las.

- Algumas ações de regra não oferecem suporte a entrada de carga útil binária, como a ação do Lambda, então é necessário decodificar cargas úteis binárias. A ação de regra do Lambda poderá receber dados binários se estiverem codificados em base64 e em uma carga útil JSON. É possível fazer isso alterando a regra para a seguinte:

  ```
  SELECT encode(*, 'base64') AS data FROM 'my_topic'
  ```

- A instrução SQL não oferece suporte a string como entrada. Para converter uma entrada de string em JSON, é possível executar o seguinte comando.
  ```

```
SELECT decode(encode(*, 'base64'), 'base64') AS payload FROM 'topic'
```

Decodificar as cargas úteis da mensagem protobuf

Protocol Buffers (protobuf) é um formato de dados de código aberto usado para serializar dados estruturados em um formato binário compacto. É usado para transmitir dados por redes ou armazená-los em arquivos. O Protobuf permite que você envie dados em pacotes pequenos e em uma taxa mais rápida do que outros formatos de mensagens. AWS IoT Core As regras oferecem suporte ao protobuf fornecendo a função SQL decode (value, decodingScheme), que permite decodificar cargas de mensagens codificadas por protobuf para o formato JSON e roteá-las para serviços downstream. Esta seção detalha o step-by-step processo para configurar a decodificação do protobuf em Regras. AWS IoT Core

Nesta seção:

- Pré-requisitos
- Criar arquivos descritores
- Upload de arquivos descritores em um bucket do S3
- Configurar a decodificação do protobuf em Regras
- Limitações
- Práticas recomendadas

Pré-requisitos

- Uma compreensão básica dos Protocol Buffers (protobuf)
- Os arquivos `.proto` que definem os tipos de mensagens e dependências relacionadas
- Instalação do compilador Protobuf (protoc) em seu sistema

Criar arquivos descritores

Se já tiver arquivos de descrição, você poderá ignorar esta etapa. Um arquivo descritor (`.desc`) é uma versão compilada de um arquivo `.proto`, que é um arquivo de texto que define as estruturas de dados e os tipos de mensagens a serem usados em uma serialização do protobuf. Para gerar um arquivo descritor, você deve definir um arquivo `.proto` e usar o compilador protoc para compilá-lo.

1. Crie arquivos `.proto` que definam os tipos de mensagem. Um exemplo de arquivo `.proto` pode ser o seguinte:

```
syntax = "proto3";

message Person {
  optional string name = 1;
  optional int32 id = 2;
  optional string email = 3;
}
```

   Neste exemplo de arquivo `.proto`, você usa a sintaxe proto3 e define o tipo de mensagem `Person`. A definição da mensagem `Person` especifica três campos (nome, ID e e-mail). Para obter mais informações sobre formatos de mensagem de arquivo `.proto`, consulte o Guia de idiomas (proto3).

2. Use o compilador protoc para compilar os arquivos `.proto` e gerar um arquivo descritor. Um exemplo de comando para criar um arquivo descritor (`.desc`) pode ser o seguinte:

```
protoc --descriptor_set_out=<FILENAME>.desc \
    --proto_path=<PATH_TO_IMPORTS_DIRECTORY> \
    --include_imports \
    <PROTO_FILENAME>.proto
```

   Esse exemplo de comando gera um arquivo descritor`<FILENAME>.desc`, que o AWS IoT Core Rules pode usar para decodificar cargas de protobuf que estejam em conformidade com a estrutura de dados definida em. `<PROTO_FILENAME>.proto`

   - `--descriptor_set_out`

     Especifica o nome do arquivo descritor (`<FILENAME>.desc`) que deve ser gerado.

   - `--proto_path`

     Especifica os locais de todos os arquivos `.proto` importados referenciados pelo arquivo que está sendo compilado. Você pode especificar o sinalizador várias vezes se tiver vários arquivos `.proto` importados com locais diferentes.

   - `--include_imports`

     Especifica que todos os arquivos `.proto` importados também devem ser compilados e incluídos no arquivo `<FILENAME>.desc` descritor.

- `<PROTO_FILENAME>.proto`

  Especifica o nome do arquivo `.proto` que você deseja compilar.

Para obter mais informações sobre a referência protoc, consulte Referência de API.

Upload de arquivos descritores em um bucket do S3

Depois de criar seus arquivos descritores`<FILENAME>.desc`, faça o upload dos arquivos `<FILENAME>.desc` descritores em um bucket do Amazon S3 usando AWS a API AWS , o SDK ou o. Console de gerenciamento da AWS

Considerações importantes

- Certifique-se de carregar os arquivos do descritor em um bucket do Amazon S3 no Região da AWS mesmo local em que você pretende configurar Conta da AWS suas regras.
- Certifique-se de conceder AWS IoT Core acesso para ler o `FileDescriptorSet` do S3. Se o seu bucket do S3 tiver a criptografia do lado do servidor (SSE) desativada ou se o bucket do S3 estiver criptografado usando chaves gerenciadas pelo Amazon S3 (SSE-S3), nenhuma configuração adicional de política será necessária. Isso pode ser feito com o exemplo de política de bucket:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "Statement1",
   "Effect": "Allow",
   "Principal": {
    "Service": "iot.amazonaws.com"
   },
   "Action": "s3:Get*",
                     "Resource": "arn:aws:s3:::<BUCKET NAME>/<FILENAME>.desc"
  }
 ]
}
```

- Se o bucket do S3 for criptografado usando uma AWS Key Management Service chave (SSE-KMS), certifique-se de conceder AWS IoT Core permissão para usar a chave ao acessar o bucket do S3. Você pode fazer isso adicionando esta declaração à sua política de chave:

```
{
 "Sid": "Statement1",
 "Effect": "Allow",
 "Principal": {
  "Service": "iot.amazonaws.com"
 },
 "Action": [
  "kms:Decrypt",
  "kms:GenerateDataKey*",
  "kms:DescribeKey"
 ],
        "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

}
```

Configurar a decodificação do protobuf em Regras

Depois de fazer o upload dos arquivos do descritor no bucket do Amazon S3, configure uma regra que possa decodificar o formato de carga útil da mensagem protobuf usando a função SQL decode(value, decodingScheme). Uma assinatura de função detalhada e um exemplo podem ser encontrados na função SQL decode(value, decodingScheme) da referência SQL de AWS IoT .

Veja a seguir um exemplo de expressão SQL usando a função decode(value, decodingScheme):

```
SELECT VALUE decode(*, 'proto', '<BUCKET NAME>', '<FILENAME>.desc', '<PROTO_FILENAME>',
 '<PROTO_MESSAGE_TYPE>') FROM '<MY_TOPIC>'
```

Neste exemplo de expressão:

- Você usa a função SQL decode(value, decodingScheme) para decodificar a carga útil da mensagem binária referenciada por *. Isso pode ser uma carga binária codificada por protobuf ou uma string JSON que representa uma carga útil protobuf codificada em base64.
- A carga útil da mensagem fornecida é codificada usando o tipo de mensagem `Person` definido em `PROTO_FILENAME.proto`.
- O bucket do Amazon S3 chamado `BUCKET NAME` contém o `FILENAME.desc` gerado de `PROTO_FILENAME.proto`.

Depois de concluir a configuração, publique uma mensagem no tópico AWS IoT Core no qual a Regra está inscrita.

Limitações

AWS IoT Core As regras oferecem suporte ao protobuf com as seguintes limitações:

- A decodificação de cargas de mensagens protobuf em modelos de substituição não é suportada.
- Ao decodificar cargas de mensagens protobuf, você pode usar a função decodificar SQL em uma única expressão SQL até duas vezes.
- O tamanho máximo da carga útil de entrada é 128 KiB (1 KiB = 1024 bytes), o tamanho máximo da carga de saída é 128 KiB e o tamanho máximo de um objeto `FileDescriptorSet` armazenado em um bucket do Amazon S3 é 32 KiB.
- Não há suporte para buckets do Amazon S3 criptografados com a criptografia SSE-C.

Práticas recomendadas

Veja a seguir algumas práticas recomendadas e dicas de solução de problemas.

- Carregue seus arquivos proto no bucket no Amazon S3.

  É uma prática recomendada fazer backup de seus arquivos proto caso algo dê errado. Por exemplo, se você modificar incorretamente os arquivos proto sem backups ao executar protoc, isso pode causar problemas em sua pilha de produção. Há várias maneiras de fazer backup de arquivos em um bucket do Amazon S3. Por exemplo, você pode usar o versionamento em buckets do S3. Para obter mais informações sobre como fazer backup de arquivos em buckets do Amazon S3, consulte o Guia do desenvolvedor do Amazon S3.
- Configure o AWS IoT registro para visualizar as entradas do registro.

  É uma boa prática configurar o AWS IoT registro para que você possa verificar AWS IoT os registros da sua conta CloudWatch. Quando a consulta SQL de uma regra chama uma função externa, o AWS IoT Core Rules gera uma entrada de registro com um `eventType` de`FunctionExecution`, que contém o campo de motivo que o ajudará a solucionar falhas. Os possíveis erros incluem um objeto do Amazon S3 não encontrado ou um descritor de arquivo protobuf inválido. Para obter mais informações sobre como configurar o registro de AWS IoT e ver as entradas de log, consulte Configurar o registro de AWS IoT e Entradas de log do mecanismo de regras.

- Atualize o `FileDescriptorSet` usando uma nova chave de objeto e atualize a chave de objeto em sua regra.

  Você pode atualizar o `FileDescriptorSet` fazendo upload de um arquivo de descritor atualizado para o seu bucket do Amazon S3. Suas atualizações do `FileDescriptorSet` podem levar até 15 minutos para serem refletidas. Para evitar esse atraso, é uma prática recomendada carregar seu `FileDescriptorSet` atualizado usando uma nova chave de objeto e atualizar a chave de objeto em sua regra.

Versões do SQL

O mecanismo de AWS IoT regras usa uma sintaxe semelhante ao SQL para selecionar dados das mensagens MQTT. As instruções SQL são interpretadas com base em uma versão do SQL especificada com a propriedade `awsIotSqlVersion` em um documento JSON que descreve a regra. Para obter mais informações sobre a estrutura de documentos de regra JSON, consulte Como criar uma regra. A `awsIotSqlVersion` propriedade permite especificar qual versão do mecanismo de regras AWS IoT SQL você deseja usar. Quando uma nova versão é implantada, você pode continuar usando uma versão mais antiga ou alterar a regra para usar a nova versão. As regras atuais continuam a usar a versão com a qual foram criadas.

O seguinte exemplo de JSON mostra como especificar a versão do SQL usando a propriedade `awsIotSqlVersion`.

```
{
    "sql": "expression",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [{
        "republish": {
            "topic": "my-mqtt-topic",
            "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
        }
    }]
}
```

AWS IoT atualmente oferece suporte às seguintes versões do SQL:

- `2016-03-23` – A versão SQL implantada em 23/03/2016 (recomendada).
- `2015-10-08` – A versão original do SQL implantada em 08/10/2015.

- `beta` – A versão beta mais recente do SQL. Essa versão pode causar alterações nas regras.

Novidades da versão 23/03/2016 do mecanismo de regras SQL

- Correções na seleção de objetos JSON aninhados.
- Correções em consultas de matrizes.
- Suporte para consulta entre objetos. Para obter mais informações, consulte Consultas de objeto aninhado.
- Suporte para emitir uma matriz como um objeto de nível superior.
- Adição da função `encode(value, encodingScheme)`, que pode ser aplicada em dados em formato JSON e não JSON. Para obter mais informações, consulte a função de codificação.

Resultado de uma **`Array`** como um objeto de nível superior

Esse recurso permite que uma regra gere uma matriz como um objeto de nível superior. Por exemplo, com base na seguinte mensagem MQTT:

```
{
    "a": {"b":"c"},
    "arr":[1,2,3,4]
}
```

E a seguinte regra:

```
SELECT VALUE arr FROM 'topic'
```

A regra gera a seguinte saída:

```
[1,2,3,4]
```

AWS IoT Serviço Device Shadow

O serviço AWS IoT Device Shadow adiciona sombras aos objetos AWS IoT físicos. As sombras podem disponibilizar o estado de um dispositivo para aplicativos e outros serviços, independentemente de o dispositivo estar conectado AWS IoT ou não. AWS IoT objetos físicos podem ter várias sombras nomeadas para que sua solução de IoT tenha mais opções para conectar seus dispositivos a outros aplicativos e serviços.

AWS IoT objetos de coisas não têm sombras até serem criados explicitamente. As sombras podem ser criadas, atualizadas e excluídas usando o AWS IoT console. Dispositivos, outros clientes da web e serviços podem criar, atualizar e excluir sombras usando o MQTT e os tópicos MQTT reservados, o HTTP usando a Sombra do Dispositivo API REST, e o AWS CLI para AWS IoT. Como as sombras são armazenadas AWS na nuvem, elas podem coletar e relatar dados de estado do dispositivo de aplicativos e outros serviços em nuvem, independentemente de o dispositivo estar conectado ou não.

Usar shadows

As sombras fornecem um armazenamento de dados confiável para dispositivos, aplicativos e outros serviços na nuvem para compartilhamento de dados. Eles permitem que dispositivos, aplicativos e outros serviços na nuvem se conectem e desconectem sem perder o estado de um dispositivo.

Enquanto dispositivos, aplicativos e outros serviços em nuvem estão conectados AWS IoT, eles podem acessar e controlar o estado atual de um dispositivo por meio de suas sombras. Por exemplo, um aplicativo pode solicitar uma alteração no estado de um dispositivo atualizando uma sombra. AWS IoT publica uma mensagem que indica a alteração no dispositivo. O dispositivo recebe essa mensagem, atualiza seu estado de forma correspondente e publica uma mensagem com o estado atualizado. O serviço Sombra do Dispositivo reflete esse estado atualizado na sombra correspondente. O aplicativo pode se inscrever na atualização da sombra ou consultar a sombra para saber seu estado atual.

Quando um dispositivo fica off-line, um aplicativo ainda pode se comunicar com AWS IoT as sombras do dispositivo. Quando o dispositivo se reconecta, ele recebe o estado atual de suas sombras para que possa atualizar seu estado de forma correspondente a elas, e possa publicar uma mensagem com seu estado atualizado. Da mesma forma, quando o estado do dispositivo muda enquanto ele está offline, o dispositivo mantém a sombra atualizada para que o aplicativo possa consultar as sombras para saber o estado atual quando ele se reconectar.

Se seus dispositivos ficam offline com frequência e você deseja configurá-los para receber mensagens delta após se reconectarem, você pode usar o atributo de sessão persistente. Para obter mais informações sobre o período de expiração da sessão persistente, consulte Período de expiração da sessão persistente.

Optar por usar sombras nomeadas ou sem nome

O serviço Sombra do Dispositivo oferece suporte a sombras nomeadas e sem nome ou clássicas. Um objeto pode ter várias sombras nomeadas e não mais do que uma sombra sem nome. O objeto também pode ter uma sombra nomeada reservada, que opera de forma semelhante a uma sombra nomeada, exceto que você não pode atualizar seu nome. Para obter mais informações, consulte Sombra nomeada reservada.

Um objeto pode ter sombras nomeadas e sem nome ao mesmo tempo. No entanto, a API usada para acessar cada uma é ligeiramente diferente, portanto, talvez seja mais eficiente decidir o tipo de sombra que funcionará melhor para sua solução e usar somente esse tipo. Para obter mais informações sobre a API para acessar as sombras, consulte Tópicos de sombra.

Com sombras nomeadas, você pode criar diferentes visualizações do estado de um objeto. Por exemplo, é possível dividir um objeto com muitas propriedades em sombras com grupos lógicos de propriedades, cada um identificado por seu nome de sombra. Também é possível limitar o acesso às propriedades agrupando-as em diferentes sombras e usando políticas para controlar o acesso. Para obter mais informações sobre políticas a serem usadas com sombras do dispositivos, consulte Ações, recursos e chaves de condição AWS IoT e AWS IoT Core políticas.

As sombras clássicas e sem nome são mais simples, mas um pouco mais limitadas do que as sombras nomeadas. Cada AWS IoT objeto pode ter apenas uma sombra sem nome. Se você espera que sua solução de IoT tenha uma necessidade limitada de dados de sombra, talvez essa seja a maneira como você quer começar a usar sombras. No entanto, se você espera adicionar sombras adicionais no futuro, considere usar sombras nomeadas desde o início.

A indexação de frota oferece suporte a sombras sem nome e sombras com nomes diferentes. Para acessar mais informações, consulte Gerenciar a indexação de frotas.

Acessar sombras

Cada sombra tem um tópico do MQTT e um URL HTTP reservados que comportam as ações `get`, `update` e `delete` na sombra.

As sombras usam documentos JSON de sombra para armazenar e recuperar dados. Um documento de sombra contém uma propriedade de estado que descreve estes aspectos do estado do dispositivo:

- `desired`

  Os aplicativos especificam os estados desejados das propriedades do dispositivo atualizando o objeto `desired`.

- `reported`

  Os dispositivos relatam seu estado atual no objeto `reported`.

- `delta`

  AWS IoT relata diferenças entre o estado desejado e o estado relatado no `delta` objeto.

Os dados armazenados em uma sombra são determinados pela propriedade de estado do corpo da mensagem da ação de atualização. As ações de atualização subsequentes podem modificar os valores de um objeto de dados existente e também adicionar e excluir chaves e outros elementos no objeto de estado da sombra. Para receber mais informações sobre como acessar sombras, consulte Usar sombras em dispositivos e Usar sombras em aplicativos e serviços.

> **Important**
>
> A permissão para fazer solicitações de atualização deve ser limitada a aplicativos e dispositivos confiáveis. Isso impede que a propriedade de estado da sombra seja alterada inesperadamente. Caso contrário, os dispositivos e aplicativos que usam a sombra devem ser projetados para esperar que as chaves na propriedade de estado sejam alteradas.

Usar sombras em dispositivos, aplicativos e outros serviços na nuvem

O uso de sombras em dispositivos, aplicativos e outros serviços na nuvem requer consistência e coordenação entre todos eles. O serviço AWS IoT Device Shadow armazena o estado de sombra, envia mensagens quando o estado de sombra muda e responde às mensagens que mudam de estado. Os dispositivos, aplicativos e outros serviços na nuvem em sua solução de IoT devem gerenciar seu estado e mantê-lo consistente com o estado da sombra do dispositivo.

Os dados de estado de sombra são dinâmicos e podem ser alterados pelos dispositivos, aplicativos e outros serviços na nuvem com permissão para acessar a sombra. Por esse motivo, é importante considerar como cada dispositivo, aplicativo e outro serviço de nuvem interagirão com a sombra. Por exemplo:

- Os dispositivos devem gravar somente na propriedade `reported` do estado de sombra ao comunicar dados de estado à sombra.
- Os aplicativos e outros serviços na nuvem devem gravar somente na propriedade `desired` ao comunicar solicitações de alteração de estado ao dispositivo por meio da sombra.

> **Important**
>
> Os dados contidos em um objeto de dados de sombra são independentes dos de outras sombras e de outras propriedades de objeto, como os atributos de um objeto e o conteúdo de mensagens MQTT que um dispositivo de objeto pode publicar. Um dispositivo pode, no entanto, relatar os mesmos dados em diferentes tópicos MQTT e sombras, se necessário. Um dispositivo compatível com várias sombras deve manter a consistência dos dados que ele relata nas diferentes sombras.

Ordem das mensagens

Não há garantia de que as mensagens do AWS IoT serviço chegarão ao dispositivo em qualquer ordem específica. O cenário a seguir mostra o que acontece nesse caso.

Documento de estado inicial:

```
{
  "state": {
    "reported": {
      "color": "blue"
    }
  },
  "version": 9,
  "timestamp": 123456776
}
```

Atualização 1:

```
{
  "state": {
    "desired": {
      "color": "RED"
    }
  },
  "version": 10,
  "timestamp": 123456777
}
```

Atualização 2:

```
{
  "state": {
    "desired": {
      "color": "GREEN"
    }
  },
  "version": 11,
  "timestamp": 123456778
}
```

Documento de estado final:

```
{
  "state": {
    "reported": {
      "color": "GREEN"
    }
  },
  "version": 12,
  "timestamp": 123456779
}
```

Isso resulta em duas mensagens delta:

```
{
  "state": {
    "color": "RED"
  },
  "version": 11,
```

```
  "timestamp": 123456778
}
```

```
{
  "state": {
    "color": "GREEN"
  },
  "version": 12,
  "timestamp": 123456779
}
```

O dispositivo pode receber essas mensagens fora de ordem. Como o estado nessas mensagens é cumulativo, um dispositivo pode descartar com segurança as mensagens que contêm um número da versão anterior ao que ele está rastreando. Se o dispositivo receber o delta para a versão 12 antes da versão 11, pele poderá descartar a mensagem da versão 11 com segurança.

Reduzir mensagens de shadow

Para reduzir o tamanho das mensagens de shadow enviadas para o dispositivo, defina uma regra que selecione apenas os campos de que o dispositivo precisa e, em seguida, publique novamente a mensagem em um tópico MQTT que o dispositivo esteja escutando.

A regra é especificada no JSON e deve ser semelhante ao seguinte:

```
{
  "sql": "SELECT state, version FROM '$aws/things/+/shadow/update/delta'",
  "ruleDisabled": false,
  "actions": [
    {
      "republish": {
        "topic": "${topic(3)}/delta",
        "roleArn": "arn:aws:iam:123456789012:role/my-iot-role"
      }
    }
  ]
}
```

A declaração de SELECT determina quais campos da mensagem serão publicados novamente no tópico especificado. Um caractere curinga "+" é usado para corresponder a todos os nomes de shadow. A regra especifica que todas as mensagens correspondentes devem ser publicadas

novamente no tópico especificado. Nesse caso, a função `"topic()"` é usada para especificar o tópico no qual a publicação deverá ser refeita. O `topic(3)` avalia o nome do objeto no tópico original. Para obter mais informações sobre como criar regras, consulte Regras para AWS IoT.

Usar sombras em dispositivos

Esta seção descreve as comunicações de dispositivos com sombras usando mensagens MQTT, o método preferido para dispositivos se comunicarem com o serviço AWS IoT Device Shadow.

As comunicações paralelas emulam um request/response modelo usando o modelo de comunicação de publicação/assinatura do MQTT. Cada ação de sombra consiste em um tópico de solicitação, um tópico de resposta bem-sucedido (`accepted`) e um tópico de resposta de erro (`rejected`).

Para que os aplicativos e serviços possam determinar se um dispositivo está conectado, consulte Detectar se um dispositivo está conectado.

> **Important**
>
> Como o MQTT usa um modelo de publish/subscribe comunicação, você deve se inscrever nos tópicos de resposta antes de publicar um tópico de solicitação. Caso contrário, você não deve receber a resposta à solicitação que publicar.
> Se você usa um AWS IoT Device SDKpara chamar o serviço Device Shadow APIs, isso é feito para você.

Os exemplos desta seção usam uma forma abreviada do tópico em que *`ShadowTopicPrefix`* podem se referir a uma sombra nomeada ou não nomeada, conforme descrito nesta tabela.

As sombras podem ser nomeadas ou sem nome (clássica). Os tópicos usados por cada uma diferem apenas no prefixo do tópico. Esta tabela mostra o prefixo do tópico usado em cada tipo de sombra.

Valor do ***`ShadowTopicPrefix`***	Tipo de sombra
`$aws/things/` *`thingName`* `/shadow`	Sombra sem nome (clássica)
`$aws/things/` *`thingName`* `/shadow/name/` *`shadowName`*	Sombra nomeada

> **Important**
>
> Verifique se o uso de sombras por seu aplicativo ou serviço é consistente e compatível com as implementações correspondentes em seus dispositivos. Por exemplo, considere como as sombras são criadas, atualizadas e excluídas. Considere também como as atualizações são processadas no dispositivo e nos aplicativos ou serviços que acessam o dispositivo por meio de uma sombra. Seu design deve ser claro sobre como o estado do dispositivo é atualizado e relatado e como seus aplicativos e serviços interagem com o dispositivo e suas sombras.

Para criar um tópico completo, selecione o *ShadowTopicPrefix* do tipo de sombra ao qual você quer fazer referência, substitua *thingName*, e *shadowName*, se aplicável, por seus valores correspondentes e acrescente isso ao stub de tópico, conforme mostrado na seguinte tabela. Lembre-se de que os tópicos diferenciam maiúsculas de minúsculas.

Consulte Tópicos de sombra para receber mais informações sobre os tópicos reservados para sombras.

Inicializando o dispositivo na primeira conexão com AWS IoT

Depois que um dispositivo se registra AWS IoT, ele deve assinar essas mensagens MQTT para as sombras que ele suporta.

Tópico	Significado	Ação que um dispositivo deve executar quando este tópico é recebido
ShadowTopicPrefix / delete/accepted	A delete solicitação foi aceita e a sombra foi AWS IoT excluída.	As ações necessárias para acomodar a sombra excluída, como interromper a publicaçã o de atualizações.
ShadowTopicPrefix / delete/rejected	A delete solicitação foi rejeitada AWS IoT e a sombra não foi excluída. O corpo da mensagem contém as informações do erro.	Responda à mensagem de erro no corpo da mensagem.

Tópico	Significado	Ação que um dispositivo deve executar quando este tópico é recebido
ShadowTopicPrefix /get/accepted	A get solicitação foi aceita por AWS IoT, e o corpo da mensagem contém o documento paralelo atual.	As ações necessárias para processar o documento de estado no corpo da mensagem.
ShadowTopicPrefix /get/rejected	A get solicitação foi rejeitada por AWS IoT, e o corpo da mensagem contém as informações do erro.	Responda à mensagem de erro no corpo da mensagem.
ShadowTopicPrefix /update/accepted	A update solicitação foi aceita por AWS IoT, e o corpo da mensagem contém o documento paralelo atual.	Confirme se os dados atualizados no corpo da mensagem correspondem ao estado do dispositivo.
ShadowTopicPrefix /update/rejected	A update solicitação foi rejeitada por AWS IoT, e o corpo da mensagem contém as informações do erro.	Responda à mensagem de erro no corpo da mensagem.
ShadowTopicPrefix /update/delta	O documento paralelo foi atualizado por meio de uma solicitação para AWS IoT, e o corpo da mensagem contém as alterações solicitadas.	Atualize o estado do dispositivo para que corresponda ao estado desejado no corpo da mensagem.
ShadowTopicPrefix /update/documents	Uma atualização na sombra foi concluída recentemente e o corpo da mensagem contém o documento de sombra atual.	Confirme se o estado atualizado no corpo da mensagem corresponde ao estado do dispositivo.

Depois de assinar as mensagens na tabela anterior para cada sombra, o dispositivo deve testar para ver se as sombras às quais oferece suporte já foram criadas pela publicação de um tópico /get para

cada sombra. Se uma mensagem `/get/accepted` for recebida, o corpo da mensagem conterá o documento de sombra que o dispositivo pode usar para inicializar seu estado. Se uma mensagem `/get/rejected` for recebida, a sombra deverá ser criada publicando uma mensagem `/update` com o estado do dispositivo atual.

Por exemplo, suponha que você tenha uma `My_IoT_Thing` objeto que não tenha sombras clássicas ou nomeadas. Se você publicar agora uma solicitação `/get` no tópico `$aws/things/My_IoT_Thing/shadow/get` reservado, será retornado um erro no tópico `$aws/things/My_IoT_Thing/shadow/get/rejected` porque o objeto não tem sombras. Para resolver esse erro, primeiro publique uma mensagem `/update` usando o tópico `$aws/things/My_IoT_Thing/shadow/update` com o estado atual do dispositivo, como a carga útil a seguir.

```
{
 "state": {
  "reported": {
   "welcome": "aws-iot",
   "color": "yellow"
  }
 }
}
```

Uma sombra clássica agora é criada para o objeto e a mensagem é publicada no tópico `$aws/things/My_IoT_Thing/shadow/update/accepted`. Se você publicar no tópico `$aws/things/My_IoT_Thing/shadow/get`, ele retornará uma resposta ao tópico `$aws/things/My_IoT_Thing/shadow/get/accepted` com o estado do dispositivo.

Para sombras nomeadas, você deve primeiro criar a sombra nomeada ou publicar uma atualização com o nome da sombra antes de usar a solicitação get. Por exemplo, para criar uma sombra nomeada `namedShadow1`, primeiro publique as informações do estado do dispositivo no tópico `$aws/things/My_IoT_Thing/shadow/name/namedShadow1/update`. Para recuperar as informações do estado, use a solicitação `/get` para a sombra nomeada, `$aws/things/My_IoT_Thing/shadow/name/namedShadow1/get`.

Processando mensagens enquanto o dispositivo está conectado ao AWS IoT

Enquanto um dispositivo está conectado AWS IoT, ele pode receber mensagens /update/delta e deve manter o estado do dispositivo compatível com as mudanças em suas sombras da seguinte forma:

1. Lendo todas as mensagens /update/delta recebidas e sincronizando o estado do dispositivo para que corresponda.
2. Publicando uma mensagem /update com um corpo de mensagem `reported` que tenha o estado atual do dispositivo, sempre que o estado do dispositivo for alterado.

Enquanto um dispositivo está conectado, ele deve publicar essas mensagens quando indicado.

Indicação	Tópico	Carga útil
O estado do dispositivo foi alterado.	*ShadowTopicPrefix* `/update`	Um documento de sombra com a propriedade `reported`.
O dispositivo pode não estar sincronizado com a sombra.	*ShadowTopicPrefix* `/get`	(empty)
Uma ação no dispositivo indica que uma sombra não será mais compatível com o dispositivo, como quando o dispositivo é removido ou substituído.	*ShadowTopicPrefix* `/delete`	(empty)

Processando mensagens quando o dispositivo se reconecta ao AWS IoT

Quando um dispositivo com uma ou mais sombras se conecta AWS IoT, ele deve sincronizar seu estado com o de todas as sombras que ele suporta por meio de:

1. Lendo todas as mensagens /update/delta recebidas e sincronizando o estado do dispositivo para que corresponda.
2. Publicando uma mensagem /update com um corpo de mensagem `reported` que tenha o estado atual do dispositivo.

Usar sombras em aplicativos e serviços

Esta seção descreve como um aplicativo ou serviço interage com o serviço AWS IoT Device Shadow. Este exemplo pressupõe que o aplicativo ou serviço esteja interagindo apenas com a

sombra e, por meio da sombra, com o dispositivo. Este exemplo não inclui nenhuma ação de gerenciamento, como criação ou exclusão de sombras.

Este exemplo usa a API REST do serviço AWS IoT Device Shadow para interagir com sombras. Ao contrário do exemplo usado emUsar sombras em dispositivos, que usa um modelo de publish/subscribe communications model, this example uses the request/response comunicação da API REST. Isso significa que o aplicativo ou serviço deve fazer uma solicitação antes de receber uma resposta de AWS IoT. Uma desvantagem deste modelo, no entanto, é que ele não é compatível com notificações. Se seu aplicativo ou serviço exigir notificações oportunas sobre mudanças no estado do dispositivo, considere os protocolos MQTT ou MQTT over WSS, que oferecem suporte ao modelo de publish/subscribe comunicação, conforme descrito em. Usar sombras em dispositivos

> **Important**
>
> Verifique se o uso de sombras por seu aplicativo ou serviço é consistente e compatível com as implementações correspondentes em seus dispositivos. Considere, por exemplo, como as sombras são criadas, atualizadas e excluídas e como as atualizações são processadas no dispositivo e nos aplicativos ou serviços que acessam a sombra. Seu design deve especificar claramente como o estado do dispositivo é atualizado e relatado, e como seus aplicativos e serviços interagem com o dispositivo e suas sombras.

O URL da API REST para sombras nomeadas é:

```
https://endpoint/things/thingName/shadow?name=shadowName
```

e para uma sombra sem nome:

```
https://endpoint/things/thingName/shadow
```

em que:

endpoint

O endpoint retornado pelo comando da CLI:

```
aws iot describe-endpoint --endpoint-type IOT:Data-ATS
```

thingName

O nome do objeto do objeto ao qual a sombra pertence

shadowName

O nome da sombra nomeada. Esse parâmetro não é usado com sombras sem nome.

Inicializando o aplicativo ou serviço em conexão com AWS IoT

Quando o aplicativo se conecta pela primeira vez AWS IoT, ele deve enviar uma solicitação HTTP GET às URLs sombras que usa para obter o estado atual das sombras que está usando. Isso permite que ele sincronize o aplicativo ou serviço com a sombra.

O estado de processamento muda enquanto o aplicativo ou serviço está conectado ao AWS IoT

Enquanto o aplicativo ou serviço está conectado AWS IoT, ele pode consultar o estado atual periodicamente enviando uma solicitação HTTP GET sobre URLs as sombras que usa.

Quando um usuário final interage com o aplicativo ou serviço para alterar o estado do dispositivo, o aplicativo ou serviço pode enviar uma solicitação HTTP POST para as URLs sombras que ele usa para atualizar o `desired` estado da sombra. Essa solicitação retorna a alteração que foi aceita, mas talvez você precise pesquisar a sombra fazendo solicitações HTTP GET até que o dispositivo atualize a sombra com seu novo estado.

Detectar se um dispositivo está conectado

Para determinar se um dispositivo está conectado no momento, inclua uma propriedade `connected` no documento de sombra e use uma mensagem MQTT Last Will and Testament (LWT) para definir a propriedade `connected` como `false` se um dispositivo for desconectado devido a um erro.

> **Note**
>
> As mensagens MQTT LWT enviadas para tópicos AWS IoT reservados (tópicos que começam com $) são ignoradas pelo serviço AWS IoT Device Shadow. No entanto, eles são processados por clientes inscritos e pelo mecanismo de AWS IoT regras, portanto, você precisará criar uma mensagem LWT que seja enviada para um tópico não reservado e uma

regra que republique a mensagem MQTT LWT como uma mensagem de atualização paralela para o tópico de atualização reservado da sombra,. *ShadowTopicPrefix*/update

Como enviar ao serviço Sombra do Dispositivo uma mensagem LWT

1. Crie uma regra que republica a mensagem MQTT LWT no tópico reservado. O exemplo a seguir é uma regra que escuta mensagens sobre o tópico my/things/myLightBulb/update e republica-a no $aws/things/myLightBulb/shadow/update.

```
{
    "rule": {
    "ruleDisabled": false,
    "sql": "SELECT * FROM 'my/things/myLightBulb/update'",
    "description": "Turn my/things/ into $aws/things/",
    "actions": [
        {
        "republish": {
            "topic": "$$aws/things/myLightBulb/shadow/update",
            "roleArn": "arn:aws:iam:123456789012:role/aws_iot_republish"
            }
        }
     ]
   }
}
```

2. Quando o dispositivo se conecta AWS IoT, ele registra uma mensagem LWT em um tópico não reservado para que a regra de republicação reconheça. Neste exemplo, esse tópico é my/things/myLightBulb/update e define a propriedade conectada como false.

```
{
    "state": {
        "reported": {
            "connected":"false"
        }
    }
}
```

3. Após a conexão, o dispositivo publica uma mensagem em seu tópico de atualização de sombra, $aws/things/myLightBulb/shadow/update, para relatar seu estado atual, o que inclui a configuração de sua propriedade connected como true.

```
{
     "state": {
        "reported": {
            "connected":"true"
        }
    }
}
```

4. Antes de desconectar-se corretamente, o dispositivo publica uma mensagem em seu tópico de atualização de sombra, `$aws/things/myLightBulb/shadow/update`, para relatar seu estado mais recente, que inclui a configuração de sua propriedade `connected` como `false`.

```
{
    "state": {
        "reported": {
            "connected":"false"
        }
    }
}
```

5. Se o dispositivo se desconectar devido a um erro, o agente de mensagens publicará a AWS IoT mensagem LWT do dispositivo em nome do dispositivo. A regra de republicação detecta essa mensagem e publica a mensagem de atualização de sombra para atualizar a `connected` propriedade da sombra do dispositivo.

> **Note**
>
> Devido à natureza assíncrona do processamento de desconexão, não é garantido que as mensagens LWT sejam enviadas em ordem durante a reconexão. Recomendamos que você use eventos de ciclo de vida para melhorar a precisão da detecção do estado de conectividade, pois esses eventos fornecem atributos para gerenciar out-of-order eventos.

Simulando comunicações de serviço Sombra do Dispositivo

Este tópico demonstra como o serviço Sombra do Dispositivo atua como intermediário e permite que dispositivos e aplicativos usem uma sombra para atualizar, armazenar e recuperar o estado de um dispositivo.

Para demonstrar a interação descrita neste tópico e explorá-la mais detalhadamente, você precisará de um Conta da AWS e um sistema no qual possa executar AWS CLI o. Se não os tiver, você ainda poderá ver a interação nos exemplos de código.

Neste exemplo, o AWS IoT console representa o dispositivo. O AWS CLI representa o aplicativo ou serviço que acessa o dispositivo por meio da sombra. A AWS CLI interface é muito semelhante à API que um aplicativo pode usar para se comunicar AWS IoT. O dispositivo neste exemplo é uma lâmpada inteligente, e o aplicativo exibe o estado da lâmpada e pode alterá-lo.

Configurar a simulação

Estes procedimentos inicializam a simulação abrindo o Console do AWS IoT, que simula o dispositivo, e a janela da linha de comando que simula o aplicativo.

Como configurar o ambiente de simulação

1. Você precisará Conta da AWS executar os exemplos deste tópico sozinho. Se você não tiver um Conta da AWS, crie um, conforme descrito emConfigurar Conta da AWS.
2. Abra o Console do AWS IoT, e, no menu esquerdo, escolha Testar para abrir o Cliente MQTT.
3. Em outra janela, abra uma janela de terminal em um sistema que tenha a AWS CLI instalada.

Você deve ter duas janelas abertas: uma com o AWS IoT console na página de teste e outra com um prompt de linha de comando.

Inicializar o dispositivo

Nesta simulação, trabalharemos com um objeto chamado, mySimulatedThing, e sua sombra chamada SimShadow1.

Criar objeto e sua política de IoT

Para criar um objeto, no AWS IoT Console:

1. Escolha Gerenciar e em seguida, Objetos.
2. Clique no botão Criar se as coisas estiverem listadas, caso contrário, clique em Registrar uma única coisa para criar uma única AWS IoT coisa.
3. Insira o nome `mySimulatedThing`, deixe outras configurações como padrão e clique em Próximo.

4. Use a criação de certificado com um clique para gerar os certificados que autenticarão a conexão do dispositivo com o AWS IoT. Clique em Ativar para ativar o certificado.
5. Você pode anexar a política `My_IoT_Policy` que daria permissão ao dispositivo para publicar e assinar os tópicos reservados do MQTT. Para obter etapas mais detalhadas sobre como criar AWS IoT algo e como criar essa política, consulteCriar um objeto.

Crie uma sombra nomeada para o objeto-objeto

Você pode criar uma sombra nomeada para algo publicando uma solicitação de atualização no tópico `$aws/things/mySimulatedThing/shadow/name/simShadow1/update` conforme descrito abaixo.

Ou crie uma sombra nomeada:

1. No AWS IoT Console, escolha seu objeto na lista de itens exibidos e, em seguida, escolha Sombras.
2. Escolha Adicionar uma sombra, insira o nome `simShadow1` e escolha Criar para adicionar a sombra nomeada.

Inscreva-se e publique em tópicos reservados do MQTT

No console, assine os shadow topics reservados do MQTT. Esses tópicos são as respostas às ações `get`, `update` e `delete` para que seu dispositivo esteja pronto para receber as respostas depois de publicar uma ação.

Como assinar um tópico MQTT no Cliente MQTT

1. No cliente MQTT, escolha Assinar um tópico.
2. Insira os tópicos `get`, `update` e `delete` para se inscrever. Copie um tópico por vez da lista a seguir, cole-o no campo Filtro de tópico e clique em Inscrever. Você deverá ver os tópicos aparecerem em Inscrições.

   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/delete/accepted`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/delete/rejected`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/accepted`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/rejected`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/accepted`

- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/rejected`
- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/delta`
- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/documents`

Neste ponto, seu dispositivo simulado está pronto para receber os tópicos conforme são publicados pelo AWS IoT.

Como publicar um tópico MQTT no Cliente MQTT

Depois que um dispositivo se inicializar e assinar tópicos de resposta, ele deve consultar as sombras compatíveis. Essa simulação suporta apenas uma sombra, a sombra que suporta um objeto chamado, mySimulatedThing, chamado SimShadow1.

Como obter o estado da sombra atual no Cliente MQTT

1. No Cliente MQTT, escolha Publicar em um tópico.
2. Em Publicar, insira o seguinte tópico e exclua qualquer conteúdo da janela do corpo da mensagem abaixo de onde você inseriu o tópico para obter. Em seguida, você pode escolher Publicar no tópico para publicar a solicitação. `$aws/things/mySimulatedThing/shadow/name/simShadow1/get`.

   Se você não criou a sombra nomeada, `simShadow1`, você receberá uma mensagem no tópico `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/rejected` e o `code` é `404`, como neste exemplo, a sombra não terá sido criada, portanto, vamos criá-la a seguir.

```
{
  "code": 404,
  "message": "No shadow exists with name: 'simShadow1'"
}
```

Como criar uma sombra com o status atual do dispositivo

1. No Cliente MQTT, escolha Publicar em um tópico e insira este tópico:

```
$aws/things/mySimulatedThing/shadow/name/simShadow1/update
```

2. Na janela de corpo da mensagem abaixo do local em que você inseriu o tópico, insira este documento de sombra para mostrar que o dispositivo está relatando seu ID e sua cor atual em valores RGB. Escolha Publicar para publicar a solicitação.

```
{
  "state": {
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        128,
        128,
        128
      ]
    }
  },
  "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
}
```

Se você receber uma mensagem no tópico:

- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/accepted`: a sombra foi criada e o corpo da mensagem contém o documento de sombra atual.
- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/rejected`: revise o erro no corpo da mensagem.
- `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/accepted`: a sombra já existirá e o corpo da mensagem terá o estado da sombra atual, como neste exemplo. Com isso, você pode definir seu dispositivo ou confirmar que ele corresponde ao estado da sombra.

```
{
  "state": {
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        128,
        128,
        128
      ]
    }
  },
  "metadata": {
```

```
    "reported": {
      "ID": {
        "timestamp": 1591140517
      },
      "ColorRGB": [
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        }
      ]
    }
  },
  "version": 3,
  "timestamp": 1591140517,
  "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
}
```

Enviar uma atualização do aplicativo

Esta seção usa o AWS CLI para demonstrar como um aplicativo pode interagir com uma sombra.

Para obter o estado atual da sombra usando o AWS CLI

Na linha de comando, insira este comando.

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1 /
dev/stdout
```

Nas plataformas Windows, você pode usar `con` ao invés de `/dev/stdout`.

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1
 con
```

Como a sombra existe e foi inicializada pelo dispositivo para refletir seu estado atual, ela deve retornar o seguinte documento de sombra.

```
{
  "state": {
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        128,
        128,
        128
      ]
    }
  },
  "metadata": {
    "reported": {
      "ID": {
        "timestamp": 1591140517
      },
      "ColorRGB": [
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        }
      ]
    }
  },
  "version": 3,
  "timestamp": 1591141111
}
```

O aplicativo pode usar essa resposta para inicializar sua representação do estado do dispositivo.

Se o aplicativo atualizar o estado, como quando um usuário final altera a cor da nossa lâmpada inteligente para amarelo, o aplicativo enviará um comando update-thing-shadow. Esse comando corresponde à API REST `UpdateThingShadow`.

Como atualizar uma sombra de um aplicativo

Na linha de comando, insira este comando.

AWS CLI v2.x

```
aws iot-data update-thing-shadow --thing-name mySimulatedThing --shadow-name
 simShadow1 \
    --cli-binary-format raw-in-base64-out \
    --payload '{"state":{"desired":{"ColorRGB":
[255,255,0]}},"clientToken":"21b21b21-bfd2-4279-8c65-e2f697ff4fab"}' /dev/stdout
```

AWS CLI v1.x

```
aws iot-data update-thing-shadow --thing-name mySimulatedThing --shadow-name
 simShadow1 \
    --payload '{"state":{"desired":{"ColorRGB":
[255,255,0]}},"clientToken":"21b21b21-bfd2-4279-8c65-e2f697ff4fab"}' /dev/stdout
```

Se for bem-sucedido, esse comando deverá retornar o seguinte documento de sombra.

```
{
  "state": {
    "desired": {
      "ColorRGB": [
        255,
        255,
        0
      ]
    }
  },
  "metadata": {
    "desired": {
      "ColorRGB": [
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        }
      ]
    }
  },
```

```
  "version": 4,
  "timestamp": 1591141596,
  "clientToken": "21b21b21-bfd2-4279-8c65-e2f697ff4fab"
}
```

Responder à atualização no dispositivo

Retornando ao cliente MQTT no AWS console, você deve ver as mensagens AWS IoT publicadas para refletir o comando de atualização emitido na seção anterior.

Como exibir as mensagens de atualização no Cliente MQTT

No cliente MQTT, escolha $ aws/things/mySimulatedThing/shadow/name/simShadow1/update/delta na coluna Assinaturas. Se o nome do tópico estiver truncado, você poderá pausar nele para ver o tópico completo. No log do tópico deste tópico, você deverá ver uma mensagem `/delta` semelhante a esta.

```
{
  "version": 4,
  "timestamp": 1591141596,
  "state": {
    "ColorRGB": [
      255,
      255,
      0
    ]
  },
  "metadata": {
    "ColorRGB": [
      {
        "timestamp": 1591141596
      },
      {
        "timestamp": 1591141596
      },
      {
        "timestamp": 1591141596
      }
    ]
  },
  "clientToken": "21b21b21-bfd2-4279-8c65-e2f697ff4fab"
}
```

Seu dispositivo processará o conteúdo desta mensagem para definir o estado do dispositivo para que corresponda ao estado `desired` na mensagem.

Depois que o dispositivo atualizar o estado para corresponder ao `desired` estado na mensagem, ele deverá enviar o novo estado relatado de volta AWS IoT publicando uma mensagem de atualização. Este procedimento simula isso no Cliente MQTT.

Como atualizar a sombra no dispositivo

1. No Cliente MQTT, escolha Publicar em um tópico.
2. Na janela do corpo da mensagem, no campo de tópico acima da janela do corpo da mensagem, insira o tópico do shadow seguido da ação `/update`: `$aws/things/mySimulatedThing/shadow/name/simShadow1/update` e no corpo da mensagem, insira este documento shadow atualizado, que descreve o estado atual do dispositivo. Clique em Publicar para publicar o estado atualizado do dispositivo.

   ```
   {
     "state": {
       "reported": {
         "ColorRGB": [255,255,0]
         }
     },
     "clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
   }
   ```

   Se a mensagem foi recebida com sucesso por AWS IoT, você deverá ver uma nova resposta no registro de aws/things/mySimulatedThing/shadow/name/simShadow1/update/accepted mensagens $ no cliente MQTT com o estado atual da sombra, como neste exemplo.

   ```
   {
     "state": {
       "reported": {
         "ColorRGB": [
           255,
           255,
           0
         ]
       }
     },
     "metadata": {
       "reported": {
   ```

```
      "ColorRGB": [
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        }
      ]
    }
  },
  "version": 5,
  "timestamp": 1591142747,
  "clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
}
```

Uma atualização bem-sucedida do estado relatado do dispositivo também faz AWS IoT com que uma descrição abrangente do estado de sombra em uma mensagem seja enviada ao `update/documents` tópico, como o corpo da mensagem resultante da atualização paralela realizada pelo dispositivo no procedimento anterior.

```
{
  "previous": {
    "state": {
      "desired": {
        "ColorRGB": [
          255,
          255,
          0
        ]
      },
      "reported": {
        "ID": "SmartLamp21",
        "ColorRGB": [
          128,
          128,
          128
        ]
      }
    },
```

```
    "metadata": {
      "desired": {
        "ColorRGB": [
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          }
        ]
      },
      "reported": {
        "ID": {
          "timestamp": 1591140517
        },
        "ColorRGB": [
          {
            "timestamp": 1591140517
          },
          {
            "timestamp": 1591140517
          },
          {
            "timestamp": 1591140517
          }
        ]
      }
    },
    "version": 4
  },
  "current": {
    "state": {
      "desired": {
        "ColorRGB": [
          255,
          255,
          0
        ]
      },
      "reported": {
        "ID": "SmartLamp21",
```

```
        "ColorRGB": [
          255,
          255,
          0
        ]
      }
    },
    "metadata": {
      "desired": {
        "ColorRGB": [
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          }
        ]
      },
      "reported": {
        "ID": {
          "timestamp": 1591140517
        },
        "ColorRGB": [
          {
            "timestamp": 1591142747
          },
          {
            "timestamp": 1591142747
          },
          {
            "timestamp": 1591142747
          }
        ]
      }
    },
    "version": 5
  },
  "timestamp": 1591142747,
  "clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
}
```

Observar a atualização no aplicativo

O aplicativo agora pode consultar o estado atual da sombra conforme relatado pelo dispositivo.

Para obter o estado atual da sombra usando o AWS CLI

1. Na linha de comando, insira este comando.

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name
 simShadow1 /dev/stdout
```

   Nas plataformas Windows, você pode usar `con` ao invés de `/dev/stdout`.

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name
 simShadow1 con
```

2. Como a sombra acaba de ser atualizada pelo dispositivo para refletir seu estado atual, ele deve retornar o seguinte documento de sombra.

```
{
  "state": {
    "desired": {
      "ColorRGB": [
        255,
        255,
        0
      ]
    },
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        255,
        255,
        0
      ]
    }
  },
  "metadata": {
    "desired": {
      "ColorRGB": [
        {
          "timestamp": 1591141596
```

```
        },
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        }
      ]
    },
    "reported": {
      "ID": {
        "timestamp": 1591140517
      },
      "ColorRGB": [
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        }
      ]
    }
  },
  "version": 5,
  "timestamp": 1591143269
}
```

Além da simulação

Experimente a interação entre a AWS CLI (representando o aplicativo) e o console (representando o dispositivo) para modelar sua solução de IoT.

Interagir com sombras

Este tópico descreve as mensagens associadas a cada um dos três métodos que o AWS IoT fornece para trabalhar com sombras. Esses métodos incluem o seguinte:

`UPDATE`

Cria uma sombra, se ela não existir, ou atualiza o conteúdo de uma sombra existente com as informações de estado fornecidas no corpo da mensagem. O AWS IoT registra um carimbo de data/hora com cada atualização para indicar quando o estado foi atualizado pela última vez. Quando o estado da sombra muda, AWS IoT envia `/delta` mensagens para todos os assinantes do MQTT com a diferença entre os `reported` estados `desired` e os. Os dispositivos ou aplicativos que recebem uma mensagem `/delta` podem executar ações com base na diferença. Por exemplo, um dispositivo pode atualizar seu estado para o estado desejado, ou um aplicativo pode atualizar sua interface do usuário (UI) para mostrar a alteração no estado do dispositivo.

`GET`

Recupera um documento de sombra atual que contém o estado completo da sombra, incluindo os metadados.

`DELETE`

Exclui a sombra do dispositivo e o seu conteúdo.

Você não pode restaurar um documento de sombra do dispositivo excluído, mas pode criar um novo documento de sombra do dispositivo com o nome de um documento de sombra do dispositivo excluído. Se você criar um documento de sombra do dispositivo com o mesmo nome daquele que foi excluído nas últimas 48 horas, o número da versão do novo documento de sombra do dispositivo seguirá o do excluído. Se um documento de sombra do dispositivo tiver sido excluído por mais de 48 horas, o número da versão de um novo documento de sombra do dispositivo com o mesmo nome será 0.

Suporte ao protocolo

AWS IoT suporta MQTT e uma API REST sobre protocolos HTTPS para interagir com sombras. AWS IoT fornece um conjunto de tópicos reservados de solicitação e resposta para ações de publicação e assinatura do MQTT. Dispositivos e aplicativos devem se inscrever nos tópicos de resposta antes de publicar em um tópico de solicitação para obter informações sobre como a solicitação AWS IoT foi tratada. Para obter mais informações, consulte Tópicos MQTT da Sombra do Dispositivo e API REST da Sombra do Dispositivo.

Solicitar e declarar o estado

Ao projetar sua solução de IoT usando AWS IoT sombras, você deve determinar os aplicativos ou dispositivos que solicitarão alterações e aqueles que as implementarão. Normalmente, um dispositivo implementa e relata alterações na sombra e os aplicativos e os serviços respondem e solicitam alterações na sombra. Sua solução pode ser diferente, mas os exemplos neste tópico pressupõem que o aplicativo cliente ou o serviço solicita alterações na sombra e o dispositivo executa as alterações e relata-as de volta à sombra.

Atualizar uma shadow

Seu aplicativo ou serviço pode atualizar o estado de uma sombra usando a API UpdateThingShadow ou publicando no tópico /update. As atualizações afetam apenas os campos especificados na solicitação.

Atualizar uma sombra quando um cliente solicita uma alteração de estado

Quando um cliente solicita uma alteração de estado em uma sombra usando o protocolo MQTT

1. O cliente deve ter um documento da sombra atual para que ele possa identificar as propriedades a serem alteradas. Consulte a ação /get para saber como obter o documento de sombra atual.
2. O cliente assina estes tópicos MQTT:
   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/update/rejected`
   - `$aws/things/thingName/shadow/name/shadowName/update/delta`
   - `$aws/things/thingName/shadow/name/shadowName/update/documents`
3. O cliente publica um tópico de solicitação `$aws/things/thingName/shadow/name/shadowName/update` com um documento de estado que contém o estado desejado da sombra. Somente as propriedades a serem alteradas precisam ser incluídas no documento. Este é um exemplo de um documento com o estado desejado.

```
{
  "state": {
    "desired": {
      "color": {
        "r": 10
      },
```

```
      "engine": "ON"
    }
  }
}
```

4. Se a solicitação de atualização for válida, AWS IoT atualiza o estado desejado na sombra e publica mensagens sobre estes tópicos:

   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/update/delta`

   A mensagem `/update/accepted` contém um documento de sombra Documento de estado da resposta /accepted, e a mensagem `/update/delta` contém um documento de sombra Documento de estado da resposta /delta.

5. Se a solicitação de atualização não for válida, AWS IoT publica uma mensagem com o `$aws/things/thingName/shadow/name/shadowName/update/rejected` tópico com um documento Documento de resposta de erro paralelo que descreve o erro.

Quando um cliente solicita uma alteração de estado em uma sombra usando a API

1. O cliente chama a API `UpdateThingShadow` com um documento de estado Documento de estado de solicitação como corpo da mensagem.
2. Se a solicitação for válida, AWS IoT retornará um código HTTP de resposta bem-sucedida e um documento Documento de estado da resposta /accepted paralelo como corpo da mensagem de resposta.

   AWS IoT também publicará uma mensagem MQTT no `$aws/things/thingName/shadow/name/shadowName/update/delta` tópico com um documento Documento de estado da resposta /delta paralelo para todos os dispositivos ou clientes que a assinarem.
3. Se a solicitação não for válida, AWS IoT retornará um código de resposta de erro HTTP e Documento de resposta de erro como corpo da mensagem de resposta.

Quando o dispositivo recebe o estado `/desired` no tópico `/update/delta`, ele faz as alterações desejadas no dispositivo. Depois, ele envia uma mensagem ao tópico `/update` para relatar seu estado atual para a sombra.

Atualizar uma sombra quando um dispositivo relata seu estado atual

Quando um dispositivo relata seu estado atual para a sombra usando o protocolo MQTT

1. O dispositivo deve assinar estes tópicos MQTT antes de atualizar a sombra:
   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/update/rejected`
   - `$aws/things/thingName/shadow/name/shadowName/update/delta`
   - `$aws/things/thingName/shadow/name/shadowName/update/documents`
2. O dispositivo relata seu estado atual publicando uma mensagem no tópico `$aws/things/thingName/shadow/name/shadowName/update` que relata o estado atual, como neste exemplo.

```
{
    "state": {
        "reported" : {
            "color" : { "r" : 10 },
            "engine" : "ON"
        }
    }
}
```

3. Se AWS IoT aceitar a atualização, ele publicará uma mensagem nos `$aws/things/thingName/shadow/name/shadowName/update/accepted` tópicos com um documento Documento de estado da resposta /accepted paralelo.
4. Se a solicitação de atualização não for válida, AWS IoT publica uma mensagem com o `$aws/things/thingName/shadow/name/shadowName/update/rejected` tópico com um documento Documento de resposta de erro paralelo que descreve o erro.

Quando um dispositivo relata seu estado atual para a sombra usando a API

1. O dispositivo chama a API `UpdateThingShadow` com um documento de estado Documento de estado de solicitação como corpo da mensagem.
2. Se a solicitação for válida, AWS IoT atualiza a sombra e retorna um código de resposta HTTP bem-sucedida com um documento Documento de estado da resposta /accepted paralelo como corpo da mensagem de resposta.

AWS IoT também publicará uma mensagem MQTT no `$aws/things/thingName/shadow/name/shadowName/update/delta` tópico com um documento Documento de estado da resposta /delta paralelo para todos os dispositivos ou clientes que a assinarem.

3. Se a solicitação não for válida, AWS IoT retornará um código de resposta de erro HTTP e Documento de resposta de erro como corpo da mensagem de resposta.

Bloqueio otimista

Você pode usar a versão do documento de estado para garantir que está atualizando a versão mais recente de um documento de sombra do dispositivo. Ao fornecer uma versão com uma solicitação de atualização, o serviço rejeitará a solicitação com um código de resposta de conflito HTTP 409 se a versão atual do documento de estado não corresponder à versão fornecida. O código de resposta de conflito também pode ocorrer em qualquer API que modifique `ThingShadow`, incluindo `DeleteThingShadow`.

Por exemplo:

Documento inicial:

```
{
  "state": {
    "desired": {
      "colors": [
        "RED",
        "GREEN",
        "BLUE"
      ]
    }
  },
  "version": 10
}
```

Atualização: (versão não corresponde, a solicitação será rejeitada)

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
```

```
      ]
    }
  },
  "version": 9
}
```

Resultado:

```
{
  "code": 409,
  "message": "Version conflict",
  "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
}
```

Atualização: (a versão corresponde; a solicitação será aceita)

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 10
}
```

Estado final:

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 11
}
```

Recuperar um documento de shadow

Você pode recuperar um documento de sombra usando a API GetThingShadow ou assinando e publicando no tópico /get. Isso recupera um documento de sombra completo, incluindo qualquer delta entre os estados `reported` e `desired`. O procedimento para esta tarefa é o mesmo para um dispositivo ou para um cliente que faz a solicitação.

Como recuperar um documento de sombra usando o protocolo MQTT

1. O dispositivo ou o cliente deve assinar estes tópicos MQTT antes de atualizar a sombra:
   - `$aws/things/thingName/shadow/name/shadowName/get/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/get/rejected`
2. O dispositivo ou o cliente publica uma mensagem no tópico `$aws/things/thingName/shadow/name/shadowName/get` com um corpo de mensagem vazio.
3. Se a solicitação for bem-sucedida, AWS IoT publica uma mensagem no `$aws/things/thingName/shadow/name/shadowName/get/accepted` tópico com um Documento de estado da resposta /accepted no corpo da mensagem.
4. Se a solicitação não for válida, AWS IoT publica uma mensagem no `$aws/things/thingName/shadow/name/shadowName/get/rejected` tópico com um Documento de resposta de erro no corpo da mensagem.

Como recuperar um documento de sombra usando uma API REST

1. O dispositivo ou o cliente chama a API `GetThingShadow` com um corpo de mensagem vazio.
2. Se a solicitação for válida, AWS IoT retornará um código HTTP de resposta bem-sucedida com um documento Documento de estado da resposta /accepted paralelo como corpo da mensagem de resposta.
3. Se a solicitação não for válida, AWS IoT retornará um código de resposta de erro HTTP e Documento de resposta de erro como corpo da mensagem de resposta.

Excluir dados de sombra

Existem duas maneiras de excluir dados de sombra: você pode excluir propriedades específicas no documento de sombra e excluir a sombra completamente.

- Para excluir propriedades específicas de uma sombra, atualize a sombra. No entanto, defina o valor das propriedades que você deseja excluir como `null`. Os campos com um valor de `null` são removidos do documento de sombra.
- Para excluir a sombra inteira, use a API DeleteThingShadow ou publique no tópico /delete.

> **Note**
>
> Excluir uma sombra não zera o número da versão imediatamente. Ele será redefinido para zero após 48 horas.

Excluir uma propriedade de um documento de sombra

Como excluir uma propriedade de uma sombra usando o protocolo MQTT

1. O dispositivo ou o cliente deve ter um documento de sombra atual para que possa identificar as propriedades a serem alteradas. Consulte Recuperar um documento de shadow para acessar informações sobre como ter o documento de sombra atual.
2. O dispositivo ou o cliente assina estes tópicos MQTT:
   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/update/rejected`
3. O dispositivo ou o cliente publica um tópico de solicitação `$aws/things/thingName/shadow/name/shadowName/update` com um documento de estado que atribui valores `null` às propriedades da sombra a ser excluída. Somente as propriedades a serem alteradas precisam ser incluídas no documento. Este é um exemplo de um documento que exclui a propriedade `engine`.

   ```
   {
     "state": {
       "desired": {
         "engine": null
       }
     }
   }
   ```
4. Se a solicitação de atualização for válida, AWS IoT excluirá as propriedades especificadas na sombra e publicará uma mensagem com o `$aws/things/thingName/shadow/`

name/*shadowName*/update/accepted tópico com um documento Documento de estado da resposta /accepted paralelo no corpo da mensagem.

5. Se a solicitação de atualização não for válida, AWS IoT publica uma mensagem com o $aws/things/*thingName*/shadow/name/*shadowName*/update/rejected tópico com um documento Documento de resposta de erro paralelo que descreve o erro.

Como excluir uma propriedade de uma sombra usando a API REST

1. O dispositivo ou o cliente chama a API UpdateThingShadow com um Documento de estado de solicitação que atribui valores null às propriedades da sombra a ser excluída. Inclua apenas as propriedades que você deseja excluir no documento. Este é um exemplo de um documento que exclui a propriedade engine.

```
{
  "state": {
    "desired": {
      "engine": null
    }
  }
}
```

2. Se a solicitação for válida, AWS IoT retornará um código HTTP de resposta bem-sucedida e um documento Documento de estado da resposta /accepted paralelo como corpo da mensagem de resposta.
3. Se a solicitação não for válida, AWS IoT retornará um código de resposta de erro HTTP e Documento de resposta de erro como corpo da mensagem de resposta.

Excluir uma shadow

Veja a seguir algumas considerações ao excluir a sombra de um dispositivo.

- Definir o estado da sombra do dispositivo como null não exclui a sombra. A versão da sombra será incrementada na próxima atualização.
- A exclusão de uma sobra de dispositivo não exclui o objeto. A exclusão de um objeto não exclui a sombra do dispositivo correspondente.
- Excluir uma sombra não zera o número da versão imediatamente. Ele será redefinido para zero após 48 horas.

Como excluir uma sombra usando o protocolo MQTT

1. O dispositivo ou o cliente assina estes tópicos MQTT:
   - `$aws/things/thingName/shadow/name/shadowName/delete/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/delete/rejected`
2. O dispositivo ou o cliente publica um `$aws/things/thingName/shadow/name/shadowName/delete` com um buffer de mensagens vazio.
3. Se a solicitação de exclusão for válida, AWS IoT excluirá a sombra e publicará uma mensagem com o `$aws/things/thingName/shadow/name/shadowName/delete/accepted` tópico e um Documento de estado da resposta /accepted documento-sombra abreviado no corpo da mensagem. Este é um exemplo de mensagem de exclusão aceita:

   ```
   {
     "version": 4,
     "timestamp": 1591057529
   }
   ```
4. Se a solicitação de atualização não for válida, AWS IoT publica uma mensagem com o `$aws/things/thingName/shadow/name/shadowName/delete/rejected` tópico com um documento Documento de resposta de erro paralelo que descreve o erro.

Como excluir uma sombra usando a API REST

1. O dispositivo ou o cliente chama a API `DeleteThingShadow` com um buffer de mensagens vazio.
2. Se a solicitação for válida, AWS IoT retornará um código de resposta de sucesso HTTP Documento de estado da resposta /accepted e um Documento de estado da resposta /accepted documento-sombra abreviado no corpo da mensagem. Este é um exemplo de mensagem de exclusão aceita:

   ```
   {
     "version": 4,
     "timestamp": 1591057529
   }
   ```
3. Se a solicitação não for válida, AWS IoT retornará um código de resposta de erro HTTP e Documento de resposta de erro como corpo da mensagem de resposta.

API REST da Sombra do Dispositivo

Um shadow expõe a seguinte URI para atualizar as informações de estado:

```
https://account-specific-prefix-ats.iot.region.amazonaws.com/things/thingName/shadow
```

O endpoint é específico para o seu Conta da AWS. Para encontrar o endpoint, você pode:

- Usar o comando describe-endpoint do AWS CLI.
- Use as configurações AWS IoT do console. Em Configurações, o endpoint está listado em Endpoint personalizado
- Use a página de detalhes do item do AWS IoT console. No console do:
  1. Abra Gerenciar e, em Gerenciar, escolha Objetos.
  2. Na lista de itens, escolha o item para o qual você deseja obter o URI do endpoint.
  3. Escolha a guia Sombras do Dispositivo e escolha sua sombra. Você pode visualizar o URI do endpoint na seção URL Sombra do Dispositivo da página de detalhes da Sombra do Dispositivo.

O formato do endpoint é o seguinte:

```
identifier.iot.region.amazonaws.com
```

A Shadow REST API segue os mesmos protocols/port mapeamentos HTTPS descritos em. Protocolos de comunicação do dispositivo

> **Note**
>
> Para usar o APIs, você deve usar `iotdevicegateway` como nome do serviço para autenticação. Para obter mais informações, consulte Io TData Plane.

Ações da API

- GetThingShadow
- UpdateThingShadow
- DeleteThingShadow
- ListNamedShadowsForThing

Você também pode usar a API para criar uma sombra nomeada fornecendo `name=`*`shadowName`* como parte do parâmetro de consulta da API.

GetThingShadow

Obtém o shadow do objeto especificada.

O documento de estado de resposta inclui o delta entre os estados `desired` e `reported`.

Solicitação

A solicitação inclui os cabeçalhos HTTP padrão, além da seguinte URI:

```
HTTP GET https://endpoint/things/thingName/shadow?name=shadowName
Request body: (none)
```

O parâmetro de consulta `name` não é necessário para sombras sem nome (clássicas).

Resposta

Após o sucesso, a resposta incluirá os cabeçalhos HTTP padrão e os seguintes código e corpo:

```
HTTP 200
Response Body: response state document
```

Para obter mais informações, consulte Exemplo de documento de estado de resposta.

Autorização

Recuperar um shadow requer uma política que permite que o chamador execute a ação `iot:GetThingShadow`. O serviço Sombra do Dispositivo aceita duas formas de autenticação: assinatura da versão 4 com credenciais do IAM ou autenticação mútua do TLS com um certificado cliente.

Veja a seguir um exemplo de política que permite que um chamador recupere uma shadow de dispositivo:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": "iot:GetThingShadow",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:thing/thing"
            ]
        }
    ]
}
```

UpdateThingShadow

Atualiza o shadow do objeto especificada.

As atualizações afetam apenas os campos especificados no documento de estado da solicitação. Os campos com um valor de `null` são removido da shadow de dispositivo.

Solicitação

A solicitação inclui os cabeçalhos HTTP padrão e os seguintes URI e corpo:

```
HTTP POST https://endpoint/things/thingName/shadow?name=shadowName
Request body: request state document
```

O parâmetro de consulta `name` não é necessário para sombras sem nome (clássicas).

Para obter mais informações, consulte Exemplo de documento de estado de solicitação.

Resposta

Após o sucesso, a resposta incluirá os cabeçalhos HTTP padrão e os seguintes código e corpo:

```
HTTP 200
Response body: response state document
```

Para obter mais informações, consulte Exemplo de documento de estado de resposta.

Autorização

Atualizar um shadow requer uma política que permite que o chamador execute a ação `iot:UpdateThingShadow`. O serviço Sombra do Dispositivo aceita duas formas de autenticação: assinatura da versão 4 com credenciais do IAM ou autenticação mútua do TLS com um certificado cliente.

Veja a seguir um exemplo de política que permite que um chamador atualize uma shadow de dispositivo:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:UpdateThingShadow",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:thing/thing"
            ]
        }
    ]
}
```

DeleteThingShadow

Apaga o shadow do objeto especificada.

Solicitação

A solicitação inclui os cabeçalhos HTTP padrão, além da seguinte URI:

```
HTTP DELETE https://endpoint/things/thingName/shadow?name=shadowName
Request body: (none)
```

O parâmetro de consulta `name` não é necessário para sombras sem nome (clássicas).

Resposta

Após o sucesso, a resposta incluirá os cabeçalhos HTTP padrão e os seguintes código e corpo:

```
HTTP 200
Response body: Empty response state document
```

Observe que a exclusão de uma sombra não redefine seu número da versão para 0.

Autorização

Apagar uma shadow de dispositivo requer uma política que permite que o chamador execute a ação `iot:DeleteThingShadow`. O serviço Sombra do Dispositivo aceita duas formas de autenticação: assinatura da versão 4 com credenciais do IAM ou autenticação mútua do TLS com um certificado cliente.

Veja a seguir um exemplo de política que permite que um chamador exclua uma shadow de dispositivo:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:DeleteThingShadow",
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:thing/thing"
            ]
        }
    ]
}
```

ListNamedShadowsForThing

Lista as sombras do objeto especificada.

Solicitação

A solicitação inclui os cabeçalhos HTTP padrão, além da seguinte URI:

```
HTTP GET /api/things/shadow/ListNamedShadowsForThing/thingName?
nextToken=nextToken&pageSize=pageSize
Request body: (none)
```

nextToken

O token para recuperação do próximo conjunto de resultados.

Esse valor é retornado nos resultados paginados e é usado na chamada que retorna a próxima página.

pageSize

O número de nomes de sombra a serem retornados em cada chamada. Consulte também `nextToken`.

thingName

O nome do objeto para a qual listar as sombras nomeadas.

Resposta

Quando bem-sucedido, a resposta incluirá os cabeçalhos HTTP padrão e o seguinte código de resposta e um Documento de resposta da lista de nomes de sombra.

> **Note**
>
> A sombra sem nome (clássica) não aparece nesta lista. A resposta é uma lista vazia se você tiver apenas uma sombra clássica ou se a `thingName` especificada não existir.

```
HTTP 200
Response body: Shadow name list document
```

Autorização

Listar uma shadow de dispositivo requer uma política que permite que o chamador execute a ação `iot:ListNamedShadowsForThing`. O serviço Sombra do Dispositivo aceita duas formas de autenticação: assinatura da versão 4 com credenciais do IAM ou autenticação mútua do TLS com um certificado cliente.

Veja a seguir um exemplo de política que permite que um chamador liste as sombras nomeadas de um objeto:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:ListNamedShadowsForThing",
```

```
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:thing/thing"
            ]
        }
    ]
}
```

Tópicos MQTT da Sombra do Dispositivo

O serviço Sombra do Dispositivo usa tópicos MQTT reservados para permitir que aplicativos e dispositivos obtenham, atualizem ou apaguem as informações de estado de um dispositivo (sombra).

Para publicar e inscrever-se em tópicos de shadow, é necessário ter autorização baseada em tópicos. A AWS IoT reserva-se o direito de adicionar novos tópicos à estrutura de tópicos existente. Por esse motivo, recomendamos que você evite assinaturas curinga para os tópicos de shadow. Por exemplo, evite assinar filtros de tópicos, `$aws/things/thingName/shadow/#` pois o número de tópicos que correspondem a esse filtro de tópicos pode aumentar à medida que novos tópicos paralelos AWS IoT são introduzidos. Para ver exemplos de mensagens publicadas nesses tópicos, consulte Interagir com sombras.

As sombras podem ser nomeadas ou sem nome (clássica). Os tópicos usados por cada uma diferem apenas no prefixo do tópico. Esta tabela mostra o prefixo do tópico usado em cada tipo de sombra.

Valor do ***ShadowTopicPrefix***	Tipo de sombra
`$aws/things/` *`thingName`* `/shadow`	Sombra sem nome (clássica)
`$aws/things/` *`thingName`* `/shadow/name/` *`shadowName`*	Sombra nomeada

Para criar um tópico completo, selecione o *`ShadowTopicPrefix`* do tipo de sombra ao qual você quer fazer referência, substitua *`thingName`* e *`shadowName`*, se aplicável, por seus valores correspondentes e acrescente isso com o stub de tópico, conforme mostrado nas seguintes seções.

Veja a seguir os tópicos do MQTT usados para interagir com shadows.

Tópicos

- /get

- /get/accepted
- /get/rejected
- /update
- /update/delta
- /update/accepted
- /update/documents
- /update/rejected
- /delete
- /delete/accepted
- /delete/rejected

/get

Publique uma mensagem vazia nesse tópico para obter a shadow de dispositivo:

```
ShadowTopicPrefix/get
```

AWS IoT responde publicando em um /get/accepted ou/get/rejected.

Exemplo de política

Veja a seguir um exemplo da política necessária:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/get"
            ]
```

```
        }
    ]
}
```

/get/accepted

AWS IoT publica um documento paralelo de resposta para este tópico ao retornar a sombra do dispositivo:

```
ShadowTopicPrefix/get/accepted
```

Para obter mais informações, consulte Documentos de estado da resposta.

Exemplo de política

Veja a seguir um exemplo da política necessária:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/get/accepted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/get/accepted"
            ]
        }
```

```
    ]
}
```

/get/rejected

AWS IoT publica um documento de resposta de erro para este tópico quando não consegue retornar a sombra do dispositivo:

```
ShadowTopicPrefix/get/rejected
```

Para obter mais informações, consulte Documento de resposta de erro.

Exemplo de política

Veja a seguir um exemplo da política necessária:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/thingName/
shadow/get/rejected"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/shadow/get/
rejected"
      ]
    }
  ]
```

```
}
```

/update

Publique um documento de estado da solicitação nesse tópico para atualizar a shadow de dispositivo:

```
ShadowTopicPrefix/update
```

O corpo da mensagem contém um documento de estado de solicitação parcial.

Ao tentar atualizar o estado de um dispositivo, um cliente envia um documento de estado de solicitação JSON com a propriedade `desired` como este:

```
{
  "state": {
    "desired": {
      "color": "red",
      "power": "on"
    }
  }
}
```

Ao atualizar sua sombra, um dispositivo envia um documento de estado de solicitação JSON com a propriedade `reported`, como este:

```
{
  "state": {
    "reported": {
      "color": "red",
      "power": "on"
    }
  }
}
```

AWS IoT responde publicando em um /update/accepted ou/update/rejected.

Exemplo de política

Veja a seguir um exemplo da política necessária:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/update"
            ]
        }
    ]
}
```

/update/delta

AWS IoT publica um documento de estado de resposta para esse tópico quando aceita uma alteração na sombra do dispositivo, e o documento de estado de resposta contém valores `desired` e `reported` estados diferentes para:

```
ShadowTopicPrefix/update/delta
```

O buffer de mensagens contém um Documento de estado da resposta /delta.

Detalhes do corpo da mensagem

- Uma mensagem publicada em `update/delta` inclui apenas os atributos desejados que diferem entre as seções `desired` e `reported`. Ela contém todos esses atributos, independentemente de esses atributos estarem contidos na mensagem de atualização atual ou de já estarem armazenados na AWS IoT. Os atributos que não diferem entre as seções `desired` e `reported` não são incluídos.
- Se um atributo estiver na seção `reported`, mas não tiver equivalente na seção `desired`, ele não será incluído.
- Se um atributo estiver na seção `desired`, mas não tiver equivalente na seção `reported`, ele será incluído.

- Se um atributo for excluído da seção `reported`, mas ainda existir na seção `desired`, ele será incluído.

Exemplo de política

Veja a seguir um exemplo da política necessária:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/update/delta"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/update/delta"
            ]
        }
    ]
}
```

/update/accepted

AWS IoT publica um documento de estado de resposta a esse tópico quando aceita uma alteração na sombra do dispositivo:

```
ShadowTopicPrefix/update/accepted
```

O buffer de mensagens contém um Documento de estado da resposta /accepted.

Exemplo de política

Veja a seguir um exemplo da política necessária:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/update/accepted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/update/accepted"
            ]
        }
    ]
}
```

/update/documents

AWS IoT publica um documento de estado neste tópico sempre que uma atualização na sombra for executada com sucesso:

```
ShadowTopicPrefix/update/documents
```

O corpo da mensagem contém um Documento de estado da resposta /documents.

Exemplo de política

Veja a seguir um exemplo da política necessária:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/update/documents"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/update/documents"
            ]
        }
    ]
}
```

/update/rejected

AWS IoT publica um documento de resposta de erro para esse tópico quando rejeita uma alteração na sombra do dispositivo:

```
ShadowTopicPrefix/update/rejected
```

O corpo da mensagem contém um Documento de resposta de erro.

Exemplo de política

Veja a seguir um exemplo da política necessária:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/update/rejected"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/update/rejected"
            ]
        }
    ]
}
```

/delete

Para excluir uma shadow de dispositivo, publique uma mensagem vazia no tópico de exclusão:

```
ShadowTopicPrefix/delete
```

O conteúdo da mensagem é ignorado.

Observe que a exclusão de uma sombra não redefine seu número da versão para 0.

AWS IoT responde publicando em um /delete/accepted ou/delete/rejected.

Exemplo de política

Veja a seguir um exemplo da política necessária:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/delete"
            ]
        }
    ]
}
```

/delete/accepted

AWS IoT publica uma mensagem neste tópico quando a sombra de um dispositivo é excluída:

```
ShadowTopicPrefix/delete/accepted
```

Exemplo de política

Veja a seguir um exemplo da política necessária:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
```

```
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/delete/accepted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/delete/accepted"
            ]
        }
    ]
}
```

/delete/rejected

AWS IoT publica um documento de resposta de erro para este tópico quando não consegue excluir a sombra do dispositivo:

```
ShadowTopicPrefix/delete/rejected
```

O corpo da mensagem contém um Documento de resposta de erro.

Exemplo de política

Veja a seguir um exemplo da política necessária:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
things/thingName/shadow/delete/rejected"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thingName/
shadow/delete/rejected"
            ]
        }
    ]
}
```

Documentos do serviço Sombra do Dispositivo

O serviço Sombra do Dispositivo respeita todas as regras da especificação JSON. Os valores, objetos e matrizes são armazenados no documento de shadow do dispositivo.

Conteúdo

- Exemplos de documentos de sombra
- Propriedades do documento
- Estado delta
- Versionamento de documentos de sombra
- Tokens de cliente em documentos de sombra
- Propriedades vazias do documento de sombra
- Valores de matriz em documentos de sombra

Exemplos de documentos de sombra

O serviço Device Shadow usa esses documentos nas operações UPDATE, GET e DELETE usando a API REST ou Pub/Sub mensagens MQTT.

Exemplos

- Documento de estado de solicitação

- Documentos de estado da resposta
- Documento de resposta de erro
- Documento de resposta da lista de nomes de sombra

Documento de estado de solicitação

Um documento de estado de solicitação tem o seguinte formato:

```
{
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        },
        "reported": {
            "attribute1": integer1,
            "attribute2": "string1",
            ...
            "attributeN": boolean1
        }
    },
    "clientToken": "token",
    "version": version
}
```

- `state` — As atualizações afetam apenas os campos especificados. Normalmente, você usará a propriedade `reported` ou `desired`, mas não ambas na mesma solicitação.
  - `desired` — As propriedades de estado e os valores solicitados para serem atualizados no dispositivo.
  - `reported` — As propriedades de estado e os valores relatados pelo dispositivo.
- `clientToken` — Se usado, é possível combinar a solicitação e a resposta correspondente pelo token do cliente.
- `version` — Se usado, o serviço Sombra do Dispositivo processará a atualização somente se a versão especificada corresponder à versão mais recente que ele tem.

Documentos de estado da resposta

Os documentos de estado de resposta têm o seguinte formato, dependendo do tipo de resposta.

Documento de estado da resposta /accepted

```
{
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        }
    },
    "metadata": {
        "desired": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        }
    },
    "timestamp": timestamp,
    "clientToken": "token",
    "version": version
}
```

Documento de estado da resposta /delta

```
{
    "state": {
        "attribute1": integer2,
        "attribute2": "string2",
        ...
        "attributeN": boolean2
    },
```

```
    "metadata": {
        "attribute1": {
            "timestamp": timestamp
        },
        "attribute2": {
            "timestamp": timestamp
        },
        ...
        "attributeN": {
            "timestamp": timestamp
        }
    },
    "timestamp": timestamp,
    "clientToken": "token",
    "version": version
}
```

Documento de estado da resposta /documents

```
{
  "previous" : {
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        },
        "reported": {
            "attribute1": integer1,
            "attribute2": "string1",
            ...
            "attributeN": boolean1
        }
    },
    "metadata": {
        "desired": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
```

```
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        },
        "reported": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        }
    },
    "version": version-1
},
"current": {
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        },
        "reported": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        }
    },
    "metadata": {
        "desired": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
```

```
            "attributeN": {
                "timestamp": timestamp
            }
        },
        "reported": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        }
    },
    "version": version
  },
  "timestamp": timestamp,
  "clientToken": "token"
}
```

Propriedades do documento de estado de resposta

- `previous` — Após uma atualização bem-sucedida, contém o `state` do objeto antes da atualização.
- `current` — Após uma atualização bem-sucedida, contém o `state` do objeto após a atualização.
- `state`
  - `reported` — Presente apenas se um objeto relatou algum dado na seção `reported` e tiver apenas campos que estavam no documento de estado da solicitação.
  - `desired` — Presente apenas se um dispositivo relatou algum dado na seção `desired` e tiver apenas campos que estavam no documento de estado da solicitação.
- `metadata` — Contém as marcações de data e hora de cada atributo nas seções `desired` e `reported` para que você possa determinar quando o estado foi atualizado.
- `timestamp`— A data e a hora da Epoch pelas quais a resposta foi gerada AWS IoT.
- `clientToken` — Presente somente se um token cliente foi usado ao publicar um JSON válido no tópico `/update`.

- `version` — A versão atual do documento da shadow de dispositivo compartilhado na AWS IoT. Ela é incrementada em um em relação à versão anterior do documento.

Documento de resposta de erro

Um documento de resposta de erro tem o seguinte formato:

```
{
    "code": error-code,
    "message": "error-message",
    "timestamp": timestamp,
    "clientToken": "token"
}
```

- `code` — Um código de resposta HTTP que indica o tipo de erro.
- `message` — Uma mensagem de texto que fornece informações adicionais.
- `timestamp`— A data e a hora pelas quais a resposta foi gerada AWS IoT. Esta propriedade não está presente em todos os documentos de resposta do erro.
- `clientToken` — Presente somente se um token de cliente foi usado na mensagem publicada.

Para obter mais informações, consulte Mensagens de erro da Sombra do Dispositivo.

Documento de resposta da lista de nomes de sombra

Um documento de resposta de lista de nomes de sombra tem o seguinte formato:

```
{
    "results": [
        "shadowName-1",
        "shadowName-2",
        "shadowName-3",
        "shadowName-n"
    ],
    "nextToken": "nextToken",
    "timestamp": timestamp
}
```

- `results` — A matriz de nomes de sombras.

- `nextToken` — O valor do token a ser usado em solicitações paginadas para obter a próxima página na sequência. Essa propriedade não está presente quando não há mais nomes de sombra para retornar.
- `timestamp`— A data e a hora pelas quais a resposta foi gerada AWS IoT.

Propriedades do documento

Um documento de shadow de dispositivo tem as seguintes propriedades:

`state`

`desired`

O estado desejado do dispositivo. Os aplicativos podem gravar nessa parte do documento para atualizar o estado de um dispositivo diretamente, sem precisar se conectar a ele.

`reported`

O estado relatado do objeto. Os dispositivos gravam nessa parte do documento para relatar seu novo estado. Os aplicativos leem essa parte do documento para determinar o último estado relatado do dispositivo.

`metadata`

Informações sobre os dados armazenados na seção `state` do documento. Isso inclui marcações de data e hora, no tempo Epoch, para cada atributo na seção `state`, que permite determinar quando foram atualizados.

> **Note**
>
> Os metadados não contribuem para o tamanho do documento para Limites de serviço ou definição de preço. Para obter mais informações, consulte Limites de serviço da AWS IoT.

`timestamp`

Indica quando a mensagem foi enviada por AWS IoT. Com o uso do carimbo de data/hora na mensagem e os carimbos de data/hora de atributos individuais na seção `desired` ou `reported`, um dispositivo pode determinar a idade de uma propriedade, mesmo que o dispositivo não tenha um relógio interno.

`clientToken`

Uma string exclusiva do dispositivo que permite associar respostas com as solicitações em um ambiente MQTT.

`version`

A versão do documento. Toda vez que o documento é atualizado, o número da versão é incrementado. Usado para garantir que a versão do documento que está sendo atualizado seja a mais recente.

Para obter mais informações, consulte Exemplos de documentos de sombra.

Estado delta

Estado delta é um tipo virtual de estado que contém a diferença entre os estados `desired` e `reported`. Os campos na seção `desired` que não estão na seção `reported` são incluídos no delta. Os campos na seção `reported` e que não estão na seção `desired` não são incluídos no delta. O delta contém metadados, e seus valores são iguais aos metadados no campo `desired`. Por exemplo:

```
{
  "state": {
    "desired": {
      "color": "RED",
      "state": "STOP"
    },
    "reported": {
      "color": "GREEN",
      "engine": "ON"
    },
    "delta": {
      "color": "RED",
      "state": "STOP"
    }
  },
  "metadata": {
    "desired": {
      "color": {
        "timestamp": 12345
      },
      "state": {
```

```
        "timestamp": 12345
      }
      },
      "reported": {
        "color": {
          "timestamp": 12345
        },
        "engine": {
          "timestamp": 12345
        }
      },
      "delta": {
        "color": {
          "timestamp": 12345
        },
        "state": {
          "timestamp": 12345
        }
      }
    },
    "version": 17,
    "timestamp": 123456789
  }
}
```

Quando os objetos aninhados diferem, o delta contém o caminho para a raiz.

```
{
  "state": {
    "desired": {
      "lights": {
        "color": {
          "r": 255,
          "g": 255,
          "b": 255
        }
      }
    },
    "reported": {
      "lights": {
        "color": {
          "r": 255,
          "g": 0,
```

```
          "b": 255
        }
      }
    },
    "delta": {
      "lights": {
        "color": {
          "g": 255
        }
      }
    }
  },
  "version": 18,
  "timestamp": 123456789
}
```

O serviço Sombra do Dispositivo calcula o delta Percorrendo cada campo no estado `desired` e comparando-o com o estado `reported`.

As matrizes são processadas como valores. Se uma matriz na seção `desired` não corresponder à matriz na seção `reported`, toda a matriz desejada será copiada para o delta.

Versionamento de documentos de sombra

O serviço Sombra do Dispositivo é compatível com o versionamento em cada mensagem de atualização, tanto de solicitação quanto de resposta. Isso indica que a cada atualização de uma sombra, a versão do documento JSON é incrementada. Isso garante duas objetos:

- Um cliente poderá receber um erro se tentar substituir um shadow usando um número da versão mais antigo. O cliente é informado de que deve sincronizar novamente antes de atualizar uma shadow de dispositivo.
- Um cliente poderá decidir não atuar em uma mensagem recebida se a mensagem tiver uma versão inferior à versão armazenada pelo cliente.

Um cliente pode ignorar a correspondência de versão não incluindo uma versão no documento de sombra.

Tokens de cliente em documentos de sombra

Você pode usar um token cliente com um sistema de mensagens com base em MQTT para verificar se o mesmo token cliente está contido em uma solicitação e em uma resposta de solicitação. Isso garante que a resposta e a solicitação estejam associadas.

> **Note**
>
> O token do cliente pode ter até 64 bytes. Um token de cliente com mais de 64 bytes resulta em uma resposta 400 (Solicitação inválida) e na mensagem de erro clientToken inválido.

Propriedades vazias do documento de sombra

As propriedades `reported` e `desired` em um documento de sombra podem estar vazias ou omitidas quando não se aplicam ao estado da sombra atual. Por exemplo, um documento de sombra contém uma propriedade `desired` somente se tiver um estado desejado. Veja a seguir um exemplo válido de um documento de estado sem a propriedade `desired`:

```
{
    "reported" : { "temp": 55 }
}
```

A propriedade `reported` também pode estar vazia, como se a sombra não tivesse sido atualizada pelo dispositivo:

```
{
    "desired" : { "color" : "RED" }
}
```

Se uma atualização fizer com que as propriedades `desired` ou `reported` se tornem nulas, ela será removida do documento. Veja a seguir como remover a propriedade `desired` definindo-a como `null`. Você pode fazer isso quando um dispositivo atualiza seu estado, por exemplo.

```
{
    "state": {
        "reported": {
            "color": "red"
        },
```

```
        "desired": null
    }
}
```

Um documento de sombra também pode ter nenhuma propriedade `desired` ou `reported`, tornando o documento de sombra vazio. Este é um exemplo de um documento de sombra vazio, mas válido.

```
{
}
```

Valores de matriz em documentos de sombra

As shadows dão suporte a matrizes, mas as trata como valores normais em que uma atualização em uma matriz substitui toda a matriz. Não é possível atualizar parte de uma matriz.

Estado inicial:

```
{
    "desired" : { "colors" : ["RED", "GREEN", "BLUE" ] }
}
```

Atualização:

```
{
    "desired" : { "colors" : ["RED"] }
}
```

Estado final:

```
{
    "desired" : { "colors" : ["RED"] }
}
```

As matrizes não podem ter valores nulos. Por exemplo, a matriz a seguir não é válida e será rejeitada.

```
{
    "desired" : {
        "colors" : [ null, "RED", "GREEN" ]
```

```
    }
}
```

Mensagens de erro da Sombra do Dispositivo

O serviço Sombra do Dispositivo publica uma mensagem no tópico de erro (por MQTT) quando uma tentativa de alterar o documento de estado falha. Essa mensagem só é emitida como uma resposta a uma solicitação de publicação em um dos tópicos `$aws` reservados. Se o cliente atualizar o documento usando a API REST, ele receberá o código de erro HTTP como parte da resposta, e nenhuma mensagem de erro MQTT será emitida.

Código de erro HTTP	Mensagens de erro
400 (Solicitação inválida)	• JSON inválido • Nó obrigatório ausente: estado • O nó de estado deve ser um objeto • O nó desejado deve ser um objeto • O nó informado deve ser um objeto • Versão inválida • Token cliente inválido **Note** Um token de cliente com mais de 64 bytes gerará essa resposta. • O JSON contém muitos níveis de aninhamento; o máximo é 6 • O estado contém um nó inválido
401 (Não autorizado)	• Não autorizado
403 (Proibido)	• Proibido
404 (Não encontrado)	• Objeto não encontrado • Não existe sombra com o nome: *shadowName*

Código de erro HTTP	Mensagens de erro
409 (Conflito)	• Conflito de versão
413 (Carga útil muito grande)	• A carga útil excede o tamanho máximo permitido
415 (Tipo de mídia incompatível)	• Codificação documentada incompatível; a codificação compatível é UTF-8
429 (Muitas solicitações)	• O serviço Sombra do Dispositivo gerará essa mensagem de erro quando houver mais de 10 solicitações em processamento em uma única conexão. Uma solicitação em andamento é uma solicitação em andamento que foi iniciada, mas ainda não concluída.
500 (Erro interno do servidor)	• Falha no serviço interno

AWS IoT Device Management Catálogo de pacotes de software

Com o AWS IoT Device Management Software Package Catalog, você pode manter um inventário dos pacotes de software e suas versões. Você pode associar versões de pacotes a itens individuais e grupos AWS IoT dinâmicos de itens e implantá-los por meio de processos ou AWS IoT trabalhos internos.

O pacote de software contém uma ou mais versões de pacote, que é uma coleção de arquivos que podem ser implantados como uma única unidade. As versões do pacote podem conter firmware, atualizações do sistema operacional, aplicativos de dispositivos, configurações e patches de segurança. À medida que o software evolui com o tempo, você pode criar uma nova versão do pacote e implantá-la em sua frota.

O hub do pacote de AWS IoT software está localizado em AWS IoT Core. Você pode usar o hub para registrar e manter centralmente seu inventário e metadados de pacotes de software, o que cria um catálogo de pacotes de software e suas versões. Você pode optar por agrupar dispositivos com base em pacotes de software e versões de pacotes implantados no dispositivo. Esse atributo oferece a oportunidade de manter o inventário de pacotes do lado do dispositivo como uma sombra nomeada, associar e agrupar dispositivos com base nas versões e visualizar a distribuição da versão do pacote em toda a frota usando métricas da frota.

Se você tiver um sistema interno de implantação de software estabelecido, poderá continuar usando esse processo para implantar suas versões de pacotes. Se você não tiver um processo de implantação estabelecido ou se preferir, recomendamos o uso de trabalhos da AWS IoT para usar os atributos do Catálogo de pacotes de software. Para obter mais informações, consulte Preparando AWS IoT trabalhos.

Este capítulo contém as seguintes seções:

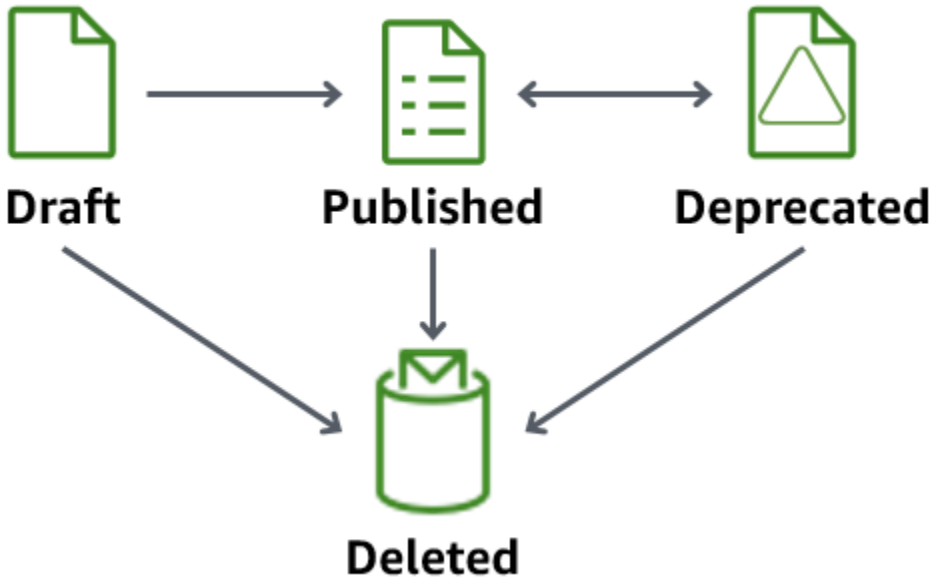
- Preparação para uso do Catálogo de pacotes de software
- Preparação de segurança
- Preparação da indexação de frota
- Preparando AWS IoT trabalhos
- Conceitos básicos do Catálogo de pacotes de software

Preparação para uso do Catálogo de pacotes de software

A seção a seguir fornece uma visão geral do ciclo de vida da versão do pacote e informações para usar o Software AWS IoT Device Management Package Catalog.

Ciclo de vida da versão do pacote

Uma versão de pacote pode evoluir nos seguintes estados do ciclo de vida:`draft`, `published` e `deprecated`. Ele também pode ser `deleted`.



- Rascunho

  Quando você cria uma versão do pacote, ela está em um estado `draft`. Esse estado indica que o pacote de software está sendo preparado ou está incompleto.

  Enquanto a versão do pacote estiver nesse estado, você não pode implantá-la. Você pode editar a descrição, os atributos e as tags da versão do pacote.

  Você pode fazer a transição de uma versão do pacote que está no `draft` estado para `published` ou está `deleted` usando o console ou emitindo as operações UpdatePackageVersionou a DeletePackageVersionAPI.

- Publicado

  Quando a versão do pacote estiver pronta para implantação, faça a transição da versão do pacote para um estado `published`. Nesse estado, você pode escolher identificar a versão do pacote

como a versão padrão editando o pacote de software no console ou por meio da operação da UpdatePackageAPI. Nesse estado, você pode editar apenas a descrição e as tags.

Você pode fazer a transição de uma versão do pacote que está no `published` estado para `deprecated` ou está `deleted` usando o console ou emitindo as operações UpdatePackageVersionou a DeletePackageVersionAPI.

- Preterido

  Se uma nova versão do pacote estiver disponível, você poderá fazer a transição de versões anteriores do pacote para `deprecated`. Você ainda pode implantar tarefas com uma versão de pacote preterida. Você também pode nomear uma versão obsoleta do pacote como a versão padrão e editar apenas a descrição e as tags.

  Considere fazer a transição de uma versão do pacote para `deprecated` quando a versão estiver desatualizada, mas você ainda tiver dispositivos em campo usando a versão mais antiga ou precisar mantê-la devido à dependência do tempo de execução.

  Você pode fazer a transição de uma versão do pacote que está no `deprecated` estado para `published` ou está `deleted` usando o console ou emitindo as operações UpdatePackageVersionou a DeletePackageVersionAPI.

- Excluído

  Quando você não pretende mais usar uma versão do pacote, pode excluí-la usando o console ou emitindo a operação da DeletePackageVersionAPI.

> **Note**
>
> Se você excluir uma versão do pacote enquanto houver tarefas pendentes que fazem referência a ela, você receberá uma mensagem de erro quando a tarefa for concluída com êxito e tentar atualizar a sombra nomeada reservada.
> Se a versão do pacote de software que você deseja excluir for nomeada como a versão padrão do pacote, você deverá primeiro atualizar o pacote para nomear outra versão como padrão ou deixar o campo sem nome. Você pode fazer isso usando o console ou a operação UpdatePackageVersionda API. (Para remover qualquer versão de pacote nomeada como padrão, defina o unsetDefaultVersionparâmetro como true ao emitir a operação da UpdatePackageAPI).

Se você excluir um pacote de software pelo console, ele excluirá todas as versões do pacote associadas a esse pacote, a menos que uma seja nomeada como a versão padrão.

Convenções de nomenclatura de versões de pacotes

Ao nomear as versões do pacote, é importante planejar e aplicar uma estratégia de nomenclatura lógica para que você e outras pessoas possam identificar facilmente a versão mais recente do pacote e a progressão da versão. Você deve fornecer um nome de versão ao criar a versão do pacote, mas a estratégia e o formato dependem muito do seu caso de negócios.

Como prática recomendada, recomendamos usar o formato de controle de versão SemVersemântico. Por exemplo, `1.2.3` onde `1` está a versão principal para alterações funcionalmente incompatíveis, `2` a versão principal para alterações funcionalmente compatíveis e `3` é a versão do patch (para correções de erros). Para obter mais informações, consulte Versionamento semântico 2.0.0. Para obter mais informações sobre os requisitos do nome da versão do pacote, consulte VersionName no guia de referência da AWS IoT API.

Versão padrão

Definir uma versão como padrão é opcional. É possível adicionar ou remover versões de pacote padrão. Você também pode implantar uma versão do pacote que não seja nomeada como a versão padrão.

Quando você cria uma versão do pacote, ela é colocada em um estado `draft` e não pode ser nomeada como a versão padrão até que você faça a transição da versão do pacote para “publicada”. O Catálogo de pacotes de software não seleciona automaticamente uma versão como padrão nem atualiza uma versão mais recente do pacote como padrão. Você deve nomear intencionalmente a versão do pacote escolhida por meio do console ou emitindo a operação da UpdatePackageVersionAPI.

Atributos de versão

Os atributos de versão e seus valores contêm informações importantes sobre as versões do seu pacote. Recomendamos que você defina atributos de uso geral para um pacote ou versão do pacote. Por exemplo, você pode criar um par nome-valor para plataforma, arquitetura, sistema operacional, data de lançamento, autor ou URL do Amazon S3.

Ao criar um AWS IoT trabalho com um documento de trabalho, você também pode optar por usar uma variável de substituição (`$parameter`) que se refere ao valor de um atributo. Para obter mais informações, consulte Preparando AWS IoT trabalhos.

Os atributos de versão usados nas versões do pacote não serão adicionados automaticamente à sombra nomeada reservada e não poderão ser indexados ou consultados diretamente por meio da indexação de frota. Para indexar ou consultar os atributos da versão do pacote por meio da indexação de frota, você pode preencher o atributo da versão na sombra nomeada reservada.

Recomendamos que o parâmetro de atributo de versão na sombra nomeada reservada capture as propriedades relatadas pelo dispositivo, como sistema operacional e horário de instalação. Elas também podem ser indexadas e consultadas por meio da indexação de frota.

Os atributos da versão não precisam seguir uma convenção de nomenclatura específica. Você pode criar pares de nome e valor para atender às necessidades da sua empresa. O tamanho combinado de todos os atributos em uma versão do pacote é limitado a 3 KB. Para obter mais informações, consulte os limites do pacote de software e das versões do pacote do Catálogo de Pacotes de Software.

Usar todos os atributos em um documento de trabalho

Você pode adicionar todos os atributos da versão do pacote automaticamente à implantação do seu trabalho para dispositivos selecionados. Para usar automaticamente todos os atributos de versão do pacote de maneira programática em um comando da API ou CLI, consulte o seguinte exemplo de documento de trabalho:

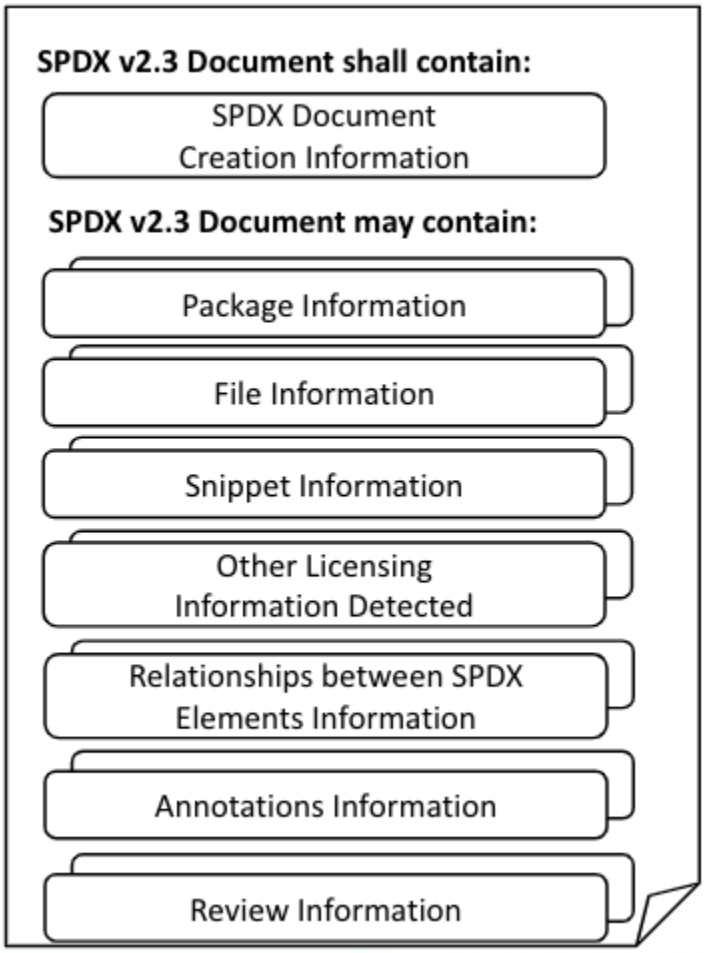
```
"TestPackage": "${aws:iot:package:TestPackage:version:PackageVersion:attributes}"
```

Lista de materiais de software

A lista de materiais de software (SBOM) fornece um repositório central para todos os aspectos do seu pacote de software. Além de armazenar pacotes e versões de pacotes de software, você pode armazenar a lista de materiais de software (SBOM) associada a cada versão do pacote no Catálogo de Pacotes de AWS IoT Device Management Software. O pacote de software contém uma ou mais versões de pacote e cada versão de pacote consiste em um ou mais componentes. Cada um desses componentes que aceitam a composição de uma versão específica do pacote pode ser descrito e catalogado usando uma lista de materiais de software. Os padrões do setor para a lista de materiais de software compatíveis são SPDX e CycloneDX. Quando uma SBOM é criada, ela passa por validação no formato padrão do setor SPDX e CycloneDX. Para obter mais informações

sobre o SPDX, consulte Troca de dados de pacote do sistema. Para obter mais informações sobre o CycloneDX, consulte CycloneDX.

A lista de materiais de software descreve todos os aspectos dos componentes de uma versão específica do pacote, como informações do pacote, informações do arquivo e outros metadados pertinentes. Veja o exemplo abaixo de uma estrutura de documento de lista de materiais de software no formato SPDX:



Benefícios da lista de materiais de software

Um dos principais benefícios de adicionar sua lista de materiais de software para uma versão de pacote no Catálogo de Pacotes de Software é o gerenciamento de vulnerabilidades.

Gerenciamento de vulnerabilidade

Avaliar e mitigar sua vulnerabilidade a riscos aparentes de segurança em componentes de software continua sendo fundamental para proteger a integridade da frota de dispositivos. Ao adicionar a lista de materiais de software armazenada no Catálogo de Pacotes de Software para cada versão

do pacote, você pode expor proativamente as lacunas na segurança sabendo quais dispositivos estão em risco com base na versão do pacote e na SBOM usando sua própria solução interna de gerenciamento de vulnerabilidades. Você pode implantar correções nos dispositivos afetados e proteger sua frota de dispositivos.

Armazenamento da lista de materiais de software

A lista de materiais de software (SBOM) de cada versão do pacote de software é armazenada em um bucket do Amazon S3 usando o atributo de versionamento do Amazon S3. O bucket do Amazon S3 que armazena a SBOM deve estar na mesma região em que a versão do pacote foi criada. O bucket do Amazon S3 usando o atributo de versionamento mantém diversas variantes de um objeto no mesmo bucket. Para obter mais informações sobre como usar versionamento em um bucket do Amazon S3, consulte Usar versionamento em buckets do Amazon S3.

> **Note**
>
> Cada versão do pacote de software pode ter vários arquivos SBOM anexados, mas os arquivos SBOM devem ser armazenados em um único arquivo zip.

A chave e o ID de versão específicos do Amazon S3 para seu bucket são usados para identificar de maneira exclusiva cada versão de uma lista de materiais de software para uma versão de pacote.

> **Note**
>
> Para uma versão de pacote com um único arquivo SBOM, você pode armazenar esse arquivo SBOM em seu bucket do Amazon S3 como um arquivo zip.
> Para uma versão de pacote com vários arquivos SBOM, você deve colocar todos os arquivos SBOM em um só arquivo zip e depois armazenar esse arquivo zip em seu bucket do Amazon S3.
> Todos os arquivos SBOM armazenados em um só arquivo zip em ambos os cenários são formatados como arquivos SPDX ou CycloneDX .json.

Política de permissões

Para AWS IoT atuar como principal especificado para acessar os arquivos zip do SBOM armazenados no bucket do Amazon S3, você precisa de uma política de permissões baseada em

recursos. Consulte o exemplo a seguir para ver a política correta de permissões com base em recursos:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "iot.amazonaws.com"
                ]
        },
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::bucketName/*"
        }
    ]
}
```

Para obter mais informações sobre políticas baseadas em recursos, consulte AWS IoT políticas baseadas em recursos

Atualizar o SBOM

Você pode atualizar a lista de materiais do software sempre que necessário para proteger e aprimorar sua frota de dispositivos. Sempre que a lista de materiais de software é atualizada em seu bucket do Amazon S3, o ID da versão muda, e você deve associar o novo URL do bucket do Amazon S3 à versão apropriada do pacote de software. Você verá o novo ID da versão na coluna ID da versão do objeto do Amazon S3 na página da versão do pacote no Console de gerenciamento da AWS. Além disso, você pode usar a operação da API `GetPackageVersion` ou o comando da CLI `get-package-version` para visualizar o novo ID da versão.

> **Note**
>
> Atualizar sua lista de materiais de software, o que causará um novo ID de versão, não levará à criação de uma nova versão do pacote.

Para obter mais informações sobre chaves de objeto do Amazon S3, consulte Criar nomes de chave de objeto.

Habilitando a indexação de AWS IoT frotas

Habilitar a indexação de AWS IoT frotas é um requisito para usar o AWS IoT Device Management Software Package Catalog. Para aproveitar a indexação de AWS IoT frotas com o Catálogo de pacotes de software, defina a sombra nomeada reservada (`$package`) como a fonte de dados para cada dispositivo no qual você deseja indexar e coletar métricas. Para obter mais informações sobre sombras nomeadas reservadas, consulte Sombra nomeada reservada.

A indexação de frotas fornece suporte que permite que AWS IoT as coisas sejam agrupadas por meio de grupos dinâmicos que são filtrados por versão do pacote de software. Por exemplo, a indexação de frota pode identificar objetos que têm ou não uma versão de pacote específica instalada, não têm nenhuma versão de pacote instalada ou correspondem a pares de nome/valor específicos. Por fim, a indexação da frota fornece métricas padrão e personalizadas que você pode usar para obter informações sobre o estado da sua frota de dispositivos. Para obter mais informações, consulte Preparação da indexação de frota.

> **Note**
>
> Habilitar a indexação de frota para o Catálogo de pacotes de software gera custos de serviço padrão. Para obter mais informações, consulte a opção AWS IoT Device Management, Preços.

Sombra nomeada reservada

A sombra nomeada reservada, `$package`, reflete o estado dos pacotes de software e das versões dos pacotes instalados no dispositivo. A indexação de frota usa a sombra nomeada reservada como fonte de dados para criar métricas padrão e personalizadas para que você possa consultar o estado da sua frota. Para obter mais informações, consulte Preparação de indexação de frota.

Uma sombra nomeada reservada é semelhante a uma sombra nomeada, com a exceção de que seu nome é predefinido e você não pode alterá-lo. Além disso, a sombra nomeada reservada não é atualizada com metadados e usa somente as palavras-chave `version` e `attributes`.

Solicitações de atualização que incluam outras palavras-chave, como `description`, receberão uma resposta de erro no tópico `rejected`. Para obter mais informações, consulte as mensagens de erro da opção Sombra do dispositivo.

Ele pode ser criado quando você cria AWS IoT algo por meio do console, quando um AWS IoT trabalho conclui e atualiza a sombra com êxito e se você executa a operação da `UpdateThingShadow`API. Para obter mais informações, consulte UpdateThingShadowo guia do AWS IoT Core desenvolvedor.

> **Note**
>
> A indexação da sombra nomeada reservada não conta para o número de sombras nomeadas que a indexação da frota pode indexar. Para obter mais informações, consulte AWS IoT Device Management Cotas e limites de indexação de frota. Além disso, se você optar por fazer com que os AWS IoT trabalhos atualizem a sombra nomeada reservada quando um trabalho for concluído com êxito, a chamada de API será contabilizada nas operações de Device Shadow e de registro e poderá ter um custo. Para obter mais informações, consulte limites e cotas de AWS IoT Device Management trabalhos e o tipo de dados IndexingFilterda API.

Estrutura da sombra **`$package`**

A sombra nomeada reservada contém o seguinte:

```
{
    "state": {
        "reported": {
            "<packageName>": {
                "version": "",
                "attributes": {
                }
            }
        }
    },
    "version" : 1
    "timestamp" : 1672531201
}
```

As propriedades da sombra são atualizadas com as seguintes informações:

- `<packageName>`: o nome do pacote de software instalado, que é atualizado com o parâmetro packageName.
- `version`: o nome da versão do pacote instalado, que é atualizado com o parâmetro versionName.
- `attributes`: metadados opcionais armazenados pelo dispositivo e indexados pela indexação da frota. Isso permite que os clientes consultem seus índices com base nos dados armazenados.
- `version`: o número da versão da sombra. É incrementado automaticamente toda vez que a sombra é atualizada e começa em 1.
- `timestamp`: indica quando a sombra foi atualizada pela última vez e foi gravada no horário Unix.

Para obter mais informações sobre o formato e o comportamento de uma sombra nomeada, consulte a opção Serviço AWS IoT Device Shadow Ordem das mensagens.

Exclusão de um pacote de software e suas versões

Antes de excluir um pacote de software, você deve fazer o seguinte:

- Confirme se o pacote e suas versões não estão sendo implantados ativamente.
- Exclua todas as versões associadas primeiro. Se uma das versões for designada como a versão padrão, você deverá remover a versão padrão nomeada do pacote. Como a designação de uma versão padrão é opcional, não há conflito em removê-la. Para remover a versão padrão do pacote de software, edite o pacote por meio do console ou use a operação da UpdatePackageVersionAPI.

Desde que não haja uma versão de pacote padrão nomeada, você pode usar o console para excluir um pacote de software e todas as versões do pacote também serão excluídas. Se você usar uma chamada de API para excluir pacotes de software, deverá excluir primeiro as versões do pacote e depois o pacote de software.

Preparação de segurança

Esta seção discute os principais requisitos de segurança do AWS IoT Device Management Software Package Catalog.

Autenticação baseada em recurso

O Catálogo de pacotes de software usa autorização baseada em recursos para fornecer segurança adicional ao atualizar o software em sua frota. Isso significa que você deve criar uma política AWS Identity and Access Management (IAM) que conceda direitos de execução`create`,`read`, `update`,`delete`, e `list` ações para pacotes e versões de pacotes de software e referenciar os pacotes de software e as versões de pacotes específicos que você deseja implantar na `Resources` seção. Você também precisa desses direitos para poder atualizar a sombra nomeada reservada. Você faz referência aos pacotes de software e às versões dos pacotes incluindo o nome do recurso da Amazon (ARN) para cada entidade.

> **Note**
>
> Se você pretende que a política conceda direitos para chamadas de API da versão do pacote (como CreatePackageVersionUpdatePackageVersion,, DeletePackageVersion), é necessário incluir o pacote de software e a versão do pacote ARNs na política. Se você pretende que a política conceda direitos para chamadas de API de pacotes de software (como CreatePackageUpdatePackage, e DeletePackage), você deve incluir somente o ARN do pacote de software na política.

Estruture o pacote de software e a versão do pacote da ARNs seguinte forma:

- Pacote de software: `arn:aws:iot:<region>:<accountID>:package/<packageName>/package`
- Versão do pacote: `arn:aws:iot:<region>:<accountID>:package/<packageName>/version/<versionName>`

> **Note**
>
> Há outros direitos relacionados que você pode incluir nesta política. Por exemplo, você pode incluir um ARN para o `job`, `thinggroup`, e `jobtemplate`. Para obter mais informações e uma lista completa das opções de política, consulte Protegendo usuários e dispositivos com AWS IoT trabalhos.

Por exemplo, se você tiver um pacote de software e uma versão do pacote com o seguinte nome:

- AWS IoT coisa: `myThing`
- Nome do pacote: `samplePackage`
- Versão `1.0.0`

A política pode ser igual a este exemplo:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:createPackage",
                "iot:createPackageVersion",
                "iot:updatePackage",
                "iot:updatePackageVersion"
            ],
            "Resource": [
               "arn:aws:iot:us-east-1:111122223333:package/samplePackage",
               "arn:aws:iot:us-east-1:111122223333:package/samplePackage/
version/1.0.0"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:GetThingShadow",
                "iot:UpdateThingShadow"
            ],
            "Resource": "arn:aws:iot:us-east-1:111122223333:thing/myThing/$package"
        }
    ]
}
```

AWS IoT Job rights para implantar versões de pacotes

Para fins de segurança, é importante conceder direitos para implantar pacotes e versões de pacotes e nomear os pacotes e versões de pacotes específicos que eles podem implantar. Para fazer isso,

você cria uma política e um perfil do IAM que concedem permissão para implantar tarefas com versões de pacotes. A política deve especificar as versões do pacote de destino como um recurso.

Política do IAM

A política do IAM concede o direito de criar uma tarefa que inclua o pacote e a versão nomeados na seção `Resource`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:CreateJob",
                "iot:CreateJobTemplate"
            ],
            "Resource":[
                "arn:aws:iot:*:111122223333:job/<jobId>",
                "arn:aws:iot:*:111122223333:thing/<thingName>/$package",
                "arn:aws:iot:*:111122223333:thinggroup/<thingGroupName>",
                "arn:aws:iot:*:111122223333:jobtemplate/<jobTemplateName>",
                "arn:aws:iot:*:111122223333:package/<packageName>/
version/<versionName>"
            ]
        }
    ]
}
```

> **Note**
>
> Se você quiser implantar uma tarefa que desinstale um pacote de software e uma versão do pacote, você deve autorizar um ARN em que a versão do pacote seja `$null`, como mostrado a seguir:

```
arn:aws:iot:<regionCode>:111122223333:package/<packageName>/version/$null
```

AWS IoT Job rights para atualizar a sombra nomeada reservada

Para permitir que as tarefas atualizem a sombra de nome reservada do objeto quando a tarefa for concluída com êxito, você deve criar um perfil e uma política do IAM. Você pode fazer isso de duas maneiras no console AWS IoT . A primeira é quando você cria um pacote de software no console. Se você vir a caixa de diálogo Habilitar dependências para gerenciamento de pacotes, poderá optar por usar uma função existente ou criar uma nova função. Ou, no console AWS IoT , escolha Configurações, em seguida Gerenciar indexação e, então, Gerenciar indexação para pacotes e versões de dispositivos.

> **Note**
>
> Se você optar por fazer com que o serviço AWS IoT Job atualize a sombra nomeada reservada quando um trabalho for concluído com êxito, a chamada da API será contabilizada nas operações de Device Shadow e de registro e poderá ter um custo. Para obter mais informações, consulte Preços do AWS IoT Core.

Quando você usa a opção Criar função, o nome da função gerada começa com `aws-iot-role-update-shadows` e contém as seguintes políticas:

Configuração de um perfil

Permissões

A política de permissões concede os direitos de consultar e atualizar a sombra do objeto. O parâmetro `$package` no ARN do recurso tem como alvo a sombra nomeada reservada.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:DescribeEndpoint",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "iot:GetThingShadow",
                "iot:UpdateThingShadow"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:111122223333:thing/<thingName>/$package"
            ]
        }
    ]
}
```

Relação de confiança

Além da política de permissões, a função exige uma relação de confiança com AWS IoT Core para que a entidade possa assumir a função e atualizar a sombra nomeada reservada.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
        },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Configuração de uma política de usuário

objetivo: PassRole permissão

Por fim, você deve ter a permissão para passar a função AWS IoT Core ao chamar a operação da UpdatePackageConfigurationAPI.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "iam:PassRole",
                "iot:UpdatePackageConfiguration"
            ],
            "Resource": "arn:aws:iam::111122223333:role/<roleName>"
        }
    ]
}
```

AWS IoT Permissões de trabalho para baixar do Amazon S3

O documento de trabalho é salvo no Amazon S3. Você se refere a esse arquivo ao enviar por meio de AWS IoT Jobs. Você deve fornecer a AWS IoT Jobs os direitos de baixar o arquivo (`s3:GetObject`). Você também deve estabelecer uma relação de confiança entre o Amazon S3 e AWS IoT Jobs. Para obter instruções sobre como criar essas políticas, consulte Pré-assinado URLs em Gerenciando trabalhos.

Permissões para atualizar a lista de materiais de software para uma versão do pacote

Para atualizar a lista de materiais de software para uma versão de pacote nos estados de `Draft`, `Published`, ou `Deprecated` ciclo de vida, você precisa de uma AWS Identity and Access Management função e políticas para localizar a nova lista de materiais de software no Amazon S3 e atualizar a versão do pacote em. AWS IoT Core

Primeiro, você colocará a lista de materiais de software atualizada em seu bucket com versionamento do Amazon S3 e chamará a operação `UpdatePackageVersion` da API com o parâmetro `sboms` incluído. Em seguida, seu diretor autorizado assumirá a função do IAM que você criou, localizará a lista de materiais de software atualizada no Amazon S3 e atualizará a versão do pacote no AWS IoT Core Software Package Catalog.

As seguintes políticas são necessárias para realizar essa atualização:

Políticas

- Política de confiança: política que estabelece uma relação de confiança com a entidade principal autorizada, assumindo o perfil do IAM, para que ela possa localizar a lista de materiais de software atualizada do seu bucket com versionamento no Amazon S3 e atualizar a versão do pacote em AWS IoT Core.

- ```
  {
      "Version":"2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Principal": {
                  "Service": "s3.amazonaws.com"
          },
              "Action": "sts:AssumeRole"
          }
      ]
  }
  ```

- ```
  {
      "Version":"2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Principal": {
                  "Service": "iot.amazonaws.com"
          },
              "Action": "sts:AssumeRole"
          }
      ]
  }
  ```

- Política de permissões: política para acessar o bucket com versionamento do Amazon S3 em que a lista de materiais do software é armazenada para uma versão do pacote e atualizá-la no AWS IoT Core.

  - ```
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "s3:GetObject"
                ],
                "Resource": [
                    "arn:aws:s3:::awsexamplebucket1"
                ]
    ```
    ```

```
        }
    ]
}
```

- ```
  {
      "Version":"2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Action": [
                  "iot:UpdatePackageVersion"
              ],
              "Resource": [
                  "arn:aws:iot:*:111122223333:package/<packageName>/
  version/<versionName>"
              ]
          }
      ]
  }
  ```

- Transmitir permissões de função: política que concede permissão para passar a função do IAM para o Amazon S3 AWS IoT Core e quando você chama [UpdatePackageVersion](#) a operação da API.

  - ```
    {
      "Version":"2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "iam:PassRole",
            "s3:GetObject"
          ],
          "Resource": [
            "arn:aws:s3:::awsexamplebucket1"
          ]
        }
      ]
    }
    ```

  - ```
    {
    ```
    ```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "iot:UpdatePackageVersion"
            ],
            "Resource": "arn:aws:iam::111122223333:role/<roleName>"
        }
    ]
}
```

> **Note**
>
> Você não pode atualizar a lista de materiais do software em uma versão do pacote que tenha passado para o estado de ciclo de vida `Deleted`.

Para obter mais informações sobre como criar uma função do IAM para um AWS serviço, consulte Criação de uma função para delegar permissão a um AWS serviço.

Para obter mais informações sobre a criação de um bucket do Amazon S3 e o upload de objetos nele, consulte Criar um bucket e Fazer upload de objetos.

Preparação da indexação de frota

Com a indexação de AWS IoT frotas, você pode pesquisar e agregar dados usando o nome reservado shadow ()`$package`. Você também pode agrupar AWS IoT coisas consultando os grupos Sombra nomeada reservada de coisas dinâmicas. Por exemplo, você pode encontrar informações sobre quais AWS IoT itens usam uma versão de pacote específica, não têm uma versão de pacote específica instalada ou não têm nenhuma versão de pacote instalada. Você pode obter mais informações combinando atributos. Por exemplo, identificar objetos que têm uma versão específica e são de um tipo específico (como a versão 1.0.0 e o tipo de objeto pump_sensor). Para obter mais informações, consulte Indexação de frota.

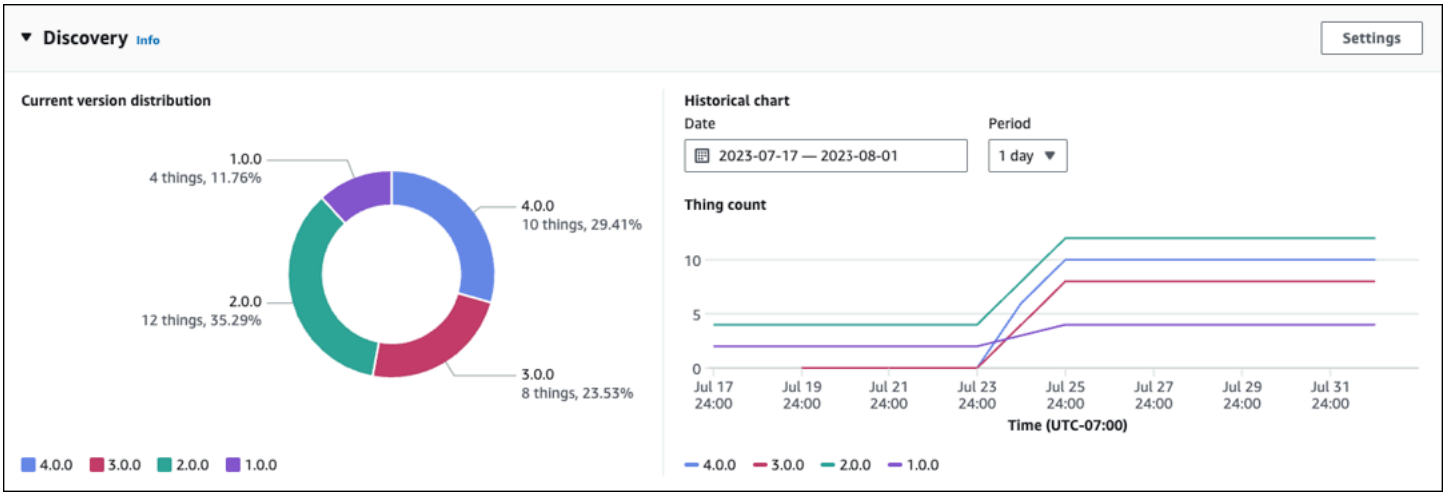
Definição da **$package** sombra como fonte de dados

Para usar a indexação de frota com o Catálogo de pacotes de software, você deve habilitar a indexação de frota, definir a sombra nomeada como a fonte de dados e definir `$package` como o filtro de sombra nomeado. Se você não ativou a indexação de frota, poderá ativá-la nesse processo. Em AWS IoT Core no console, abra Configurações, selecione Gerenciar indexação, em seguida Adicionar sombras nomeadas, Adicionar pacotes de software do dispositivo e versões, e Atualizar. Para obter mais informações, consulte Como gerenciar indexação de objetos.

Como alternativa, você pode ativar a indexação da frota ao criar seu primeiro pacote. Quando a caixa de diálogo Habilitar dependências para gerenciamento de pacotes for exibida, escolha a opção de adicionar pacotes e versões de software do dispositivo como fontes de dados à indexação da frota. Ao selecionar essa opção, você também ativa a indexação da frota.

> **Note**
>
> Habilitar a indexação de frota para o Catálogo de pacotes de software gera custos de serviço padrão. Para obter mais informações, consulte a opção AWS IoT Device Management, Preços.

Métricas serão exibidos no console



Na página de detalhes do pacote de software do AWS IoT console, o painel Discovery exibe métricas padrão ingeridas pela `$package` sombra.

- O gráfico de distribuição da versão atual mostra o número de dispositivos e a porcentagem das 10 versões mais recentes do pacote que estão associadas a AWS IoT alguma coisa de todos os dispositivos associados a esse pacote de software. Nota: Se o pacote de software tiver mais versões do pacote do que as identificadas no gráfico, você poderá encontrá-las agrupadas em Outros.
- O gráfico Histórico mostra o número de dispositivos associados às versões selecionadas do pacote em um período de tempo especificado. O gráfico fica inicialmente vazio até que você selecione até cinco versões do pacote e defina o intervalo de datas e o intervalo de tempo. Para selecionar os parâmetros do gráfico, escolha Configurações. Os dados exibidos no gráfico Histórico podem ser diferentes do gráfico de distribuição da versão atual devido à diferença no número de versões do pacote que eles exibem e também porque você pode escolher quais versões do pacote analisar no gráfico Histórico. Nota: quando você seleciona uma versão do pacote para visualizar, ela conta para o número máximo de limites de métricas da frota. Para obter mais informações, consulte Cotas e limites de indexação de frota.

Para outro método para obter informações sobre como coletar a distribuição da versão do pacote, consulte Como coletar a distribuição da versão do pacote por meio de `getBucketsAggregation`.

Padrões de consulta

A indexação de frota com o Catálogo de pacotes de software usa a maioria dos atributos aceitos (por exemplo, termos, frases e campos de pesquisa). No entanto, operadores de comparação (por exemplo, menor que < e maior que >) e consultas `range` não estão disponíveis para a chave `version` de sombra reservada nomeada (`$package`). Essas consultas estão disponíveis para a chave `attributes`. Para obter mais informações, consulte Sintaxe de consulta.

Exemplo de dados

Nota: para obter informações sobre a sombra nomeada reservada e sua estrutura, consulte Sombra nomeada reservada.

Neste exemplo, um primeiro dispositivo é nomeado `AnyThing` e tem os seguintes pacotes instalados:

- Pacote de software: `SamplePackage`

  Versão do pacote: `1.0.0`

  ID de pacote: `1111`

A sombra se parece com esta a seguir:

```
{
    "state": {
        "reported": {
            "SamplePackage": {
                "version": "1.0.0",
                "attributes": {
                    "s3UrlForSamplePackage": "https://EXAMPIEBUCKET.s3.us-
west-2.amazonaws.com/exampleCodeFile1",
                    "packageID": "1111"
                    }
            }
        }
    }
}
```

Um segundo dispositivo tem um nome `AnotherThing` e tem o seguinte pacote instalado:

- Pacote de software: `SamplePackage`

  Versão do pacote: `1.0.0`

  ID de pacote: `1111`
- Pacote de software: `OtherPackage`

  Versão do pacote: `1.2.5`

  ID de pacote: `2222`

A sombra se parece com esta a seguir:

```
{
    "state": {
        "reported": {
            "SamplePackage": {
                "version": "1.0.0",
                "attributes": {
                    "s3UrlForSamplePackage": "https://EXAMPIEBUCKET.s3.us-
west-2.amazonaws.com/exampleCodeFile1",
                    "packageID": "1111"
                }
```

```
            },
            "OtherPackage": {
                "version": "1.2.5",
                "attributes": {
                    "s3UrlForOtherPackage": "https://EXAMPIEBUCKET.s3.us-
west-2.amazonaws.com/exampleCodeFile2",
                    "packageID": "2222"
                    }
            },
        }
    }
}
```

Consultas de exemplo

A tabela a seguir lista exemplos de consultas com base nos exemplos de sombras de dispositivos para `AnyThing` e `AnotherThing`. Para obter mais informações, consulte a opção Consultas de exemplo do objeto.

Versão mais recente do AWS IoT Device Tester for FreeRTOS

Informações solicitadas	Consulta	Resultado
objetos que têm uma versão de pacote específica instalada	`shadow.name.$package.reported.SamplePackage.version:1.0.0`	`AnyThing`, `OtherThing`
objetos que não têm uma versão específica do pacote instalada	`NOT shadow.name.$package.reported.OtherPackage.version:1.2.5`	`AnyThing`
Qualquer dispositivo usando uma versão de pacote cujo ID do pacote seja maior que 1.500	`shadow.name.$package.reported.*.attributes.packageID>1500"`	`OtherThing`
Objetos que têm um pacote específico instalado e com mais de um pacote instalado	`shadow.name.$package.reported.SamplePackage.version:1.`	`OtherThing`

Informações solicitadas	Consulta	Resultado
	`0.0 AND shadow.name.$package.reported.totalCount:2`	

Coleta da distribuição da versão do pacote por meio de `getBucketsAggregation`

Além do painel Discovery no AWS IoT console, você também pode obter informações sobre a distribuição da versão do pacote usando a operação da `GetBucketsAggregation`API. Para obter as informações de distribuição da versão do pacote, você precisa fazer o seguinte:

- Defina um campo personalizado na indexação da frota para cada pacote de software. Nota: Criação dos campos personalizados conta para AWS IoT service quotas para indexação de frota.
- Formate o campo personalizado da seguinte forma:

  `shadow.name.$package.reported.<packageName>.version`

Para obter mais informações, consulte a seção Campos personalizados na indexação de AWS IoT frotas.

Preparando AWS IoT trabalhos

AWS IoT Device Management O Software Package Catalog estende os AWS IoT trabalhos por meio de parâmetros de substituição e integração com indexação de AWS IoT frotas, grupos dinâmicos de coisas e o item AWS IoT reservado chamado shadow.

> **Note**
>
> Para usar toda a funcionalidade que o Software Package Catalog oferece, você deve criar estas funções e políticas AWS Identity and Access Management (IAM): direitos de AWS IoT trabalhos para implantar versões de pacotes e direitos de AWS IoT trabalhos para atualizar a sombra nomeada reservada. Para obter mais informações, consulte Preparação de segurança.

Parâmetros de substituição para trabalhos AWS IoT

Você pode usar parâmetros de substituição como um espaço reservado em seu documento de AWS IoT trabalho. Quando o serviço de tarefa encontra um parâmetro de substituição, ele aponta a tarefa para um atributo de versão de software nomeado para o valor do parâmetro. Você pode usar esse processo para criar um único documento de trabalho e passar os metadados para a tarefa por meio de atributos de uso geral. Por exemplo, você pode passar por um URL do Amazon Simple Storage Service (Amazon S3), um pacote de software do nome do recurso da Amazon (ARN) ou uma assinatura no documento de trabalho por meio dos atributos da versão do pacote.

Os parâmetros de substituição devem ser formatados no documento de trabalho da seguinte forma:

- Nome do pacote de software e versão do pacote
  - A string vazia entre `package::version` representa o parâmetro de substituição do nome do pacote de software. A string vazia entre `version::attribute` representa o parâmetro de substituição da versão do pacote de software. Consulte o exemplo a seguir para usar o nome do pacote e os parâmetros de substituição da versão do pacote em um documento de trabalho: `${aws:iot:package::version::attributes:<attributekey>}`.
  - O documento do trabalho preencherá automaticamente esses parâmetros de substituição usando o ARN da versão com base nos detalhes da versão do pacote. Se você estiver criando um trabalho ou um modelo de trabalho para uma implantação de pacote único usando um comando de API ou CLI, o ARN da versão para uma versão do pacote será representado pelo parâmetro `destinationPackageVersions` em `CreateJob` e `DescribeJob`.
- Todos os atributos de uma versão do pacote de software
  - Consulte o exemplo a seguir para usar todos os atributos de um parâmetro de substituição de versão de pacote de software em um documento de trabalho: `${aws:iot:package:<packageName>:version:<versionName>:attributes}`

> **Note**
>
> O nome do pacote, a versão do pacote e todos os parâmetros de substituição de atributos podem ser usados juntos. Consulte o exemplo a seguir para usar todos os três parâmetros de substituição em um documento de trabalho: `${aws:iot:package::version::attributes}`

No exemplo a seguir, há um pacote de software chamado `samplePackage`, e ele tem uma versão de pacote chamada `2.1.5` que tem os seguintes atributos:

- nome: `s3URL`, valor: `https://EXAMPIEBUCKET.s3.us-west-2.amazonaws.com/exampleCodeFile`
  - Esse atributo identifica a localização do arquivo de código armazenado no Amazon S3.
- nome: `signature`, valor: `aaaaabbbbbccccddddeeeeeffffffgggggghhhhhiiiiijjjj`
  - Esse atributo fornece um valor de assinatura de código que o dispositivo exige como medida de segurança. Para obter mais informações, consulte Assinatura de código para tarefas. Nota: Esse atributo é um exemplo e não é obrigatório como parte do Catálogo de pacotes de software ou das tarefas.

Para `s3URL`, o parâmetro do documento de trabalho é escrito da seguinte forma:

```
{
"samplePackage": "${aws:iot:package:samplePackage1:version:2.1.5:attributes:s3URL}"
}
```

Para `signature`, o parâmetro do documento de trabalho é escrito da seguinte forma:

```
{
"samplePackage": "${aws:iot:package:samplePackage1:version:2.1.5:attributes:signature}"
}
```

O documento de trabalho completo está escrito da seguinte forma:

```
{
  ...
  "Steps": {
    "uninstall": ["samplePackage"],
    "download": [
      {
        "samplePackage":
 "${aws:iot:package:samplePackage1:version:2.1.5:attributes:s3URL}"
      },
    ],
    "signature": [
      "samplePackage" :
 "${aws:iot:package:samplePackage1:version:2.1.5:attributes:signature}"
```

```
    ]
  }
}
```

Depois que a substituição é feita, o seguinte documento de trabalho é implantado nos dispositivos:

```
{
  ...
  "Steps": {
    "uninstall": ["samplePackage"],
    "download": [
      {
        "samplePackage": "https://EXAMPIEBUCKET.s3.us-west-2.amazonaws.com/
exampleCodeFile"
      },
    ],
    "signature": [
      "samplePackage" : "aaaaabbbbbcccccdddddeeeeeffffffgggggghhhhhiiiiijjjj"
    ]
  }
}
```

Parâmetros de substituição (visualização antes e depois)

Os parâmetros de substituição simplificam a criação de um documento de trabalho usando vários sinalizadores, como `$default` para a versão padrão do pacote. Isso elimina a necessidade de inserir manualmente metadados específicos da versão do pacote para cada implantação de trabalho, pois esses sinalizadores são preenchidos de modo automático com os metadados referenciados na versão específica do pacote. Para obter mais informações sobre os atributos da versão do pacote, como `$default` para a versão padrão do pacote, consulte Preparação do documento de trabalho e versão do pacote para implantação.

No Console de gerenciamento da AWS, ative o botão Visualizar substituição na janela do editor do arquivo de instruções de implantação durante a implantação de um trabalho para uma versão do pacote para visualizar o documento do trabalho com e sem os parâmetros de substituição.

Usando o parâmetro “antes da substituição” em `DescribeJob` e `GetJobDocument` APIs, você pode visualizar a resposta da API antes e depois da remoção dos parâmetros de substituição. Consulte os exemplos a seguir com o `DescribeJob` e `GetJobDocument` APIs:

- `DescribeJob`

- Visualização padrão

```
{
    "jobId": "<jobId>",
    "description": "<description>",
    "destinationPackageVersions": ["arn:aws:iot:us-west-2:123456789012:package/
TestPackage/version/1.0.2"]
}
```

- Antes da visualização de substituição

```
{
    "jobId": "<jobId>",
    "description": "<description>",
    "destinationPackageVersions": ["arn:aws:iot:us-west-2:123456789012:package/
TestPackage/version/$default"]
}
```

- `GetJobDocument`
  - Visualização padrão

```
{
    "attributes": {
        "location": "prod-artifacts.s3.us-east-1.amazonaws.com/mqtt-core",
        "signature": "IQoJb3JpZ2luX2VjEIrwEaCXVzLWVhc3QtMSJHMEUCIAofPNPpZ9cI",
        "streamName": "mqtt-core",
        "fileId": "0"
    },
}
```

  - Antes da visualização de substituição

```
{
    "attributes": "${aws:iot:package:TestPackage:version:$default:attributes}",
}
```

Para obter mais informações sobre AWS IoT trabalhos, criação de documentos de trabalho e implantação de trabalhos, consulte Trabalhos.

Preparação do documento de trabalho e versão do pacote para implantação

Quando uma versão do pacote é criada, ela está em um estado `draft` para indicar que está sendo preparada para implantação. Para preparar a versão do pacote para implantação, crie um documento de trabalho, salve-o em um local em que a tarefa possa ser acessada (como o Amazon S3) e confirme se a versão do pacote tem os valores de atributos que você deseja que o documento de trabalho use. (Nota: você pode atualizar os atributos de uma versão do pacote apenas enquanto ela estiver no estado `draft`.)

Ao criar um AWS IoT Job ou um modelo de Job para uma implantação de pacote único, você tem as seguintes opções para personalizar seu documento de trabalho:

Arquivo de instruções de implantação (**`recipe`**)

- O arquivo de instruções de implantação de uma versão do pacote contém as instruções de implantação, incluindo um documento de trabalho embutido, para implantar uma versão do pacote em vários dispositivos. O arquivo associa instruções de implantação específicas a uma versão do pacote para uma implantação rápida e eficiente do trabalho.

  No Console de gerenciamento da AWS, você pode criar o arquivo na janela de visualização do arquivo de instruções de implantação na guia Configurações de implantação da versão do fluxo de trabalho de criação de novo pacote. Você pode aproveitar AWS IoT para gerar automaticamente um arquivo de instruções a partir dos atributos da versão do pacote usando Iniciar a partir do arquivo AWS IoT recomendado ou usar seu documento de trabalho existente armazenado em um bucket do Amazon S3 usando Use seu próprio arquivo de instruções de implantação.

  > **Note**
  >
  > Se você usar seu próprio documento de trabalho, poderá atualizá-lo diretamente na janela de Visualização do arquivo de instruções de implantação, mas ele não atualizará automaticamente seu documento de trabalho original armazenado no bucket do Amazon S3.

  Ao usar o AWS CLI ou, um comando de API`CreatePackageVersion`, como, ou `GetPackageVersionUpdatePackageVersion`, `recipe` representa o arquivo de instruções de implantação, que inclui um documento de trabalho embutido.

Para obter mais informações sobre o que é um documento de trabalho, consulte Conceitos básicos.

Consulte o exemplo a seguir para o arquivo de instruções de implantação, conforme representado por`recipe`:

```
{
    "packageName": "sample-package-name",
    "versionName": "sample-package-version",
    ...
    "recipe": "{...}"
}
```

> **Note**
>
> O arquivo de instrução de implantação, conforme representado por `recipe`, pode ser atualizado quando uma versão do pacote está no estado de status de `published`, pois está separada dos metadados da versão do pacote. Ele se torna imutável durante a implantação do trabalho.

Atributos de versão do **`Artifact`**

- Usando o atributo de versão `artifact` na versão do pacote de software, você pode adicionar a localização do Amazon S3 aos artefatos da versão do pacote. Quando uma implantação de trabalho para a versão do pacote é acionada usando AWS IoT Jobs, o espaço reservado de URL pré-assinado `${aws:iot:package:<packageName>:version:<versionName>:artifact-location:s3-presigned-url}` no documento de trabalho será atualizado usando o bucket do Amazon S3, a chave do bucket e a versão do arquivo armazenado no bucket do Amazon S3. O bucket do Amazon S3 que armazena os artefatos de versão do pacote deve estar na mesma região em que a versão do pacote foi criada.

> **Note**
>
> Para armazenar várias versões de objetos do mesmo arquivo em seu bucket do Amazon S3, habilite o versionamento em seu bucket. Para obter mais informações, consulte Habilitar o versionamento em buckets.

Para acessar os artefatos da versão do pacote no bucket do Amazon S3 ao usar a operação de API `CreatePackageVersion` ou `UpdatePackageVersion`, você deve ter as seguintes permissões:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObjectVersion",
            "Resource": "arn:aws:s3:::bucket-name/key-name"
        }
    ]
}
```

Para obter mais informações sobre o atributo de versão `artifact` nas operações `UpdatePackageVersion` da API `CreatePackageVersion` CreatePackageVersione, consulte UpdatePackageVersione.

Consulte o exemplo a seguir que mostra o atributo de versão `artifact` compatível com a localização do artefato no Amazon S3 ao criar uma nova versão do pacote:

```
{
    "packageName": "sample package name",
    "versionName": "1.0",
    "artifact": {
        "s3Location": {
            "bucket": "firmware",
            "key": "image.bin",
            "version": "12345"
        }
    }
```

```
}
```

> **Note**
>
> Quando uma versão do pacote é atualizada de um estado de status `draft` para um estado de status `published`, os atributos da versão do pacote e a localização dos artifícios se tornam imutáveis. Para atualizar essas informações, você precisa criar uma nova versão do pacote e realizar essas atualizações enquanto estiver no estado de status `draft`.

Versão do pacote

- Uma versão padrão do pacote de software pode ser indicada nas versões disponíveis do pacote de software, fornecendo uma versão de pacote segura e estável. Isso serve como a versão básica do pacote de software ao implantar a versão padrão do pacote em sua frota de dispositivos usando AWS IoT Jobs. Ao criar um trabalho para implantar a versão do pacote `$default` para um pacote de software, a versão do pacote no documento do trabalho e na nova implantação do trabalho deve corresponder a `$default`. A versão do pacote na implantação do trabalho é representada por `destinationPackageVersions` para comandos de API e CLI e `VersionARN` no Console de gerenciamento da AWS. A versão do pacote no documento de trabalho é representada pelo seguinte espaço reservado para documento de trabalho mostrado abaixo:

  ```
  arn:aws:iot:<regionCode>:111122223333:package/<packageName>/version/$default
  ```

  Para criar um trabalho ou modelo de trabalho usando a versão padrão do pacote, use a sinalização `$default` no comando de API `CreateJob` ou `CreateJobTemplate`, conforme mostrado abaixo:

  ```
  "$ aws iot create-job \
      --destination-package-versions "arn:aws:iot:us-west-2:123456789012:package/
  TestPackage/version/$default"
      --document file://jobdoc.json
  ```

> **Note**
>
> O atributo `$default` da versão do pacote que faz referência à versão padrão é um atributo opcional que só é necessário ao fazer referência à versão do pacote padrão para uma implantação de trabalho por meio de AWS IoT Jobs.

Quando estiver satisfeito com a versão do pacote, publique-a por meio da página de detalhes do pacote de software no AWS IoT console ou emitindo a operação da UpdatePackageVersionAPI. Em seguida, você pode referenciar a versão do pacote ao criar o trabalho por meio do AWS IoT console ou emitindo a operação da CreateJobAPI.

Nomeação dos pacotes e das versões durante a implantação

Para implantar uma versão do pacote de software em um dispositivo, confirme se o pacote de software e a versão do pacote referenciados no documento do trabalho correspondem ao pacote de software e à versão do pacote declarados no parâmetro `destinationPackageVersions` na operação da API `CreateJob`. Se não corresponderem, você receberá uma mensagem de erro solicitando que as duas referências correspondam. Para obter mais informações sobre mensagens de erro do Catálogo de pacotes de software, consulte Mensagens de erro de solução de problemas geral.

Além dos pacotes de software e das versões de pacotes referenciados no documento de trabalho, você pode incluir pacotes de software e versões de pacotes adicionais no parâmetro `destinationPackageVersions` na operação da API `CreateJob` não referenciados no documento de trabalho. Inclua as informações de instalação necessárias no documento de trabalho para que os dispositivos instalem adequadamente as versões adicionais do pacote de software. Para obter mais informações sobre a operação `CreateJob` da API, consulte CreateJob.

Segmentação de empregos por meio de AWS IoT grupos dinâmicos

O Catálogo de pacotes de software trabalha com indexação de frota, trabalhos da AWS IoT e grupos de coisas dinâmicas da AWS IoT para filtrar e direcionar dispositivos na frota a fim de selecionar qual versão do pacote implantar nos dispositivos. Você pode executar uma consulta de indexação de frota com base nas informações atuais do pacote do seu dispositivo e direcionar essas coisas para um AWS IoT trabalho. Você também pode lançar atualizações de software, mas somente para dispositivos de destino qualificados. Por exemplo, você pode especificar que deseja implantar uma

configuração somente nos dispositivos que atualmente executam o `iot-device-client 1.5.09`. Para obter mais informações, consulte a opção Criar um grupo de objetos dinâmicas.

Versões reservadas nomeadas de sombra e pacote

Se configurado, o AWS IoT Jobs pode atualizar uma coisa reservada chamada shadow (`$package`) quando o trabalho for concluído com êxito. Se você fizer isso, não precisará associar manualmente uma versão do pacote a uma sombra nomeada reservado do objeto.

Você pode optar por associar ou atualizar manualmente uma versão do pacote à sombra nomeada reservada do objeto nas seguintes situações:

- Você registra algo AWS IoT Core sem associar a versão do pacote instalado.
- AWS IoT Jobs não está configurado para atualizar a coisa reservada chamada shadow.
- Você usa um processo interno para enviar versões de pacotes para sua frota e esse processo não é atualizado AWS IoT Core quando concluído.

> **Note**
>
> Recomendamos que você use AWS IoT Jobs para atualizar a versão do pacote no reservado chamado shadow (`$package`). Atualizar o parâmetro de versão na `$package` sombra por meio de outros processos (como chamadas de API manuais ou programáticas) quando o AWS IoT Jobs também está configurado para atualizar a sombra, pode causar inconsistências entre a versão real no dispositivo e a versão relatada à sombra nomeada reservada.

Você pode adicionar ou atualizar uma versão de pacote para uma sombra nomeada reservada do objeto (`$package`) por meio do console ou da operação da API `UpdateThingShadow`. Para obter mais informações, consulte Associando uma versão de pacote a uma AWS IoT coisa.

> **Note**
>
> Associar uma versão do pacote a AWS IoT algo não atualiza diretamente o software do dispositivo. Você deve implantar a versão do pacote no dispositivo para atualizar o software do dispositivo.

Como desinstalar um pacote de software e sua versão do pacote

`$null`é um espaço reservado que solicita que o serviço AWS IoT Jobs remova o pacote de software e a versão do pacote existentes do dispositivo chamado shadow. `$package` Para obter mais informações, consulte Sombra nomeada reservada.

Para usar esse recurso, substitua o nome da versão no final do destinationPackageVersionAmazon Resource Name (ARN) por. `$null` Depois disso, você deve instruir seu serviço a remover o software do dispositivo.

O ARN autorizado usa o seguinte formato:

```
arn:aws:iot:<regionCode>:111122223333:package/<packageName>/version/$null
```

Por exemplo,

```
$ aws iot create-job \
    ... \
    --destinationPackageVersions ["arn:aws:iot:us-east-1:111122223333:package/
samplePackage/version/$null"]
```

Conceitos básicos do Catálogo de pacotes de software

Você pode criar e manter o Catálogo AWS IoT Device Management de Pacotes de Software por meio das Console de gerenciamento da AWS operações de AWS IoT Core API e AWS Command Line Interface (AWS CLI).

> **Note**
>
> Habilitar a indexação de AWS IoT frotas é um requisito para usar o Software Package Catalog. Operações básicas, como criar uma versão de pacote de software no Console de gerenciamento da AWS e usar o comando da `CreatePackage` API, falharão sem a indexação de AWS IoT frota ativada.
> Para obter mais informações sobre como usar a indexação de AWS IoT frotas com o Software Package Catalog, consultePreparação da indexação de frota.

Como usar o console

Para usar o Console de gerenciamento da AWS, faça login em sua AWS conta e navegue até AWS IoT Core. No painel de navegação, escolha a opção Pacotes do software. Você pode criar e gerenciar pacotes e suas versões nesta seção.

Uso de operações de API ou CLI

Você pode usar as operações de AWS IoT Core API para criar e gerenciar recursos do Software Package Catalog. Para obter mais informações, consulte Referência AWS SDKs e kits de ferramentas de AWS IoT API. Os AWS CLI comandos também gerenciam seu catálogo. Para obter mais informações, consulte a opção AWS IoT Referência de comandos da CLI.

Este capítulo contém as seguintes seções:

- Criação de um pacote de software e uma versão do pacote
- Implantando uma versão do pacote por meio de trabalhos AWS IoT
- Associando uma versão de pacote a qualquer coisa AWS IoT

Criação de um pacote de software e uma versão do pacote

Você pode usar as etapas a seguir para criar um pacote e uma versão inicial por meio do Console de gerenciamento da AWS.

Para criar um pacote de software

1. Faça login na sua AWS conta e navegue até o AWS IoT console.
2. No painel de navegação lateral, selecione Pacotes do software.
3. Na página do AWS IoT pacote de software, selecione Criar pacote. A caixa de diálogo Habilitar dependências para gerenciamento de pacotes é exibida.
4. Em indexação de frota, selecione Adicionar pacotes de software e versão do dispositivo. Isso é necessário para o Catálogo de pacotes de software e fornece indexação de frota e métricas sobre sua frota.
5. [Opcional] Se você quiser que os AWS IoT trabalhos atualizem a sombra nomeada reservada quando os trabalhos forem concluídos com êxito, selecione Atualizar automaticamente as sombras dos trabalhos. Se você não quiser que os AWS IoT trabalhos façam essa atualização, deixe essa caixa de seleção desmarcada.
6. [Opcional] Para conceder aos AWS IoT trabalhos o direito de atualizar a sombra nomeada reservada, em Selecionar função, escolha Criar função. Se você não quiser que os AWS IoT trabalhos façam essa atualização, essa função não é necessária.

7. Crie ou selecione uma função.

   a. Se você não tiver uma função para essa finalidade: Quando a caixa de diálogo Criar função for exibida, insira um Nome de função e escolha Criar.

   b. Se você tiver uma função para essa finalidade: em Selecionar função, escolha sua função e, em seguida, certifique-se de que a caixa de seleção Anexar política ao perfil do IAM esteja marcada.

8. Selecione a opção Confirmar. A página Criar novo pacote é exibida.
9. Em Detalhes do pacote, insira um Nome do pacote.
10. Em Descrição do pacote, insira as informações para ajudá-lo a identificar e gerenciar esse pacote.
11. [Opcional] Você pode usar tags para ajudar a categorizar e gerenciar esse pacote. Para adicionar tags, expanda Tags, escolha Adicionar tag e insira um par de valores-chave. É possível inserir até 50 tags. Para obter mais informações, consulte Como marcar seus AWS IoT recursos.

Para adicionar uma versão do pacote ao criar um novo pacote

1. Em Versão inicial, insira o Nome da versão.

   Recomendamos usar o SemVer formato (por exemplo,`1.0.0`) para identificar de forma exclusiva a versão do seu pacote. Você também pode usar uma estratégia de formatação diferente que melhor se adapte ao seu caso de uso. Para obter mais informações, consulte Ciclo de vida da versão do pacote.

2. Em Descrição da versão, insira as informações que ajudarão você a identificar e gerenciar essa versão do pacote.

   > **Note**
   >
   > A caixa de seleção Versão padrão está desativada porque as versões do pacote são criadas em um estado `draft`. Você pode nomear a versão padrão após criar a versão do pacote e quando você altera o estado para `published`. Para obter mais informações, consulte Ciclo de vida da versão do pacote.

3. [Opcional] Para ajudá-lo a gerenciar essa versão ou comunicar informações aos seus dispositivos, insira um ou mais pares de nome-valor para os atributos da versão. Escolha a

opção Adicionar atributo para cada par nome-valor inserido. Para obter mais informações, consulte Atributos de versão.

4. [Opcional] Você pode usar tags para ajudar a categorizar e gerenciar esse pacote. Para adicionar tags, expanda Tags, escolha Adicionar tag e insira um par de valores-chave. É possível inserir até 50 tags. Para obter mais informações, consulte Como marcar seus AWS IoT recursos.
5. Escolha Próximo.

Associar a lista de materiais de software a uma versão do pacote (opcional)

1. Na Etapa 3: Versão SBOMs (opcional) na janela de configurações do SBOM, escolha o formato de arquivo SBOM padrão e o modo de validação usados para validar sua lista de materiais de software antes que ela seja associada à versão do pacote.
2. Na janela Adicionar arquivo da SBOM, insira o nome do recurso da Amazon (ARN) representando seu bucket do Amazon S3 com versionamento e o formato de arquivo da SBOM preferido se o tipo padrão não funcionar.

> **Note**
>
> Você pode adicionar um único arquivo SBOM ou um único arquivo zip contendo vários SBOMs se tiver mais de uma lista de materiais de software para a versão do seu pacote.

3. Na janela Arquivo da SBOM adicionado, você pode visualizar o arquivo SBOM adicionado para a versão do pacote.
4. Escolha Criar pacote e versão. A página da versão do pacote é exibida e você pode ver o status de validação do arquivo da SBOM na janela Arquivo da SBOM adicionado. O status inicial será `In progress` quando o arquivo da SBOM for validado.

> **Note**
>
> Os status de validação do arquivo da SBOM são `Invalid file`, `Not started`, `In progress`, `Validated (SPDX)`, `Validated (CycloneDX)` e os motivos da falha de validação.

Implantando uma versão do pacote por meio de trabalhos AWS IoT

Você pode usar as etapas a seguir para implantar uma versão do pacote por meio do Console de gerenciamento da AWS.

Pré-requisitos:

Antes de começar, faça o seguinte:

- Registre AWS IoT coisas com AWS IoT Core. Para obter instruções sobre como adicionar seus dispositivos AWS IoT Core, consulte Criar um objeto.
- [Opcional] Crie um grupo de AWS IoT coisas ou grupo dinâmico para atingir os dispositivos nos quais você implantará a versão do pacote. Para obter instruções sobre como criar um grupo de objetos, consulte Criar um grupo de objetos estático. Para obter instruções sobre como criar um grupo de objetos dinâmicas, consulte Criar um grupo de objetos dinâmicas.
- Crie um pacote de software e uma versão do pacote. Para obter mais informações, consulte Criação de um pacote de software e uma versão do pacote.
- Criar um documento de trabalho. Para obter mais informações, consulte Preparação do documento de trabalho e a versão do pacote para implantação.

Para implantar um AWS IoT trabalho

1. No AWS IoT console, escolha Pacotes de software.
2. Escolha o pacote de software que você deseja implantar. A página de detalhes do pacote de software é exibida.
3. Escolha a versão do pacote que você deseja implantar, em Versões, e escolha Implantar versão do trabalho.
4. Se esta for sua primeira vez implantando um trabalho por meio deste portal, uma caixa de diálogo que descreve os requisitos será exibida. Revise as informações e selecione Confirmar.
5. Insira um nome para a implantação ou deixe o nome gerado automaticamente no campo Nome.
6. [Opcional] No campo Descrição, insira uma descrição que identifique a finalidade ou o conteúdo da implantação ou deixe as informações geradas automaticamente.

   Nota: recomendamos que você não use informações de identificação pessoal nos campos Nome e descrição do trabalho.
7. [Opcional] Adicione qualquer tag para associar a essa tarefa.

8. Escolha Próximo.
9. Em Objetivos da tarefa, escolha as objetos ou grupos de objetos que devem receber a tarefa.
10. No campo Arquivo de tarefa, especifique o arquivo JSON do documento de trabalho.
11. Integração do Open Jobs com o serviço Catálogo de pacotes.
12. Selecione os pacotes e as versões especificados em seu documento de trabalho.

> **Note**
>
> É necessário escolher os mesmos pacotes e versões de pacotes especificados no documento de trabalho. Você pode incluir mais, mas a tarefa emitirá instruções somente para os pacotes e versões incluídos no documento de trabalho. Para obter mais informações, consulte Nomeando os pacotes e as versões durante a implantação.

13. Escolha Próximo.
14. Na página Configuração da tarefa, selecione um dos seguintes tipos de tarefa na caixa de diálogo Configuração da tarefa:
    - Trabalho de snapshot: um trabalho de instantâneo é concluído quando termina sua execução nos dispositivos e grupos de destino.
    - Trabalho contínuo: um trabalho contínuo se aplica a grupos de objetos e é executado em qualquer dispositivo que você adicione posteriormente a um grupo-alvo especificado.
15. Na caixa de diálogo Configurações adicionais - opcionais, revise as seguintes configurações de Trabalho opcionais e faça suas seleções. Para obter mais informações, consulte Configurações de implantação, agendamento e cancelamento de tarefas e Tempo limite de execução do trabalho e novas configurações.
    - Configuração de distribuição
    - Configuração de agendamento
    - Configuração de tempo limite de execução de trabalhos
    - Configuração de repetição de execuções de tarefas
    - Configuração de anulação
16. Revise as seleções de tarefas e escolha Enviar.

Depois de criar a tarefa, o console gera uma assinatura JSON e a coloca no seu documento de trabalho. Você pode usar o AWS IoT console para ver o status de um trabalho, cancelar ou excluir um trabalho. Para gerenciar trabalhos, acesse o Hub de trabalhos do console.

Associando uma versão de pacote a qualquer coisa AWS IoT

Depois de instalar o software em seu dispositivo, você pode associar uma versão do pacote a uma AWS IoT coisa reservada chamada shadow. Se os AWS IoT trabalhos tiverem sido configurados para atualizar o item chamado shadow reservado após a implantação e conclusão bem-sucedida do trabalho, você não precisará concluir esse procedimento. Para obter mais informações, consulte Sombra nomeada reservada.

Pré-requisitos:

Antes de começar, faça o seguinte:

- Crie qualquer AWS IoT coisa, ou coisas, e estabeleça a telemetria por meio de. AWS IoT Core Para obter mais informações, consulte Introdução ao AWS IoT Core.
- Crie um pacote de software e uma versão do pacote. Para obter mais informações, consulte Criação de um pacote de software e uma versão do pacote.
- Instale o software da versão do pacote no dispositivo.

> **Note**
>
> Associar uma versão do pacote a AWS IoT algo não atualiza nem instala o software no dispositivo físico. A versão do pacote deve ser implantada no dispositivo.

Para associar uma versão de pacote a qualquer AWS IoT coisa

1. No painel de navegação do AWS IoT console, expanda o menu Todos os dispositivos e escolho objetos.
2. Identifique o AWS IoT item que você deseja atualizar na lista e escolha o nome do item para exibir sua página de detalhes.
3. Na seção Detalhes, escolha Pacotes e versões.
4. Escolha Adicionar ao pacote e à versão.
5. Em Escolher um pacote de dispositivo, escolha o pacote de software desejado.

6. Em Escolher uma versão, escolha a versão do software que você deseja.
7. Escolha Adicionar pacote do dispositivo.

   O pacote e a versão aparecem na lista Pacotes e versões selecionados.
8. Repita essas etapas para cada pacote e versão que você quer associar a esse objeto.
9. Ao terminar, escolha Adicionar detalhes do pacote e da versão. A página Detalhes do objeto é aberta e você pode ver o novo pacote e a nova versão na lista.

AWS IoT Empregos

Use AWS IoT Jobs para definir um conjunto de operações remotas que podem ser enviadas e executadas em um ou mais dispositivos conectados AWS IoT a. Por exemplo, você pode definir um trabalho que instrui um conjunto de dispositivos a baixar e instalar aplicativos, executar atualizações de firmware, reinicializar, alternar certificados ou executar operações de solução de problemas remotamente.

Acessando AWS IoT empregos

Você pode começar a usar o AWS IoT Jobs usando o console ou a AWS IoT Core API.

Utilizar o console

Faça login no Console de gerenciamento da AWS e acesse o AWS IoT console. No painel de navegação, escolha Gerenciar e, depois, Trabalhos. Você pode criar e gerenciar trabalhos nesta seção. Se você quiser criar e gerenciar modelos de trabalho, no painel de navegação, escolha Modelos de trabalho. Para obter mais informações, consulte Crie e gerencie trabalhos usando o Console de gerenciamento da AWS.

Como usar a API ou a CLI

Você pode começar usando as operações da AWS IoT Core API. Para saber mais, consulte Referência de API do AWS IoT. A AWS IoT Core API na qual os AWS IoT trabalhos são criados é suportada pelo AWS SDK. Para obter mais informações, consulte Kits AWS SDKs de ferramentas.

Você pode usar os comandos AWS CLI to run para criar e gerenciar trabalhos e modelos de trabalho. Para obter mais informações, consulte referência de CLI de AWS IoT.

AWS IoT Empregos, regiões e endpoints

AWS IoT O Jobs oferece suporte a endpoints de API de plano de controle e plano de dados que são específicos para você Região da AWS. Os endpoints da API do plano de dados são específicos para seu Conta da AWS e. Região da AWS Para obter mais informações sobre os endpoints de AWS IoT jobs, consulte AWS IoT Device Management - endpoints de dados de jobs na Referência AWS geral.

O que é uma operação remota?

> **Note**
>
> Over-the-air As atualizações (OTA) não estão disponíveis nas seguintes regiões:
>
> - Europa (Espanha)
> - Ásia-Pacífico (Malásia)

Uma operação remota é qualquer atualização ou ação que você pode realizar em um dispositivo físico, dispositivo virtual ou endpoint que pode ser feita remotamente sem a necessidade da presença física de um operador ou técnico. A operação remota é realizada usando uma atualização over-the-air (OTA) para que seus dispositivos não precisem estar fisicamente presentes. Gerenciar sua frota de dispositivos no Nuvem AWS permite que você execute operações remotas em seus dispositivos quando eles são registrados no AWS IoT Core.

AWS IoT Device Management O Jobs oferece uma abordagem escalável para realizar ações remotas em seus dispositivos registrados no. AWS IoT Core Uma tarefa é criada Nuvem AWS e enviada para todos os dispositivos de destino usando uma atualização OTA por meio do protocolo MQTT ou HTTP.

AWS IoT Device Management Os trabalhos oferecem a capacidade de realizar operações remotas, como redefinições de fábrica, reinicializações de dispositivos e atualizações OTA de software, de forma segura, escalável e econômica.

Para obter mais informações sobre AWS IoT Core, consulteO que é AWS IoT?.

Para obter mais informações sobre Trabalhos de AWS IoT Device Management , consulte O que é AWS IoT Jobs?.

Benefícios de usar o AWS IoT Device Management Jobs para operações remotas

Usar o AWS IoT Device Management Jobs para realizar suas operações remotas simplifica o gerenciamento da sua frota de dispositivos. A lista a seguir destaca alguns dos principais benefícios de usar trabalhos AWS IoT Device Management para realizar operações remotas:

- Integração perfeita com outros Serviços da AWS

- AWS IoT Device Management O Jobs se integra estreitamente aos seguintes recursos e valor agregado: Serviços da AWS
  - Amazon S3: armazene suas instruções de operação remota em um bucket seguro do Amazon S3, em que você controla as permissões de acesso para esse conteúdo. O uso de um bucket do Amazon S3 fornece uma solução de armazenamento escalável e durável que se integra de forma nativa ao Device AWS IoT Management Software Package Catalog, permitindo que AWS IoT Device Management Jobs faça referência e substitua nas instruções de atualização. Para obter mais informações, consulte O que é a Amazon S3?.
  - Amazon CloudWatch: monitore e registre o status de implementação da operação remota da execução do trabalho para cada dispositivo, além de outras atividades do dispositivo, para rastrear e analisar o desempenho geral do trabalho em AWS IoT Device Management Jobs. Para obter mais informações, consulte O que é a Amazon CloudWatch? Monitorar logs de trabalhos e capturar dados históricos para solução de problemas. Como funcionam com trabalhos.
  - Serviço AWS IoT Device Shadow: mantenha uma representação digital de sua AWS IoT coisa por meio de uma sombra de dispositivo usando o AWS IoT Device Management Jobs para que o estado do seu dispositivo esteja disponível para aplicativos e outros serviços, independentemente da conectividade do dispositivo. Para obter mais informações, consulte Serviço AWS IoT Device Shadow.
- Práticas recomendadas de segurança
  - Controle de permissão: controle as permissões de acesso às suas instruções operacionais remotas usando o Amazon S3 e determine quais usuários do IAM podem implantar suas instruções operacionais remotas em sua frota de dispositivos usando AWS IoT políticas e funções de usuário do IAM.
    - Para obter mais informações sobre AWS IoT políticas, consulteCrie uma AWS IoT política.
    - Para obter mais informações sobre funções de usuário do IAM, consulte Gerenciamento de identidade e acesso para AWS IoT.
- Escalabilidade
  - Implantação de trabalho direcionado: controle quais dispositivos recebem o documento de trabalho de um trabalho com uma implantação de trabalho direcionada usando critérios específicos de agrupamento de dispositivos inseridos em seu documento de trabalho ao criar o trabalho. Criar AWS IoT algo para cada dispositivo e armazenar essas informações no AWS IoT registro permite que você realize pesquisas direcionadas usando a indexação de frotas. Você pode criar grupos personalizados com base nos resultados da pesquisa de indexação

da frota para apoiar a implantação do seu trabalho de destino. Para obter mais informações, consulte Gerenciando dispositivos com AWS IoT. Use trabalhos para capturar instantâneos versus trabalhos contínuos.

  - Status do trabalho: acompanhe o status da distribuição do documento de trabalho em sua frota de dispositivos e o status geral do trabalho em um nível de frota de dispositivos, além do status de implementação individual do documento de trabalho em cada dispositivo. Para obter mais informações, consulte Trabalhos e estados de execução de trabalhos.
  - Nova escalabilidade do dispositivo: implante facilmente seu documento de trabalho em um novo dispositivo adicionando-o a um grupo personalizado criado usando a indexação de frotas por meio de um trabalho contínuo. Isso economizará seu tempo ao não precisar implantar o documento de trabalho em cada novo dispositivo separadamente. Você também pode adotar uma abordagem mais direcionada com uma captura instantânea implantando um documento de trabalho em um grupo predeterminado de dispositivos uma vez e então o trabalho será concluído.
- Flexibilidade
  - Configurações de trabalho: personalize seu trabalho e seu documento de trabalho com implementação, agendamento, cancelamento, tempo limite e repetição das configurações de trabalho opcionais para atender às suas necessidades específicas. Para obter mais informações, consulte Configurações de trabalho.
- Econômica
  - Apresente uma estrutura de custos mais eficiente para manter sua frota de dispositivos aproveitando o AWS IoT Device Management Jobs para implantar atualizações críticas e realizar tarefas de manutenção de rotina. Uma solução do-it-yourself (DIY) para manter sua frota de dispositivos inclui custos recorrentes e variáveis, como a infraestrutura necessária para hospedar e gerenciar a solução DIY, custos de mão de obra para desenvolver, manter e escalar a solução DIY e custos de transmissão de dados. Aproveitando a estrutura transparente e de custos fixos dos AWS IoT Device Management Jobs, você sabe exatamente quanto custará cada execução de trabalho de um dispositivo, além dos custos de transmissão de dados necessários para facilitar a implantação do documento de trabalho em sua frota de dispositivos e rastrear o status de execução do trabalho em cada dispositivo. Para obter mais informações, consulte Preços do AWS IoT Core.

O que é AWS IoT Jobs?

Use AWS IoT Jobs para definir um conjunto de operações remotas que podem ser enviadas e executadas em um ou mais dispositivos conectados AWS IoT a.

Para criar trabalhos, primeiro defina um documento de trabalho que contenha uma lista de instruções descrevendo as operações que o dispositivo deve executar remotamente. Para realizar essas operações, especifique uma lista de destinos, que são itens individuais, grupos de objetos ou ambos. Juntos, o documento de trabalho e as metas constituem uma implantação.

Cada implantação pode ter configurações adicionais:

- Distribuição: essa configuração define quantos dispositivos recebem o documento de trabalho a cada minuto.
- Anular: se um determinado número de dispositivos não receber a notificação do trabalho, use essa configuração para cancelar o trabalho. Isso evita o envio de uma atualização incorreta para uma frota inteira.
- Tempo limite: se uma resposta não for recebida de suas metas de trabalho dentro de um determinado período, o trabalho pode falhar. Você pode acompanhar o trabalho que está sendo executado nesses dispositivos.
- Tentar novamente: se um dispositivo relatar uma falha ou um trabalho expirar, você pode usar AWS IoT Jobs para reenviar o documento do trabalho para o dispositivo automaticamente.
- Agendamento: essa configuração permite que você agende um trabalho para uma data e hora futuras. Ela também permite que você crie janelas de manutenção recorrentes que atualizam dispositivos durante períodos predefinidos e de baixo tráfego.

AWS IoT Jobs envia uma mensagem para informar aos alvos que um trabalho está disponível. O destino inicia a execução do trabalho baixando o documento do trabalho, executando as operações que ele especifica e relatando seu progresso. AWS IoT Você pode acompanhar o progresso de um trabalho para um alvo específico ou para todos os destinos executando comandos fornecidos por AWS IoT Jobs. Quando um trabalho é iniciado, ele tem o status Em andamento. Os dispositivos então relatam atualizações incrementais enquanto exibem esse status até que o trabalho seja bem-sucedido, falhe ou expire.

Os tópicos a seguir descrevem alguns conceitos-chave de trabalhos e do ciclo de vida de trabalhos e execuções de trabalhos.

Tópicos

- Principais conceitos sobre trabalhos
- Trabalhos e estados de execução de trabalhos

Principais conceitos sobre trabalhos

Os conceitos a seguir fornecem detalhes sobre AWS IoT trabalhos e como criar e implantar trabalhos para executar operações remotas em seus dispositivos.

Conceitos básicos

A seguir estão os conceitos básicos que você deve conhecer ao usar AWS IoT Jobs.

Trabalho

Um trabalho é uma operação remota que é enviada e executada em um ou mais dispositivos conectados à AWS IoT. Por exemplo, você pode definir um trabalho que instrui um conjunto de dispositivos a baixar e instalar um aplicativo ou executar atualizações de firmware, reinicializar, alternar certificados ou executar operações de solução de problemas remotamente.

Documento de trabalho

Para criar um trabalho, você deve primeiro criar um documento de trabalho, que consiste na descrição das operações remotas a serem realizadas pelos dispositivos.

Os documentos de trabalho são documentos JSON codificados em UTF-8 e devem conter as informações das quais seus dispositivos precisam para executar um trabalho. Um documento de trabalho contém um ou mais dos URLs quais o dispositivo pode baixar uma atualização ou outros dados. O documento de trabalho pode ser armazenado em um bucket do Amazon S3 ou ser incluído em linha com o comando que cria o trabalho.

> **Note**
>
> Nas respostas do MQTT, o campo `jobDocument` é um objeto JSON. Nas respostas HTTP, é uma representação de string do objeto JSON.

Destino

Ao criar um trabalho, você especifica uma lista de destinos, que são os dispositivos que devem executar as operações. Os destinos podem ser objetos, grupos de objetos ou ambos. O serviço AWS IoT Jobs envia uma mensagem para cada alvo para informá-lo de que um trabalho está disponível.

Implantação

Depois de criar um trabalho fornecendo o documento de trabalho e especificando sua lista de destinos, o documento de trabalho é então implantado nos dispositivos de destino remotos para os quais você deseja realizar a atualização. Para trabalhos de snapshot, o trabalho será concluído após a implantação nos dispositivos de destino. Para trabalhos contínuos, um trabalho é implantado em um grupo de dispositivos à medida que eles são adicionados aos grupos.

Execução de trabalho

Uma execução de trabalho é uma instância de um trabalho em um dispositivo de destino. O destino começa a execução de um trabalho ao fazer download do documento de trabalho. Em seguida, ele executa as operações especificadas no documento e relata seu progresso para AWS IoT o. Um número de execução é um identificador exclusivo da execução de um trabalho em um destino específico. O serviço AWS IoT Jobs fornece comandos para rastrear o progresso da execução de um trabalho em um destino e o progresso de um trabalho em todos os alvos.

Conceitos dos tipos de trabalho

Os conceitos a seguir podem ajudá-lo a entender mais sobre os diferentes tipos de trabalhos que você pode criar com AWS IoT Jobs.

Trabalho de snapshot

Por padrão, um trabalho é enviado para todos os destinos que você especifica ao criar o trabalho. Depois que esses destinos concluem o trabalho (ou relatam que não puderam concluí-lo), o trabalho está concluído.

Trabalho contínuo

Um trabalho contínuo é enviado a todos os destinos que você especifica ao criar o trabalho. Ele continua a executar e é enviado a todos os novos dispositivos (objetos) que são adicionados ao grupo de destino. Por exemplo, um trabalho contínuo pode ser usado para integrar ou atualizar dispositivos conforme são adicionados a um grupo. Você pode criar um trabalho contínuo definindo um parâmetro opcional ao criá-lo.

> **Note**
>
> Ao segmentar sua frota de IoT usando grupos de objetos dinâmicas, recomendamos que você use trabalhos contínuos em vez de trabalhos de snapshot. Ao usar trabalhos contínuos, os dispositivos que se juntam ao grupo recebem a execução do trabalho mesmo após a criação do trabalho.

Pré-assinado URLs

Para acesso seguro e por tempo limitado aos dados que não estão incluídos no documento de trabalho, você pode usar o Amazon S3 pré-assinado. URLs Coloque seus dados em um bucket do Amazon S3 e adicione um link de espaço reservado para os dados no documento de trabalho. Quando AWS IoT Jobs recebe uma solicitação para o documento de trabalho, ele analisa o documento procurando os links de espaço reservado e, em seguida, substitui os links pelo Amazon S3 pré-assinado. URLs

O link de espaço reservado tem o seguinte formato:

```
${aws:iot:s3-presigned-url:https://s3.amazonaws.com/bucket/key}
```

onde *`bucket`* é o nome do seu bucket e *`key`* é o objeto no bucket ao qual você está vinculando.

Nas regiões de Pequim e Ningxia, o trabalho pré-assinado só URLs funciona se o proprietário do recurso tiver uma licença ICP (Internet Content Provider). Para obter mais informações, consulte Amazon Simple Storage Service na documentação Getting Started with AWS Services in China.

Conceitos da configuração de trabalho

Os conceitos a seguir podem ajudar você a entender como configurar trabalhos.

Distribuições

Você pode especificar a rapidez com que os destinos são notificados sobre a execução de um trabalho pendente. Isso permite que você crie uma distribuição em etapas para gerenciar melhor as atualizações, reinicializações e outras operações. Você pode criar uma configuração de distribuição usando uma taxa de distribuição estática ou uma exponencial. Para especificar o número máximo de destinos de trabalho a serem informados por minuto, use uma taxa de distribuição estática.

Para obter exemplos de como definir taxas de distribuição e obter mais informações sobre como configurar distribuições de trabalhos, consulte Configurações de implantação, agendamento e anulação de trabalhos.

Programação

O agendamento de trabalhos permite que você agende o cronograma de distribuição de um documento de trabalho em todos os dispositivos do grupo de destino para trabalhos contínuos e de snapshot. Além disso, você pode criar uma janela de manutenção opcional contendo datas e horários específicos em que um trabalho distribuirá o documento de trabalho em todos os dispositivos do grupo de destino. Uma janela de manutenção é uma instância recorrente com uma frequência de datas e horários diários, semanais, mensais ou personalizados selecionados durante a criação do trabalho inicial ou do modelo de trabalho. Somente trabalhos contínuos podem ser programados para realizar uma distribuição durante uma janela de manutenção.

O agendamento de trabalhos é específico para seu trabalho. Execuções de trabalhos individuais não podem ser agendadas. Para obter mais informações, consulte Configurações de implantação, agendamento e anulação de trabalhos.

Anular

Você pode criar um conjunto de condições para cancelar distribuições quando critérios que você especificou forem atendidos. Para obter mais informações, consulte Configurações de implantação, agendamento e anulação de trabalhos.

Tempos limite

O tempo limite do trabalho notifica você sempre que uma implantação de trabalho ficar paralisada no estado `IN_PROGRESS` por um período inesperadamente longo. Há dois tipos de temporizadores: em andamento e de etapa. Quando o trabalho estiver `IN_PROGRESS`, você poderá monitorar e acompanhar o progresso da implantação do trabalho.

As configurações de distribuição e anulação são específicas para seu trabalho, enquanto a configuração de tempo limite é específica para a implantação de um trabalho. Para obter mais informações, consulte Configurações de novas tentativas e de tempo limite de execuções de trabalhos.

Repetições

As repetições de trabalho possibilitam repetir a execução do trabalho quando um trabalho falha, atinge o tempo limite ou ambos. É possível ter até dez novas tentativas para executar a trabalho.

Você pode monitorar e acompanhar o progresso de sua nova tentativa e se a execução do trabalho foi bem-sucedida.

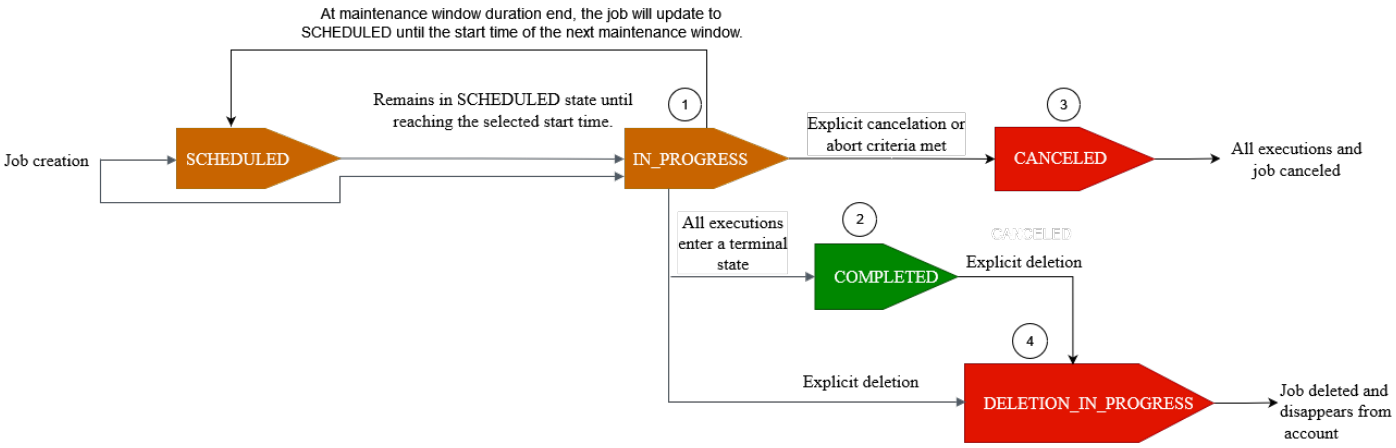
As configurações de distribuição e anulação são específicas para seu trabalho, enquanto a configuração de tempo limite e repetição é específica para a execução de um trabalho. Para obter mais informações, consulte Configurações de novas tentativas e de tempo limite de execuções de trabalhos.

Trabalhos e estados de execução de trabalhos

As seções a seguir descrevem o ciclo de vida de um AWS IoT trabalho e o ciclo de vida de uma execução do trabalho.

Estados do trabalho

O diagrama a seguir mostra os diferentes estados de um AWS IoT trabalho.



Um trabalho que você cria usando AWS IoT Jobs pode estar em um dos seguintes estados:

- SCHEDULED

  Durante a criação inicial do trabalho ou do modelo de trabalho usando o AWS IoT console, CreateJoba CreateJobTemplateAPI ou a API, você pode selecionar a configuração de agendamento opcional no AWS IoT console ou `SchedulingConfig` na CreateJobAPI ou CreateJobTemplateAPI. Quando você inicia um trabalho agendado contendo um determinado `startTime`, `endTime` e `endBehavior`, o status do trabalho é atualizado para `SCHEDULED`. Quando o trabalho atingir o `startTime` ou o `startTime` selecionado da próxima janela de manutenção (se você selecionou a distribuição do trabalho durante uma janela de manutenção), o

status será atualizado de `SCHEDULED` para `IN_PROGRESS` e iniciará a distribuição do documento de trabalho em todos os dispositivos do grupo de destino.

- IN_PROGRESS

  Quando você cria um trabalho usando o AWS IoT console ou a CreateJobAPI, o status do trabalho é atualizado para`IN_PROGRESS`. Durante a criação do trabalho, o trabalho de AWS IoT começa a implementar execuções de trabalhos nos dispositivos do seu grupo de destino. Depois que todas as execuções de trabalhos forem distribuídas, o trabalho de AWS IoT espera que os dispositivos concluam a ação remota.

  Para obter informações sobre simultaneidade e limites que se aplicam a trabalhos em andamento, consulte. AWS IoT Limites de trabalhos

  > **Note**
  >
  > Quando um trabalho `IN_PROGRESS` chegar ao final da janela de manutenção atual, a distribuição do documento do trabalho será interrompida. O trabalho será atualizado para `SCHEDULED` até a `startTime` da próxima janela de manutenção.

- CONCLUÍDO

  Um trabalho contínuo é processado de uma das seguintes maneiras:

  - Para um trabalho contínuo sem a configuração de agendamento opcional selecionada, ele está sempre em andamento e continua sendo executado em todos os novos dispositivos adicionados ao grupo de destino. Nunca alcançará um estado de status de `COMPLETED`.
  - Para um trabalho contínuo com a configuração de agendamento opcional selecionada, o seguinte é verdadeiro:
    - Se um `endTime` foi fornecido, um trabalho contínuo alcançará o status `COMPLETED` quando o `endTime` for aprovado e todas as execuções do trabalho tenham atingido um estado de status terminal.
    - Se um `endTime` não tiver sido fornecido na configuração de agendamento opcional, o trabalho contínuo continuará executando a distribuição do documento de trabalho.

  Para um trabalho de snapshot, o status do trabalho muda para `COMPLETED` quando todas as execuções do trabalho entram em um estado terminal, como `SUCCEEDED`, `FAILED`, `TIMED_OUT`, `REMOVED` ou `CANCELED`.

- CANCELED

Quando você cancela um trabalho usando o AWS IoT console, a CancelJobAPI ou oConfiguração de anulação de trabalho, o status do trabalho muda paraCANCELED. Durante o cancelamento do trabalho, o AWS IoT Jobs começa a cancelar as execuções de tarefas criadas anteriormente.

Para obter informações sobre simultaneidade e limites que se aplicam a trabalhos que estão sendo cancelados, consulte AWS IoT Limites de trabalhos.

- DELETION_IN_PROGRESS

  Quando você exclui um trabalho usando o AWS IoT console ou a DeleteJobAPI, o status do trabalho muda paraDELETION_IN_PROGRESS. Durante a exclusão do trabalho, o AWS IoT Jobs começa a excluir execuções de trabalhos criados anteriormente. Depois que todas as execuções de tarefas forem excluídas, a tarefa desaparecerá da sua AWS conta.

Estados de execução de trabalho

A tabela a seguir mostra os diferentes estados da execução de um AWS IoT trabalho e se a alteração de estado é iniciada pelo dispositivo ou pelos AWS IoT Jobs.

Estados e origem da execução do trabalho

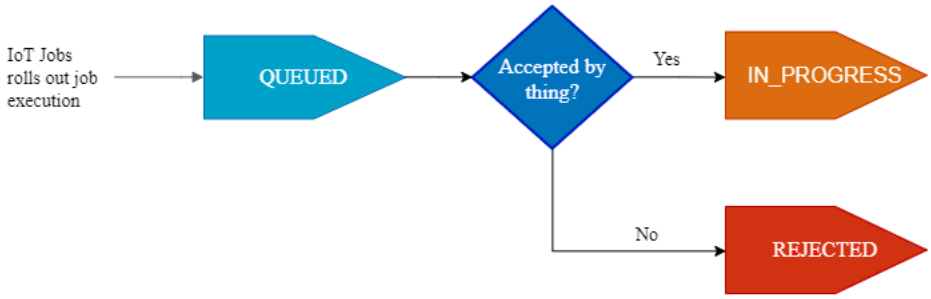
Estado de execução do trabalho	Iniciado pelo dispositivo?	Iniciado por AWS IoT Jobs?	Status do terminal?	Pode ser tentado novamente?
QUEUED	Não	Sim	Não	Não aplicável
IN_PROGRESS	Sim	Não	Não	Não aplicável
SUCCEEDED	Sim	Não	Sim	Não aplicável
FAILED	Sim	Não	Sim	Sim
TIMED_OUT	Não	Sim	Sim	Sim
REJECTED	Sim	Não	Sim	Não
REMOVED	Não	Sim	Sim	Não
CANCELED	Não	Sim	Sim	Não

A seção a seguir descreve mais sobre os estados de execução de um trabalho que é implementado quando você cria um trabalho com AWS IoT Jobs.

- QUEUED

  Quando AWS IoT Jobs lança uma execução de trabalho para um dispositivo de destino, o status de execução do trabalho é definido como`QUEUED`. A execução do trabalho permanece no estado `QUEUED` até:

  - Seu dispositivo receber a execução do trabalho, invocar as operações da API de trabalho e relatar o status como `IN_PROGRESS`.
  - Você cancelar o trabalho ou a execução do trabalho, ou quando os critérios de cancelamento especificados são atendidos e o status muda para `CANCELED`.
  - Seu dispositivo ser removido do grupo de destino e o status mudar para `REMOVED`.



- IN_PROGRESS

  Se seu dispositivo de IoT assinar o reservado Tópicos de trabalhos `$notify` e `$notify-next` invocar a `StartNextPendingJobExecution` API ou a `UpdateJobExecution` API com um status de`IN_PROGRESS`, AWS IoT Jobs definirá o status de execução do trabalho como. `IN_PROGRESS`

  A API `UpdateJobExecution` pode ser invocada várias vezes com o status de `IN_PROGRESS`. Você pode especificar detalhes adicionais sobre as etapas de execução usando o objeto `statusDetails`.

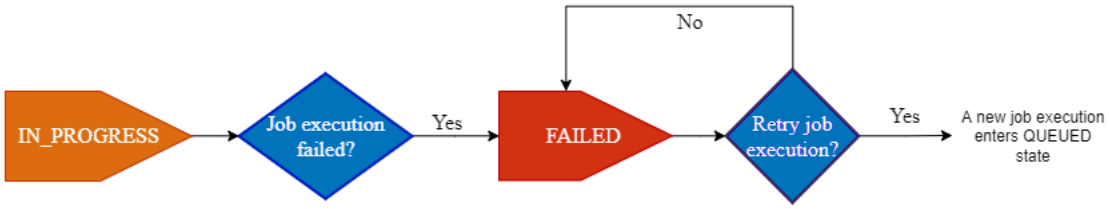
  > **Note**
  >
  > Se você criar várias tarefas para cada dispositivo, as AWS IoT tarefas e o protocolo MQTT não garantem a ordem de entrega.

- SUCCEEDED

Quando seu dispositivo conclui com êxito a operação remota, ele deve invocar a `UpdateJobExecution` API com o status de `SUCCEEDED` para indicar que a execução do trabalho foi bem-sucedida. AWS IoT Em seguida, os trabalhos atualizam e retornam o status de execução do trabalho como`SUCCEEDED`.

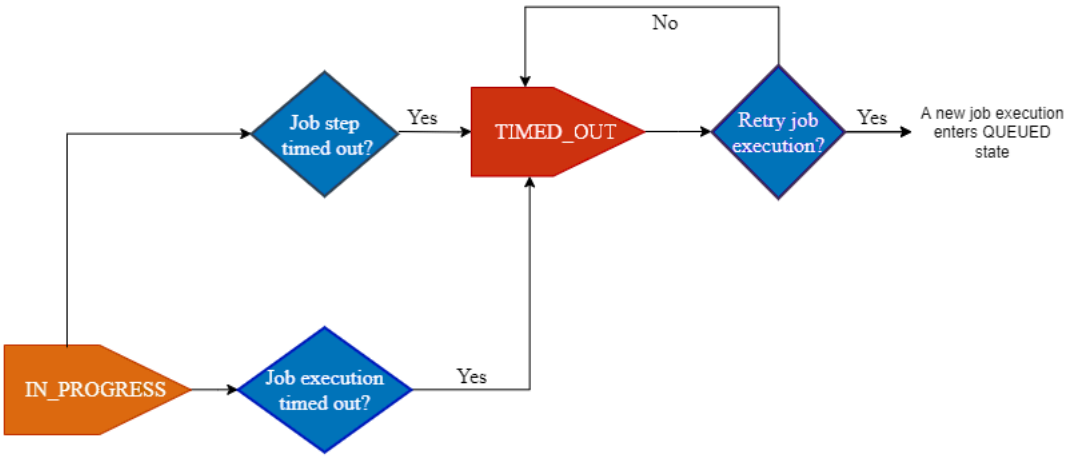
- FAILED

  Quando seu dispositivo não consegue concluir a operação remota, ele deve invocar a `UpdateJobExecution` API com o status de `Failed` para indicar que a execução do trabalho falhou. AWS IoT Em seguida, os trabalhos atualizam e retornam o status de execução do trabalho como`Failed`. Você pode repetir a execução desse trabalho para o dispositivo usando o. Configuração de repetição de execução de trabalho



- TEMPORIZADO_LIMITE

  Quando seu dispositivo não consegue concluir uma etapa do trabalho quando o status é`IN_PROGRESS`, ou quando não consegue concluir a operação remota dentro do tempo limite do cronômetro em andamento, o AWS IoT Jobs define o status de execução do trabalho como. `TIMED_OUT` Você também tem um temporizador para cada etapa de um trabalho em andamento e se aplica somente à execução do trabalho. A duração do temporizador em andamento é especificada usando a propriedade `inProgressTimeoutInMinutes` do Configuração de tempo limite de execução de trabalhos. Você pode repetir a execução desse trabalho para o dispositivo usando o. Configuração de repetição de execução de trabalho



- REJEITADO

  Quando seu dispositivo recebe uma solicitação inválida ou incompatível, ele deve invocar a `UpdateJobExecution` API com o status de. `REJECTED` AWS IoT Em seguida, os trabalhos atualizam e retornam o status de execução do trabalho como`REJECTED`.

- REMOVIDO

  Quando seu dispositivo não é mais um destino válido para a execução do trabalho, como quando está separado de um grupo dinâmico de objetos, o trabalho de AWS IoT define o status de execução do trabalho como `REMOVED`. Você pode reconectar o item ao seu grupo de destino e reiniciar a execução do trabalho no dispositivo.

- CANCELED

  Quando você cancela um trabalho ou cancela a execução de um trabalho usando o console ou a `CancelJobExecution` API, `CancelJob` ou quando os critérios de cancelamento especificados usando o Configuração de anulação de trabalho são atendidos, AWS IoT Jobs cancela o trabalho e define o status de execução do trabalho como. `CANCELED`

Gerenciar trabalhos

Use trabalhos para notificar os dispositivos sobre uma atualização de software ou firmware. Você pode usar o AWS IoT console, o Operações de API de gerenciamento e controle de trabalhos AWS Command Line Interface, o ou o AWS SDKspara criar e gerenciar trabalhos.

Assinatura de código para trabalhos

Ao enviar código para dispositivos, para que os dispositivos detectem se o código foi modificado em trânsito, recomendamos que você assine o arquivo de código usando a AWS CLI. Para obter instruções, consulte Criar e gerenciar trabalhos usando a AWS CLI.

Para obter mais informações, consulte Para que serve a assinatura de código AWS IoT? .

Documento de trabalho

Antes de criar um trabalho, você deve criar um documento de trabalho. Se você estiver usando a assinatura de código para AWS IoT, deverá carregar seu documento de trabalho em um bucket Amazon S3 versionado. Para mais informações sobre a criação de um bucket do Amazon S3 e o upload de um arquivo para ele, consulte Conceitos básicos do Amazon Simple Storage Service, no Guia de conceitos básicos do Amazon S3.

> **Tip**
>
> Para exemplos de documentos de trabalho, consulte o exemplo de jobs-agent.js no AWS IoT SDK para JavaScript.

Pré-assinado URLs

Seu documento de trabalho pode conter um URL pré-assinado do Amazon S3 que aponta para o arquivo de código (ou outro arquivo). Os Amazon URLs S3 pré-assinados são válidos somente por um período limitado de tempo e são gerados quando um dispositivo solicita um documento de trabalho. Posto que o URL pré-assinado não é criado junto com o documento de trabalho, use um URL de espaço reservado no seu documento de trabalho. Um URL de espaço reservado tem a seguinte aparência:

```
${aws:iot:s3-presigned-url-v2:https://
s3.region.amazonaws.com/<bucket>/<code file>}
```

onde:

- *bucket*é o bucket do Amazon S3 que contém o arquivo de código.
- *code file*é a chave Amazon S3 do arquivo de código.

Quando um dispositivo solicita o documento de trabalho, AWS IoT gera o URL pré-assinado e substitui o URL do espaço reservado pelo URL pré-assinado. Seu documento de trabalho é, então, enviado para o dispositivo.

Perfil do IAM para conceder permissão para baixar arquivos do S3

Ao criar um trabalho que usa o Amazon URLs S3 pré-assinado, você deve fornecer uma função do IAM. O perfil deve conceder permissão para fazer download de arquivos do bucket do Amazon S3, onde os dados ou atualizações são armazenados. A função também deve conceder permissão para a AWS IoT assumir a função.

Você pode especificar um limite de tempo opcional para o o URL pré-assinado. Para obter mais informações, consulte CreateJob.

Conceda permissão a AWS IoT Jobs para assumir sua função

1. Vá para o hub Perfis do console do IAM e escolha seu perfil.
2. Na guia Relações de Confiança, escolha Editar Relação de Confiança e substitua o documento de política pelo seguinte JSON. Selecione Atualizar política de confiança.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iot.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

3. Para proteger do problema de confused deputy, adicione as chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount` à política.

> **Important**
>
> Seu `aws:SourceArn` deve estar em conformidade com o formato: `arn:aws:iot:region:account-id:*`. Certifique-se de que *`region`* corresponda à sua AWS IoT região e *`account-id`* corresponda ao ID da sua conta de cliente. Para obter mais informações, consulte Prevenção de confused deputy entre serviços.

```
{
  "Effect": "Allow",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service":
          "iot.amazonaws.com"
       },
      "Action": "sts:AssumeRole",
      "Condition": {
         "StringEquals": {
            "aws:SourceAccount": "123456789012"
         },
         "ArnLike": {
              "aws:SourceArn": "arn:aws:iot:*:123456789012:job/*"
         }
       }
     }
   ]
}
```

4. Se seu trabalho usa um documento de trabalho que é um objeto do Amazon S3, escolha Permissões e use o seguinte JSON. Isso adiciona uma política que concede permissão para baixar arquivos do seu bucket do Amazon S3:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::your_S3_bucket/*"
        }
    ]
}
```

URL pré-assinada para upload de arquivo

Se seus dispositivos precisarem fazer upload de arquivos para um bucket do Amazon S3 durante a implantação de um trabalho, você poderá incluir o seguinte espaço reservado de URL pré-assinado em seu documento de trabalho:

```
${aws:iot:s3-presigned-url-upload:https://s3.region.amazonaws.com/<bucket>/<key>}
```

Você pode usar no máximo duas de cada uma das palavras-chave `${thingName}`, `${jobId}` e `${executionNumber}` como reservadas dentro do atributo `key` no URL do espaço reservado para upload de arquivo localizado em seu documento de trabalho. O espaço reservado local que representa essas palavras-chave reservadas no atributo `key` será analisado e substituído quando a execução do trabalho for criada. Usar um espaço reservado local com palavras-chave reservadas específicas para cada dispositivo garante que cada arquivo carregado de um dispositivo seja específico desse dispositivo e não seja substituído por um arquivo similar carregado de outro dispositivo destinado à mesma implantação de trabalho. Para obter informações sobre como solucionar problemas de espaços reservados locais em um espaço reservado de URL pré-assinado para carregar arquivos durante a implantação de um trabalho, consulte Mensagens de erro de solução de problemas geral.

> **Note**
>
> O nome do bucket do Amazon S3 não pode conter o espaço reservado local que representa as palavras-chave reservadas para o arquivo carregado. O espaço reservado local deve estar localizado no atributo `key`.

Esse espaço reservado de URL pré-assinado será convertido em um URL de upload pré-assinado do Amazon S3 em seu documento de trabalho quando um dispositivo o receber. Seus dispositivos usarão isso para fazer upload de arquivos em bucket do Amazon S3 de destino.

> **Note**
>
> Quando o bucket e a chave do Amazon S3 não forem fornecidos na URL de espaço reservado acima, o AWS IoT Jobs gerará automaticamente uma chave para cada dispositivo usando até duas de cada `${thingName}`, `${jobId}` e `${executionNumber}`.

URL pré-assinada usando o versionamento do Amazon S3

Proteger a integridade de um arquivo armazenado em um bucket do Amazon S3 é fundamental para garantir implantações seguras de trabalhos usando esse arquivo em sua frota de dispositivos. Com o uso do versionamento do Amazon S3, você pode adicionar um identificador de versão para cada variante do arquivo armazenado em seu bucket do Amazon S3 para rastrear cada versão do arquivo. Isso fornece uma visão sobre qual versão do arquivo é implantada em sua frota de dispositivos usando o AWS IoT Jobs. Para obter mais informações sobre os buckets do Amazon S3 usando versionamento, consulte Usar versionamento em buckets do Amazon S3.

Se o arquivo estiver armazenado no Amazon S3 e o documento de trabalho contiver um espaço reservado de URL pré-assinado, AWS IoT Jobs gerará uma URL pré-assinada no documento de trabalho usando o bucket do Amazon S3, a chave do bucket e a versão do arquivo armazenado no bucket do Amazon S3. Esse URL pré-assinado gerado no documento do trabalho substituirá o espaço reservado do URL pré-assinado originalmente no documento do trabalho. Se você atualizar o arquivo armazenado em seu bucket do Amazon S3, uma nova versão do arquivo e `versionId` subsequentes serão criados para sinalizar as atualizações feitas e possibilitar que esse arquivo específico seja focado em futuras implantações de trabalhos.

Consulte os exemplos a seguir para ver antes e durante o Amazon S3 pré-assinado URLs em seu documento de trabalho usando o: `versionId`

Espaço reservado para URL pré-assinado do Amazon S3 (antes da implantação do trabalho)

```
//Virtual-hosted style URL
${aws:iot:s3-presigned-url-v2:https://bucket-name.s3.region-code.amazonaws.com/key-name%3FversionId%3Dversion-id}

//Path-style URL
${aws:iot:s3-presigned-url-v2:https://s3.region-code.amazonaws.com/bucket-name/key-name%3FversionId%3Dversion-id}
```

URL pré-assinado do Amazon S3 (durante a implantação do trabalho)

```
//Virtual-hosted style URL
${aws:iot:s3-presigned-url-v2:https://sample-bucket-name.s3.us-
west-2.amazonaws.com/sample-code-file.png%3FversionId%3Dversion1}

//Path-style
${aws:iot:s3-presigned-url-v2:https://s3.us-west-2.amazonaws.com/sample-bucket-
name/sample-code-file.png%3FversionId%3Dversion1}
```

Para obter mais informações sobre objetos hospedados virtualmente e no estilo de caminho do Amazon S3 URLs, consulte solicitações e solicitações no estilo Path. Virtual-hosted-style

> **Note**
>
> Se você quiser anexar a uma URL pré-assinada `versionId` do Amazon S3, ela deve estar em conformidade com o suporte à codificação de URL AWS SDK for Java 2.x. Para obter mais informações, consulte Alterações na análise do Amazon URIs S3 da versão 1 para a versão 2.

Diferenças de versões de espaços reservados de URLs pré-assinados do Amazon S3

A lista a seguir descreve as diferenças entre espaços reservados de URLs pré-assinados do Amazon S3 `${aws:iot:s3-presigned-url-v1` (versão 1) e `${aws:iot:s3-presigned-url-v2` (versão 2):

- O espaço reservado `${aws:iot:s3-presigned-url-v1` de URL pré-assinado do Amazon S3 não comporta `version-id`.
- O espaço reservado `${aws:iot:s3-presigned-url-v1` de URL pré-assinado do Amazon S3 recebe o URL do Amazon S3 como não codificado. O espaço reservado `${aws:iot:s3-presigned-url-v2` de URL pré-assinado do Amazon S3 exige que o URL do Amazon S3 seja codificado para estar em conformidade com o padrão do SDK do Amazon S3.

Tópicos

- Crie e gerencie trabalhos usando o Console de gerenciamento da AWS
- Crie e gerencie trabalhos usando o AWS CLI

Crie e gerencie trabalhos usando o Console de gerenciamento da AWS

Esta seção descreve como você pode criar e gerenciar trabalhos a partir do AWS IoT console. Depois de criar um trabalho, você pode ver as informações sobre o trabalho na página de detalhes e gerenciar o trabalho.

> **Note**
>
> Se você quiser realizar a assinatura de código para AWS IoT trabalhos, use AWS CLI o. Para obter mais informações, consulte Criar e gerenciar trabalhos usando a AWS CLI.

Tópicos

- Crie e gerencie trabalhos usando o Console de gerenciamento da AWS
- Visualize e gerencie trabalhos usando o Console de gerenciamento da AWS

Crie e gerencie trabalhos usando o Console de gerenciamento da AWS

Para criar um trabalho, faça login no AWS IoT console e acesse o hub de trabalhos na seção Ações remotas. Em seguida, execute as seguintes etapas.

1. Na página Trabalhos, na caixa de diálogo Trabalhos, escolha Criar trabalho.
2. Dependendo do dispositivo que você está usando, você pode criar um trabalho personalizado, um trabalho de atualização do FreeRTOS OTA ou um trabalho. AWS IoT Greengrass Para este exemplo, selecione Criar trabalho personalizado. Escolha Próximo.
3. Na página Propriedades do trabalho personalizado, na caixa de diálogo Propriedades do trabalho, insira suas informações nos seguintes campos:

   - Nome: insira um nome de trabalho alfanumérico exclusivo.
   - Descrição - opcional: insira uma descrição opcional sobre seu trabalho.
   - Etiquetas - opcionais:

   > **Note**
   >
   > Recomendamos que você não use informações de identificação pessoal em seu trabalho IDs e descrição.

Escolha Próximo.

4. Na página Configuração do arquivo na caixa de diálogo Destinos do trabalho, selecione as Coisas ou Grupos de coisas em que você deseja executar esse trabalho.

   Na caixa de diálogo Documento de trabalho, selecione uma das seguintes opções:

   - Do arquivo: um arquivo de trabalho do JSON que você carregou anteriormente para um bucket do Amazon S3
     - Assinatura de código

       No documento de trabalho localizado em sua URL do Amazon S3, `${aws:iot:code-sign-signature:s3://region.bucket/code-file@code-file-version-id}` é necessário como espaço reservado até que seja substituído pelo caminho do arquivo de código assinado usando seu Perfil de assinatura de código. O novo arquivo de código assinado aparecerá inicialmente em uma pasta `SignedImages` no seu bucket de origem do Amazon S3. Um novo documento de trabalho contendo um prefixo `Codesigned_` será criado com o caminho do arquivo de código assinado substituindo o espaço reservado para assinatura de código e colocado em sua URL do Amazon S3 para criar um novo trabalho.
     - Recurso de pré-assinatura URLs

       No menu suspenso Função de pré-assinatura, escolha a função do IAM que você criou em URLsPresigned. Usar `${aws:iot:s3-presigned-url:` para pré-assinar URLs objetos localizados no Amazon S3 é a melhor prática de segurança para dispositivos que baixam objetos do Amazon S3.

       Se você quiser usar pré-assinado URLs para um espaço reservado para assinatura de código, use o seguinte modelo de exemplo:

       ```
       ${aws:iot:s3-presigned-url:${aws:iot:code-sign-signature:<S3 URL>}
       ```

   - Do modelo: um modelo de trabalho contendo um documento de trabalho e configurações de trabalho. O modelo de trabalho pode ser um modelo de trabalho personalizado que você criou ou um modelo AWS gerenciado.

     Se você estiver criando um trabalho para realizar ações remotas usadas com frequência, como reinicializar seu dispositivo, poderá usar um modelo AWS gerenciado. Esses modelos já foram pré-configurados para uso. Para obter mais informações, consulte Crie um modelo

de trabalho personalizado e Crie modelos de trabalho personalizados a partir de modelos gerenciados.

5. Na página Configuração do trabalho, na caixa de diálogo Configuração do trabalho, selecione um dos seguintes tipos de trabalho:

   - Trabalho de snapshot: um trabalho de instantâneo é concluído quando termina sua execução nos dispositivos e grupos de destino.
   - Trabalho contínuo: um trabalho contínuo se aplica a grupos de objetos e é executado em qualquer dispositivo que você adicione posteriormente a um grupo-alvo especificado.

6. Na caixa de diálogo Configurações adicionais - opcionais, revise as seguintes configurações de Trabalho opcionais e faça suas seleções:

   - Configuração de distribuição
   - Configuração de agendamento
   - Configuração de tempo limite de execução de trabalhos
   - Configuração de repetição de execuções de trabalhos - nova
   - Configuração de anulação

   Consulte as seções a seguir para obter informações adicionais sobre as configurações de Trabalho:

   - Configurações de implantação, agendamento e anulação de trabalhos
   - Configurações de novas tentativas e de tempo limite de execuções de trabalhos

   Revise todas as suas seleções de trabalho e, em seguida, escolha Enviar para criar seu trabalho.

Visualize e gerencie trabalhos usando o Console de gerenciamento da AWS

Depois de criar a tarefa, o console gera uma assinatura JSON e a coloca no seu documento de trabalho. Você pode usar o console da AWS IoT para visualizar o status, cancelar ou excluir um trabalho.

Se você escolher o trabalho que criou, você poderá encontrar:

- Detalhes gerais do trabalho, como nome, descrição, tipo, hora em que foi criado, última atualização e hora de início estimada.
- Todas as configurações de trabalho que você especificou e seu status.
- O documento de trabalho.
- As execuções do trabalho e todas as tags opcionais que você especificou.

Para gerenciar trabalhos, acesse o Hub de trabalhos do console e escolha se você quer editar, excluir ou cancelar o trabalho.

Crie e gerencie trabalhos usando o AWS CLI

Esta seção descreve como criar e gerenciar trabalhos.

Criar trabalhos

Para criar um AWS IoT trabalho, use o CreateJob comando. O trabalho é enfileirado para execução nos destinos (objetos ou grupos de objetos) que você especifica. Para criar um AWS IoT trabalho, você precisa de um documento de trabalho que possa ser incluído no corpo da solicitação ou como um link para um documento do Amazon S3. Se o trabalho incluir o download de arquivos usando o Amazon URLs S3 pré-assinado, você precisará de uma função do IAM Amazon Resource Name (ARN) que tenha permissão para baixar o arquivo e conceda permissão ao serviço Jobs para assumir AWS IoT a função.

Para obter mais informações sobre a sintaxe ao inserir a data e a hora usando um comando da API ou o AWS CLI, consulte Timestamp.

Assinatura de código com trabalhos

Se você estiver usando a assinatura de código para AWS IoT, deverá iniciar um trabalho de assinatura de código e incluir a saída em seu documento de trabalho. Isso substituirá o espaço reservado para assinatura de código em seu documento de trabalho, que é necessário como espaço reservado até que seja substituído pelo caminho do arquivo de código assinado usando seu perfil de assinatura de código. O espaço reservado para assinatura de código será exibido da seguinte forma:

```
${aws:iot:code-sign-signature:s3://region.bucket/code-file@code-file-version-id}
```

Use o start-signing-jobcomando para criar um trabalho de assinatura de código. `start-signing-job`retorna uma ID de trabalho. Use o comando describe-signing-job para obter o local do Amazon

S3 onde a assinatura é armazenada. Em seguida, você pode baixar a assinatura do Amazon S3. Para obter mais informações sobre trabalhos de assinatura de código, consulte Assinatura de código do AWS IoT.

Seu documento de trabalho deve conter um espaço reservado de pré-assinatura do URL para o arquivo de código e a saída da assinatura JSON inserida em um bucket do Amazon S3 usando o comando start-signing-job:

```
{
    "presign": "${aws:iot:s3-presigned-url:https://s3.region.amazonaws.com/bucket/
image}",
}
```

Criar um trabalho com um documento de trabalho

O comando a seguir mostra como criar um trabalho usando um documento de trabalho (*job-document.json*) armazenado em um bucket do Amazon S3 (*jobBucket*) e uma função com permissão para baixar arquivos do Amazon *S3DownloadRole* S3 ().

```
aws iot create-job  \
      --job-id 010  \
      --targets arn:aws:iot:us-east-1:123456789012:thing/thingOne  \
      --document-source https://s3.amazonaws.com/amzn-s3-demo-bucket/job-document.json
 \
      --timeout-config inProgressTimeoutInMinutes=100 \
      --job-executions-rollout-config "{ \"exponentialRate\": { \"baseRatePerMinute
\": 50, \"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings
\": 1000, \"numberOfSucceededThings\": 1000}}, \"maximumPerMinute\": 1000}" \
      --abort-config "{ \"criteriaList\": [ { \"action\": \"CANCEL\", \"failureType
\": \"FAILED\", \"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20},
 { \"action\": \"CANCEL\", \"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings
\": 200, \"thresholdPercentage\": 50}]}" \
      --presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/
S3DownloadRole\", \"expiresInSec\":3600}"
```

O trabalho continua*thingOne*.

O parâmetro `timeout-config` opcional especifica o tempo que cada dispositivo tem para concluir a execução do trabalho. O temporizador é iniciado quando o status da execução do trabalho é definido como `IN_PROGRESS`. Se o status da execução do trabalho não estiver definido como outro estado terminal antes que o tempo expire, ele será definido como `TIMED_OUT`.

O temporizador em andamento não pode ser atualizado e é aplicado a todas as execuções do trabalho. Sempre que a execução de uma tarefa permanece no `IN_PROGRESS` estado por mais tempo do que esse intervalo, ela falha e muda para o `TIMED_OUT` status do terminal. AWS IoT também publica uma notificação MQTT.

Para obter mais informações sobre como criar configurações para distribuir e anular trabalhos, consulte Configuração para Distribuir e Anular Trabalhos.

> **Note**
>
> Os documentos de trabalho que são especificados como arquivos do Amazon S3 são recuperados no momento em que você cria o trabalho. Se você alterar o conteúdo do arquivo do Amazon S3 que você usou como a origem de seu documento de trabalho depois de ter criado o documento de trabalho, o que é enviado para os destinos do trabalho não é alterado.

Atualizar um trabalho

Para atualizar um trabalho, use o comando UpdateJob. Você pode atualizar os campos `description`, `presignedUrlConfig`, `jobExecutionsRolloutConfig`, `abortConfig` e `timeoutConfig` de um trabalho.

```
aws iot update-job  \
  --job-id 010  \
  --description "updated description" \
  --timeout-config inProgressTimeoutInMinutes=100 \
  --job-executions-rollout-config "{ \"exponentialRate\": { \"baseRatePerMinute\": 50,
 \"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings\": 1000,
 \"numberOfSucceededThings\": 1000}, \"maximumPerMinute\": 1000}}" \
  --abort-config "{ \"criteriaList\": [ { \"action\": \"CANCEL\", \"failureType
\": \"FAILED\", \"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20},
 { \"action\": \"CANCEL\", \"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings
\": 200, \"thresholdPercentage\": 50}]}" \
  --presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/
S3DownloadRole\", \"expiresInSec\":3600}"
```

Para obter mais informações, consulte Configuração de distribuição e anulação de trabalhos.

Cancelar um trabalho

Para cancelar um trabalho, use o comando CancelJob. O cancelamento de um trabalho AWS IoT impede a implementação de novas execuções de trabalho para o trabalho. Também cancela qualquer execução de trabalho que esteja em um `QUEUED` estado. AWS IoT mantém todas as execuções de trabalho em um estado terminal intocadas porque o dispositivo já concluiu o trabalho. Se o status de uma execução de trabalho for `IN_PROGRESS`, ele também permanecerá inalterado, a menos que você use o parâmetro opcional `--force`.

O comando a seguir mostra como cancelar um trabalho com ID 010.

```
aws iot cancel-job --job-id 010
```

O comando exibe a seguinte saída:

```
{
    "jobArn": "string",
    "jobId": "string",
    "description": "string"
}
```

Ao cancelar um trabalho, as execuções de trabalho em estado `QUEUED` serão canceladas. As execuções de trabalho em estado `IN_PROGRESS` serão canceladas, mas somente se você especificar o parâmetro opcional `--force`. As execuções de trabalhos em estado terminal não são canceladas.

> **Warning**
>
> O cancelamento de um trabalho no estado `IN_PROGRESS` (por meio da definição do parâmetro `--force`) cancelará qualquer execução de trabalho em andamento e impedirá que o dispositivo que está executando o trabalho atualize o status da execução do trabalho. Tenha cuidado e certifique-se de que todo dispositivo que esteja executando um trabalho cancelado possa ser recuperado para um estado válido.

O status de um trabalho cancelado ou de uma de suas execuções de trabalho é eventualmente consistente. AWS IoT interrompe o agendamento de novas execuções de `QUEUED` tarefas e

execuções de tarefas para esses dispositivos o mais rápido possível. A alteração do status de uma execução de trabalho para `CANCELED` pode levar algum tempo, dependendo do número de dispositivos e de outros fatores.

Se um trabalho for cancelado porque atende aos critérios definidos por um objeto `AbortConfig`, o serviço adicionará valores de preenchimento automático para os campos `comment` e `reasonCode`. É possível criar seus próprios valores de `reasonCode` se o cancelamento de trabalho for acionado pelo usuário.

Cancelar uma execução de trabalho

Use o comando CancelJobExecution para cancelar uma execução de trabalho em um dispositivo. Ele cancelará uma execução de trabalho que esteja em estado `QUEUED`. Se desejar cancelar uma execução de trabalho em andamento, deverá usar o parâmetro `--force`.

O comando a seguir mostra como cancelar a execução de trabalho 010 em execução em `myThing`.

```
aws iot cancel-job-execution --job-id 010 --thing-name myThing
```

O comando não exibe nenhuma saída.

Uma execução de trabalho que esteja em estado `QUEUED` será cancelada. Uma execução de trabalho em estado `IN_PROGRESS` será cancelada, mas somente se você especificar o parâmetro opcional `--force`. As execuções de trabalho em estado terminal não podem ser canceladas.

> **Warning**
>
> Cancelar uma execução de trabalho em estado `IN_PROGRESS` impedirá que o dispositivo atualize o status da execução do trabalho. Tenha cuidado e certifique-se de que o dispositivo possa ser recuperado para um estado válido.

Se a execução do trabalho estiver em estado terminal ou em estado `IN_PROGRESS` e o parâmetro `--force` não estiver definido como `true`, esse comando gerará `InvalidStateTransitionException`.

O status de uma execução de trabalho cancelada é eventualmente consistente. A alteração do status de uma execução de trabalho para `CANCELED` pode levar algum tempo, dependendo de vários fatores.

Excluir um trabalho

Use o comando DeleteJob para excluir um trabalho e suas execuções de trabalho. Por padrão, você só pode excluir um trabalho em estado terminal (`SUCCEEDED` ou `CANCELED`). Caso contrário, ocorrerá uma exceção. Você pode excluir um trabalho em estado `IN_PROGRESS`, mas somente se o parâmetro `force` for definido como `true`.

Para excluir um trabalho, execute o comando a seguir:

```
aws iot delete-job --job-id 010 --force|--no-force
```

O comando não exibe nenhuma saída.

> **Warning**
>
> Ao excluir um trabalho no estado `IN_PROGRESS`, o dispositivo que está executando o trabalho não pode acessar informações do trabalho nem atualizar o status da execução do trabalho. Tenha cuidado e certifique-se de que todo dispositivo executando um trabalho que tenha sido excluído possa ser recuperado para um estado válido.

A exclusão de um trabalho pode levar algum tempo, dependendo do número de execuções criadas para o trabalho e de outros fatores. Enquanto o trabalho está sendo excluído, `DELETION_IN_PROGRESS` aparece como status do trabalho. Ocorrerá erro se você tentar excluir ou cancelar um trabalho cujo status já seja `DELETION_IN_PROGRESS`.

Somente 10 trabalhos podem ter, simultaneamente, o status `DELETION_IN_PROGRESS`. Caso contrário, um `LimitExceededException` ocorre.

Obter um documento de trabalho

Use o comando GetJobDocument para recuperar um documento de trabalho para um trabalho. Um documento de trabalho é uma descrição das operações remotas a serem realizadas pelos dispositivos.

Execute o seguinte comando para obter um documento de trabalho:

```
aws iot get-job-document --job-id 010
```

O comando retorna o documento de trabalho para o trabalho especificado:

```
{
    "document": "{\n\t\"operation\":\"install\",\n\t\"url\":\"http://amazon.com/
firmWareUpate-01\",\n\t\"data\":\"${aws:iot:s3-presigned-url:https://s3.amazonaws.com/
amzn-s3-demo-bucket/datafile}\"\n}"
}
```

> **Note**
>
> Quando você usa esse comando para recuperar um documento de trabalho, os espaços reservados URLs não são substituídos pelo Amazon S3 pré-assinado. URLs Quando um dispositivo chama a operação da GetPendingJobExecutionsAPI, o espaço reservado URLs é substituído pelo Amazon URLs S3 pré-assinado no documento de trabalho.

Listar tarefas

Para obter uma lista de todos os trabalhos em seu Conta da AWS, use o ListJobs comando. Os dados do trabalho e os dados de execução do trabalho são retidos por um tempo limitado. Execute o comando a seguir para listar todos os trabalhos em seu Conta da AWS:

```
aws iot list-jobs
```

O comando retorna todos os trabalhos em sua conta, classificados pelo status do trabalho:

```
{
    "jobs": [
        {
            "status": "IN_PROGRESS",
            "lastUpdatedAt": 1486687079.743,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/013",
            "createdAt": 1486687079.743,
            "targetSelection": "SNAPSHOT",
            "jobId": "013"
        },
        {
            "status": "SUCCEEDED",
            "lastUpdatedAt": 1486685868.444,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/012",
            "createdAt": 1486685868.444,
            "completedAt": 148668789.690,
```

```
            "targetSelection": "SNAPSHOT",
            "jobId": "012"
        },
        {
            "status": "CANCELED",
            "lastUpdatedAt": 1486678850.575,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/011",
            "createdAt": 1486678850.575,
            "targetSelection": "SNAPSHOT",
            "jobId": "011"
        }
    ]
}
```

Descrever um trabalho

Execute o comando DescribeJob para obter o status de um trabalho. O comando a seguir mostra como descrever um trabalho:

```
$ aws iot describe-job --job-id 010
```

O comando retorna o status do trabalho especificado. Por exemplo:

```
{
    "documentSource": "https://s3.amazonaws.com/amzn-s3-demo-bucket/job-
document.json",
    "job": {
        "status": "IN_PROGRESS",
        "jobArn": "arn:aws:iot:us-east-1:123456789012:job/010",
        "targets": [
            "arn:aws:iot:us-east-1:123456789012:thing/myThing"
        ],
        "jobProcessDetails": {
            "numberOfCanceledThings": 0,
            "numberOfFailedThings": 0,
            "numberOfInProgressThings": 0,
            "numberOfQueuedThings": 0,
            "numberOfRejectedThings": 0,
            "numberOfRemovedThings": 0,
            "numberOfSucceededThings": 0,
            "numberOfTimedOutThings": 0,
            "processingTargets": [
                arn:aws:iot:us-east-1:123456789012:thing/thingOne,
```

```
                arn:aws:iot:us-east-1:123456789012:thinggroup/thinggroupOne,
                arn:aws:iot:us-east-1:123456789012:thing/thingTwo,
                arn:aws:iot:us-east-1:123456789012:thinggroup/thinggroupTwo
            ]
        },
        "presignedUrlConfig": {
            "expiresInSec": 60,
            "roleArn": "arn:aws:iam::123456789012:role/S3DownloadRole"
        },
        "jobId": "010",
        "lastUpdatedAt": 1486593195.006,
        "createdAt": 1486593195.006,
        "targetSelection": "SNAPSHOT",
        "jobExecutionsRolloutConfig": {
            "exponentialRate": {
                "baseRatePerMinute": integer,
                "incrementFactor": integer,
                "rateIncreaseCriteria": {
                    "numberOfNotifiedThings": integer, // Set one or the other
                    "numberOfSucceededThings": integer // of these two values.
                },
            "maximumPerMinute": integer
         }
        },
        "abortConfig": {
            "criteriaList": [
                {
                    "action": "string",
                    "failureType": "string",
                    "minNumberOfExecutedThings": integer,
                    "thresholdPercentage": integer
                }
            ]
        },
        "timeoutConfig": {
           "inProgressTimeoutInMinutes": number
          }
    }
}
```

Listar execuções para um trabalho

Um trabalho em execução em um dispositivo específico é representado por um objeto de execução de trabalho. Execute o comando ListJobExecutionsForJob para listar todas as execuções de um trabalho. O exemplo a seguir mostra como listar as execuções de um trabalho:

```
aws iot list-job-executions-for-job --job-id 010
```

O comando retorna uma lista de execuções do trabalho:

```
{
    "executionSummaries": [
    {
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingOne",
        "jobExecutionSummary": {
            "status": "QUEUED",
            "lastUpdatedAt": 1486593196.378,
            "queuedAt": 1486593196.378,
            "executionNumber": 1234567890
        }
    },
    {
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingTwo",
        "jobExecutionSummary": {
            "status": "IN_PROGRESS",
            "lastUpdatedAt": 1486593345.659,
            "queuedAt": 1486593196.378,
            "startedAt": 1486593345.659,
            "executionNumber": 4567890123
        }
    }
    ]
}
```

Listar execuções de trabalho para um objeto

Execute o comando ListJobExecutionsForThing para listar todas as execuções de trabalho em execução em um objeto. O exemplo a seguir mostra como listar execuções de trabalho para um objeto:

```
aws iot list-job-executions-for-thing --thing-name thingOne
```

O comando retorna uma lista de execuções de trabalho que estão em execução ou que executaram no objeto especificada:


```
{
    "executionSummaries": [
    {
        "jobExecutionSummary": {
            "status": "QUEUED",
            "lastUpdatedAt": 1486687082.071,
            "queuedAt": 1486687082.071,
            "executionNumber": 9876543210
        },
        "jobId": "013"
    },
    {
        "jobExecutionSummary": {
            "status": "IN_PROGRESS",
            "startAt": 1486685870.729,
            "lastUpdatedAt": 1486685870.729,
            "queuedAt": 1486685870.729,
            "executionNumber": 1357924680
        },
        "jobId": "012"
    },
    {
        "jobExecutionSummary": {
            "status": "SUCCEEDED",
            "startAt": 1486678853.415,
            "lastUpdatedAt": 1486678853.415,
            "queuedAt": 1486678853.415,
            "executionNumber": 4357680912
        },
        "jobId": "011"
    },
    {
        "jobExecutionSummary": {
            "status": "CANCELED",
            "startAt": 1486593196.378,
            "lastUpdatedAt": 1486593196.378,
            "queuedAt": 1486593196.378,
            "executionNumber": 2143174250
        },
        "jobId": "010"
```

```
    }
    ]
}
```

Descrever a execução do trabalho

Execute o comando DescribeJobExecution para obter o status de execução de um trabalho. Você deve especificar um ID de trabalho e um nome de objeto e, opcionalmente, um número de execução para identificar a execução do trabalho. O comando a seguir mostra como descrever a execução de um trabalho:

```
aws iot describe-job-execution --job-id 017 --thing-name thingOne
```

O comando retorna a `JobExecution`. Por exemplo:

```
{
    "execution": {
        "jobId": "017",
        "executionNumber": 4516820379,
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingOne",
        "versionNumber": 123,
        "createdAt": 1489084805.285,
        "lastUpdatedAt": 1489086279.937,
        "startedAt": 1489086279.937,
        "status": "IN_PROGRESS",
        "approximateSecondsBeforeTimedOut": 100,
        "statusDetails": {
            "status": "IN_PROGRESS",
            "detailsMap": {
                "percentComplete": "10"
            }
        }
    }
}
```

Excluir uma execução de trabalho

Execute o comando DeleteJobExecution para excluir a execução de um trabalho. Você deve especificar um ID de trabalho, um nome de objeto e um número de execução para identificar a execução do trabalho. O comando a seguir mostra como excluir a execução de um trabalho:

```
aws iot delete-job-execution --job-id 017 --thing-name thingOne --execution-number
 1234567890 --force|--no-force
```

O comando não exibe nenhuma saída.

Por padrão, o status de execução do trabalho deve ser `QUEUED` ou em estado terminal (`SUCCEEDED`, `FAILED`, `REJECTED`, `TIMED_OUT`, `REMOVED` ou `CANCELED`). Caso contrário, ocorrerá um erro. Para excluir uma execução de trabalho com status `IN_PROGRESS`, você pode definir o parâmetro `force` como `true`.

> **Warning**
>
> Ao excluir uma execução de trabalho com status `IN_PROGRESS`, o dispositivo que está executando o trabalho não pode acessar informações do trabalho nem atualizar o status da execução do trabalho. Tenha cuidado e certifique-se de que o dispositivo possa ser recuperado para um estado válido.

Modelos de trabalho

Use modelos de trabalho para pré-configurar trabalhos que você pode implantar em vários conjuntos de dispositivos de destino. Para implantar ações remotas executadas com frequência em seus dispositivos, como reinicializar ou instalar um aplicativo, você pode usar modelos para definir configurações padrão. Para realizar operações como a implantação de patches de segurança e correções de bugs, você pode criar modelos a partir de trabalhos existentes.

Ao criar um modelo de trabalho, especifique as seguintes configurações e recursos adicionais.

- Propriedades do trabalho
- Documentos e metas de trabalho
- Critérios de implantação, agendamento e cancelamento
- Critérios de tempo limite e repetição

Modelos personalizados e AWS gerenciados

Dependendo da ação remota que você deseja realizar, você pode criar um modelo de trabalho personalizado ou usar um modelo AWS gerenciado. Use modelos de trabalho personalizados

para fornecer seu próprio documento de trabalho personalizado e criar trabalhos reutilizáveis para implantar em seus dispositivos. AWS modelos gerenciados são modelos de trabalho fornecidos pelo AWS IoT Jobs para ações comumente executadas. Esses modelos têm um documento de trabalho predefinido para algumas ações remotas, para que você não precise criar seu próprio documento de trabalho. Os modelos gerenciados ajudam você a criar trabalhos reutilizáveis para um lançamento mais rápido em seus dispositivos.

Tópicos

- Use modelos AWS gerenciados para implantar operações remotas comuns
- Crie modelos de trabalho personalizados

Use modelos AWS gerenciados para implantar operações remotas comuns

AWS modelos gerenciados são modelos de trabalho fornecidos pela AWS. Eles são usados para ações remotas executadas com frequência, como reinicializar, baixar um arquivo ou instalar um aplicativo em seus dispositivos. Esses modelos têm um documento de trabalho predefinido para cada ação remota, para que você não precise criar seu próprio documento de trabalho.

Você pode escolher entre um conjunto de configurações predefinidas e criar trabalhos usando esses modelos sem escrever nenhum código adicional. Usando modelos gerenciados, você pode visualizar o documento de trabalho implantado em suas frotas. Você pode criar um trabalho usando esses modelos e criar um modelo de trabalho personalizado que pode ser reutilizado em suas operações remotas.

O que os modelos gerenciados contêm?

Cada modelo AWS gerenciado contém:

- O ambiente para executar os comandos no documento de trabalho.
- Um documento de trabalho que especifica o nome da operação e seus parâmetros. Por exemplo, se você usar um modelo de Baixar arquivo, o nome da operação será Baixar arquivo e os parâmetros poderão ser:
  - O URL do arquivo que você deseja baixar no seu dispositivo. Isso pode ser um recurso da Internet ou um URL público ou pré-assinado do Amazon Simple Storage Service (Amazon S3).
  - Um caminho de arquivo local no dispositivo para armazenar o arquivo baixado.

  Para obter mais informações sobre os documentos de trabalho e seus respectivos parâmetros de configuração, consulte Ações remotas e documentos de trabalho do modelo gerenciado.

Pré-requisitos

Para que seus dispositivos executem as ações remotas especificadas pelo modelo de documento de trabalho gerenciado, você deve:

- Instalar o software específico em seu dispositivo

  Use o software e os gerenciadores de tarefas do seu próprio AWS IoT dispositivo ou o Device Client. Dependendo do seu caso comercial, você também pode executar os dois para que desempenhem funções diferentes.

  - Use o software e os gerenciadores de trabalho do seu próprio dispositivo

    Você pode escrever seu próprio código para os dispositivos usando o AWS IoT Device SDK e sua biblioteca de gerenciadores que oferecem suporte às operações remotas. Para implantar e executar trabalhos, verifique se as bibliotecas do atendente do dispositivo foram instaladas corretamente e estão sendo executadas nos dispositivos.

    Você também pode optar por utilizar seus próprios gerenciadores compatíveis com as operações remotas. Para obter mais informações, consulte Exemplos de manipuladores de tarefas no GitHub repositório do AWS IoT Device Client.

  - Use o cliente AWS IoT do dispositivo

    Ou você pode instalar e executar o AWS IoT Device Client em seus dispositivos porque, por padrão, ele suporta o uso de todos os modelos gerenciados diretamente do console.

    O cliente do dispositivo é um software de código aberto escrito em C++ que você pode compilar e instalar em seus dispositivos de IoT baseados em Linux incorporados. O cliente do dispositivo tem um cliente base e discretos recursos do lado do cliente. O cliente base estabelece conectividade com AWS IoT o protocolo MQTT e pode se conectar com os diferentes recursos do lado do cliente.

    Para realizar operações remotas em seus dispositivos, use o atributo Trabalho do lado do cliente do cliente do dispositivo. Esse atributo contém um analisador para receber o documento do trabalho e os gerenciadores de trabalho que implementam as ações remotas especificadas no documento do trabalho. Para obter mais informações sobre o cliente do dispositivo e seus recursos, consulte Cliente do dispositivo de AWS IoT.

    Ao ser executado em dispositivos, o cliente do dispositivo recebe o documento do trabalho e tem uma implementação específica da plataforma que usa para executar comandos no documento.

Para obter mais informações sobre a configuração do cliente do dispositivo e o uso do atributo dos Trabalhos, consulte os tutoriais de AWS IoT.

- Use um ambiente compatível

  Para cada modelo gerenciado, você encontrará informações sobre o ambiente que você pode usar para executar as ações remotas. Recomendamos que você use o modelo com um ambiente Linux compatível, conforme especificado no modelo. Use o AWS IoT Device Client para executar as ações remotas do modelo gerenciado porque ele suporta microprocessadores comuns e ambientes Linux, como Debian e Ubuntu.

Ações remotas e documentos de trabalho do modelo gerenciado

A seção a seguir lista os diferentes modelos AWS gerenciados para AWS IoT trabalhos e descreve as ações remotas que podem ser executadas nos dispositivos. A seção a seguir contém informações sobre o documento de trabalho e uma descrição dos parâmetros do documento de trabalho para cada ação remota. O software do lado do dispositivo usa o nome do modelo e os parâmetros para realizar a ação remota.

AWS os modelos gerenciados aceitam parâmetros de entrada para os quais você especifica um valor ao criar um trabalho usando o modelo. Todos os modelos gerenciados têm dois parâmetros de entrada opcionais em comum: `runAsUser` e `pathToHandler`. Com exceção do modelo `AWS-Reboot`, os modelos exigem parâmetros de entrada adicionais para os quais você deve especificar um valor ao criar um trabalho usando o modelo. Esses parâmetros de entrada necessários variam dependendo do modelo que você escolher. Por exemplo, se você escolher o modelo `AWS-Download-File`, deverá especificar uma lista de pacotes a serem instalados e uma URL para baixar arquivos.

Especifique um valor para os parâmetros de entrada ao usar o AWS IoT console ou o AWS Command Line Interface (AWS CLI) para criar um trabalho que usa um modelo gerenciado. Ao usar a CLI, forneça esses valores usando o objeto `document-parameters`. Para obter mais informações, consulte DocumentParameters.

> **Note**
>
> Use `document-parameters` somente ao criar trabalhos a partir de modelos gerenciados da AWS . Esse parâmetro não pode ser usado com modelos de trabalho personalizados nem para criar trabalhos a partir deles.

A seguir, é apresentada uma descrição dos parâmetros de entrada opcionais comuns. Você verá uma descrição de outros parâmetros de entrada que cada modelo gerenciado exige na próxima seção.

`runAsUser`

Esse parâmetro especifica se o gerenciador de trabalho deve ser executado como outro usuário. Se não for especificado durante a criação do trabalho, o gerenciador de trabalho será executado como o mesmo usuário do cliente do dispositivo. Ao executar o gerenciador de trabalho como outro usuário, especifique um valor de string que não tenha mais de 256 caracteres.

`pathToHandler`

O caminho para o gerenciador de trabalho em execução no dispositivo. Se não for especificado durante a criação do trabalho, o cliente do dispositivo usa o diretório de trabalho atual.

Veja a seguir as diferentes ações remotas, seus documentos de trabalho e parâmetros que aceitam. Todos esses modelos são compatíveis com o ambiente Linux para executar a operação remota no dispositivo.

AWS–Download–File

Nome do modelo

`AWS-Download-File`

Descrição do modelo

Um modelo gerenciado fornecido pelo AWS para baixar um arquivo.

Parâmetros de entrada

Esse modelo tem os seguintes parâmetros necessários. Você também pode especificar os parâmetros opcionais `runAsUser` e `pathToHandler`.

`downloadUrl`

O URL para fazer download do arquivo. Isso pode ser um recurso da Internet, um objeto no Amazon S3 que pode ser acessado publicamente ou um objeto no Amazon S3 que só pode ser acessado pelo seu dispositivo usando um URL pré-assinado. Para obter mais informações sobre como usar permissões pré-assinadas URLs e conceder permissões, consulte. Pré-assinado URLs

`filePath`

Um caminho de arquivo local que mostra a localização no dispositivo para armazenar o arquivo baixado.

Comportamento do dispositivo

O dispositivo baixa o arquivo do local especificado, verifica se o download foi concluído e o armazena localmente.

Documento de trabalho

Veja a seguir o documento do trabalho e sua versão mais recente. O modelo mostra o caminho para o gerenciador de trabalho e o script de shell, `download-file.sh`, que o gerenciador de trabalho deve executar para baixar o arquivo. Também mostra os parâmetros necessários `downloadUrl` e `filePath`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Download-File",
        "type": "runHandler",
        "input": {
          "handler": "download-file.sh",
          "args": [
            "${aws:iot:parameter:downloadUrl}",
            "${aws:iot:parameter:filePath}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Install–Application

Nome do modelo

`AWS-Install-Application`

Descrição do modelo

Um modelo gerenciado fornecido pela AWS para a instalação de um ou mais aplicativos.

Parâmetros de entrada

Esse modelo tem o seguinte parâmetro necessário, `packages`. Você também pode especificar os parâmetros opcionais `runAsUser` e `pathToHandler`.

`packages`

Uma lista separada por espaços de um ou mais aplicativos a serem instalados.

Comportamento do dispositivo

O dispositivo instala os aplicativos conforme especificado no documento do trabalho.

Documento de trabalho

Veja a seguir o documento do trabalho e sua versão mais recente. O modelo mostra o caminho para o gerenciador de trabalho e o script de shell, `install-packages.sh`, que o gerenciador de trabalho deve executar para baixar o arquivo. Também mostra o parâmetro `packages` necessário.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Install-Application",
        "type": "runHandler",
        "input": {
          "handler": "install-packages.sh",
          "args": [
            "${aws:iot:parameter:packages}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
```

```
}
```

AWS–Reboot

Nome do modelo

`AWS-Reboot`

Descrição do modelo

Um modelo gerenciado fornecido por AWS para reinicializar seu dispositivo.

Parâmetros de entrada

Esse modelo não tem parâmetros necessários. Você pode especificar os parâmetros opcionais `runAsUser` e `pathToHandler`.

Comportamento do dispositivo

O dispositivo é reinicializado com êxito.

Documento de trabalho

Veja a seguir o documento do trabalho e sua versão mais recente. O modelo mostra o caminho para o gerenciador de trabalho e o script de shell, `reboot.sh`, que o gerenciador de trabalho deve executar para reinicializar o dispositivo.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Reboot",
        "type": "runHandler",
        "input": {
          "handler": "reboot.sh",
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
```

```
}
```

AWS–Remove–Application

Nome do modelo

`AWS-Remove-Application`

Descrição do modelo

Um modelo gerenciado fornecido pela AWS para desinstalar um ou mais aplicativos.

Parâmetros de entrada

Esse modelo tem o seguinte parâmetro necessário, `packages`. Você também pode especificar os parâmetros opcionais `runAsUser` e `pathToHandler`.

`packages`

Uma lista separada por espaços de um ou mais aplicativos a serem desinstalados.

Comportamento do dispositivo

O dispositivo desinstala os aplicativos conforme especificado no documento do trabalho.

Documento de trabalho

Veja a seguir o documento do trabalho e sua versão mais recente. O modelo mostra o caminho para o gerenciador de trabalho e o script de shell, `remove-packages.sh`, que o gerenciador de trabalho deve executar para baixar o arquivo. Também mostra o parâmetro `packages` necessário.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Remove-Application",
        "type": "runHandler",
        "input": {
          "handler": "remove-packages.sh",
          "args": [
            "${aws:iot:parameter:packages}"
          ],
```

```
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Restart–Application

Nome do modelo

`AWS-Restart-Application`

Descrição do modelo

Um modelo gerenciado fornecido pela AWS para interromper e reiniciar um ou mais serviços.

Parâmetros de entrada

Esse modelo tem o seguinte parâmetro necessário, `services`. Você também pode especificar os parâmetros opcionais `runAsUser` e `pathToHandler`.

Services

Uma lista separada por espaços de um ou mais aplicativos a serem reiniciados.

Comportamento do dispositivo

Os aplicativos especificados são interrompidos e depois reiniciados no dispositivo.

Documento de trabalho

Veja a seguir o documento do trabalho e sua versão mais recente. O modelo mostra o caminho para o gerenciador de trabalho e o script de shell, `restart-services.sh`, que o gerenciador de trabalho deve executar para reiniciar os serviços do sistema. Também mostra o parâmetro `services` necessário.

```
{
  "version": "1.0",
  "steps": [
    {
```

```
      "action": {
        "name": "Restart-Application",
        "type": "runHandler",
        "input": {
          "handler": "restart-services.sh",
          "args": [
            "${aws:iot:parameter:services}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Start–Application

Nome do modelo

`AWS-Start-Application`

Descrição do modelo

Um modelo gerenciado fornecido por AWS para iniciar um ou mais serviços.

Parâmetros de entrada

Esse modelo tem o seguinte parâmetro necessário, `services`. Você também pode especificar os parâmetros opcionais `runAsUser` e `pathToHandler`.

`services`

Uma lista separada por espaços de um ou mais aplicativos a serem iniciados.

Comportamento do dispositivo

Os aplicativos especificados começam a ser executados no dispositivo.

Documento de trabalho

Veja a seguir o documento do trabalho e sua versão mais recente. O modelo mostra o caminho para o gerenciador de trabalho e o script de shell,`start-services.sh`, que o gerenciador de

trabalho deve executar para iniciar os serviços do sistema. Também mostra o parâmetro `services` necessário.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Start-Application",
        "type": "runHandler",
        "input": {
          "handler": "start-services.sh",
          "args": [
            "${aws:iot:parameter:services}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Stop–Application

Nome do modelo

`AWS-Stop-Application`

Descrição do modelo

Um modelo gerenciado fornecido por AWS para interromper um ou mais serviços.

Parâmetros de entrada

Esse modelo tem o seguinte parâmetro necessário, `services`. Você também pode especificar os parâmetros opcionais `runAsUser` e `pathToHandler`.

`services`

Uma lista separada por espaços de um ou mais aplicativos a serem interrompidos.

Comportamento do dispositivo

Os aplicativos especificados param de ser executados no dispositivo.

Documento de trabalho

Veja a seguir o documento do trabalho e sua versão mais recente. O modelo mostra o caminho para o gerenciador de trabalho e o script de shell, `stop-services.sh`, que o gerenciador de trabalho deve executar para interromper os serviços do sistema. Também mostra o parâmetro `services` necessário.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Stop-Application",
        "type": "runHandler",
        "input": {
          "handler": "stop-services.sh",
          "args": [
            "${aws:iot:parameter:services}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Run–Command

Nome do modelo

`AWS-Run-Command`

Descrição do modelo

Um modelo gerenciado fornecido por AWS para executar um comando shell.

Parâmetros de entrada

Esse modelo tem o seguinte parâmetro necessário, `command`. Você também pode especificar o parâmetro opcional `runAsUser`.

`command`

Uma sequência de comando separada por vírgula. Qualquer vírgula contida no comando em si deve ser recuada.

Comportamento do dispositivo

O dispositivo executa o comando shell conforme especificado no documento do trabalho.

Documento de trabalho

Veja a seguir o documento do trabalho e sua versão mais recente. O modelo mostra o caminho para o comando de trabalho e o comando que você forneceu e que o dispositivo executará.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Run-Command",
        "type": "runCommand",
        "input": {
          "command": "${aws:iot:parameter:command}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

Tópicos

- Crie um trabalho a partir de modelos AWS gerenciados usando o Console de gerenciamento da AWS
- Crie um trabalho a partir de modelos AWS gerenciados usando o AWS CLI

Crie um trabalho a partir de modelos AWS gerenciados usando o Console de gerenciamento da AWS

Use o Console de gerenciamento da AWS para obter informações sobre modelos AWS gerenciados e criar um trabalho usando esses modelos. Em seguida, você pode salvar o trabalho criado como seu próprio modelo personalizado.

Obter detalhes sobre modelos gerenciados

Você pode obter informações sobre os diferentes modelos gerenciados que estão disponíveis para uso no AWS IoT console.

1. Para ver seus modelos gerenciados disponíveis, acesse o hub Job templates do AWS IoT console e escolha a guia Managed templates.
2. Para ver os detalhes, escolha um modelo gerenciado.

A página Detalhes contém as seguintes informações:

- Nome, descrição e nome do recurso da Amazon (ARN) do modelo gerenciado.
- O ambiente no qual as operações remotas podem ser executadas, como o Linux.
- O documento de trabalho JSON que especifica o caminho para o gerenciador de trabalho e os comandos a serem executados no dispositivo. Por exemplo, o seguinte mostra um exemplo de documento de trabalho para o modelo de reinicialização da AWS. O modelo mostra o caminho para o gerenciador de trabalho e o script de shell, `reboot.sh`, que o gerenciador de trabalho deve executar para reinicializar o dispositivo.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Reboot",
        "type": "runHandler",
        "input": {
          "handler": "reboot.sh",
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

Para obter mais informações sobre o documento de trabalho e seus parâmetros de configuração para várias ações remotas, consulte Ações remotas e documentos de trabalho do modelo gerenciado.

- A versão mais recente do documento de trabalho.

Crie um trabalho usando modelos gerenciados

Você pode usar o console AWS de gerenciamento para escolher um modelo AWS gerenciado para criar um trabalho. Esta seção explica como fazer isso.

Você também pode iniciar o fluxo de trabalho de criação de tarefas e, em seguida, escolher o modelo AWS gerenciado que deseja usar ao criar a tarefa. Para obter mais informações sobre esse fluxo de trabalho, consulte Crie e gerencie trabalhos usando o Console de gerenciamento da AWS.

1. Escolha seu modelo AWS gerenciado

   Acesse o hub Job templates do AWS IoT console, escolha a guia Managed templates e, em seguida, escolha seu modelo.

2. Crie um trabalho usando seu modelo gerenciado

   1. Na página Detalhes do modelo de trabalho, escolha Criar trabalho.

      O console muda para a etapa Propriedades do trabalho personalizado do fluxo de trabalho de Criar trabalho, na qual sua configuração de modelo foi adicionada.

   2. Insira um nome de trabalho alfanumérico exclusivo e uma descrição e tags opcionais e, em seguida, escolha Próximo.

   3. Escolha as objetos ou grupos de objetos como destinos de trabalho que você deseja executar nesse trabalho.

   4. Na seção Documento de trabalho, seu modelo é exibido com suas configurações e parâmetros de entrada. Insira valores para os parâmetros de entrada do modelo escolhido. Por exemplo, se você escolheu o modelo AWS-Download-File:

      - Para downloadUrl, insira o URL do arquivo a ser baixado, por exemplo: `https://example.com/index.html`.
      - Para filePath, insira o caminho no dispositivo para armazenar o arquivo baixado, por exemplo: *`path/to/file`*.

      Opcionalmente, você também pode inserir valores para os parâmetros `runAsUser` e `pathToHandler`. Para obter mais informações sobre parâmetros de entrada de cada modelo, consulte Ações remotas e documentos de trabalho do modelo gerenciado.

5. Na página Configuração do trabalho, escolha o tipo de trabalho como contínuo ou de snapshot. Um trabalho de snapshot é concluído quando termina sua execução nos dispositivos e grupos de destino. Um trabalho contínuo se aplica a grupos de objetos e é executado em qualquer dispositivo que você adicione posteriormente a um grupo-alvo especificado.
6. Continue adicionando configurações adicionais ao seu trabalho e, em seguida, revise e crie seu trabalho. Para obter informações adicionais sobre as configurações adicionais, consulte:
   - Configurações de implantação, agendamento e anulação de trabalhos
   - Configurações de novas tentativas e de tempo limite de execuções de trabalhos

Crie modelos de trabalho personalizados a partir de modelos gerenciados

Você pode usar um modelo AWS gerenciado e um trabalho personalizado como ponto de partida para criar seu próprio modelo de trabalho personalizado. Para criar um modelo de trabalho personalizado, primeiro crie um trabalho a partir do seu modelo AWS gerenciado, conforme descrito na seção anterior.

Em seguida, você pode salvar o trabalho personalizado como modelo para criar seu próprio modelo personalizado. Para salvar como modelo:

1. Acesse o Job hub do AWS IoT console e escolha o trabalho que contém seu modelo gerenciado.
2. Escolha Salvar como modelo de trabalho e, em seguida, crie seu modelo de trabalho personalizado. Para obter mais informações sobre como criar um modelo de trabalho personalizado, consulte Crie um modelo de trabalho a partir de um trabalho existente.

Crie um trabalho a partir de modelos AWS gerenciados usando o AWS CLI

Use o AWS CLI para obter informações sobre modelos AWS gerenciados e criar um trabalho usando esses modelos. Em seguida, você pode salvar o trabalho como um modelo e criar seu próprio modelo personalizado.

Liste modelos gerenciados

O `list-managed-job-templates` AWS CLI comando lista todos os modelos de trabalho em seu Conta da AWS.

```
aws iot list-managed-job-templates
```

Por padrão, a execução desse comando exibe todos os modelos AWS gerenciados disponíveis e seus detalhes.

```
{
    "managedJobTemplates": [
        {
            "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Reboot:1.0",
            "templateName": "AWS-Reboot",
            "description": "A managed job template for rebooting the device.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        },
        {
            "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Remove-
Application:1.0",
            "templateName": "AWS-Remove-Application",
            "description": "A managed job template for uninstalling one or more
 applications.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        },
        {
            "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Stop-Application:1.0",
            "templateName": "AWS-Stop-Application",
            "description": "A managed job template for stopping one or more system
 services.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        },

        ...

        {
            "templateArn": "arn:aws:iot:us-east-1::jobtemplate/AWS-Restart-
Application:1.0",
```

```
            "templateName": "AWS-Restart-Application",
            "description": "A managed job template for restarting one or more system
 services.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        }
    ]
}
```

Para obter mais informações, consulte ListManagedJobTemplates.

Obtenha detalhes sobre um modelo gerenciado

O `describe-managed-job-template` AWS CLI comando obtém detalhes sobre um modelo de trabalho especificado. Especifique o nome do modelo de trabalho e uma versão opcional do modelo. Se a versão do modelo não for especificada, a versão padrão predefinida retornará. Veja a seguir um exemplo de execução do comando para obter detalhes sobre o modelo de `AWS-Download-File`.

```
aws iot describe-managed-job-template \
    --template-name AWS-Download-File
```

O comando exibe os detalhes do modelo e o ARN, seu documento de trabalho e o parâmetro `documentParameters`, que é uma lista de pares de valores-chave dos parâmetros de entrada do modelo. Para obter mais informações sobre os diferentes modelos e parâmetros de entrada, consulte Ações remotas e documentos de trabalho do modelo gerenciado.

> **Note**
>
> O `documentParameters` objeto retornado quando você usa essa API só deve ser usado ao criar trabalhos a partir de modelos AWS gerenciados. O objeto não deve ser usado para modelos de trabalho personalizados. Para obter um exemplo que mostra como usar esse parâmetro, consulte Crie um trabalho usando modelos gerenciados.

```
{
    "templateName": "AWS-Download-File",
    "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Download-File:1.0",
```

```
    "description": "A managed job template for downloading a file.",
    "templateVersion": "1.0",
    "environments": [
        "LINUX"
    ],
    "documentParameters": [
        {
            "key": "downloadUrl",
            "description": "URL of file to download.",
            "regex": "(.*?)",
            "example": "http://www.example.com/index.html",
            "optional": false
        },
        {
            "key": "filePath",
            "description": "Path on the device where downloaded file is written.",
            "regex": "(.*?)",
            "example": "/path/to/file",
            "optional": false
        },
        {
            "key": "runAsUser",
            "description": "Execute handler as another user. If not specified, then
 handler is executed as the same user as device client.",
            "regex": "(.){0,256}",
            "example": "user1",
            "optional": true
        },
        {
            "key": "pathToHandler",
            "description": "Path to handler on the device. If not specified, then
 device client will use the current working directory.",
            "regex": "(.){0,4096}",
            "example": "/path/to/handler/script",
            "optional": true
        }
    ],
    "document": "{\"version\":\"1.0\",\"steps\":[{\"action\":{\"name
\":\"Download-File\",\"type\":\"runHandler\",\"input\":{\"handler\":
\"download-file.sh\",\"args\":[\"${aws:iot:parameter:downloadUrl}\",
\"${aws:iot:parameter:filePath}\"],\"path\":\"${aws:iot:parameter:pathToHandler}\"},
\"runAsUser\":\"${aws:iot:parameter:runAsUser}\"}}]}"
}
```

Para obter mais informações, consulte DescribeManagedJobTemplate.

Crie um trabalho usando modelos gerenciados

O `create-job` AWS CLI comando pode ser usado para criar um trabalho a partir de um modelo de trabalho. Ele tem como destino um dispositivo chamado `thingOne` e especifica o nome do recurso da Amazon (ARN) do modelo gerenciado a ser usado como base para o trabalho. Você pode substituir configurações avançadas, como configurações de tempo limite e cancelamento, passando os parâmetros associados do comando `create-job`.

O exemplo mostra como criar um trabalho que usa o modelo `AWS-Download-File`. Também mostra como especificar os parâmetros de entrada do modelo usando o parâmetro `document-parameters`.

> **Note**
>
> Use o `document-parameters` objeto somente com modelos AWS gerenciados. Este objeto não deve ser usado com modelos de trabalho personalizados.

```
aws iot create-job \
   --targets arn:aws:iot:region:account-id:thing/thingOne \
   --job-id "new-managed-template-job" \
   --job-template-arn arn:aws:iot:region::jobtemplate/AWS-Download-File:1.0 \
   --document-parameters downloadUrl=https://example.com/index.html,filePath=path/to/
file
```

em que:

- *region*é Região da AWS o.
- *account-id*é o Conta da AWS número exclusivo.
- *thingOne* é o nome do objeto de IoT para a qual o trabalho é direcionado.
- *AWS-Download-File:1.0*é o nome do modelo gerenciado.
- `https://`*example.com/index*`.html` é o URL para fazer download do arquivo.
- `https://`*pathto/file/index* é o caminho de arquivo local no dispositivo para armazenar o arquivo baixado.

Execute o comando a seguir para criar um trabalho para o modelo, *AWS-Download-File*.

```
{
    "jobArn": "arn:aws:iot:region:account-id:job/new-managed-template-job",
    "jobId": "new-managed-template-job",
    "description": "A managed job template for downloading a file."
}
```

Crie um modelo de trabalho personalizado a partir de modelos gerenciados

1. Crie um trabalho usando um modelo gerenciado, conforme descrito na seção anterior.
2. Crie um modelo de trabalho personalizado usando o ARN do trabalho que você criou. Para obter mais informações, consulte Crie um modelo de trabalho a partir de um trabalho existente.

Crie modelos de trabalho personalizados

Você pode criar modelos de trabalho usando o AWS IoT console AWS CLI e. Você também pode criar trabalhos a partir de modelos de trabalho usando o AWS CLI AWS IoT console e os aplicativos web do Fleet Hub for AWS IoT Device Management. Para obter mais informações sobre como trabalhar com modelos de trabalho nos aplicativos do Fleet Hub, consulte Trabalhando com modelos de trabalho no Fleet Hub para gerenciamento de AWS IoT dispositivos.

> **Note**
>
> O número total de padrões de substituição em um documento de trabalho precisa ser dez ou menos.

Tópicos

- Crie modelos de trabalho personalizados usando o Console de gerenciamento da AWS
- Crie modelos de trabalho personalizados usando o AWS CLI

Crie modelos de trabalho personalizados usando o Console de gerenciamento da AWS

Este tópico explica como criar, excluir e visualizar detalhes sobre modelos de trabalho usando o AWS IoT console.

Crie um modelo de trabalho personalizado

Você pode criar um modelo de trabalho personalizado original ou criar um modelo de trabalho a partir de um trabalho existente. Você também pode criar um modelo de trabalho personalizado a partir de um trabalho existente que foi criado usando um modelo AWS gerenciado. Para obter mais informações, consulte Crie modelos de trabalho personalizados a partir de modelos gerenciados.

Crie um modelo de trabalho original

1. Comece a criar seu modelo de trabalho

   1. Acesse o hub Job templates do AWS IoT console e escolha a guia Custom templates.
   2. Escolha Criar modelo de trabalho.

   > **Note**
   >
   > Você também pode navegar até a página Modelos de trabalho na página Serviços relacionados em Fleet Hub.

2. Especifique propriedades do modelo de trabalho

   Na página Criar modelo de trabalho, insira um identificador alfanumérico para o nome do trabalho e uma descrição alfanumérica para fornecer detalhes adicionais sobre o modelo.

   > **Note**
   >
   > Não recomendamos o uso de informações de identificação pessoal em seu trabalho IDs ou descrições.

3. Forneça o documento de trabalho

   Forneça um arquivo de trabalho JSON que seja armazenado em um bucket do S3 ou um documento integrado que seja especificado no trabalho. Esse arquivo de trabalho se tornará o documento de trabalho quando você criar um trabalho usando esse modelo.

   Se o arquivo do trabalho estiver armazenado em um bucket do S3, insira o URL do S3 ou escolha Procurar no S3 e, em seguida, navegue até o documento de trabalho e selecione-o.

> **Note**
>
> Você pode selecionar apenas buckets do S3 em sua Região atual.

4. Continue adicionando configurações adicionais ao seu trabalho e, em seguida, revise e crie seu trabalho. Para obter informações sobre as configurações adicionais e opcionais, consulte os seguintes links:

   - Configurações de implantação, agendamento e anulação de trabalhos
   - Configurações de novas tentativas e de tempo limite de execuções de trabalhos

Crie um modelo de trabalho a partir de um trabalho existente

1. Selecione o trabalho

   1. Acesse o Job hub do AWS IoT console e escolha o trabalho que você deseja usar como base para seu modelo de trabalho.
   2. Selecione Salvar como modelo de trabalho.

   > **Note**
   >
   > Opcionalmente, você pode escolher um documento de trabalho diferente ou editar as configurações avançadas do trabalho original e, em seguida, selecionar Criar modelo de trabalho. Seu novo modelo de trabalho aparece na página Modelos de trabalho.

2. Especifique propriedades do modelo de trabalho

   Na página Criar modelo de trabalho, insira um identificador alfanumérico para o nome do trabalho e uma descrição alfanumérica para fornecer detalhes adicionais sobre o modelo.

   > **Note**
   >
   > O documento de trabalho é o arquivo de trabalho que você especificou ao criar o modelo. Se o documento de trabalho for especificado no trabalho em vez de em um local do S3, você poderá ver o documento de trabalho na página de detalhes desse trabalho.

3. Continue adicionando configurações adicionais ao seu trabalho e, em seguida, revise e crie seu trabalho. Para obter informações adicionais sobre as configurações adicionais, consulte:

   - Configurações de implantação, agendamento e anulação de trabalhos
   - Configurações de novas tentativas e de tempo limite de execuções de trabalhos

Crie um trabalho com base em um modelo de trabalho personalizado

Você pode criar um trabalho a partir de um modelo de trabalho personalizado acessando a página de detalhes do seu modelo de trabalho, conforme descrito neste tópico. Você também pode criar um trabalho ou escolher o modelo de trabalho que deseja usar ao executar o fluxo de trabalho de criação de trabalhos. Para obter mais informações, consulte Crie e gerencie trabalhos usando o Console de gerenciamento da AWS.

Este tópico mostra como criar um trabalho na página de detalhes de um modelo de trabalho personalizado. Você também pode criar um trabalho a partir de um modelo AWS gerenciado. Para obter mais informações, consulte Crie um trabalho usando modelos gerenciados.

1. Escolha seu modelo de trabalho personalizado

   Acesse o hub Job templates do AWS IoT console, escolha a guia Custom templates e, em seguida, escolha seu modelo.

2. Crie um trabalho usando seu modelo personalizado

   Como criar um trabalho:

   1. Na página Detalhes do modelo de trabalho, escolha Criar trabalho.

      O console muda para a etapa Propriedades do trabalho personalizado do fluxo de trabalho de Criar trabalho, na qual sua configuração de modelo foi adicionada.

   2. Insira um nome de trabalho alfanumérico exclusivo e uma descrição e tags opcionais e, em seguida, escolha Próximo.

   3. Escolha as objetos ou grupos de objetos como destinos de trabalho que você deseja executar nesse trabalho.

      Na seção Documento de trabalho, seu modelo é exibido com suas configurações. Se você quiser usar um documento de trabalho diferente, escolha Procurar e selecione um bucket e um documento diferentes. Escolha Próximo.

4. Na página Configuração do trabalho, escolha o tipo de trabalho como contínuo ou de snapshot. Um trabalho de snapshot é concluído quando termina sua execução nos dispositivos e grupos de destino. Um trabalho contínuo se aplica a grupos de objetos e é executado em qualquer dispositivo que você adicione posteriormente a um grupo-alvo especificado.
5. Continue adicionando configurações adicionais ao seu trabalho e, em seguida, revise e crie seu trabalho. Para obter informações adicionais sobre as configurações adicionais, consulte:
   - Configurações de implantação, agendamento e anulação de trabalhos
   - Configurações de novas tentativas e de tempo limite de execuções de trabalhos

> **Note**
>
> Quando um trabalho criado a partir de um modelo de trabalho atualiza os parâmetros existentes fornecidos pelo modelo de trabalho, esses parâmetros atualizados substituirão os parâmetros existentes fornecidos pelo modelo de trabalho para esse trabalho.

Você também pode criar trabalhos a partir de modelos de trabalho com os aplicativos web do Fleet Hub. Para obter informações sobre a criação de trabalhos no Fleet Hub, consulte Trabalhando com modelos de trabalho no Fleet Hub para gerenciamento de AWS IoT dispositivos.

Exclua um modelo de trabalho

Para excluir um modelo de trabalho, primeiro acesse o hub de Modelos de trabalho do AWS IoT console e escolha a guia Modelos personalizados. Em seguida, escolha o modelo que você deseja excluir e selecione Próximo.

> **Note**
>
> A exclusão é permanente e o modelo de trabalho não aparece mais na guia Modelos personalizados.

Crie modelos de trabalho personalizados usando o AWS CLI

Este tópico explica como criar, excluir e recuperar detalhes sobre modelos de trabalho usando a AWS CLI.

Crie um modelo de trabalho do zero

O AWS CLI comando a seguir mostra como criar um trabalho usando um documento de trabalho (*job-document.json*) armazenado em um bucket do S3 e uma função com permissão para baixar arquivos do Amazon S3 *S3DownloadRole* ().

```
aws iot create-job-template  \
      --job-template-id 010  \
      --description "My custom job template for updating the device firmware"
      --document-source https://s3.amazonaws.com/amzn-s3-demo-bucket/job-document.json
 \
      --timeout-config inProgressTimeoutInMinutes=100 \
      --job-executions-rollout-config "{ \"exponentialRate\": { \"baseRatePerMinute\":
 50, \"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings\":
 1000, \"numberOfSucceededThings\": 1000}}, \"maximumPerMinute\": 1000}" \
      --abort-config "{ \"criteriaList\": [ { \"action\": \"CANCEL\", \"failureType
\": \"FAILED\", \"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20},
 { \"action\": \"CANCEL\", \"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings
\": 200, \"thresholdPercentage\": 50}]}" \
      --presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/
S3DownloadRole\", \"expiresInSec\":3600}"
```

O parâmetro opcional `timeout-config` especifica o tempo que cada dispositivo tem para concluir a execução do trabalho. O temporizador é iniciado quando o status da execução do trabalho é definido como `IN_PROGRESS`. Se o status da execução do trabalho não estiver definido como outro estado terminal antes que o tempo expire, ele será definido como `TIMED_OUT`.

O temporizador em andamento não pode ser atualizado e é aplicado a todos os lançamentos de trabalho para o trabalho. Sempre que o lançamento de um trabalho permanece no `IN_PROGRESS` estado por mais tempo do que esse intervalo, o lançamento do trabalho falha e muda para o `TIMED_OUT` status do terminal. AWS IoT também publica uma notificação MQTT.

Para obter mais informações sobre como criar configurações para distribuir e anular trabalhos, consulte Configuração para distribuir e anular trabalhos.

> **Note**
>
> Os documentos de trabalho que são especificados como arquivos do Amazon S3 são recuperados no momento em que você cria o trabalho. Se você alterar o conteúdo do arquivo

do Amazon S3 que você usou como a origem de seu documento de trabalho depois de ter criado o trabalho, o que é enviado para os destinos do trabalho não é alterado.

Crie um modelo de trabalho a partir de um trabalho existente

O AWS CLI comando a seguir cria um modelo de trabalho especificando o Amazon Resource Name (ARN) de um trabalho existente. O novo modelo de trabalho usa todas as configurações especificadas no trabalho. Opcionalmente, você pode alterar qualquer configuração no trabalho existente usando qualquer um dos parâmetros opcionais.

```
aws iot create-job-template  \
      --job-arn arn:aws:iot:region:123456789012:job/job-name  \
      --timeout-config inProgressTimeoutInMinutes=100
```

Obtenha detalhes sobre um modelo de trabalho

O AWS CLI comando a seguir obtém detalhes sobre um modelo de trabalho especificado.

```
aws iot describe-job-template \
      --job-template-id template-id
```

O comando exibe a seguinte saída.

```
{
   "abortConfig": {
      "criteriaList": [
         {
            "action": "string",
            "failureType": "string",
            "minNumberOfExecutedThings": number,
            "thresholdPercentage": number
         }
      ]
   },
```

```
   "createdAt": number,
   "description": "string",
   "document": "string",
   "documentSource": "string",
   "jobExecutionsRolloutConfig": {
      "exponentialRate": {
         "baseRatePerMinute": number,
         "incrementFactor": number,
         "rateIncreaseCriteria": {
            "numberOfNotifiedThings": number,
            "numberOfSucceededThings": number
         }
      },
      "maximumPerMinute": number
   },
   "jobTemplateArn": "string",
   "jobTemplateId": "string",
   "presignedUrlConfig": {
      "expiresInSec": number,
      "roleArn": "string"
   },
   "timeoutConfig": {
      "inProgressTimeoutInMinutes": number
   }
}
```

Liste modelos de trabalho

O AWS CLI comando a seguir lista todos os modelos de trabalho em seu Conta da AWS.

```
aws iot list-job-templates
```

O comando exibe a seguinte saída.

```
{
   "jobTemplates": [
      {
         "createdAt": number,
         "description": "string",
```

```
         "jobTemplateArn": "string",
         "jobTemplateId": "string"
      }
   ],
   "nextToken": "string"
}
```

Para recuperar páginas adicionais de resultados, use o valor do campo `nextToken`.

Exclua um modelo de trabalho

O AWS CLI comando a seguir exclui um modelo de trabalho especificado.

```
aws iot delete-job-template \
      --job-template-id template-id
```

O comando não exibe nenhuma saída.

Crie um trabalho com base em um modelo de trabalho personalizado

O AWS CLI comando a seguir cria um trabalho a partir de um modelo de trabalho personalizado. Ele tem como destino um dispositivo chamado `thingOne` e especifica o nome do recurso da Amazon (ARN) do modelo de trabalho a ser usado como base para o trabalho. Você pode substituir configurações avançadas, como configurações de tempo limite e cancelamento, passando os parâmetros associados do comando `create-job`.

> **Warning**
>
> O objeto `document-parameters` deve ser usado com o comando `create-job` somente ao criar trabalhos a partir de modelos gerenciados da AWS . Este objeto não deve ser usado com modelos de trabalho personalizados. Para obter um exemplo que mostra como criar trabalhos usando esse parâmetro, consulte Crie um trabalho usando modelos gerenciados.

```
aws iot create-job \
      --targets arn:aws:iot:region:123456789012:thing/thingOne  \
```

```
    --job-template-arn arn:aws:iot:region:123456789012:jobtemplate/template-id
```

Configurações de trabalho

Você pode ter as seguintes configurações adicionais para cada trabalho que você implanta nos destinos especificados.

- Distribuição: define quantos dispositivos recebem o documento de trabalho a cada minuto.
- Agendamento: programa um trabalho para uma data e hora futuras, além de usar janelas de manutenção recorrentes.
- Interromper: cancela um trabalho em casos como quando alguns dispositivos não recebem a notificação do trabalho ou quando seus dispositivos relatam falhas na execução do trabalho.
- Tempo limite: se não houver resposta de suas metas de trabalho dentro de um determinado período após o início das execuções de trabalho, o trabalho pode falhar.
- Tentar novamente: repita a execução do trabalho se o dispositivo relatar uma falha ao tentar concluir a execução de um trabalho ou se a execução do trabalho atingir o tempo limite.

Ao usar essas configurações, você pode monitorar o status da execução do trabalho e evitar que uma atualização incorreta seja enviada para uma frota inteira.

Tópicos

- Como as configurações de trabalho funcionam
- Especificar configurações adicionais

Como as configurações de trabalho funcionam

As configurações de distribuição e anulação são usadas ao implantar um trabalho e as configurações de tempo limite e repetição para execução do trabalho. As seções a seguir mostram mais informações sobre como essas configurações funcionam.

Tópicos

- Configurações de implantação, agendamento e anulação de trabalhos
- Configurações de novas tentativas e de tempo limite de execuções de trabalhos

Configurações de implantação, agendamento e anulação de trabalhos

Você pode usar as configurações de distribuição, agendamento e anulação do trabalho para definir quantos dispositivos recebem o documento do trabalho, agendar uma distribuição do trabalho e determinar os critérios para cancelar um trabalho.

Configuração de distribuição de trabalho

Você pode especificar a rapidez com que os destinos são notificados sobre a execução de um trabalho pendente. Você também pode criar uma distribuição em etapas para gerenciar as atualizações, reinicializações e outras operações. Para especificar como seus destinos são notificados, use as taxas de distribuição de trabalhos.

Taxas de distribuição de trabalho

Você pode criar uma configuração de distribuição usando uma taxa de distribuição constante ou uma exponencial. Para especificar o número máximo de destinos de trabalho a serem informados por minuto, use uma taxa de distribuição constante.

AWS IoT os trabalhos podem ser implantados usando taxas de implantação exponenciais à medida que vários critérios e limites são atendidos. Se o número de trabalhos com falha corresponder a um conjunto de critérios que você especificar, você poderá cancelar a distribuição do trabalho. Você define os critérios da taxa de distribuição do trabalho ao criar um trabalho usando o objeto `JobExecutionsRolloutConfig`. Você também define os critérios de anulação do trabalho na criação do trabalho usando o objeto `AbortConfig`.

O exemplo a seguir mostra como as taxas de distribuição funcionam. Por exemplo, uma distribuição de trabalho com uma taxa básica de 50 por minuto, fator de incremento de 2 e número de dispositivos notificados e bem-sucedidos, cada um, de 1.000, funcionaria da seguinte forma: o trabalho começará com uma taxa de 50 execuções de trabalho por minuto e continuará nessa taxa até que 1.000 objetos tenham recebido notificações de execução de trabalho ou 1.000 execuções de trabalho bem-sucedidas tenham ocorrido.

A tabela a seguir ilustra como a implantação continuaria nos primeiros quatro incrementos.

Taxa de implementação por minuto	50	100	200	400
Número de dispositivos notificados ou execuções de trabalho bem-sucedidas para satisfazer um aumento na taxa	1.000	2.000	3.000	4.000

> **Note**
>
> Se você estiver no limite máximo simultâneo de 500 trabalhos (`isConcurrent = True`), todos os trabalhos ativos permanecerão com o status de `IN-PROGRESS` e não implementarão nenhuma nova execução de trabalho até que o número de trabalhos simultâneos seja 499 ou menos (`isConcurrent = False`). Isso se aplica a trabalhos contínuos e de snapshot.
> Se `isConcurrent = True`, o trabalho está implementando atualmente execuções de trabalhos em todos os dispositivos do seu grupo de destino. Se `isConcurrent = False`, o trabalho concluiu a distribuição de todas as execuções de trabalhos em todos os dispositivos do seu grupo de destino. Ele atualizará seu estado de status quando todos os dispositivos do grupo de destino atingirem um estado terminal ou uma porcentagem limite do grupo de destino, caso você tenha selecionado uma configuração de anulação de trabalho. Os estados de status do nível de trabalho para `isConcurrent = True` e `isConcurrent = False` são ambos `IN_PROGRESS`.
> Para obter mais informações sobre limites de trabalho ativos e simultâneos, consulte Limites de trabalho ativos e simultâneos.

Taxas de distribuição de trabalhos para trabalhos contínuos usando grupos dinâmicos de objetos

Quando você usa um trabalho contínuo para implantar operações remotas em sua frota, o AWS IoT Jobs implementa execuções de trabalhos para dispositivos em seu grupo-alvo. Para novos dispositivos adicionados ao grupo dinâmico de objetos, essas execuções de trabalhos continuam sendo distribuídas nesses dispositivos mesmo após a criação do trabalho.

A configuração de distribuição pode controlar as taxas de distribuição somente para dispositivos adicionados ao grupo até a criação do trabalho. Depois que um trabalho é criado, para qualquer novo dispositivo, as execuções do trabalho são criadas quase em tempo real assim que os dispositivos se juntam ao grupo de destino.

Configuração de agendamento de trabalho

Você pode agendar um trabalho contínuo ou de snapshot com até um ano de antecedência usando um horário de início, horário de término e comportamento de término predeterminados para o que acontecerá com a execução de cada trabalho ao atingir o horário de término. Além disso, você pode criar uma janela de manutenção recorrente opcional com frequência, horário de início e duração

flexíveis para que trabalhos contínuos possam distribuir um documento de trabalho em todos os dispositivos do grupo de destino.

Configurações de agendamento de trabalho

Horário de início

O horário de início de um trabalho agendado é a data e a hora futuras em que o trabalho iniciará a distribuição do documento de trabalho em todos os dispositivos do grupo de destino. O horário de início de um trabalho agendado se aplica a trabalhos contínuos e trabalhos de snapshot. Quando um trabalho agendado é criado inicialmente, ele mantém um estado de status de `SCHEDULED`. Ao chegar ao `startTime` que você selecionou, ele é atualizado para `IN_PROGRESS` e inicia a distribuição do documento de trabalho. O `startTime` deve ser menor ou igual a um ano a partir da data e hora iniciais em que você criou o trabalho agendado.

Para obter mais informações sobre a sintaxe para `startTime` usar um comando de API ou o AWS CLI, consulte Timestamp.

Para um trabalho com a configuração de agendamento opcional que ocorre durante uma janela de manutenção recorrente em um local que observa o horário de verão, o horário mudará em uma hora ao mudar do horário de verão para o horário padrão e do horário padrão para o horário de verão.

> **Note**
>
> O fuso horário exibido no Console de gerenciamento da AWS é o fuso horário atual do sistema. No entanto, esses fusos horários serão convertidos em UTC no sistema.

Horário de término

O horário de término de um trabalho agendado é a data e a hora futuras em que o trabalho interromperá a distribuição do documento de trabalho em qualquer dispositivo restante no grupo de destino. A hora de término de um trabalho agendado se aplica a trabalhos contínuos e trabalhos de snapshot. Depois que um trabalho agendado chega ao `endTime` selecionado e todas as execuções do trabalho atingem um estado terminal, ele atualiza seu estado de status de `IN_PROGRESS` para `COMPLETED`. O `endTime` deve ser menor ou igual a dois anos a partir da data e hora iniciais em que você criou o trabalho agendado. A duração mínima entre o `startTime` e o `endTime` é de 30 minutos. As tentativas de repetição da execução do trabalho ocorrerão até que o trabalho alcance o `endTime`; então, o `endBehavior` ditará como proceder.

Para obter mais informações sobre a sintaxe para `endTime` usar um comando de API ou o AWS CLI, consulte Timestamp.

Para um trabalho com a configuração de agendamento opcional que ocorre durante uma janela de manutenção recorrente em um local que observa o horário de verão, o horário mudará em uma hora ao mudar do horário de verão para o horário padrão e do horário padrão para o horário de verão.

> **Note**
>
> O fuso horário exibido no Console de gerenciamento da AWS é o fuso horário atual do sistema. No entanto, esses fusos horários serão convertidos em UTC no sistema.

Comportamento final

O comportamento final de um trabalho agendado determina o que acontece com o trabalho e com todas as execuções de trabalhos inacabados quando o trabalho chega ao `endTime` selecionado.

A seguir, são listados os comportamentos finais que você pode selecionar ao criar o trabalho ou o modelo de trabalho:

- `STOP_ROLLOUT`
  - `STOP_ROLLOUT` interrompe a distribuição do documento de trabalho em todos os dispositivos restantes no grupo de destino do trabalho. Além disso, todas as execuções de trabalhos `QUEUED` e `IN_PROGRESS` continuarão até atingirem um estado terminal. Esse é o comportamento final padrão, a menos que você selecione `CANCEL` ou `FORCE_CANCEL`.
- `CANCEL`
  - `CANCEL` interrompe a distribuição do documento de trabalho em todos os dispositivos restantes no grupo de destino do trabalho. Além disso, todas as execuções de trabalhos `QUEUED` serão canceladas, enquanto todas as execuções de trabalho `IN_PROGRESS` continuarão até atingirem um estado terminal.
- `FORCE_CANCEL`
  - `FORCE_CANCEL` interrompe a distribuição do documento de trabalho em todos os dispositivos restantes no grupo de destino do trabalho. Além disso, todas as execuções de trabalhos `QUEUED` e `IN_PROGRESS` serão canceladas.

> **Note**
>
> Para selecionar um `endbehavior`, selecione um `endtime`

Duração máxima

A duração máxima de um trabalho agendado deve ser menor ou igual a dois anos, independentemente do `startTime` e `endTime`.

A tabela a seguir lista cenários comuns de duração de um trabalho agendado:

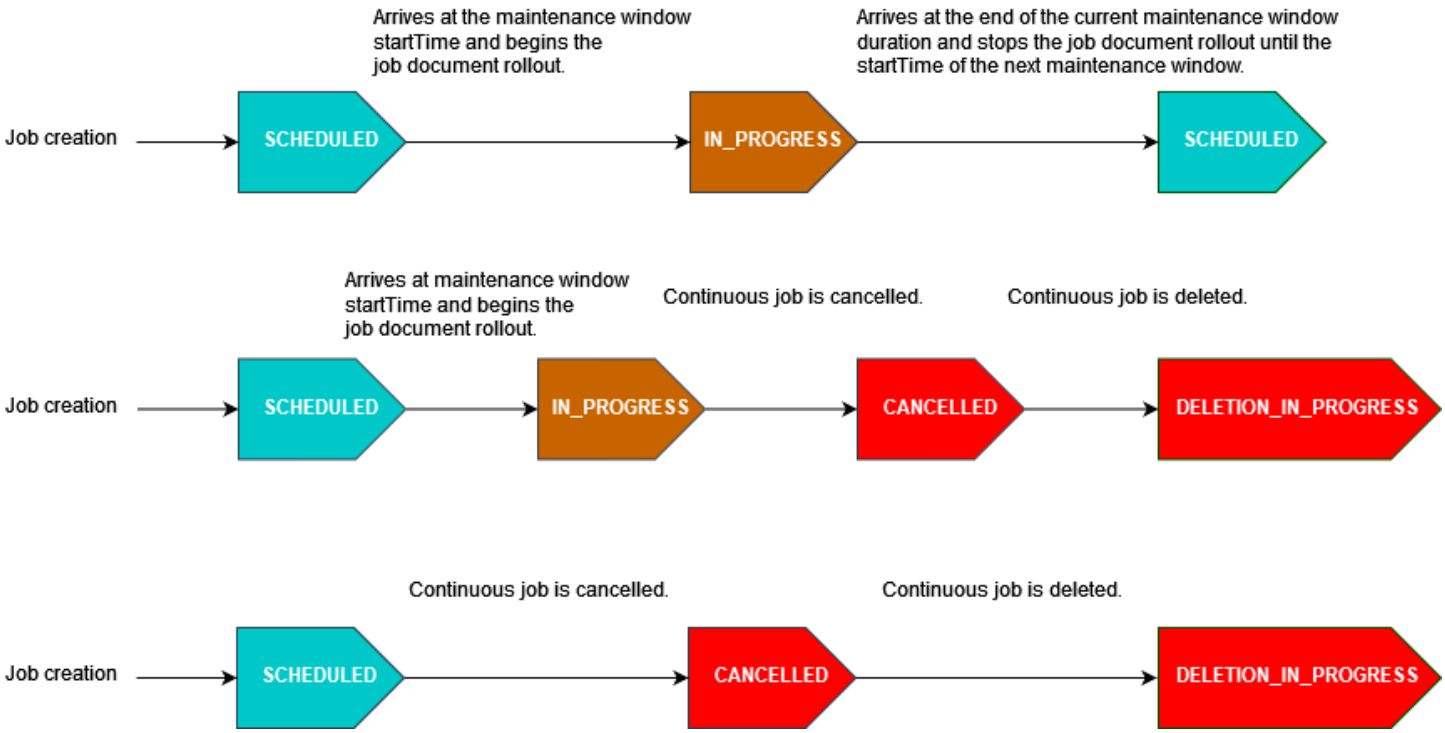
Número de exemplo de trabalho agendado	startTime	endTime	Duração máxima
1	Imediatamente após a criação inicial do trabalho.	Um ano após a criação inicial do trabalho.	Um ano
2	Um mês após a criação inicial do trabalho.	13 meses após a criação inicial do trabalho.	Um ano
3	Um ano após a criação inicial do trabalho.	Dois anos após a criação inicial do trabalho.	Um ano
4	Imediatamente após a criação inicial do trabalho.	Dois anos após a criação inicial do trabalho.	Dois anos

Janela de manutenção recorrente

A janela de manutenção é uma configuração opcional dentro da configuração de agendamento do Console de gerenciamento da AWS e `SchedulingConfig` dentro do `CreateJob` e. `CreateJobTemplate` APIs Você pode configurar uma janela de manutenção recorrente com uma hora de início, duração e frequência predeterminadas (diária, semanal ou mensal) em que a janela

de manutenção ocorre. As janelas de manutenção só se aplicam a trabalhos contínuos. A duração máxima de uma janela de manutenção recorrente é de 23 horas e 50 minutos.

O diagrama a seguir ilustra os estados do status de trabalho para vários cenários de trabalho programados com uma janela de manutenção opcional:



Para obter mais informações sobre o status de trabalho, consulte Trabalhos e estados de execução de trabalhos.

> **Note**
>
> Se um trabalho chegar ao `endTime` durante uma janela de manutenção, ele será atualizado de `IN_PROGRESS` para `COMPLETED`. Além disso, todas as execuções de trabalho restantes seguirão `endBehavior` do trabalho.

Expressão cron

Para trabalhos agendados que distribuem o documento de trabalho durante uma janela de manutenção com uma frequência personalizada, a frequência personalizada é inserida usando uma expressão cron. Uma expressão cron têm seis campos obrigatórios, que são separados por um espaço em branco.

Sintaxe

```
cron(fields)
```

Campo	Valores	Curingas
minutos	0-59	, - * /
Horas	0-23	, - * /
D ay-of-month	1-31	, - * ? / L W
Mês	1-12 ou JAN-DEZ	, - * /
D ay-of-week	1-7 ou DOM-SÁB	, - * ? L #
Ano	1970-2199	, - * /

Curingas

- A , (vírgula) curinga inclui valores adicionais. No campo Mês, JAN, FEV, MAR incluiria janeiro, fevereiro e março.
- O - (traço) curinga especifica intervalos. No campo Dia, 1-15 incluiria dias 1 a 15 do mês especificado.
- O * (asterisco) curinga inclui todos os valores no campo. No campo Horas, * incluiria cada hora. Você não pode usar * nos Day-of-week campos Day-of-month e. Se você usá-lo em um deles, utilize ? no outro.
- A / (barra) curinga especifica incrementos. No campo Minutos, você pode inserir 1/10 para especificar cada décimo minuto a partir do primeiro minuto da hora (por exemplo, o 11º, 21º e 31º minuto, etc.).
- O curinga ? (interrogação) especifica um ou outro. No Day-of-month campo, você poderia inserir 7 e se não se importasse em que dia da semana era o 7º, você poderia inserir? no Day-of-week campo.
- O curinga L nos campos Day-of-month ou Day-of-week especifica o último dia do mês ou da semana.

- O **`W`** curinga no Day-of-month campo especifica um dia da semana. No Day-of-month campo, **`3W`** especifica o dia da semana mais próximo do terceiro dia do mês.
- O caractere curinga # no Day-of-week campo especifica uma determinada instância do dia da semana especificado em um mês. Por exemplo, 3#2 seria a segunda terça-feira do mês: o 3 refere-se a terça-feira, porque é o terceiro dia de cada semana, e o 2 refere-se ao segundo dia desse tipo dentro do mês.

> **Note**
>
> Se você usar um caractere '#', poderá definir somente uma expressão no day-of-week campo. Por exemplo, o valor `"3#1,6#3"` não é válido porque é interpretado como duas expressões.

Restrições

- Não é possível especificar os campos Day-of-month e Day-of-week na mesma expressão cron. Se você especificar um valor (ou um *) em um dos campos, deverá usar um ? no outro.

Exemplos

Consulte os exemplos de strings cron a seguir ao usar uma expressão cron para o `startTime` de uma janela de manutenção recorrente.

Minutos	Horas	Dia do mês	Mês	Dia da semana	Ano	Significado
0	10	*	*	?	*	Executada às 10h (UTC) todos os dias
15	12	*	*	?	*	Executada às 12h15 (UTC)

Minutos	Horas	Dia do mês	Mês	Dia da semana	Ano	Significado
						todos os dias
0	18	?	*	SEG-SEX	*	Executada às 18h (UTC) de segunda a sexta
0	8	1	*	?	*	Executada às 8h (UTC) todo o primeiro dia do mês

Lógica final da duração da janela de manutenção recorrente

Quando a distribuição de um trabalho durante uma janela de manutenção atinge o final da duração da ocorrência da janela de manutenção atual, as seguintes ações ocorrerão:

- O trabalho encerrará todas as distribuições do documento de trabalho em todos os dispositivos restantes em seu grupo de destino. A operação será retomada no `startTime` da próxima janela de manutenção.
- Todas as execuções de trabalhos com um status de `QUEUED` permanecerão `QUEUED` até o `startTime` da próxima ocorrência da janela de manutenção. Na próxima janela, eles podem alternar para `IN_PROGRESS` quando o dispositivo estiver pronto para começar a executar as ações especificadas no documento do trabalho.
- Todas as execuções de trabalhos com um status de `IN_PROGRESS` continuarão realizando as ações especificadas no documento do trabalho até atingirem um estado terminal. Qualquer nova tentativa, conforme especificado em `JobExecutionsRetryConfig`, ocorrerá no `startTime` da próxima janela de manutenção.

Configuração de anulação de trabalho

Use essa configuração para criar um critério para cancelar um trabalho quando uma porcentagem limite de dispositivos atender a esses critérios. Por exemplo, você pode usar essa configuração para cancelar um trabalho nos seguintes casos:

- Quando uma porcentagem mínima de dispositivos não recebe as notificações de execução do trabalho, como quando seu dispositivo é incompatível com uma atualização Over-The-Air (OTA). Nesse caso, seu dispositivo pode relatar um status `REJECTED`.
- Quando uma porcentagem mínima de dispositivos relata falhas na execução de trabalhos, como quando seu dispositivo encontra uma desconexão ao tentar baixar o documento de trabalho de um URL do Amazon S3. Nesses casos, seu dispositivo deve estar programado para relatar o status `FAILURE` ao AWS IoT.
- Quando um status `TIMED_OUT` é relatado porque a execução do trabalho expira para uma porcentagem limite de dispositivos após o início das execuções do trabalho.
- Quando há várias falhas de nova tentativa. Quando você adiciona uma configuração de nova tentativa, cada nova tentativa pode gerar cobranças adicionais para sua Conta da AWS. Nesses casos, o cancelamento do trabalho pode cancelar execuções de trabalhos em fila e evitar novas tentativas para essas execuções. Para obter mais informações sobre a configuração de nova tentativa e como usá-la com a configuração de anulação, consulte Configurações de novas tentativas e de tempo limite de execuções de trabalhos.

Você pode configurar uma condição de cancelamento do trabalho usando o AWS IoT console ou a API AWS IoT Jobs.

Configurações de novas tentativas e de tempo limite de execuções de trabalhos

Use a configuração de tempo limite de execução do trabalho para enviar a você Notificações de trabalhos quando a execução de um trabalho estiver em andamento por mais tempo do que a duração definida. Use a configuração de repetição da execução do trabalho para repetir a execução quando o trabalho falhar ou atingir o tempo limite.

Configuração de tempo limite de execução de trabalhos

O tempo limite do trabalho permite que você receba uma notificação sempre que uma execução de trabalho ficar paralisada no estado `IN_PROGRESS` por um período inesperadamente longo. Quando o trabalho estiver `IN_PROGRESS`, você poderá monitorar o progresso da execução do trabalho.

Temporizadores para tempos limite de trabalhos

Há dois tipos de temporizadores: em andamento e de etapa.

Temporizadores em andamento

Ao criar um trabalho ou um modelo de trabalho, você pode especificar um valor para o temporizador em andamento que esteja entre 1 minuto e 7 dias. Você pode atualizar o valor desse temporizador até o início da execução do trabalho. Depois que o temporizador é iniciado, ele não pode ser atualizado e o valor do temporizador se aplica a todas as execuções do trabalho. Sempre que a execução de uma tarefa permanece no `IN_PROGRESS` status por mais tempo do que esse intervalo, a execução da tarefa falha e muda para o `TIMED_OUT` status do terminal. AWS IoT também publica uma notificação MQTT.

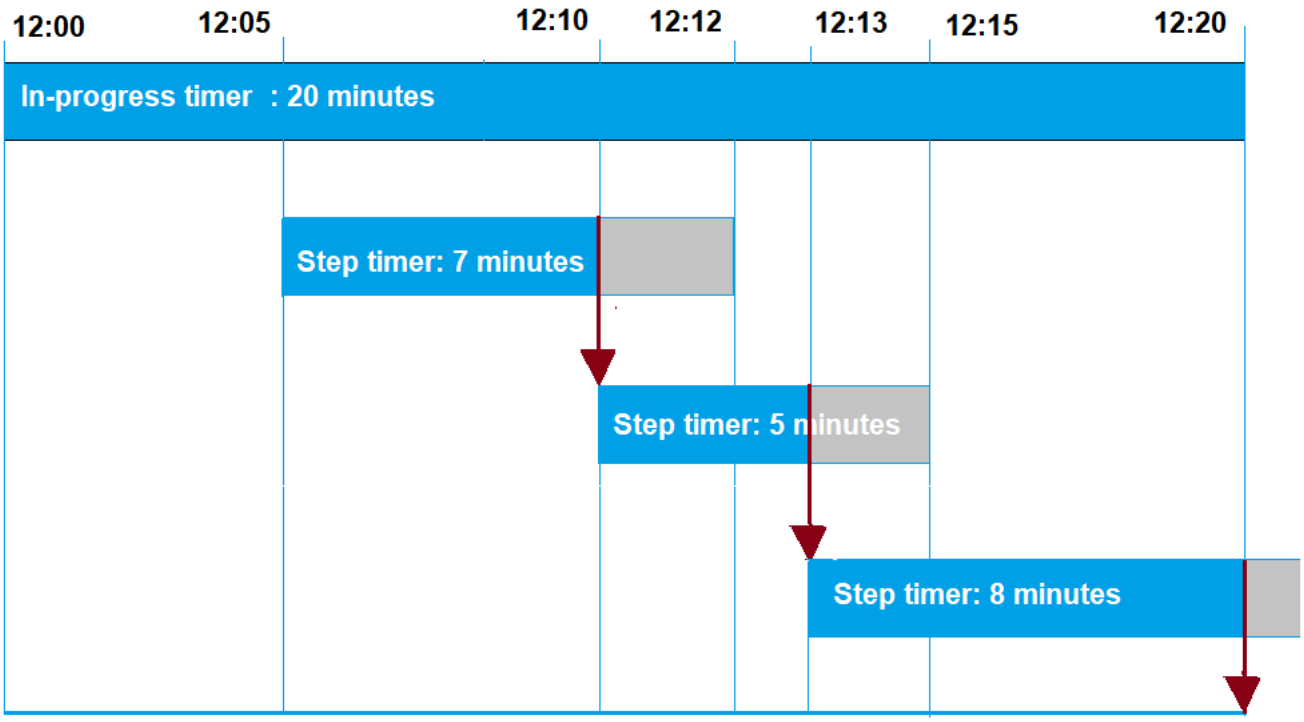
Temporizador de etapas

Você também pode definir um temporizador de etapa que se aplique somente à execução do trabalho que você deseja atualizar. Esse temporizador não tem efeito sobre o temporizador em andamento. Cada vez que você atualizar uma execução de trabalho, é possível definir um novo valor para o temporizador de etapa. Você também pode criar um novo temporizador de etapa ao iniciar a próxima execução de trabalho pendente de um objeto. Se a execução de trabalho permanecer com o status `IN_PROGRESS` por mais tempo que o intervalo do temporizador de etapa, ela falhará e alternará para o status `TIMED_OUT` terminal.

> **Note**
>
> Você pode definir o cronômetro em andamento usando o AWS IoT console ou a API AWS IoT Jobs. Para especificar o temporizador da etapa, use a API.

Como funcionam os temporizadores para o tempo limite de trabalho

A seguir, uma ilustração de como os tempos limite em andamento e de etapa interagem um com o outro em um período de tempo limite de 20 minutos.



A seguir, são mostradas as diferentes etapas:

1. 12:00

   Um novo trabalho é criado e um temporizador em andamento de vinte minutos é iniciado ao criar um trabalho. O temporizador em andamento começa a ser executado e a execução do trabalho muda para o status `IN_PROGRESS`.

2. 12h05

   Um novo temporizador de etapa com um valor de 7 minutos é criado. A execução do trabalho agora expirará às 12h12.

3. 12h10

   Um novo temporizador de etapa com um valor de 5 minutos é criado. Quando um novo temporizador de etapa é criado, o temporizador de etapa anterior é descartado e a execução do trabalho agora atingirá o tempo limite às 12h15.

4. 12h13

   Um novo temporizador de etapa com um valor de 9 minutos é criado. O temporizador de etapa anterior é descartado e a execução do trabalho agora atingirá o tempo limite às 12h20, pois o

temporizador em andamento expira às 12h20. O temporizador de etapa não pode exceder o limite absoluto do temporizador em andamento.

Configuração de repetição de execução de trabalho

Você pode usar a configuração de repetição para repetir a execução do trabalho quando um determinado conjunto de critérios for atendido. Uma nova tentativa pode ser feita quando um trabalho atinge o tempo limite ou quando o dispositivo falha. Para repetir a execução devido a uma falha de tempo limite, você deve habilitar a configuração de tempo limite.

Como usar a configuração de repetição

Siga as etapas a seguir para repetir essa configuração:

1. Determine se deve usar a configuração de repetição para `FAILED`, `TIMED_OUT` ou ambos os critérios de falha. Para o `TIMED_OUT` status, depois que o status é relatado, o AWS IoT Jobs repete automaticamente a execução do trabalho para o dispositivo.
2. Para o status `FAILED`, verifique se a falha na execução do trabalho pode ser repetida. Se for possível repetir, programe seu dispositivo para relatar um status `FAILURE` ao AWS IoT. A seção a seguir descreve mais sobre falhas que podem ser repetidas e não repetidas.
3. Especifique o número de novas tentativas a serem usadas para cada tipo de falha usando as informações anteriores. Para um único dispositivo, você pode especificar até dez novas tentativas para os dois tipos de falha combinados. As tentativas de repetição são interrompidas automaticamente quando uma execução é bem-sucedida ou quando atinge o número especificado de tentativas.
4. Adicione uma configuração de anulação para cancelar o trabalho se houver repetidas falhas de repetição para evitar cobranças adicionais com um grande número de tentativas de repetição.

> **Note**
>
> Quando um trabalho chega ao final de uma ocorrência recorrente da janela de manutenção, todas as execuções do trabalho `IN_PROGRESS` continuarão executando as ações identificadas no documento do trabalho até chegarem ao estado terminal. Se a execução de um trabalho atingir um estado terminal `FAILED` ou `TIMED_OUT` fora de uma janela de manutenção, uma nova tentativa ocorrerá na próxima janela se as tentativas não forem esgotadas. No próximo `startTime` da ocorrência da janela de manutenção, uma nova

execução de trabalho será criada e entrará em um estado de status `QUEUED` até que o dispositivo esteja pronto para começar.

Configuração de repetição e anulação

Cada tentativa de nova tentativa incorre em cobranças adicionais para você. Conta da AWS Para evitar cobranças adicionais devido a falhas repetidas de tentativas, recomendamos adicionar uma configuração de anulação. Para saber mais sobre preços, consulte Preços do AWS IoT Device Management.

Você pode encontrar várias falhas de repetição quando uma porcentagem alta de seus dispositivos atinge o tempo limite ou relata uma falha. Nesse caso, você pode usar a configuração de anulação para cancelar o trabalho e evitar qualquer execução de trabalho na fila ou novas tentativas.

> **Note**
>
> Quando os critérios de anulação são atendidos para cancelar a execução de um trabalho, somente as execuções de trabalhos `QUEUED` são canceladas. Nenhuma tentativa de repetição do dispositivo na fila será feita. No entanto, as execuções de trabalhos atuais que tenham um status `IN_PROGRESS` não serão canceladas.

Antes de repetir uma execução de trabalho com falha, também recomendamos que você verifique se a falha na execução do trabalho pode ser repetida, conforme descrito na seção a seguir.

Tipo de **`FAILED`** repetir em caso de falha

Para tentar repetir no tipo de falha `FAILED`, seus dispositivos devem ser programados para relatar o status `FAILURE` de uma falha na execução de um trabalho no AWS IoT. Defina a configuração de repetição com os critérios para repetir as execuções de trabalhos `FAILED` e especifique o número de novas tentativas a serem executadas. Quando o AWS IoT Jobs detectar o `FAILURE` status, ele tentará automaticamente repetir a execução do trabalho no dispositivo. As repetições continuam até que a execução do trabalho seja bem-sucedida ou quando atingir o número máximo de tentativas de repetição.

Você pode acompanhar cada tentativa de repetição e o trabalho que está sendo executado nesses dispositivos. Ao rastrear o status da execução, após o número especificado de tentativas, você pode usar seu dispositivo para relatar falhas e iniciar outra tentativa.

Falhas que podem ser repetidas e não repetidas

Sua falha na execução do trabalho pode ser repetida ou não. Cada nova tentativa de repetição pode gerar cobranças em sua Conta da AWS. Para evitar cobranças adicionais decorrentes de várias tentativas, primeiro considere verificar se a falha na execução do trabalho pode ser repetida. Um exemplo de falha que pode ser repetida inclui um erro de conexão que seu dispositivo encontra ao tentar baixar o documento de trabalho de um URL do Amazon S3. Se a falha na execução do trabalho puder ser repetida, programe seu dispositivo para relatar um status `FAILURE` caso a execução do trabalho falhe. Em seguida, defina a configuração de nova tentativa para repetir as execuções `FAILED`.

Se a execução não puder ser repetida, para evitar novas tentativas e possivelmente incorrer em cobranças adicionais em sua conta, recomendamos que você programe o dispositivo para relatar um status `REJECTED` ao AWS IoT. Exemplos de falhas que não podem ser repetidas incluem quando seu dispositivo é incompatível com o recebimento de uma atualização de trabalho ou quando ocorre um erro de memória ao executar um trabalho. Nesses casos, o AWS IoT Jobs não repetirá a execução do trabalho porque ele tentará novamente a execução do trabalho somente quando detectar um `FAILED` status ou. `TIMED_OUT`

Depois de determinar que uma falha na execução do trabalho pode ser repetida, se uma tentativa de repetição ainda falhar, considere verificar os logs do dispositivo.

> **Note**
>
> Quando um trabalho com a configuração de agendamento opcional chegar ao seu `endTime`, o `endBehavior` selecionado interromperá a distribuição do documento de trabalho em todos os dispositivos restantes no grupo de destino e ditará como proceder com as execuções restantes do trabalho. As tentativas são repetidas se selecionadas por meio da configuração de repetição.

Tipo de **`TIMEOUT`** repetir em caso de falha

Se você ativar o tempo limite ao criar um trabalho, o AWS IoT Jobs tentará repetir a execução do trabalho para o dispositivo quando o status mudar de `IN_PROGRESS` para. `TIMED_OUT` Essa alteração de status pode ocorrer quando o temporizador em andamento expira ou quando um temporizador de etapa que você especifica está `IN_PROGRESS` e, em seguida, expira. As repetições continuam até que a execução do trabalho seja bem-sucedida ou quando atingir o número máximo de tentativas de repetição para esse tipo de falha.

Trabalhos contínuos e atualizações de membros de grupos

Para trabalhos contínuos com status de trabalho `IN_PROGRESS`, o número de tentativas de repetição é redefinido para zero quando há atualizações na associação ao grupo de um objeto. Por exemplo, considere que você especificou cinco tentativas de repetição e três tentativas já foram realizadas. Se um objeto agora for removida do grupo de objetos e depois voltar ao grupo, como acontece com grupos de objetos dinâmicos, o número de tentativas de repetição é redefinido para zero. Agora você pode realizar cinco tentativas de repetição para seu grupo de objetos, em vez das duas tentativas restantes. Além disso, quando um objeto é removida do grupo de objetos, outras tentativas são canceladas.

Especificar configurações adicionais

Ao criar um trabalho ou um modelo de trabalho, você pode especificar essas configurações adicionais. Veja a seguir quando você pode especificar essas configurações.

- Ao criar um modelo de trabalho personalizado. As configurações adicionais que você especificar serão salvas quando você criar um trabalho a partir do modelo.
- Ao criar um trabalho personalizado usando um arquivo de trabalho. O arquivo de trabalho pode ser um arquivo JSON carregado em um bucket do S3.
- Ao criar um trabalho personalizado usando um arquivo de trabalho personalizado. Se o modelo já tiver essas configurações especificadas, você poderá reutilizá-las ou substituí-las especificando novas configurações.
- Ao criar um trabalho personalizado usando um modelo AWS gerenciado.

Tópicos

- Especifique as configurações do trabalho usando o Console de gerenciamento da AWS
- Especifique as configurações do trabalho usando a API AWS IoT Jobs

Especifique as configurações do trabalho usando o Console de gerenciamento da AWS

Você pode adicionar as diferentes configurações para seu trabalho usando o AWS IoT console. Depois de criar um trabalho, você pode ver os detalhes do status das configurações do seu trabalho na página de detalhes do trabalho. Para obter mais informações sobre as diferentes configurações e como elas funcionam, consulte Como as configurações de trabalho funcionam.

Adicione as configurações do trabalho ao criar um trabalho ou um modelo de trabalho.

Ao criar um modelo de trabalho personalizado

Para especificar a configuração de distribuição ao criar um modelo de trabalho personalizado

1. Acesse o hub Job templates do AWS IoT console e escolha Create job template.
2. Especifique as propriedades do modelo de trabalho, forneça o documento do trabalho, expanda a configuração que você deseja adicionar e, em seguida, especifique os parâmetros de configuração.

Ao criar um trabalho personalizado

Para especificar a configuração de distribuição ao criar um trabalho personalizado

1. Acesse o hub Job do AWS IoT console e escolha Create job.
2. Escolha Criar um trabalho personalizado e especifique as propriedades do trabalho, os destinos e se deseja usar um arquivo de trabalho ou um modelo para o documento do trabalho. Você pode usar um modelo personalizado ou AWS gerenciado.
3. Escolha a configuração do trabalho e, em seguida, expanda a Configuração de distribuição para especificar se deseja usar uma Taxa constante ou uma Taxa exponencial. Em seguida, especifique os parâmetros de configuração.

A próxima seção mostra os parâmetros que você pode especificar para cada configuração.

Configuração de distribuição

Você pode especificar se deseja usar uma taxa de distribuição constante ou uma taxa exponencial.

- Defina uma taxa de distribuição constante

  Para definir uma taxa constante para execuções de trabalhos, escolha Taxa constante e, em seguida, especifique o Máximo por minuto para o limite superior da taxa. Esse valor é opcional e varia de 1 a 1000. Se você não definir, ele usará 1000 como valor padrão.
- Defina uma taxa de distribuição exponencial

  Para definir uma taxa exponencial, escolha Taxa exponencial e, em seguida, especifique os parâmetros:
  - Taxa básica por minuto

A taxa na qual os trabalhos são executados até que o limite de Número de dispositivos notificados ou Número de dispositivos bem-sucedidos seja atingido para os Critérios de aumento de taxa.

- Fator de incremento

  O fator exponencial pelo qual a taxa de distribuição aumenta após o limite do Número de dispositivos notificados ou do Número de dispositivos bem-sucedidos ser atingido para os Critérios de aumento da taxa.

- Critérios de aumento de taxa

  O limite para o Número de dispositivos notificados ou o Número de dispositivos bem-sucedidos.

Configuração de anulação

Escolha Adicionar nova configuração e especifique os seguintes parâmetros para cada configuração:

- Tipo de falha

  Especifica os tipos de falha que iniciam a anulação do trabalho. Isso inclui FAILED, REJECTED, TIMED_OUT ou ALL.

- Fator de incremento

  Especifica o número de execuções de trabalho concluídas que devem ocorrer antes de os critérios de anulação do trabalho serem atendidos.

- Porcentagem de limite

  Especifica o número total de objetos executadas que iniciam a anulação de um trabalho.

Configuração de agendamento

Cada trabalho pode começar imediatamente após a criação inicial, programado para começar em uma data e hora posteriores ou ocorrer durante uma janela de manutenção recorrente.

Escolha Adicionar nova configuração e especifique os seguintes parâmetros para cada configuração:

- Início do trabalho

  Especifique a data e a hora de início do trabalho.

- Janela de manutenção recorrente

  Uma janela de manutenção recorrente define a data e a hora específicas em que um trabalho pode implantar o documento do trabalho nos dispositivos de destino do trabalho. A janela de manutenção pode ser repetida diariamente, semanalmente, mensalmente ou com uma recorrência de dia e hora personalizada.
- Fim do trabalho

  Especifique a data e a hora de término do trabalho.
- Comportamento ao término do trabalho

  Selecione um comportamento final para todas as execuções de trabalhos inacabadas quando o trabalho terminar.

> **Note**
>
> Quando um trabalho com a configuração de agendamento opcional e o horário de término selecionado atinge o horário de término, o trabalho interrompe a distribuição em todos os dispositivos restantes no grupo de destino. Ele também aproveita o comportamento final selecionado sobre como prosseguir com as execuções de trabalhos restantes e suas tentativas conforme a configuração de repetição.

Configuração de tempo limite

Por padrão, não há tempo limite e sua execução de trabalho é cancelada ou excluída. Para usar tempos limite, escolha Habilitar tempo limite e, em seguida, especifique um valor de tempo limite entre 1 minuto e 7 dias.

Configuração de repetição

> **Note**
>
> Depois que um trabalho é criado, o número de repetições não pode ser atualizado. Você só pode remover a configuração de repetição para todos os tipos de falha. Ao criar um trabalho, considere o número apropriado de repetições a serem usadas em sua configuração. Para evitar custos excessivos devido a possíveis falhas de repetição, adicione uma configuração de anulação.

Escolha Adicionar nova configuração e especifique os seguintes parâmetros para cada configuração:

- Tipo de falha

  Especifica os tipos de falha que devem acionar uma nova tentativa de execução do trabalho. Isso inclui Falha, Tempo limite e Todos.

- Número de novas tentativas

  Especifica o número de novas tentativas para o Tipo de falha escolhido. Para os dois tipos de falha combinados, é possível tentar até dez novas tentativas.

Especifique as configurações do trabalho usando a API AWS IoT Jobs

Você pode usar a CreateJobou a CreateJobTemplateAPI para especificar as diferentes configurações de trabalho. As seções a seguir descrevem como adicionar essas configurações. Depois de adicionar as configurações, você pode usar JobExecutionSummarye visualizar JobExecutionSummaryForJobo status delas.

Para obter mais informações sobre as diferentes configurações e como elas funcionam, consulte Como as configurações de trabalho funcionam.

Configuração de distribuição

Você pode especificar uma taxa de distribuição constante ou exponencial para sua configuração de distribuição.

- Defina uma taxa de distribuição constante

  Para definir uma taxa de distribuição constante, use o objeto `JobExecutionsRolloutConfig` para adicionar o parâmetro `maximumPerMinute` à solicitação `CreateJob`. Esse parâmetro especifica o limite superior da taxa em que as execuções de trabalho podem ocorrer. Esse valor é opcional e varia de 1 a 1000. Se você não definir o valor, ele usará 1000 como valor padrão.

  ```
  "jobExecutionsRolloutConfig": {
      "maximumPerMinute": 1000
  }
  ```

- Defina uma taxa de distribuição exponencial

Para definir uma taxa variável de distribuição de trabalhos, use o objeto `JobExecutionsRolloutConfig`. Você pode configurar a propriedade `ExponentialRolloutRate` ao executar a operação da API `CreateJob`. O exemplo a seguir define uma taxa de distribuição exponencial usando o parâmetro `exponentialRate`. Para obter mais informações sobre os parâmetros, consulte `ExponentialRolloutRate`.

```
{
...
  "jobExecutionsRolloutConfig": {
    "exponentialRate": {
      "baseRatePerMinute": 50,
      "incrementFactor": 2,
      "rateIncreaseCriteria": {
        "numberOfNotifiedThings": 1000,
        "numberOfSucceededThings": 1000
      },
      "maximumPerMinute": 1000
    }
  }
...
}
```

Em que o parâmetro:

baseRatePerMinuto

Especifica a taxa em que os trabalhos são executados até que o limite `numberOfNotifiedThings` ou `numberOfSucceededThings` seja atingido.

incrementFactor

Especifica o fator exponencial em que a taxa de distribuição aumenta após o limite `numberOfNotifiedThings` ou `numberOfSucceededThings` ser atingido.

rateIncreaseCriteria

Especifica o limite `numberOfNotifiedThings` ou `numberOfSucceededThings`.

Configuração de anulação

Para adicionar essa configuração usando a API, especifique o parâmetro `AbortConfig` ao executar a `CreateJob`, ou a operação da API `CreateJobTemplate`. O exemplo a seguir mostra uma configuração de anulação para uma distribuição de trabalho que estava passando por várias execuções malsucedidas, conforme especificado na operação da API `CreateJob`.

> **Note**
>
> Excluir uma execução de trabalho afeta o valor computacional da execução total concluída. Quando um trabalho é anulado, o serviço cria um `comment` automatizado e um `reasonCode` para diferenciar um cancelamento orientado pelo usuário de um cancelamento por anulação de trabalho.

```
"abortConfig": {
   "criteriaList": [
      {
         "action": "CANCEL",
         "failureType": "FAILED",
         "minNumberOfExecutedThings": 100,
         "thresholdPercentage": 20
      },
      {
         "action": "CANCEL",
         "failureType": "TIMED_OUT",
         "minNumberOfExecutedThings": 200,
         "thresholdPercentage": 50
      }
   ]
 }
```

Em que o parâmetro:

ação

Especifica a ação a ser tomada quando os critérios de anulação são atendidos. Esse parâmetro é necessário, e `CANCEL` é o único valor válido.

failureType

Especifica quais tipos de falha devem iniciar uma anulação de trabalho. Os valores válidos são `FAILED`, `REJECTED`, `TIMED_OUT` e `ALL`.

minNumberOfExecutedThings

Especifica o número de execuções de trabalho concluídas que devem ocorrer antes de os critérios de anulação do trabalho serem atendidos. Neste exemplo, a AWS IoT não verifica se a anulação de um trabalho deve ocorrer até que pelo menos 100 dispositivos tenham concluído as execuções do trabalho.

thresholdPercentage

Especifica o número total de objetos para as quais trabalhos são executados que podem iniciar uma anulação de trabalho. Neste exemplo, AWS IoT verifica sequencialmente e inicia um aborto de trabalho se a porcentagem limite for atingida. Se pelo menos 20% das execuções completas falharem após a conclusão de cem execuções, a distribuição do trabalho será cancelada. Se esse critério não AWS IoT for atendido, verifique se pelo menos 50% das execuções concluídas atingiram o tempo limite após a conclusão de 200 execuções. Se for esse o caso, ele cancela a distribuição do trabalho.

Configuração de agendamento

Para adicionar essa configuração usando a API, especifique a `SchedulingConfig` opcional ao executar a `CreateJob`, ou a operação da API `CreateJobTemplate`.

```
  "SchedulingConfig": {
    "endBehavior": string
    "endTime": string
    "maintenanceWindows": string
    "startTime": string
  }
```

Em que o parâmetro:

startTime

Especifica a data e a hora de início do trabalho.

endTime

Especifica a data e a hora em que o trabalho terminará.

maintenanceWindows

Especifica se uma janela de manutenção opcional foi selecionada para o trabalho agendado para distribuir o documento do trabalho em todos os dispositivos no grupo de destino. O formato de string para `maintenanceWindow` é YYYY/MM/DD para a data e hh:mm para a hora.

endBehavior

Especifica o comportamento do trabalho para um trabalho agendado ao chegar ao `endTime`.

> **Note**
>
> O opcional `SchedulingConfig` para um trabalho pode ser visualizado no `DescribeJob`e. `DescribeJobTemplate` APIs

Configuração de tempo limite

Para adicionar essa configuração usando a API, especifique o parâmetro `TimeoutConfig` ao executar a `CreateJob`, ou a operação da API `CreateJobTemplate`.

Para usar a configuração de tempo limite

1. Para definir o cronômetro em andamento ao criar um trabalho ou modelo de trabalho, defina um valor para a `inProgressTimeoutInMinutes` propriedade do objeto opcional TimeoutConfig.

   ```
   "timeoutConfig": {
     "inProgressTimeoutInMinutes": number
   }
   ```

2. Para especificar um cronômetro de etapas para a execução de um trabalho, defina um valor para `stepTimeoutInMinutes` quando você chama UpdateJobExecution. O temporizador de etapa se aplica apenas à execução do trabalho que você atualizar. É possível definir um novo valor para esse temporizador cada vez que você atualizar uma execução de trabalho.

   > **Note**
   >
   > `UpdateJobExecution` também descarta um temporizador de etapa que já foi criado por meio da criação de um novo temporizador de etapa com um valor de -1.

```
{
   ...
    "statusDetails": {
      "string" : "string"
   },
   "stepTimeoutInMinutes": number
}
```

3. Para criar um novo cronômetro de etapas, você também pode chamar a operação da StartNextPendingJobExecutionAPI.

Configuração de repetição

> **Note**
>
> Ao criar um trabalho, considere o número apropriado de repetições a serem usadas em sua configuração. Para evitar custos excessivos devido a possíveis falhas de repetição, adicione uma configuração de anulação. Depois que um trabalho é criado, o número de repetições não pode ser atualizado. Você só pode definir o número de novas tentativas como 0 usando a operação da UpdateJobAPI.

Para adicionar essa configuração usando a API, especifique o parâmetro `jobExecutionsRetryConfig` ao executar a `CreateJob`, ou a operação da API `CreateJobTemplate`.

```
{
...
  "jobExecutionsRetryConfig": {
      "criteriaList": [
         {
            "failureType": "string",
            "numberOfRetries": number
         }
      ]
  }
...
}
```

Onde criteriaList é uma matriz que especifica a lista de critérios que determina o número de novas tentativas permitidas para cada tipo de falha em um trabalho.

Dispositivos e trabalhos

Os dispositivos podem se comunicar com AWS IoT trabalhos usando MQTT, HTTP Signature versão 4 ou HTTP TLS. Para determinar o endpoint a ser usado quando seu dispositivo se comunica com AWS IoT Jobs, execute o DescribeEndpoint comando. Por exemplo, se você executar este comando:

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

você obterá um resultado semelhante ao seguinte:

```
{
    "endpointAddress": "a1b2c3d4e5f6g7-ats.iot.us-west-2.amazonaws.com"
}
```

Uso do protocolo MQTT

Os dispositivos podem se comunicar com AWS IoT Jobs usando o protocolo MQTT. Os dispositivos se inscrevem nos tópicos do MQTT para serem notificados sobre novos trabalhos e receber respostas do serviço AWS IoT Jobs. Os dispositivos publicam em tópicos MQTT para consultar ou atualizar o estado do lançamento de um trabalho. Cada dispositivo tem seu próprio tópico geral MQTT. Para obter mais informações sobre publicação e assinatura em tópicos MQTT, consulte Protocolos de comunicação do dispositivo.

Com esse método de comunicação, seu dispositivo usa o certificado e a chave privada específicos do dispositivo para se autenticar com o Jobs. AWS IoT

Seus dispositivos podem se inscrever nos tópicos a seguir. `thing-name` é o nome do objeto associado ao dispositivo.

- **`$aws/things/thing-name/jobs/notify`**

  Inscreva-se neste tópico para receber uma notificação quando um lançamento de trabalho for adicionado ou removido da lista de lançamentos de trabalhos pendentes.
- **`$aws/things/thing-name/jobs/notify-next`**

  Inscreva-se neste tópico para receber uma notificação quando a próxima execução de trabalho pendente for alterada.

- **`$aws/things/thing-name/jobs/request-name/accepted`**

  O serviço AWS IoT Jobs publica mensagens de sucesso e falha em um tópico do MQTT. O tópico é formado anexando `accepted` ou `rejected` ao tópico usado para fazer a solicitação. Aqui, `request-name` está o nome de uma solicitação, como `Get` e o tópico pode ser:`$aws/things/myThing/jobs/get`. AWS IoTEm seguida, Jobs publica mensagens de sucesso sobre o `$aws/things/myThing/jobs/get/accepted` tópico.
- **`$aws/things/thing-name/jobs/request-name/rejected`**

  Aqui, `request-name` é nome de uma solicitação, como `Get`. Se a solicitação falhar, AWS IoT Jobs publicará mensagens de falha sobre o `$aws/things/myThing/jobs/get/rejected` tópico.

Você também pode usar as seguintes operações de API de HTTPS:

- Atualizar o status de uma execução de trabalho chamando a API `UpdateJobExecution`.
- Consultar o status da execução de um trabalho chamando a API `DescribeJobExecution`.
- Recuperar uma lista de execuções de trabalhos pendentes, chamando a API `GetPendingJobExecutions`.
- Recuperar a próxima execução de trabalho pendente chamando a API `DescribeJobExecution` com o `jobId` `$next`.
- Acessar e iniciar a próxima execução de trabalho pendente chamando a API `StartNextPendingJobExecution`.

Uso do HTTP Signature Versão 4

Os dispositivos podem se comunicar com AWS IoT Jobs usando a assinatura HTTP versão 4 na porta 443. Esse é o método usado pela AWS SDKs e CLI. Para obter mais informações sobre essas ferramentas, consulte Referência de AWS CLI Comandos: iot-jobs-data ou Ferramentas AWS SDKs e consulte a IotJobsDataPlane seção do seu idioma preferido.

Com esse método de comunicação, seu dispositivo usa credenciais do IAM para se autenticar com AWS IoT Jobs.

Os comandos a seguir estão disponíveis usando esse método:

- DescribeJobExecution

```
aws iot-jobs-data describe-job-execution ...
```

- GetPendingJobExecutions

```
aws iot-jobs-data get-pending-job-executions ...
```

- StartNextPendingJobExecution

```
aws iot-jobs-data start-next-pending-job-execution ...
```

- UpdateJobExecution

```
aws iot-jobs-data update-job-execution ...
```

Uso do HTTP TLS

Os dispositivos podem se comunicar com AWS IoT Jobs usando HTTP TLS na porta 8443 usando um cliente de software de terceiros que ofereça suporte a esse protocolo.

Com esse método, seu dispositivo usa a autenticação baseada em certificado X.509 (por exemplo, o certificado específico para o dispositivo e a chave privada).

Os comandos a seguir estão disponíveis usando esse método:

- DescribeJobExecution
- GetPendingJobExecutions
- StartNextPendingJobExecution
- UpdateJobExecution

Programar dispositivos para funcionarem com trabalhos

Os exemplos nesta seção usam MQTT para ilustrar como um dispositivo funciona com serviço Jobs da AWS IoT. Se desejar, você pode usar os comandos da API ou CLI correspondentes. Nesses exemplos, estamos supondo que um dispositivo chamado `MyThing` assinará os seguintes tópicos MQTT:

- `$aws/things/MyThing/jobs/notify` (ou `$aws/things/MyThing/jobs/notify-next`)
- `$aws/things/MyThing/jobs/get/accepted`
- `$aws/things/MyThing/jobs/get/rejected`

- `$aws/things/MyThing/jobs/jobId/get/accepted`
- `$aws/things/MyThing/jobs/jobId/get/rejected`

Se você estiver usando a assinatura de código paraAWS IoT, o código do dispositivo deverá verificar a assinatura do seu arquivo de código. A assinatura está no documento de trabalho na propriedade `codesign`. Para obter mais informações sobre como verificar uma assinatura de arquivo de código, consulte Device Agent Sample.

Tópicos

- Fluxo de trabalho do dispositivo
- Fluxo de trabalho
- Notificações de trabalhos

Fluxo de trabalho do dispositivo

Um dispositivo pode lidar com trabalhos executados usando uma das seguintes formas.

- Consiga o próximo trabalho

  1. Quando um dispositivo se torna online pela primeira vez, ele deve assinar o tópico `notify-next` do dispositivo.
  2. Chamar a API MQTT DescribeJobExecution com o jobId `$next` para obter o próximo trabalho, seu documento de trabalho e outros detalhes, incluindo qualquer estado salvo em `statusDetails`. Se o documento de trabalho tiver uma assinatura de arquivo de código, você deverá verificar a assinatura antes de continuar com o processamento da solicitação de trabalho.
  3. Chamar a API MQTT UpdateJobExecution para atualizar o status do trabalho. Ou, para combinar essa e a etapa anterior em uma chamada, o dispositivo pode chamar StartNextPendingJobExecution.
  4. (Opcional) Você pode adicionar um temporizador de etapa. Para isso, defina um valor para `stepTimeoutInMinutes` ao chamar UpdateJobExecution ou StartNextPendingJobExecution.
  5. Executar as ações especificadas pelo documento de trabalho usando a API MQTT UpdateJobExecution para relatar o andamento do trabalho.

6. Continue monitorando a execução do trabalho chamando a DescribeJobExecution API MQTT com esse jobId. Se a execução do trabalho for excluída, DescribeJobExecution retornará um `ResourceNotFoundException`.

   O dispositivo deve conseguir voltar para um estado válido se a execução do trabalho for cancelada ou excluída enquanto o dispositivo estiver executando o trabalho.

7. Chame a UpdateJobExecution API MQTT quando finalizar o trabalho para atualizar o respectivo status e relatar êxito ou falha.
8. Como o status de execução desse trabalho foi alterado para o estado final, o próximo trabalho disponível para execução (se houver) será alterado. O dispositivo é notificado sobre a alteração da próxima execução de trabalho pendente. Neste ponto, o dispositivo deve continuar conforme descrito na etapa 2.

Se o dispositivo permanecer on-line, ele continuará recebendo notificações da próxima execução do trabalho pendente. Isso inclui seus dados de execução do trabalho, quando ele conclui um trabalho ou quando uma nova execução de trabalho pendente é adicionada. Quando isso ocorre, o dispositivo continua, conforme descrito na etapa 2.

- Selecione entre os trabalhos disponíveis

1. Quando um dispositivo se torna online pela primeira vez, ele deve assinar o tópico `notify` do objeto.
2. Chamar a API MQTT GetPendingJobExecutions para obter uma lista de execuções de trabalhos pendentes.
3. Se a lista contiver uma ou mais execuções de trabalho, selecione uma.
4. Chamar a API MQTT DescribeJobExecution para obter o documento de trabalho e outros detalhes, incluindo qualquer estado salvo em `statusDetails`.
5. Chamar a API MQTT UpdateJobExecution para atualizar o status do trabalho. Se o campo `includeJobDocument` estiver definido como `true` nesse comando, o dispositivo poderá ignorar a etapa anterior e recuperar o documento de trabalho neste ponto.
6. Você pode adicionar um temporizador de etapa. Para isso, defina um valor para `stepTimeoutInMinutes` ao chamar UpdateJobExecution.
7. Executar as ações especificadas pelo documento de trabalho usando a API MQTT UpdateJobExecution para relatar o andamento do trabalho.

8. Continue monitorando a execução do trabalho chamando a DescribeJobExecution API MQTT com esse jobId. Se a execução de trabalho for cancelada ou excluída enquanto o dispositivo estiver executando o trabalho, o dispositivo deve conseguir voltar para um estado válido.
9. Chame a UpdateJobExecution API MQTT quando finalizar o trabalho para atualizar o respectivo status e relatar êxito ou falha.

Se o dispositivo permanecer online, ele será notificado de todas as execuções de trabalho pendentes quando uma nova execução de trabalho se tornar disponível. Quando isso ocorrer, o dispositivo poderá continuar, conforme descrito na etapa 2.

Se o dispositivo não puder executar o trabalho, ele deverá chamar a API MQTT UpdateJobExecution para atualizar o status do trabalho para `REJECTED`.

Fluxo de trabalho

A seguir, são mostradas as diferentes etapas no fluxo de trabalho, desde o início de um novo trabalho até o relatório do status de conclusão da execução de um trabalho.

Iniciar um novo trabalho

Quando um novo trabalho é criado, o AWS IoT Jobs publica uma mensagem sobre o `$aws/things/`*`thing-name`*`/jobs/notify` tópico para cada dispositivo de destino.

A mensagem contém as seguintes informações:

```
{
    "timestamp":1476214217017,
    "jobs":{
        "QUEUED":[{
            "jobId":"0001",
            "queuedAt":1476214216981,
            "lastUpdatedAt":1476214216981,
            "versionNumber" : 1
        }]
    }
}
```

O dispositivo recebe essa mensagem no tópico `'$aws/things/`*`thingName`*`/jobs/notify'` quando a execução do trabalho é enfileirada.

> **Note**
>
> Para trabalhos com a `SchedulingConfig` opcional, o trabalho manterá um estado de status inicial de `SCHEDULED`. Quando o trabalho atingir o `startTime` selecionado, ocorrerá o seguinte:
>
> - O estado do status do trabalho será atualizado para `IN_PROGRESS`.
> - O trabalho iniciará a distribuição do documento de trabalho em todos os dispositivos do grupo de destino.

Obter informações do trabalho

Para obter mais informações sobre a execução de um trabalho, o dispositivo chama a API MQTT DescribeJobExecution com o campo `includeJobDocument` definido como `true` (o padrão).

Se a solicitação for bem-sucedida, o serviço AWS IoT Jobs publicará uma mensagem sobre o `$aws/things/MyThing/jobs/0023/get/accepted` tópico:

```
{
    "clientToken" : "client-001",
    "timestamp" : 1489097434407,
    "execution" : {
        "approximateSecondsBeforeTimedOut": number,
        "jobId" : "023",
        "status" : "QUEUED",
        "queuedAt" : 1489097374841,
        "lastUpdatedAt" : 1489097374841,
        "versionNumber" : 1,
        "jobDocument" : {
            < contents of job document >
        }
    }
}
```

Se a solicitação falhar, o serviço AWS IoT Jobs publicará uma mensagem sobre o `$aws/things/MyThing/jobs/0023/get/rejected` tópico.

O dispositivo agora tem o documento de trabalho, que ele pode usar para executar as operações remotas do trabalho. Se o documento de trabalho contiver um URL pré-assinado do Amazon S3, o dispositivo poderá usar esse URL para fazer download de todos os arquivos necessários do trabalho.

Relatar o status da execução de um trabalho

Enquanto executa o trabalho, o dispositivo pode chamar a API MQTT UpdateJobExecution para atualizar o status da execução do trabalho.

Por exemplo, um dispositivo pode atualizar o status da execução do trabalho para `IN_PROGRESS` publicando a seguinte mensagem no tópico `$aws/things/MyThing/jobs/0023/update`:

```
{
    "status":"IN_PROGRESS",
    "statusDetails": {
        "progress":"50%"
    },
    "expectedVersion":"1",
    "clientToken":"client001"
}
```

O serviço trabalhos responde publicando uma mensagem no tópico `$aws/things/MyThing/jobs/0023/update/accepted` ou `$aws/things/MyThing/jobs/0023/update/rejected`:

```
{
    "clientToken":"client001",
    "timestamp":1476289222841
}
```

O dispositivo pode combinar as duas solicitações anteriores chamando StartNextPendingJobExecution. Isso obtém e inicia a próxima execução de trabalho pendente e permite que o dispositivo atualize o status de execução do trabalho. Essa solicitação também retorna o documento de trabalho quando há uma execução de trabalho pendente.

Se o trabalho contiver um TimeoutConfig, o cronômetro em andamento começará a ser executado. Você também pode definir um cronômetro de etapas para a execução de um trabalho definindo um valor para `stepTimeoutInMinutes` quando você chama UpdateJobExecution. O temporizador de etapa se aplica apenas à execução do trabalho que você atualizar. É possível definir um novo valor para esse temporizador cada vez que você atualizar uma execução de trabalho. Você também pode criar um cronômetro de etapas ao ligar StartNextPendingJobExecution. Se a execução de trabalho permanecer com o status `IN_PROGRESS` por mais tempo que o intervalo do temporizador de etapa, ela falhará e alternará para o status `TIMED_OUT` terminal. O temporizador de etapa não tem efeito sobre o temporizador em andamento que você define ao criar um trabalho.

O campo `status` pode ser definido como `IN_PROGRESS`, `SUCCEEDED` ou `FAILED`. Você não pode atualizar o status da execução de um trabalho que já esteja em um estado terminal.

Relatar a conclusão da execução

Quando o dispositivo conclui a execução do trabalho, ele chama a API MQTT UpdateJobExecution. Se o trabalho tiver sido bem-sucedido, defina `status` como `SUCCEEDED` e, na carga da mensagem, em `statusDetails`, adicione outras informações sobre o trabalho, como pares nome/valor. Os temporizadores de etapa e em andamento terminam quando a execução do trabalho é concluída.

Por exemplo:

```
{
    "status":"SUCCEEDED",
    "statusDetails": {
        "progress":"100%"
    },
    "expectedVersion":"2",
    "clientToken":"client-001"
}
```

Se o trabalho não foi bem-sucedido, defina `status` como `FAILED` e, em `statusDetails`, adicione informações sobre o erro que ocorreu:

```
{
    "status":"FAILED",
    "statusDetails": {
        "errorCode":"101",
        "errorMsg":"Unable to install update"
    },
    "expectedVersion":"2",
    "clientToken":"client-001"
}
```

> **Note**
>
> O atributo `statusDetails` pode conter qualquer número de pares de nome e valor.

Quando o serviço AWS IoT Jobs recebe essa atualização, ele publica uma mensagem sobre o `$aws/things/MyThing/jobs/notify` tópico para indicar que a execução do trabalho foi concluída:

```
{
    "timestamp":1476290692776,
    "jobs":{}
}
```

Trabalhos adicionais

Se houver outras execuções de trabalho pendentes para o dispositivo, elas serão incluídas na mensagem publicada em `$aws/things/MyThing/jobs/notify`.

Por exemplo:

```
{
    "timestamp":1476290692776,
    "jobs":{
        "QUEUED":[{
            "jobId":"0002",
            "queuedAt":1476290646230,
            "lastUpdatedAt":1476290646230
        }],
        "IN_PROGRESS":[{
            "jobId":"0003",
            "queuedAt":1476290646230,
            "lastUpdatedAt":1476290646230
        }]
    }
}
```

Notificações de trabalhos

O serviço AWS IoT Jobs publica mensagens MQTT em tópicos reservados quando os trabalhos estão pendentes ou quando a primeira execução do trabalho na lista é alterada. Os dispositivos podem acompanhar trabalhos pendentes assinando esses tópicos.

Tipos de notificação de trabalhos

As notificações de trabalhos são publicadas em tópicos MQTT como cargas JSON. Há dois tipos de notificações:

ListNotification

Uma `ListNotification` contém uma lista de até 15 execuções de trabalhos pendentes. Elas são classificados por status (execuções de trabalho de `IN_PROGRESS` antes de execuções de trabalhos de `QUEUED`) e, em seguida, pelo número de vezes em que foram colocadas em fila.

Uma `ListNotification` é publicada sempre que um dos seguintes critérios são atendidos.

- Uma nova execução de trabalho é colocada na fila ou muda para um status terminal (`IN_PROGRESS` ou `QUEUED`).
- Uma execução de trabalho anterior muda para um status final (`FAILED`, `SUCCEEDED`, `CANCELED`, `TIMED_OUT`, `REJECTED` ou `REMOVED`).

Para acessar mais informações sobre os limites com e sem a configuração de agendamento, consulte Limites de execuções de trabalhos.

NextNotification

- Uma `NextNotification` contém informações resumidas sobre a próxima execução de trabalho da fila.

  Uma `NextNotification` é publicada sempre que a primeira execução de trabalho da lista é alterada.

  - Uma nova execução é adicionada à lista como `QUEUED` e é a primeira da lista.
  - O status de uma execução de trabalho existente que não é a primeira da lista é alterado de `QUEUED` para `IN_PROGRESS` e torna-se a primeira na lista. (Isso acontece quando não há outras execuções de trabalho de `IN_PROGRESS` na lista ou quando a execução de trabalho cujo status é alterado de `QUEUED` para `IN_PROGRESS` foi colocado na fila antes de qualquer outra execução de `IN_PROGRESS` na lista.)
  - O status da primeira execução de trabalho da lista é alterado para um status terminal e é removido da lista.

Para obter mais informações sobre publicação e assinatura em tópicos MQTT, consulte the section called “Protocolos de comunicação do dispositivo”.

> **Note**
>
> As notificações não estão disponíveis quando você usa HTTP Signature Versão 4 ou HTTP TLS para se comunicar com os trabalhos.

Trabalho pendente

O serviço AWS IoT Jobs publica uma mensagem em um tópico do MQTT quando um trabalho é adicionado ou removido da lista de execuções pendentes de um item ou quando a primeira execução de um trabalho na lista muda:

- `$aws/things/thingName/jobs/notify`
- `$aws/things/thingName/jobs/notify-next`

As mensagens contêm as seguintes cargas de exemplo:

`$aws/things/thingName/jobs/notify`:

```
{
  "timestamp" : 10011,
  "jobs" : {
    "IN_PROGRESS" : [ {
      "jobId" : "other-job",
      "queuedAt" : 10003,
      "lastUpdatedAt" : 10009,
      "executionNumber" : 1,
      "versionNumber" : 1
    } ],
    "QUEUED" : [ {
      "jobId" : "this-job",
      "queuedAt" : 10011,
      "lastUpdatedAt" : 10011,
      "executionNumber" : 1,
      "versionNumber" : 0
    } ]
  }
```

```
}
```

Se a execução do trabalho chamada `this-job` tiver origem em um trabalho com a configuração de agendamento opcional selecionada e a distribuição do documento de trabalho programada para ocorrer durante uma janela de manutenção, ela aparecerá somente durante uma janela de manutenção recorrente. Fora de uma janela de manutenção, o trabalho chamado `this-job` será excluído da lista de execuções de trabalhos pendentes, conforme mostrado no exemplo a seguir.

```
{
  "timestamp" : 10011,
  "jobs" : {
    "IN_PROGRESS" : [ {
      "jobId" : "other-job",
      "queuedAt" : 10003,
      "lastUpdatedAt" : 10009,
      "executionNumber" : 1,
      "versionNumber" : 1
    } ],
    "QUEUED" : []
  }
}
```

`$aws/things/`*`thingName`*`/jobs/notify-next`:

```
{
  "timestamp" : 10011,
  "execution" : {
    "jobId" : "other-job",
    "status" : "IN_PROGRESS",
    "queuedAt" : 10009,
    "lastUpdatedAt" : 10009,
    "versionNumber" : 1,
    "executionNumber" : 1,
    "jobDocument" : {"c":"d"}
  }
}
```

Se a execução do trabalho chamada `other-job` tiver origem em um trabalho com a configuração de agendamento opcional selecionada e a distribuição do documento de trabalho programada para ocorrer durante uma janela de manutenção, ela aparecerá somente durante uma janela de

manutenção recorrente. Fora de uma janela de manutenção, o trabalho chamado `other-job` não será listado como a próxima execução do trabalho, conforme mostrado no exemplo a seguir.

```
{} //No other pending jobs
```

```
{
  "timestamp" : 10011,
  "execution" : {
      "jobId" : "this-job",
      "queuedAt" : 10011,
      "lastUpdatedAt" : 10011,
      "executionNumber" : 1,
      "versionNumber" : 0,
      "jobDocument" : {"a":"b"}
  }
} // "this-job" is pending next to "other-job"
```

Os possíveis valores do status da execução do trabalho são `QUEUED`, `IN_PROGRESS`, `FAILED`, `SUCCEEDED`, `CANCELED`, `TIMED_OUT`, `REJECTED` e `REMOVED`.

A série de exemplos a seguir mostra as notificações publicadas em cada tópico quando as execuções de um trabalho são criadas e mudam de um estado para outro.

Primeiro, é criado um trabalho chamado `job1`. Essa notificação é publicada no tópico `jobs/notify`:

```
{
  "timestamp": 1517016948,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517016947,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

Essa notificação é publicada no tópico `jobs/notify-next`:

```
{
  "timestamp": 1517016948,
  "execution": {
    "jobId": "job1",
    "status": "QUEUED",
    "queuedAt": 1517016947,
    "lastUpdatedAt": 1517016947,
    "versionNumber": 1,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
    }
  }
}
```

Ao criar outro trabalho (`job2`), essa notificação é publicada no tópico `jobs/notify`:

```
{
  "timestamp": 1517017192,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517016947,
        "executionNumber": 1,
        "versionNumber": 1
      },
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

Não é publicada uma notificação no tópico `jobs/notify-next` porque o próximo trabalho da fila (`job1`) não sofreu alteração. Quando `job1` começa a executar, o status muda para `IN_PROGRESS`.

Não são publicadas notificações porque a lista de trabalhos e o próximo trabalho na fila não sofreram alterações.

Ao adicionar um terceiro trabalho (`job3`), essa notificação é publicada no tópico `jobs/notify`:

```
{
  "timestamp": 1517017906,
  "jobs": {
    "IN_PROGRESS": [
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517017472,
        "startedAt": 1517017472,
        "executionNumber": 1,
        "versionNumber": 2
      }
    ],
    "QUEUED": [
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      },
      {
        "jobId": "job3",
        "queuedAt": 1517017905,
        "lastUpdatedAt": 1517017905,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

Não é publicada uma notificação no tópico `jobs/notify-next` porque o próximo trabalho da fila ainda é `job1`.

Quando `job1` for concluído, o status mudará para `SUCCEEDED` e essa notificação será publicada no tópico `jobs/notify`:

```
{
  "timestamp": 1517186269,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      },
      {
        "jobId": "job3",
        "queuedAt": 1517017905,
        "lastUpdatedAt": 1517017905,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

A essa altura, `job1` já terá sido removido da fila e o próximo trabalho a ser executado será `job2`. Essa notificação é publicada no tópico `jobs/notify-next`:

```
{
  "timestamp": 1517186269,
  "execution": {
    "jobId": "job2",
    "status": "QUEUED",
    "queuedAt": 1517017191,
    "lastUpdatedAt": 1517017191,
    "versionNumber": 1,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
    }
  }
}
```

Se o `job3` precisar começar antes do `job2` (o que não é recomendado), o status do `job3` será alterado para `IN_PROGRESS`. Se isso acontecer, o `job2` não será mais o próximo na fila, e essa notificação será publicada no tópico `jobs/notify-next`:

```
{
  "timestamp": 1517186779,
  "execution": {
    "jobId": "job3",
    "status": "IN_PROGRESS",
    "queuedAt": 1517017905,
    "startedAt": 1517186779,
    "lastUpdatedAt": 1517186779,
    "versionNumber": 2,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
    }
  }
}
```

Nenhuma notificação é publicada no tópico `jobs/notify` porque nenhum trabalho foi adicionado ou removido.

Se o dispositivo rejeitar o `job2` e atualizar seu status para `REJECTED`, essa notificação será publicada no tópico `jobs/notify`:

```
{
  "timestamp": 1517189392,
  "jobs": {
    "IN_PROGRESS": [
      {
        "jobId": "job3",
        "queuedAt": 1517017905,
        "lastUpdatedAt": 1517186779,
        "startedAt": 1517186779,
        "executionNumber": 1,
        "versionNumber": 2
      }
    ]
  }
}
```

Se a exclusão do `job3` (que ainda está em andamento) for forçada, essa notificação será publicada no tópico `jobs/notify`:

```
{
  "timestamp": 1517189551,
  "jobs": {}
}
```

Nesse ponto, a fila está vazia. Essa notificação é publicada no tópico `jobs/notify-next`:

```
{
  "timestamp": 1517189551
}
```

AWS IoT operações de API de empregos

AWS IoT A API Jobs pode ser usada para qualquer uma das seguintes categorias:

- Tarefas administrativas, como gerenciamento e controle de trabalhos. Esse é o ambiente de gerenciamento.
- Dispositivos que realizam esses trabalhos. Esse é o plano de dados, que permite enviar e receber dados.

O gerenciamento e o controle de trabalhos usam uma API do protocolo HTTPS. Os dispositivos podem usar uma API MQTT ou de protocolo HTTPS. A API do ambiente de gerenciamento é projetada para um volume baixo de chamadas típicas durante a criação e o acompanhamento de trabalhos. Normalmente, ela abre uma conexão para uma única solicitação e, em seguida, fecha a conexão depois que a resposta é recebida. O plano de dados HTTPS e a API MQTT permitem pesquisas longas. Essas operações de API foram projetadas para grandes quantidades de tráfego que podem ser escaladas para milhões de dispositivos.

Cada API HTTPS do AWS IoT Jobs tem um comando correspondente que permite que você chame a API a partir do AWS Command Line Interface (AWS CLI). Os comandos são em letras minúsculas, com hifens entre as palavras que compõem o nome da API. Por exemplo, você pode chamar a API `CreateJob` na CLI, digitando:

```
aws iot create-job ...
```

No caso de erro durante uma operação, você recebe uma resposta de erro que contém as informações sobre o erro.

ErrorResponse

Contém informações sobre um erro que ocorreu durante uma operação do serviço Jobs da AWS IoT .

O exemplo a seguir mostra a sintaxe dessa operação:

```
{
    "code": "ErrorCode",
    "message": "string",
    "clientToken": "string",
    "timestamp": timestamp,
    "executionState": JobExecutionState
}
```

A seguir está uma descrição desse `ErrorResponse`:

`code`

ErrorCode pode ser configurado para:

InvalidTopic

A solicitação foi enviada para um tópico no namespace AWS IoT Jobs que não é mapeado para nenhuma operação de API.

InvalidJson

O conteúdo da solicitação não pôde ser interpretado como JSON codificado em UTF-8 válido.

InvalidRequest

O conteúdo da solicitação não era válido. Por exemplo, esse código é retornado quando uma solicitação `UpdateJobExecution` contém detalhes do status inválido. A mensagem contém detalhes sobre o erro.

InvalidStateTransition

Uma atualização tentou alterar a execução do trabalho para um estado que não é válido devido ao estado atual da execução do trabalho. Por exemplo, uma tentativa de alterar uma solicitação no estado SUCCEEDED para o estado IN_PROGRESS. Nesse caso, o corpo da mensagem de erro também contém o campo `executionState`.

ResourceNotFound

A `JobExecution` especificada pelo tópico da solicitação não existe.

VersionMismatch

A versão esperada especificada na solicitação não corresponde à versão da execução do trabalho no serviço AWS IoT Jobs. Nesse caso, o corpo da mensagem de erro também contém o campo `executionState`.

InternalError

Ocorreu um erro interno durante o processamento da solicitação.

RequestThrottled

A solicitação foi acelerada.

TerminalStateReached

Ocorre quando um comando para descrever um trabalho é executado em um trabalho que está em um estado terminal.

`message`

A sequência de uma mensagem de erro.

`clientToken`

Uma sequência arbitrária usada para correlacionar uma solicitação com sua resposta.

`timestamp`

O tempo, em segundos, desde a epoch.

`executionState`

Um objeto `JobExecutionState`. Esse campo é incluído apenas quando o campo `code` tem o valor `InvalidStateTransition` ou `VersionMismatch`. Nesses casos, torna-se desnecessário executar uma solicitação `DescribeJobExecution` separada para obter os dados do status da execução do trabalho atual.

A seguir estão listadas as operações e os tipos de dados da API Jobs.

- Tipos de dados e API de controle e gerenciamento de trabalhos
- Tipos de dados e operações da API MQTT e HTTPS do dispositivo de trabalhos

Tipos de dados e API de controle e gerenciamento de trabalhos

Os comandos a seguir estão disponíveis para aplicativos de gerenciamento e controle de trabalhos na CLI e por meio do protocolo HTTPS.

- Tipos de dados de gerenciamento e controle de trabalhos
- Operações de API de gerenciamento e controle de trabalhos

Para determinar o *endpoint-url* parâmetro para seus comandos da CLI, execute esse comando.

```
aws iot describe-endpoint --endpoint-type=iot:Jobs
```

Este comando retorna a seguinte saída.

```
{
"endpointAddress": "account-specific-prefix.jobs.iot.aws-region.amazonaws.com"
}
```

> **Note**
>
> O endpoint do serviço Jobs não é compatível com o `x-amzn-http-ca` ALPN.
> Se você estiver usando endpoints de pilha dupla (IPv6 e IPv6), use o endpoint. `iot:Data-ATS` Somente IPv4 o `iot:Jobs` endpoint oferece suporte.

Tipos de dados de gerenciamento e controle de trabalhos

Os tipos de dados a seguir são usados por aplicativos de gerenciamento e controle para se comunicar com AWS IoT Jobs.

Trabalho

O objeto `Job` contém detalhes sobre um trabalho. O exemplo a seguir mostra a sintaxe:

```
{
    "jobArn": "string",
    "jobId": "string",
    "status": "IN_PROGRESS|CANCELED|SUCCEEDED",
```

```
    "forceCanceled": boolean,
    "targetSelection": "CONTINUOUS|SNAPSHOT",
    "comment": "string",
    "targets": ["string"],
    "description": "string",
    "createdAt": timestamp,
    "lastUpdatedAt": timestamp,
    "completedAt": timestamp,
    "jobProcessDetails": {
        "processingTargets": ["string"],
        "numberOfCanceledThings": long,
        "numberOfSucceededThings": long,
        "numberOfFailedThings": long,
        "numberOfRejectedThings": long,
        "numberOfQueuedThings": long,
        "numberOfInProgressThings": long,
        "numberOfRemovedThings": long,
        "numberOfTimedOutThings": long
    },
    "presignedUrlConfig": {
        "expiresInSec": number,
        "roleArn": "string"
    },
    "jobExecutionsRolloutConfig": {
        "exponentialRate": {
           "baseRatePerMinute": integer,
           "incrementFactor": integer,
           "rateIncreaseCriteria": {
              "numberOfNotifiedThings": integer, // Set one or the other
              "numberOfSucceededThings": integer // of these two values.
           },
           "maximumPerMinute": integer
      }
    },
    "abortConfig": {
       "criteriaList": [
          {
             "action": "string",
             "failureType": "string",
             "minNumberOfExecutedThings": integer,
             "thresholdPercentage": integer
          }
       ]
    },
```

```
    "SchedulingConfig": {
      "startTime": string
      "endTime": string
      "timeZone": string


      "endTimeBehavior": string

   },
    "timeoutConfig": {
        "inProgressTimeoutInMinutes": long
    }
}
```

Para acessar mais informações, consulte Job ou job.

JobSummary

O objeto JobSummary contém um resumo do trabalho. O exemplo a seguir mostra a sintaxe:

```
{
    "jobArn": "string",
    "jobId": "string",
    "status": "IN_PROGRESS|CANCELED|SUCCEEDED|SCHEDULED",
    "targetSelection": "CONTINUOUS|SNAPSHOT",
    "thingGroupId": "string",
    "createdAt": timestamp,
    "lastUpdatedAt": timestamp,
    "completedAt": timestamp
}
```

Para acessar mais informações, consulte JobSummary ou job-summary.

JobExecution

O objeto JobExecution representa a execução de um trabalho em um dispositivo. O exemplo a seguir mostra a sintaxe:

> **Note**
>
> Quando você usa as operações da API do ambiente de gerenciamento, o tipo de dados JobExecution não contém um campo JobDocument. Para obter essas informações,

você pode usar a operação da API GetJobDocument ou o comando da CLI get-job-document.

```
{
    "approximateSecondsBeforeTimedOut": 50,
    "executionNumber": 1234567890,
    "forceCanceled": true|false,
    "jobId": "string",
    "lastUpdatedAt": timestamp,
    "queuedAt": timestamp,
    "startedAt": timestamp,
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|
REMOVED",
    "forceCanceled": boolean,
    "statusDetails": {
        "detailsMap": {
            "string": "string" ...
        },
        "status": "string"
    },
    "thingArn": "string",
    "versionNumber": 123
}
```

Para acessar mais informações, consulte JobExecution ou job-execution.

JobExecutionSummary

O objeto JobExecutionSummary contém informações resumidas sobre a execução do trabalho. O exemplo a seguir mostra a sintaxe:

```
{
    "executionNumber": 1234567890,
    "queuedAt": timestamp,
    "lastUpdatedAt": timestamp,
    "startedAt": timestamp,
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|REMOVED"
}
```

Para acessar mais informações, consulte JobExecutionSummary ou job-execution-summary.

JobExecutionSummaryForJob

O objeto `JobExecutionSummaryForJob` contém um resumo das informações sobre execuções de trabalho de um trabalho específico. O exemplo a seguir mostra a sintaxe:

```
{
    "executionSummaries": [
        {
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyThing",
            "jobExecutionSummary": {
                "status": "IN_PROGRESS",
                "lastUpdatedAt": 1549395301.389,
                "queuedAt": 1541526002.609,
                "executionNumber": 1
            }
        },
        ...
    ]
}
```

Para acessar mais informações, consulte `JobExecutionSummaryForJob` ou `job-execution-summary-for-job`.

JobExecutionSummaryForThing

O `JobExecutionSummaryForThing` objeto contém um resumo das informações sobre a execução de um trabalho em uma coisa específica. FThe o exemplo a seguir mostra a sintaxe:

```
{
    "executionSummaries": [
        {
            "jobExecutionSummary": {
                "status": "IN_PROGRESS",
                "lastUpdatedAt": 1549395301.389,
                "queuedAt": 1541526002.609,
                "executionNumber": 1
            },
            "jobId": "MyThingJob"
        },
        ...
    ]
}
```

Para acessar mais informações, consulte JobExecutionSummaryForThing ou job-execution-summary-for-thing.

Operações de API de gerenciamento e controle de trabalhos

Use um dos seguintes comandos da CLI ou operações da API:

AssociateTargetsWithJob

Associa um grupo a um trabalho contínuo. Os seguintes critérios devem ser atendidos:

- O trabalho deve ter sido criado com o campo `targetSelection` definido como `CONTINUOUS`.
- O status do trabalho deve ser `IN_PROGRESS`.
- O número total de destinos associados a um trabalho não deve ultrapassar 100.

HTTPS request

```
POST /jobs/jobId/targets

{
"targets": [ "string" ],
"comment": "string"
}
```

Para obter mais informações, consulte AssociateTargetsWithJob.

CLI syntax

```
aws iot  associate-targets-with-job \
--targets <value> \
--job-id <value> \
[--comment <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
"targets": [
"string"
],
```

```
"jobId": "string",
"comment": "string"
}
```

Para obter mais informações, consulte [associate-targets-with-job](#).

CancelJob

Cancela um trabalho.

HTTPS request

```
PUT /jobs/jobId/cancel

{
"force": boolean,
"comment": "string",
"reasonCode": "string"
}
```

Para obter mais informações, consulte [CancelJob](#).

CLI syntax

```
aws iot cancel-job \
    --job-id <value> \
    [--force <value>]  \
    [--comment <value>]  \
    [--reasonCode <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
    "jobId": "string",
    "force": boolean,
    "comment": "string"
}
```

Para obter mais informações, consulte [cancel-job](#).

CancelJobExecution

Cancela uma execução de trabalho em um dispositivo.

HTTPS request

```
PUT /things/thingName/jobs/jobId/cancel

{
"force": boolean,
"expectedVersion": "string",
"statusDetails": {
    "string": "string"
    ...
}
}
```

Para obter mais informações, consulte CancelJobExecution.

CLI syntax

```
aws iot cancel-job-execution \
--job-id <value> \
--thing-name <value> \
[--force | --no-force] \
[--expected-version <value>] \
[--status-details <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
"jobId": "string",
"thingName": "string",
"force": boolean,
"expectedVersion": long,
"statusDetails": {
"string": "string"
}
}
```

Para obter mais informações, consulte `cancel-job-execution`.

CreateJob

Cria um trabalho. Você pode fornecer o documento de trabalho como um link para um arquivo em um bucket do Amazon S3; (parâmetro `documentSource`) ou no corpo da solicitação (parâmetro `document`).

Um trabalho pode se tornar contínuo definindo o parâmetro opcional `targetSelection` como `CONTINUOUS` (o padrão é `SNAPSHOT`). Um trabalho contínuo pode ser usado para integrar ou atualizar dispositivos à medida que são adicionados a um grupo, pois ele continua em execução e é iniciado em itens recém-adicionados. Isso pode ocorrer mesmo após as objetos do grupo no momento em que o trabalho foi criado terem concluído o trabalho.

Um trabalho pode ter um opcional TimeoutConfig, que define o valor do cronômetro em andamento. O temporizador em andamento não pode ser atualizado e é aplicado a todas as execuções do trabalho.

As seguintes validações são realizadas em argumentos para a API `CreateJob`:

- O `targets` argumento deve ser uma lista de coisas ou grupos de coisas válidos ARNs. Todas as coisas e grupos de coisas devem estar em seu Conta da AWS.
- O argumento `documentSource` deve ser um URL válido do Amazon S3 para um documento de trabalho. O Amazon S3 URLs está no formato:. `https://s3.amazonaws.com/`*`bucketName`*`/`*`objectName`*
- O documento armazenado na URL especificada pelo argumento `documentSource` deve ser um documento JSON codificado em UTF-8.
- O tamanho de um documento de trabalho é limitado a 32 KB devido ao limite do tamanho de uma mensagem MQTT (128 KB) e da criptografia.
- `jobId`Deve ser único em seu Conta da AWS.

HTTPS request

```
PUT /jobs/jobId

{
"targets": [ "string" ],
"document": "string",
"documentSource": "string",
"description": "string",
```

```
"jobTemplateArn": "string",
"presignedUrlConfigData": {
    "roleArn": "string",
    "expiresInSec": "integer"
},
"targetSelection": "CONTINUOUS|SNAPSHOT",
"jobExecutionsRolloutConfig": {
    "exponentialRate": {
       "baseRatePerMinute": integer,
       "incrementFactor": integer,
       "rateIncreaseCriteria": {
          "numberOfNotifiedThings": integer, // Set one or the other
          "numberOfSucceededThings": integer // of these two values.
       },
       "maximumPerMinute": integer
  }
},
"abortConfig": {
   "criteriaList": [
      {
         "action": "string",
         "failureType": "string",
         "minNumberOfExecutedThings": integer,
         "thresholdPercentage": integer
      }
   ]
},
"SchedulingConfig": {
    "startTime": string
    "endTime": string
    "timeZone": string


    "endTimeBehavior": string

   }
"timeoutConfig": {
  "inProgressTimeoutInMinutes": long
}
}
```

Para obter mais informações, consulte `CreateJob`.

CLI syntax

```
aws iot create-job \
    --job-id <value> \
    --targets <value> \
    [--document-source <value>] \
    [--document <value>] \
    [--description <value>] \
    [--job-template-arn <value>] \
    [--presigned-url-config <value>] \
    [--target-selection <value>] \
    [--job-executions-rollout-config <value>] \
    [--abort-config <value>] \
    [--timeout-config <value>] \
    [--document-parameters <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
    "jobId": "string",
    "targets": [ "string" ],
    "documentSource": "string",
    "document": "string",
    "description": "string",
    "jobTemplateArn": "string",
    "presignedUrlConfig": {
        "roleArn": "string",
        "expiresInSec": long
     },
    "targetSelection": "string",
    "jobExecutionsRolloutConfig": {
          "exponentialRate": {
              "baseRatePerMinute": integer,
              "incrementFactor": integer,
              "rateIncreaseCriteria": {
                 "numberOfNotifiedThings": integer, // Set one or the other
                 "numberOfSucceededThings": integer // of these two values.
              },
      "maximumPerMinute": integer
      }
    },
```

```
    "abortConfig": {
    "criteriaList": [
        {
           "action": "string",
           "failureType": "string",
           "minNumberOfExecutedThings": integer,
           "thresholdPercentage": integer
         }
      ]
    },
    "timeoutConfig": {
          "inProgressTimeoutInMinutes": long
    },
    "documentParameters": {
    "string": "string"
    }
}
```

Para obter mais informações, consulte `create-job`.

DeleteJob

Exclui um trabalho e as execuções de trabalho correspondentes.

A exclusão de um trabalho pode levar tempo, dependendo do número de execuções criadas para o trabalho e de vários outros fatores. Enquanto o trabalho está sendo excluído, seu status é mostrado como "DELETION_IN_PROGRESS". A tentativa de excluir ou cancelar um trabalho cujo status já seja "DELETION_IN_PROGRESS" resulta em um erro.

HTTPS request

```
DELETE /jobs/jobId?force=force
```

Para obter mais informações, consulte `DeleteJob`.

CLI syntax

```
aws iot  delete-job \
--job-id <value> \
[--force | --no-force]  \
[--cli-input-json <value>] \
```

```
[--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
"jobId": "string",
"force": boolean
}
```

Para obter mais informações, consulte delete-job.

DeleteJobExecution

Exclui uma execução de trabalho.

HTTPS request

```
DELETE /things/thingName/jobs/jobId/executionNumber/executionNumber?force=force
```

Para obter mais informações, consulte DeleteJobExecution.

CLI syntax

```
aws iot  delete-job-execution \
--job-id <value> \
--thing-name <value> \
--execution-number <value> \
[--force | --no-force]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
"jobId": "string",
"thingName": "string",
"executionNumber": long,
"force": boolean
}
```

Para obter mais informações, consulte delete-job-execution.

DescribeJob

Obtém os detalhes da execução do trabalho.

HTTPS request

```
GET /jobs/jobId
```

Para obter mais informações, consulte DescribeJob.

CLI syntax

```
aws iot describe-job \
--job-id <value>  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
"jobId": "string"
}
```

Para obter mais informações, consulte describe-job.

DescribeJobExecution

Obtém os detalhes da execução de um trabalho. O status da execução do trabalho deve ser `SUCCEEDED` ou `FAILED`.

HTTPS request

```
GET /things/thingName/jobs/jobId?executionNumber=executionNumber
```

Para obter mais informações, consulte DescribeJobExecution.

CLI syntax

```
aws iot  describe-job-execution \
--job-id <value> \
```

```
--thing-name <value> \
[--execution-number <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
"jobId": "string",
"thingName": "string",
"executionNumber": long
}
```

Para obter mais informações, consulte `describe-job-execution`.

GetJobDocument

Obtém o documento de trabalho para um trabalho.

> **Note**
>
> Os espaços reservados não URLs são substituídos pelo Amazon URLs S3 pré-assinado no documento devolvido. Os pré-assinados URLs são gerados somente quando o serviço AWS IoT Jobs recebe uma solicitação pelo MQTT.

HTTPS request

```
GET /jobs/jobId/job-document
```

Para obter mais informações, consulte `GetJobDocument`.

CLI syntax

```
aws iot get-job-document \
--job-id <value>  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
"jobId": "string"
}
```

Para obter mais informações, consulte get-job-document.

ListJobExecutionsForJob

Obtém uma lista de execuções de trabalhos para um trabalho.

HTTPS request

```
GET /jobs/jobId/things?status=status&maxResults=maxResults&nextToken=nextToken
```

Para obter mais informações, consulte ListJobExecutionsForJob.

CLI syntax

```
aws iot  list-job-executions-for-job \
--job-id <value> \
[--status <value>] \
[--max-results <value>] \
[--next-token <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
"jobId": "string",
"status": "string",
"maxResults": "integer",
"nextToken": "string"
}
```

Para obter mais informações, consulte list-job-executions-for-job.

ListJobExecutionsForThing

Obtém uma lista de execuções de trabalhos para um objeto.

HTTPS request

```
GET /things/thingName/jobs?status=status&maxResults=maxResults&nextToken=nextToken
```

Para obter mais informações, consulte ListJobExecutionsForThing.

CLI syntax

```
aws iot list-job-executions-for-thing \
--thing-name <value> \
[--status <value>] \
[--max-results <value>] \
[--next-token <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
"thingName": "string",
"status": "string",
"maxResults": "integer",
"nextToken": "string"
}
```

Para obter mais informações, consulte list-job-executions-for-thing.

ListJobs

Obtém uma lista de empregos em seu Conta da AWS.

HTTPS request

```
GET /jobs?
status=status&targetSelection=targetSelection&thingGroupName=thingGroupName&thingGroupId=thi
```

Para obter mais informações, consulte ListJobs.

CLI syntax

```
aws iot list-jobs \
[--status <value>] \
```

```
[--target-selection <value>] \
[--max-results <value>] \
[--next-token <value>] \
[--thing-group-name <value>] \
[--thing-group-id <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
"status": "string",
"targetSelection": "string",
"maxResults": "integer",
"nextToken": "string",
"thingGroupName": "string",
"thingGroupId": "string"
}
```

Para obter mais informações, consulte `list-jobs`.

UpdateJob

Atualiza campos compatíveis do trabalho especificado. Valores atualizados para `timeoutConfig` entram em vigor somente para novas execuções em andamento. Atualmente, os lançamentos em andamento continuam sendo lançados com a configuração de tempo limite anterior.

HTTPS request

```
PATCH /jobs/jobId
{
"description": "string",
"presignedUrlConfig": {
  "expiresInSec": number,
  "roleArn": "string"
},
"jobExecutionsRolloutConfig": {
  "exponentialRate": {
     "baseRatePerMinute": number,
     "incrementFactor": number,
     "rateIncreaseCriteria": {
        "numberOfNotifiedThings": number,
```

```
        "numberOfSucceededThings": number
     },
  "maximumPerMinute": number
  },
"abortConfig": {
  "criteriaList": [
     {
        "action": "string",
        "failureType": "string",
        "minNumberOfExecutedThings": number,
        "thresholdPercentage": number
     }
  ]
},
"timeoutConfig": {
  "inProgressTimeoutInMinutes": number
}
}
```

Para obter mais informações, consulte UpdateJob.

CLI syntax

```
aws iot  update-job \
--job-id <value> \
[--description <value>] \
[--presigned-url-config <value>] \
[--job-executions-rollout-config <value>] \
[--abort-config <value>] \
[--timeout-config <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
"description": "string",
"presignedUrlConfig": {
  "expiresInSec": number,
  "roleArn": "string"
},
"jobExecutionsRolloutConfig": {
  "exponentialRate": {
```

```
      "baseRatePerMinute": number,
      "incrementFactor": number,
      "rateIncreaseCriteria": {
         "numberOfNotifiedThings": number,
         "numberOfSucceededThings": number
      }
   },
   "maximumPerMinute": number
},
"abortConfig": {
   "criteriaList": [
      {
         "action": "string",
         "failureType": "string",
         "minNumberOfExecutedThings": number,
         "thresholdPercentage": number
      }
   ]
},
"timeoutConfig": {
   "inProgressTimeoutInMinutes": number
}
}
```

Para obter mais informações, consulte [update-job](#).

Tipos de dados e operações da API MQTT e HTTPS do dispositivo de trabalhos

Os seguintes comandos estão disponíveis por meio dos protocolos MQTT e HTTPS. Use essas operações de API no plano de dados para dispositivos que executam os trabalhos.

Tipos de dados MQTT e HTTPS de dispositivos de trabalho

Os tipos de dados a seguir são usados para se comunicar com o serviço AWS IoT Jobs pelos protocolos MQTT e HTTPS.

JobExecution

O objeto `JobExecution` representa a execução de um trabalho em um dispositivo. O exemplo a seguir mostra a sintaxe:

> **Note**
>
> Quando você usa as operações da API do plano de dados MQTT e HTTP, o tipo de dados `JobExecution` contém um campo `JobDocument`. Seus dispositivos podem usar essas informações para recuperar o documento de trabalho de uma execução de trabalho.

```
{
    "jobId" : "string",
    "thingName" : "string",
    "jobDocument" : "string",
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|
REMOVED",
    "statusDetails": {
        "string": "string"
    },
    "queuedAt" : "timestamp",
    "startedAt" : "timestamp",
    "lastUpdatedAt" : "timestamp",
    "versionNumber" : "number",
    "executionNumber": long
}
```

Para acessar mais informações, consulte `JobExecution` ou `job-execution`.

JobExecutionState

O `JobExecutionState` contém informações sobre o estado da execução de um trabalho. O exemplo a seguir mostra a sintaxe:

```
{
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|
REMOVED",
    "statusDetails": {
        "string": "string"
        ...
    }
    "versionNumber": "number"
}
```

Para acessar mais informações, consulte `JobExecutionState` ou `job-execution-state`.

JobExecutionSummary

Contém um subconjunto de informações sobre a execução de um trabalho. O exemplo a seguir mostra a sintaxe:

```
{
    "jobId": "string",
    "queuedAt": timestamp,
    "startedAt": timestamp,
    "lastUpdatedAt": timestamp,
    "versionNumber": "number",
    "executionNumber": long
}
```

Para acessar mais informações, consulte `JobExecutionSummary` ou `job-execution-summary`.

Saiba mais sobre as operações da API de MQTT e HTTPS nas seções a seguir:

- Operações da API MQTT do dispositivo de trabalhos
- API HTTP do dispositivo do Jobs

Operações da API MQTT do dispositivo de trabalhos

Você pode emitir comandos do dispositivo de trabalhos publicando mensagens MQTT nos Tópicos reservados usados para comandos de trabalhos.

Seu cliente do lado do dispositivo deve estar inscrito nos tópicos das mensagens de resposta desses comandos. Se você usar o AWS IoT Device Client, seu dispositivo se inscreverá automaticamente nos tópicos de resposta. Isso indica que agente de mensagens publicará tópicos da mensagem de resposta para o cliente que publicou a mensagem de comando, independentemente de seu cliente ter assinado ou não os tópicos da mensagem de resposta. Essas mensagens de resposta não passam pelo agente de mensagens e não podem ser assinadas por outros clientes ou regras.

Ao assinar os tópicos de trabalhos e tópicos do evento `jobExecution` de sua solução de monitoramento de frota, primeiro habilite os eventos de trabalho e execução de trabalhos para receber quaisquer eventos no lado da nuvem. As mensagens de progresso do trabalho que são processadas por meio do agente de mensagens e podem ser usadas pelas regras de AWS IoT são publicadas como Eventos de trabalho. Como o agente de mensagens publica mensagens de resposta, mesmo sem uma assinatura explícita, seu cliente deve estar configurado para receber e

identificar as mensagens que recebe. Seu cliente também deve confirmar que o tópico *thingName* na mensagem recebida se aplica ao nome da coisa do cliente antes que o cliente aja na mensagem.

> **Note**
>
> As mensagens AWS IoT enviadas em resposta às mensagens de comando da API MQTT Jobs são cobradas em sua conta, independentemente de você ter se inscrito ou não explicitamente.

Veja a seguir as operações da API MQTT e sua sintaxe de solicitação e resposta. Todas as operações da API MQTT têm os seguintes parâmetros:

clientToken

Um token do cliente opcional usado para correlacionar solicitações e respostas. Insira um valor arbitrário aqui e ele será refletido na resposta.

`timestamp`

O tempo, em segundos, desde a epoch em que a mensagem foi enviada.

GetPendingJobExecutions

Obtém a lista de todos os trabalhos que não estão em um status terminal, para um objeto específica.

Para invocar essa API, publique uma mensagem em `$aws/things/`*thingName*`/jobs/get`.

Carga da solicitação:

```
{ "clientToken": "string" }
```

O agente de mensagens publicará `$aws/things/`*thingName*`/jobs/get/accepted` e `$aws/things/`*thingName*`/jobs/get/rejected` mesmo sem uma assinatura específica. No entanto, para que seu cliente receba as mensagens, ele deve estar ouvindo. Para obter mais informações, consulte a observação sobre mensagens de API do Jobs.

Carga da resposta:

```
{
```

```
"inProgressJobs" : [ JobExecutionSummary ... ],
"queuedJobs" : [ JobExecutionSummary ... ],
"timestamp" : 1489096425069,
"clientToken" : "client-001"
}
```

Onde `inProgressJobs` e `queuedJobs` retornam uma lista de objetos JobExecutionSummary que têm status de `IN_PROGRESS` ou `QUEUED`.

StartNextPendingJobExecution

Obtém e começa a próxima execução de trabalho pendente para um objeto (status `IN_PROGRESS` ou `QUEUED`).

- Todas as execuções de trabalho com o status `IN_PROGRESS` são retornadas primeiro.
- As execuções de trabalho são retornadas na ordem em que foram colocadas em fila. Quando um objeto for adicionada ou removida do grupo de destino do seu trabalho, confirme a ordem de distribuição de qualquer nova execução de trabalho em comparação com as execuções de trabalhos existentes.
- Se a execução do próximo trabalho pendente for `QUEUED`, seu estado será alterado para `IN_PROGRESS` e os detalhes do status da execução do trabalho serão definidos conforme especificado.
- Se a execução do próximo trabalho pendente já for `IN_PROGRESS`, os detalhes de seu status não serão alterados.
- Se nenhuma execução de trabalho estiver pendente, a resposta não incluirá o campo `execution`.
- Se desejar, você pode criar um temporizador de etapa definindo um valor para a propriedade `stepTimeoutInMinutes`. Se você não atualizar o valor dessa propriedade executando `UpdateJobExecution`, a execução do trabalho atingirá o tempo limite quando o temporizador de etapa expirar.

Para invocar essa API, publique uma mensagem em `$aws/things/`*`thingName`*`/jobs/start-next`.

Carga da solicitação:

```
{
"statusDetails": {
    "string": "job-execution-state"
```

```
    ...
},
"stepTimeoutInMinutes": long,
"clientToken": "string"
}
```

`statusDetails`

Uma coleção de pares nome e valor que descrevem o status da execução do trabalho. Se não especificado, o `statusDetails` não será alterado.

`stepTimeOutInMinutes`

Especifica o tempo que este dispositivo tem para concluir a execução do trabalho. Se o status de execução do trabalho não estiver definido como um estado terminal antes que o temporizador expire ou seja redefinido (ao chamar `UpdateJobExecution`, definir o status como `IN_PROGRESS` e especificar um novo valor de tempo limite no campo `stepTimeoutInMinutes`), o status de execução do trabalho será definido como `TIMED_OUT`. A configuração do tempo limite não tem efeito sobre o tempo limite de execução desse trabalho, que pode ter sido especificado quando o trabalho foi criado (`CreateJob` usando o campo `timeoutConfig`).

Os valores válidos para este parâmetro variam de 1 a 10.080 (1 minuto a 7 dias). Um valor de -1 também é válido e cancelará o cronômetro da etapa atual (criado por um uso anterior do UpdateJobExecutionRequest).

O agente de mensagens publicará `$aws/things/`*`thingName`*`/jobs/start-next/accepted` e `$aws/things/`*`thingName`*`/jobs/start-next/rejected` mesmo sem uma assinatura específica. No entanto, para que seu cliente receba as mensagens, ele deve estar ouvindo. Para obter mais informações, consulte a observação sobre mensagens de API do Jobs.

Carga da resposta:

```
{
"execution" : JobExecutionData,
"timestamp" : timestamp,
"clientToken" : "string"
}
```

Onde `execution` é um objeto JobExecution. Por exemplo:

```
{
"execution" : {
    "jobId" : "022",
    "thingName" : "MyThing",
    "jobDocument" : "< contents of job document >",
    "status" : "IN_PROGRESS",
    "queuedAt" : 1489096123309,
    "lastUpdatedAt" : 1489096123309,
    "versionNumber" : 1,
    "executionNumber" : 1234567890
},
"clientToken" : "client-1",
"timestamp" : 1489088524284,
}
```

DescribeJobExecution

Obtém informações detalhadas sobre uma execução de trabalho.

Você pode definir `jobId` como `$next` para retornar a próxima execução de trabalho pendente para um objeto (com status `IN_PROGRESS` ou `QUEUED`).

Para invocar essa API, publique uma mensagem em `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/get`.

Carga da solicitação:

```
{
"jobId" : "022",
"thingName" : "MyThing",
"executionNumber": long,
"includeJobDocument": boolean,
"clientToken": "string"
}
```

`thingName`

O nome do objeto associada ao dispositivo.

`jobId`

O identificador exclusivo atribuído a este trabalho quando ele foi criado.

Ou usar `$next` para retornar a próxima execução de trabalho pendente para um objeto (com status `IN_PROGRESS` ou `QUEUED`). Nesse caso, todas as execuções de trabalho com o status `IN_PROGRESS` são retornadas primeiro. As execuções de trabalho são retornadas na ordem em que foram criadas.

`executionNumber`

(Opcional) Um número que identifica a execução de um trabalho em um dispositivo. Se não especificado, a execução do trabalho mais recente será retornada.

`includeJobDocument`

(Opcional) A menos que definido como `false`, a resposta conterá o documento de trabalho. O padrão é `true`.

O agente de mensagens publicará `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/get/accepted` e `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/get/rejected` mesmo sem uma assinatura específica. No entanto, para que seu cliente receba as mensagens, ele deve estar ouvindo. Para obter mais informações, consulte a observação sobre mensagens de API do Jobs.

Carga da resposta:

```
{
"execution" : JobExecutionData,
"timestamp": "timestamp",
"clientToken": "string"
}
```

Onde `execution` é um objeto JobExecution.

UpdateJobExecution

Atualiza o status de uma execução de trabalho. Você pode criar um temporizador de etapa. Para isso, defina um valor para a propriedade `stepTimeoutInMinutes`. Se você não atualizar o valor dessa propriedade executando `UpdateJobExecution` novamente, a execução do trabalho atingirá o tempo limite quando o temporizador de etapa expirar.

Para invocar essa API, publique uma mensagem em `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/update`.

Carga da solicitação:

```
{
"status": "job-execution-state",
"statusDetails": {
    "string": "string"
    ...
},
"expectedVersion": "number",
"executionNumber": long,
"includeJobExecutionState": boolean,
"includeJobDocument": boolean,
"stepTimeoutInMinutes": long,
"clientToken": "string"
}
```

`status`

O novo status da execução do trabalho (`IN_PROGRESS`, `FAILED`, `SUCCEEDED` ou`REJECTED`). Isso deve ser especificado em todas as atualizações.

`statusDetails`

Uma coleção de pares nome e valor que descrevem o status da execução do trabalho. Se não especificado, o `statusDetails` não será alterado.

`expectedVersion`

A versão esperada atual da execução do trabalho. Cada vez que você atualiza a execução do trabalho, sua versão é incrementada. Se a versão da execução do trabalho armazenada no serviço AWS IoT Jobs não corresponder, a atualização será rejeitada com um `VersionMismatch` erro. Um ErrorResponse que contém os dados atuais do status de execução do trabalho também é retornado. Isso torna desnecessário executar uma solicitação `DescribeJobExecution` separada para obter os dados do status da execução do trabalho.

`executionNumber`

(Opcional) Um número que identifica a execução de um trabalho em um dispositivo. Se não especificado, a execução do trabalho mais recente será usada.

`includeJobExecutionState`

(Opcional) Quando incluído e definido como `true`, a resposta conterá o campo `JobExecutionState`. O padrão é `false`.

`includeJobDocument`

(Opcional) Quando incluído e definido como `true`, a resposta conterá o `JobDocument`. O padrão é `false`.

`stepTimeoutInMinutes`

Especifica o tempo que este dispositivo tem para concluir a execução do trabalho. Se o status de execução do trabalho não for definido como um estado terminal antes que o temporizador expire ou antes que o temporizador seja redefinido, o status da execução do trabalho será definido como `TIMED_OUT`. A configuração ou a reconfiguração do tempo limite não tem efeito sobre o tempo limite de execução do trabalho que pode ter sido especificado quando o trabalho foi criado.

O agente de mensagens publicará `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/update/accepted` e `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/update/rejected` mesmo sem uma assinatura específica. No entanto, para que seu cliente receba as mensagens, ele deve estar ouvindo. Para obter mais informações, consulte a observação sobre mensagens de API do Jobs.

Carga da resposta:

```
{
"executionState": JobExecutionState,
"jobDocument": "string",
"timestamp": timestamp,
"clientToken": "string"
}
```

`executionState`

Um objeto JobExecutionState.

`jobDocument`

Um objeto documento de trabalho.

> **Note**
>
> Nas respostas do MQTT, o campo `jobDocument` é um objeto JSON. Nas respostas HTTP, é uma representação de string do objeto JSON.

`timestamp`

O tempo, em segundos, desde a epoch em que a mensagem foi enviada.

`clientToken`

Um token do cliente usado para correlacionar solicitações e respostas.

Ao usar o protocolo MQTT, você também pode realizar as seguintes atualizações:

JobExecutionsChanged

Enviado sempre que uma execução de trabalho é adicionada ou removida da lista de execuções de trabalhos pendentes para um objeto.

Use o tópico :

`$aws/things/thingName/jobs/notify`

Carga da mensagem:

```
{
"jobs" : {
    "JobExecutionState": [ JobExecutionSummary ... ]
         },
    "timestamp": timestamp
}
```

NextJobExecutionChanged

Enviado sempre que houver uma alteração em qual execução de trabalho é a seguinte na lista de execuções de trabalhos pendentes para um objeto, conforme definido para `DescribeJobExecution` com o `jobId` `$next`. Essa mensagem não é enviada quando os detalhes da próxima execução de trabalho são alterados, apenas quando o próximo trabalho que seria retornado por `DescribeJobExecution` com o `jobId` `$next` tiver sido alterado. Considere as execuções de trabalho J1 e J2 com status `QUEUED`. J1 é a próxima na lista de execuções de trabalhos pendentes. Se o status de J2 for alterado para `IN_PROGRESS` enquanto o estado de J1 permanecer inalterado, essa notificação será enviada e conterá os detalhes do J2.

Use o tópico :

`$aws/things/thingName/jobs/notify-next`

Carga da mensagem:

```
{
"execution" : JobExecution,
"timestamp": timestamp,
}
```

API HTTP do dispositivo do Jobs

Os dispositivos podem se comunicar com AWS IoT Jobs usando a assinatura HTTP versão 4 na porta 443. Esse é o método usado pela AWS SDKs e CLI. Para obter mais informações sobre essas ferramentas, consulte Referência de AWS CLI comandos: iot-jobs-data ou AWS SDKs e Ferramentas.

Os seguintes comandos estão disponíveis para dispositivos que executam os trabalhos. Para obter informações sobre o uso de operações de API com o protocolo MQTT, consulte Operações da API MQTT do dispositivo de trabalhos.

GetPendingJobExecutions

Obtém a lista de todos os trabalhos que não estão em um status terminal, para um objeto específica.

HTTPS request

```
GET /things/thingName/jobs
```

Resposta:

```
{
"inProgressJobs" : [ JobExecutionSummary ... ],
"queuedJobs" : [ JobExecutionSummary ... ]
}
```

Para obter mais informações, consulte `GetPendingJobExecutions`.

CLI syntax

```
aws iot-jobs-data get-pending-job-executions \
--thing-name <value>  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
"thingName": "string"
}
```

Para obter mais informações, consulte `get-pending-job-executions`.

StartNextPendingJobExecution

Obtém e começa a próxima execução de trabalho pendente para um objeto (com um status de `IN_PROGRESS` ou `QUEUED`).

- Todas as execuções de trabalho com o status `IN_PROGRESS` são retornadas primeiro.
- As execuções de trabalho são retornadas na ordem em que foram criadas.
- Se a execução do próximo trabalho pendente for `QUEUED`, seu status será alterado para `IN_PROGRESS` e os detalhes do status da execução do trabalho serão definidos conforme especificado.
- Se a execução do próximo trabalho pendente já for `IN_PROGRESS`, os detalhes de seu status não se alteram.
- Se nenhuma execução de trabalho estiver pendente, a resposta não incluirá o campo `execution`.
- Se desejar, você pode criar um temporizador de etapa definindo um valor para a propriedade `stepTimeoutInMinutes`. Se você não atualizar o valor dessa propriedade executando `UpdateJobExecution`, a execução do trabalho atingirá o tempo limite quando o temporizador de etapa expirar.

HTTPS request

O exemplo a seguir mostra a sintaxe da solicitação:

```
PUT /things/thingName/jobs/$next
{
"statusDetails": {
    "string": "string"
    ...
},
"stepTimeoutInMinutes": long
}
```

Para obter mais informações, consulte StartNextPendingJobExecution.

CLI syntax

Resumo:

```
aws iot-jobs-data  start-next-pending-job-execution \
--thing-name <value> \
{--step-timeout-in-minutes <value>] \
[--status-details <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
"thingName": "string",
"statusDetails": {
"string": "string"
},
"stepTimeoutInMinutes": long
}
```

Para obter mais informações, consulte start-next-pending-job-execution.

DescribeJobExecution

Obtém informações detalhadas sobre uma execução de trabalho.

Você pode definir o `jobId` como `$next` para retornar a próxima execução de trabalho pendente para um objeto. O status da execução do trabalho deve ser `QUEUED` ou `IN_PROGRESS`.

HTTPS request

Solicitação:

```
GET /things/thingName/jobs/jobId?
executionNumber=executionNumber&includeJobDocument=includeJobDocument
```

Resposta:

```
{
```

```
"execution" : JobExecution,
}
```

Para obter mais informações, consulte DescribeJobExecution.

CLI syntax

Resumo:

```
aws iot-jobs-data  describe-job-execution \
--job-id <value> \
--thing-name <value> \
[--include-job-document | --no-include-job-document] \
[--execution-number <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
"jobId": "string",
"thingName": "string",
"includeJobDocument": boolean,
"executionNumber": long
}
```

Para obter mais informações, consulte describe-job-execution.

UpdateJobExecution

Atualiza o status de uma execução de trabalho. Se desejar, você pode criar um temporizador de etapa definindo um valor para a propriedade `stepTimeoutInMinutes`. Se você não atualizar o valor dessa propriedade executando `UpdateJobExecution` novamente, a execução do trabalho atingirá o tempo limite quando o temporizador de etapa expirar.

HTTPS request

Solicitação:

```
POST /things/thingName/jobs/jobId
{
"status": "job-execution-state",
```

```
"statusDetails": {
    "string": "string"
    ...
},
"expectedVersion": "number",
"includeJobExecutionState": boolean,
"includeJobDocument": boolean,
"stepTimeoutInMinutes": long,
"executionNumber": long
}
```

Para obter mais informações, consulte [UpdateJobExecution](#).

CLI syntax

Resumo:

```
aws iot-jobs-data  update-job-execution \
--job-id <value> \
--thing-name <value> \
--status <value> \
[--status-details <value>] \
[--expected-version <value>] \
[--include-job-execution-state | --no-include-job-execution-state] \
[--include-job-document | --no-include-job-document] \
[--execution-number <value>]  \
[--cli-input-json <value>] \
[--step-timeout-in-minutes <value>] \
[--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
"jobId": "string",
"thingName": "string",
"status": "string",
"statusDetails": {
"string": "string"
},
"stepTimeoutInMinutes": number,
"expectedVersion": long,
"includeJobExecutionState": boolean,
"includeJobDocument": boolean,
"executionNumber": long
```

```
}
```

Para obter mais informações, consulte `update-job-execution`.

Protegendo usuários e dispositivos com AWS IoT o Jobs

Para autorizar os usuários a usar AWS IoT Jobs com seus dispositivos, você deve conceder permissões a eles usando as políticas do IAM. Em seguida, os dispositivos devem ser autorizados usando AWS IoT Core políticas para se conectar com segurança AWS IoT, receber execuções de trabalhos e atualizar o status da execução.

Tipo de política necessária para AWS IoT empregos

A tabela a seguir mostra os diferentes tipos de políticas que você deve usar para autorização. Para obter mais informações sobre a política necessária a ser usada, consulte Autorização.

Tipo de política necessária

Caso de uso	Protocol	Autenticação	plane/data Plano de controle	Tipo de identidade	Tipo de política necessária
Autorize um administrador, operador ou serviço de nuvem a trabalhar com segurança com trabalhos	HTTPS	AWS Autenticação Signature versão 4 (porta 443)	Tanto ambiente de gerenciamento como plano de dados	Identidade do Amazon Cognito, IAM ou usuário federado	Política do IAM
Autorize seu dispositivo de IoT a funcionar de forma	MQTT/HTTPS	Autenticação mútua TCP ou TLS (porta 8883 ou 443)	Plano de dados	Certificados X.509	AWS IoT Core política

Caso de uso	Protocol	Autenticação	plane/dat a Plano de controle	Tipo de identidade	Tipo de política necessária
segura com trabalhos					

Para autorizar operações de AWS IoT Jobs que podem ser executadas no plano de controle e no plano de dados, você deve usar políticas do IAM. As identidades devem ter sido autenticadas com AWS IoT para realizar essas operações, que devem ser Identidades do Amazon Cognito ou Usuários, grupos e funções do IAM. Para obter mais informações sobre a autenticação, consulte Autenticação.

Agora, os dispositivos devem ser autorizados no plano de dados usando AWS IoT Core políticas para se conectar com segurança ao gateway do dispositivo. O gateway do dispositivo permite que os dispositivos se comuniquem com segurança AWS IoT, recebam execuções de trabalhos e atualizem o status de execução do trabalho. A comunicação do dispositivo é protegida usando protocolos seguros MQTT ou Publicação HTTPS de comunicação. Esses protocolos usam Certificados do cliente X.509 os fornecidos por AWS IoT para autenticar as conexões do dispositivo.

Veja a seguir como você autoriza seus usuários, serviços de nuvem e dispositivos a usar o AWS IoT Jobs. Para obter mais informações sobre as operações de API do ambiente de gerenciamento e plano de dados, consulte AWS IoT operações de API de empregos.

Tópicos

- Autorizando usuários e serviços em nuvem a usar o Jobs AWS IoT
- Autorizando seus dispositivos a usar AWS IoT Jobs com segurança no plano de dados

Autorizando usuários e serviços em nuvem a usar o Jobs AWS IoT

Para autorizar seus usuários e serviços em nuvem, você deve usar políticas do IAM no ambiente de gerenciamento e no plano de dados. As políticas devem ser usadas com o protocolo HTTPS e devem usar a autenticação AWS Signature Version 4 (porta 443) para autenticar os usuários.

> **Note**
>
> AWS IoT Core as políticas não devem ser usadas no plano de controle. Somente as políticas do IAM são usadas para autorizar usuários ou serviços em nuvem. Para obter mais informações sobre a política necessária a ser usada, consulte Tipo de política necessária para AWS IoT empregos.

As políticas do IAM são documentos JSON que contêm declarações de política. As declarações de política usam os elementos Efeito, Ação e Recurso para especificar recursos, ações permitidas ou negadas, além das condições sob as quais as ações são permitidas ou negadas. Para obter mais informações, consulte Referência de elementos de política JSON do IAM no Guia do usuário do IAM.

> **Warning**
>
> Recomendamos que você não use permissões curinga, como `"Action": ["iot:*"]` em suas políticas ou AWS IoT Core políticas do IAM. Usar permissões curinga não é uma prática recomendada de segurança. Para obter mais informações, consulte Política de AWS IoT excessivamente permissiva.

Políticas do IAM no ambiente de gerenciamento

No ambiente de gerenciamento, as políticas do IAM usam o prefixo `iot:` com a ação para autorizar a operação correspondente da API de trabalhos. Por exemplo, a ação de política `iot:CreateJob` concede ao usuário permissão para usar a API `CreateJob`.

Ações de políticas

A tabela a seguir mostra uma lista de ações da política do IAM e permissões para utilizar as ações da API. Para obter informações sobre tipos de recursos, consulte Tipos de recursos definidos por AWS IoT. Para obter mais informações sobre AWS IoT ações, consulte Ações definidas por AWS IoT.

Ações de política do IAM no ambiente de gerenciamento

Ação de política	Operação de API	Resource types	Description
`iot:AssociateTargetsWithJob`	AssociateTargetsWithJob	• trabalho • objeto • thinggroup	Representa a permissão para associar um grupo a um trabalho contínuo. A permissão `iot:AssociateTargetsWithJob` é verificada sempre que é feita uma solicitação para associar destinos.
`iot:CancelJob`	CancelJob	trabalho	Representa a permissão para cancelar um trabalho. A permissão `iot:CancelJob` é verificada sempre que é feita uma solicitação para cancelar um trabalho.
`iot:CancelJobExecution`	CancelJobExecution	• trabalho • objeto	Representa a permissão para cancelar uma execução de trabalho. A permissão `iot: CancelJobExecution` é verificada sempre que é feita uma solicitação para cancelar uma execução de trabalho.
`iot:CreateJob`	CreateJob	• trabalho • objeto • thinggroup • jobtemplate • pacote	Representa a permissão para criar um trabalho. A permissão `iot: CreateJob` é verificada sempre que é feita uma solicitação para criar um trabalho.
`iot:CreateJobTemplate`	CreateJobTemplate	• trabalho • jobtemplate • pacote	Representa a permissão para criar um modelo de trabalho. A permissão `iot: CreateJobTemplate` é verificada sempre que é feita uma solicitação para criar um modelo de trabalho.

Ação de política	Operação de API	Resource types	Description
iot:DeleteJob	DeleteJob	trabalho	Representa a permissão para excluir um trabalho. A permissão iot: DeleteJob é verificada sempre que é feita uma solicitação para excluir um trabalho.
iot:DeleteJobTemplate	DeleteJobTemplate	jobtemplate	Representa a permissão para excluir um modelo de trabalho. A permissão iot: CreateJobTemplate é verificada sempre que é feita uma solicitação para excluir um modelo de trabalho.
iot:DeleteJobExecution	DeleteJobTemplate	• trabalho • objeto	Representa a permissão para excluir uma execução de trabalho. A permissão iot: DeleteJobExecution é verificada sempre que é feita uma solicitação para excluir uma execução de trabalho.
iot:DescribeJob	DescribeJob	trabalho	Representa a permissão para descrever um trabalho. A permissão iot: DescribeJob é verificada sempre que é feita uma solicitação para descrever um trabalho.
iot:DescribeJobExecution	DescribeJobExecution	• trabalho • objeto	Representa a permissão para descrever uma execução de trabalho. A permissão iot: DescribeJobExecution é verificada sempre que é feita uma solicitação para descrever uma execução de trabalho.
iot:DescribeJobTemplate	DescribeJobTemplate	jobtemplate	Representa a permissão para descrever um modelo de trabalho. A permissão iot: DescribeJobTemplate é verificada sempre que é feita uma solicitação para descrever um modelo de trabalho.

Ação de política	Operação de API	Resource types	Description
iot:DescribeManagedJobTemplate	DescribeManagedJobTemplate	jobtemplate	Representa a permissão para descrever um modelo de trabalho gerenciado. A permissão iot: DescribeManagedJobTemplate é verificada sempre que é feita uma solicitação para descrever um modelo de trabalho gerenciado.
iot:GetJobDocument	GetJobDocument	trabalho	Representa a permissão para obter o documento de trabalho de um trabalho. A permissão iot:GetJobDocument é verificada sempre que é feita uma solicitação para obter um documento de trabalho.
iot:ListJobExecutionsForJob	ListJobExecutionsForJob	trabalho	Representa a permissão para listar as execuções de trabalho para um trabalho. A permissão iot:ListJobExecutionsForJob é verificada sempre que é feita uma solicitação para listar as execuções de trabalho para um trabalho.
iot:ListJobExecutionsForThing	ListJobExecutionsForThing	objeto	Representa a permissão para listar as execuções de trabalho para um trabalho. A permissão iot:ListJobExecutionsForThing é verificada sempre que é feita uma solicitação para listar as execuções de trabalho para um objeto.
iot:ListJobs	ListJobs	nenhuma	Representa a permissão para listar os trabalhos. A permissão iot:ListJobs é verificada sempre que é feita uma solicitação para listar os trabalhos.

Ação de política	Operação de API	Resource types	Description
`iot:ListJobTemplates`	ListJobTemplates	nenhuma	Representa a permissão para listar os modelos de trabalho. A permissão `iot:ListJobTemplates` é verificada sempre que é feita uma solicitação para listar os modelos de trabalho.
`iot:ListManagedJobTemplates`	ListManagedJobTemplates	nenhuma	Representa a permissão para listar os modelos de trabalho gerenciados. A permissão `iot:ListManagedJobTemplates` é verificada sempre que é feita uma solicitação para listar os modelos de trabalho gerenciados.
`iot:UpdateJob`	UpdateJob	trabalho	Representa a permissão para atualizar um trabalho. A permissão `iot:UpdateJob` é verificada sempre que é feita uma solicitação para atualizar um trabalho.
`iot:TagResource`	TagResource	• trabalho • jobtemplate • objeto	Concede permissão para atribuir uma tag a um recurso específico.
`iot:UntagResource`	UntagResource	• trabalho • jobtemplate • objeto	Concede permissão para remover uma tag de um recurso específico.

Exemplos básicos de políticas do IAM

O exemplo a seguir mostra uma política do IAM que concede ao usuário permissão para realizar as seguintes ações para suo objeto ou grupo de objetos de IoT.

No exemplo, substitua:

- *region*com o seu Região da AWS, como`us-east-1`.
- *account-id*com seu Conta da AWS número, como`57EXAMPLE833`.
- *thing-group-name*com o nome do seu grupo de coisas de IoT para o qual você está segmentando trabalhos, como. `FirmwareUpdateGroup`
- *thing-name*com o nome da sua coisa de IoT para a qual você está segmentando trabalhos, como. `MyIoTThing`

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "iot:CreateJobTemplate",
                "iot:CreateJob"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:us-east-1:123456789012:thinggroup/thing-group-
name"
        },
        {
            "Action": [
                "iot:DescribeJob",
                "iot:CancelJob",
                "iot:DeleteJob"
            ],
            "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:job/*"
        },
        {
            "Action": [
                "iot:DescribeJobExecution",
                "iot:CancelJobExecution",
                "iot:DeleteJobExecution"
            ],
            "Effect": "Allow",
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:thing/thing-123",
                "arn:aws:iot:us-east-1:123456789012:job/*"
            ]
        }
```

```
    ]
}
```

Exemplo de política do IAM para autorização baseada em IP

Você pode impedir que as entidades principais façam chamadas de API para o endpoint do seu ambiente de gerenciamento a partir de endereços IP específicos. Para especificar os endereços IP que podem ser permitidos, no elemento Condição da sua política do IAM, use a chave de condição global `aws:SourceIp`.

O uso dessa chave de condição também pode impedir que outras AWS service (Serviço da AWS) pessoas façam essas chamadas de API em seu nome, como AWS CloudFormation. Para permitir o acesso a esses serviços, use a chave de condição `aws:ViaAWSService`global com a SourceIp chave aws:. Isso garante que a restrição de acesso ao endereço IP de origem se aplique somente a solicitações feitas diretamente por uma entidade principal. Para obter mais informações, consulte AWS: Nega acesso AWS com base no IP de origem.

O exemplo a seguir mostra como permitir somente um endereço IP específico que pode fazer chamadas de API para o endpoint do ambiente de gerenciamento. A chave `aws:ViaAWSService` está definida como `true`, o que permite que outros serviços façam chamadas de API em seu nome.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:CreateJobTemplate",
                "iot:CreateJob"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                 "IpAddress": {
                     "aws:SourceIp": "123.45.167.89"
                 },
                 "Bool": {
                     "aws:ViaAWSService": "false"
        }
```

```
        }
    }]
}
```

Políticas do IAM no plano de dados

As políticas do IAM no plano de dados usam o prefixo `iotjobsdata:` para autorizar trabalhos e operações de API que os usuários podem realizar. No plano de dados, você pode conceder permissão ao usuário para usar a API `DescribeJobExecution` usando a ação de política `iotjobsdata:DescribeJobExecution`.

> **Warning**
>
> Não é recomendável usar políticas do IAM no plano de dados ao segmentar trabalhos de AWS IoT para seus dispositivos. Recomendamos que você use políticas do IAM no ambiente de gerenciamento para que os usuários criem e gerenciem trabalhos. No plano de dados, para autorizar dispositivos a recuperar execuções de trabalhos e atualizar o status de execução, use AWS IoT Core políticas para o protocolo HTTPS.

Exemplos básicos de políticas do IAM

As operações de API que devem ser autorizadas geralmente são executadas digitando comandos da CLI. Veja a seguir um exemplo de um usuário realizando uma operação `DescribeJobExecution`.

No exemplo, substitua:

- *`region`*com o seu Região da AWS, como`us-east-1`.
- *`account-id`*com seu Conta da AWS número, como`57EXAMPLE833`.
- *`thing-name`*com o nome da sua coisa de IoT para a qual você está segmentando trabalhos, como. `myRegisteredThing`
- *`job-id`* é o identificador exclusivo do trabalho que é direcionado usando a API.

```
aws iot-jobs-data describe-job-execution \
    --endpoint-url "https://account-id.jobs.iot.region.amazonaws.com" \
    --job-id jobID --thing-name thing-name
```

Veja a seguir um exemplo de política do IAM que autoriza essa ação:

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": [
            "iotjobsdata:DescribeJobExecution"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:thing/thing-123"
    }
}
```

Exemplos de política do IAM para autorização baseada em IP

Você pode impedir que as entidades principais façam chamadas de API para o endpoint do seu plano de dados a partir de endereços IP específicos. Para especificar os endereços IP que podem ser permitidos, no elemento Condição da sua política do IAM, use a chave de condição global `aws:SourceIp`.

O uso dessa chave de condição também pode impedir que outras AWS service (Serviço da AWS) pessoas façam essas chamadas de API em seu nome, como AWS CloudFormation. Para permitir o acesso a esses serviços, use a chave de condição global `aws:ViaAWSService` com a chave de condição `aws:SourceIp`. Isso garante que a restrição de acesso ao endereço IP se aplique somente a solicitações feitas diretamente pela entidade principal. Para obter mais informações, consulte AWS: Nega acesso AWS com base no IP de origem.

O exemplo a seguir mostra como permitir somente um endereço IP específico que pode fazer chamadas de API para o endpoint do plano de dados.

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iotjobsdata:*"
        ],
        "Resource": [
            "*"
```

```
        ],
        "Condition": {
            "IpAddress": {
                "aws:SourceIp": "123.45.167.89"
        },
            "Bool": {
                "aws:ViaAWSService": "false"
        }
        }
    }]
}
```

O exemplo a seguir mostra como impedir que endereços IP ou intervalos de endereços específicos façam chamadas de API para o endpoint do plano de dados.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iotjobsdata:*"
            ],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "123.45.167.89",
                        "192.0.2.0/24",
                        "203.0.113.0/24"
                    ]
            }
        },
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Exemplo de política do IAM para ambiente de gerenciamento e plano de dados

Se você executar uma operação de API no ambiente de gerenciamento e no plano de dados, a ação da política do ambiente de gerenciamento deverá usar o prefixo `iot:` e a ação da política do plano de dados deverá usar o prefixo `iotjobsdata:`.

Por exemplo, a API `DescribeJobExecution` pode ser usada tanto no ambiente de gerenciamento quanto no plano de dados. No plano de controle, a DescribeJobExecutionAPI é usada para descrever a execução de um trabalho. No plano de dados, a DescribeJobExecutionAPI é usada para obter detalhes da execução de um trabalho.

A política do IAM a seguir autoriza o usuário a usar a API `DescribeJobExecution` no ambiente de gerenciamento e no plano de dados.

No exemplo, substitua:

- *`region`*com o seu Região da AWS, como`us-east-1`.
- *`account-id`*com seu Conta da AWS número, como`57EXAMPLE833`.
- *`thing-name`*com o nome da sua coisa de IoT para a qual você está segmentando trabalhos, como. `MyIoTThing`

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "iotjobsdata:DescribeJobExecution"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:us-east-1:123456789012:thing/thing-123"
        },
        {
            "Action": [
                "iot:DescribeJobExecution",
                "iot:CancelJobExecution",
                "iot:DeleteJobExecution"
            ],
            "Effect": "Allow",
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:thing/thing-123",
```

```
    "arn:aws:iot:us-east-1:123456789012:job/*"
            ]
        }
    ]
}
```

Autorize a marcação de recursos de IoT

Para um melhor controle sobre trabalhos e modelos de trabalho que você pode criar, modificar ou usar, você pode anexar tags aos trabalhos ou modelos de trabalho. As tags também ajudam você a discernir a propriedade, atribuir e alocar custos, colocando-os em grupos de cobrança e anexando tags a eles.

Quando um usuário quiser marcar seus trabalhos ou modelos de trabalho criados usando o Console de gerenciamento da AWS ou o AWS CLI, sua política do IAM deve conceder ao usuário permissões para marcá-los. Para conceder permissões, sua política do IAM; deve usar a ação `iot:TagResource`.

> **Note**
>
> Se sua política do IAM não incluir a ação `iot:TagResource`, qualquer `CreateJob` ou `CreateJobTemplate` com uma tag exibirá um erro `AccessDeniedException`.

Quando você quiser marcar seus trabalhos ou modelos de trabalho que você criou usando o Console de gerenciamento da AWS ou o AWS CLI, sua política do IAM deve conceder permissão para marcá-los. Para conceder permissões, sua política do IAM; deve usar a ação `iot:TagResource`.

Para obter informações gerais sobre a marcação de recursos, consulte Marcando seus recursos AWS IoT.

Exemplo de política do IAM

Consulte os seguintes exemplos de políticas do IAM que concedem permissões de marcação:

Exemplo 1

Um usuário que executa o comando a seguir para criar um trabalho e marcá-lo em um ambiente específico.

Neste exemplo, substitua:

- *region*com o seu Região da AWS, como`us-east-1`.
- *account-id*com seu Conta da AWS número, como`57EXAMPLE833`.
- *thing-name*com o nome da sua coisa de IoT para a qual você está segmentando trabalhos, como. `MyIoTThing`

```
aws iot create-job
    --job-id test_job
    --targets "arn:aws:iot:region:account-id:thing/thingOne"
    --document-source "https://s3.amazonaws.com/amzn-s3-demo-bucket/job-document.json"
    --description "test job description"
    --tags Key=environment,Value=beta
```

Para este exemplo, você deve usar a seguinte política do IAM:

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": [
            "iot:CreateJob",
            "iot:CreateJobTemplate",
            "iot:TagResource"
        ],
        "Effect": "Allow",
        "Resource": [
           "arn:aws:iot:us-east-1:123456789012:job/*",
           "arn:aws:iot:us-east-1:123456789012:jobtemplate/*"
        ]
    }
}
```

Autorizando seus dispositivos a usar AWS IoT Jobs com segurança no plano de dados

Para autorizar seus dispositivos a interagir de forma segura com AWS IoT Jobs no plano de dados, você deve usar políticas. AWS IoT Core AWS IoT Core políticas para trabalhos são documentos JSON contendo declarações de políticas. Essas políticas também usam elementos Efeito, Ação e Recurso e seguem uma convenção semelhante às políticas do IAM. Para obter mais informações

sobre os elementos, consulte Referência de elementos de política JSON do IAM no Guia do usuário do IAM.

As políticas podem ser usadas com os protocolos MQTT e HTTPS e devem usar a autenticação mútua TCP ou TLS para autenticar os dispositivos. Veja a seguir como usar essas políticas nos diferentes protocolos de comunicação.

> **Warning**
>
> Recomendamos que você não use permissões curinga, como `"Action": ["iot:*"]` em suas políticas ou AWS IoT Core políticas do IAM. Usar permissões curinga não é uma prática recomendada de segurança. Para obter mais informações, consulte Política de AWS IoT excessivamente permissiva.

AWS IoT Core políticas para o protocolo MQTT

AWS IoT Core as políticas do protocolo MQTT concedem a você permissões para usar as ações da API MQTT do dispositivo de tarefas. As operações da API MQTT são usadas para trabalhar com tópicos do MQTT reservados para comandos de trabalho. Para obter mais informações sobre essas operações de API, consulte Operações da API MQTT do dispositivo de trabalhos.

As políticas do MQTT usam ações de política como `iot:Connect`, `iot:Publish`, `iot:Subscribe` e `iot:Receieve` para trabalhar com os tópicos de trabalho. Essas políticas permitem que você se conecte ao agente de mensagens, assine os tópicos do MQTT de trabalhos e envie e receba mensagens do MQTT entre seus dispositivos e a nuvem. Para obter mais informações sobre essas ações, consulte AWS IoT Core ações políticas.

Para obter informações sobre tópicos para AWS IoT trabalhos, consulteTópicos de trabalhos.

Exemplos básicos de políticas do MQTT

O exemplo a seguir mostra como você pode usar `iot:Publish` e `iot:Subscribe` para publicar e assinar trabalhos e execuções de trabalhos.

No exemplo, substitua:

- *`region`*com o seu Região da AWS, com`us-east-1`.
- *`account-id`*com seu Conta da AWS número, com`57EXAMPLE833`.

- *thing-name*com o nome da sua coisa de IoT para a qual você está segmentando trabalhos, como. `MyIoTThing`

```
{
    "Version":"2012-10-17",

    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Subscribe"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/$aws/events/job/*",
    "arn:aws:iot:us-east-1:123456789012:topic/$aws/events/jobExecution/*",
    "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/thing-123/jobs/*"
            ]
        }
    ]
}
```

AWS IoT Core políticas para o protocolo HTTPS

AWS IoT Core as políticas no plano de dados também podem usar o protocolo HTTPS com o mecanismo de autenticação TLS para autorizar seus dispositivos. No plano de dados, as políticas usam o prefixo `iotjobsdata:` para autorizar operações de API de trabalhos que os dispositivos podem realizar. Por exemplo, a ação de política `iotjobsdata:DescribeJobExecution` concede ao usuário permissão para usar a API `DescribeJobExecution`.

> **Note**
>
> As ações da política do plano de dados devem usar o prefixo `iotjobsdata:`. No ambiente de gerenciamento, as ações devem usar o prefixo `iot:`. Para ver um exemplo de política do IAM quando as ações de política do ambiente de gerenciamento e do plano de dados são usadas, consulte Exemplo de política do IAM para ambiente de gerenciamento e plano de dados.

Ações de políticas

A tabela a seguir mostra uma lista de ações AWS IoT Core políticas e permissões para autorizar dispositivos a usar as ações da API. Para obter uma lista das operações de API que você pode realizar no plano de dados, consulte API HTTP do dispositivo do Jobs.

> **Note**
>
> Essas ações da política de execução de trabalhos se aplicam apenas ao endpoint HTTP TLS. Se você usar o endpoint MQTT, deverá usar as ações de política do MQTT definidas anteriormente.

AWS IoT Core ações políticas no plano de dados

Ação de política	Operação de API	Resourc types	Description
`iotjobsdata:DescribeJobExecution`	DescribeJobExecution	• trabal • objeto	Representa a permissão para recuperar uma execução de trabalho. A permissão `iotjobsdata:DescribeJobExecution` é verificada sempre que é feita uma solicitação para recuperar uma execução de trabalho.
`iotjobsdata:GetPendingJobExecutions`	GetPendingJobExecutions	objeto	Representa a permissão para recuperar a lista de trabalhos que não estão em status terminal para um objeto. A permissão `iotjobsdata:GetPendingJobExecutions` é verificada sempre que é feita uma solicitação para recuperar a lista.
`iotjobsdata:StartN`	StartNextPendingJobExecution	objeto	Representa a permissão para obter e iniciar a próxima execução de trabalho pendente para um objeto. Isto é, para atualizar uma

Ação de política	Operação de API	Resourc types	Description
extPendin gJobExecution			execução de trabalho com status QUEUED para IN_PROGRESS . A permissão iotjobsdata:StartN extPendingJobExecution é verificada sempre que é feita uma solicitação para iniciar a próxima execução de trabalho pendente.
iotjobsda ta:Update JobExecution	UpdateJob Execution	objeto	Representa a permissão para atualizar uma execução de trabalho. A permissão iotjobsdata:Update JobExecution é verificada sempre que é feita uma solicitaç ão para atualizar o estado de uma execução de trabalho.

Exemplo básico de política

Veja a seguir um exemplo de uma AWS IoT Core política que concede permissão para realizar as ações nas operações da API do plano de dados para qualquer recurso. Você pode definir o escopo de sua política para um recurso específico, como um objeto de IoT. No seu exemplo, substitua:

- *region*com seus Região da AWS , comous-east-1.
- *account-id*com seu Conta da AWS número, como57EXAMPLE833.
- *thing-name*com o nome da coisa de IoT, como. MyIoTthing

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "iotjobsdata:GetPendingJobExecutions",
                "iotjobsdata:StartNextPendingJobExecution",
                "iotjobsdata:DescribeJobExecution",
```

```
                "iotjobsdata:UpdateJobExecution"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:us-east-1:123456789012:thing/thing-123"
        }
    ]
}
```

Um exemplo de quando você deve usar essas políticas pode ser quando seus dispositivos de IoT usam uma política AWS IoT Core para acessar uma dessas operações de API, como o exemplo a seguir da API `DescribeJobExecution`:

```
GET /things/thingName/jobs/jobId?
executionNumber=executionNumber&includeJobDocument=includeJobDocument&namespaceId=namespaceId
 HTTP/1.1
```

AWS IoT Limites de trabalhos

AWS IoT O Jobs tem cotas ou limites de serviço que correspondem ao número máximo de recursos ou operações de serviço para você Conta da AWS.

Tópicos

- Limites de execuções de trabalhos
- Limites de trabalho ativos e simultâneos

Limites de execuções de trabalhos

Esta seção fornece informações sobre os limites de execução de trabalhos para o AWS IoT Device Management.

> **Note**
>
> Esses limites não fazem parte das cotas de serviço que você pode encontrar na documentação sobre Service Quotas do AWS IoT Device Management.

Para obter informações sobre o número de execuções de trabalhos pendentes, você pode usar a `GetPendingJobExecutions` API ou assinar os tópicos reservados do MQTT para AWS IoT trabalhos e receber. Tipos de notificação de trabalhos

O número de execuções de trabalhos pendentes em sua conta pode variar dependendo de a configuração de agendamento estar ou não habilitada e se você usa uma janela de manutenção recorrente.

Número máximo de execuções de trabalhos pendentes

Nome da notificação/ API	Description	Sem configuração de agendamento	Com configuração de agendamento
`ListNotification`	A `ListNotification` é publicada sempre que uma execução de trabalho antiga entra em um status terminal, ou quando uma nova execução de trabalho é colocada na fila ou muda para um status não terminal. Ela pode exibir até 15 execuções de trabalhos pendentes que estão `QUEUED` ou `IN_PROGRESS` .	10	15 (Somente até 5 execuções de trabalhos aparecem na `ListNotification` durante a janela de manutenção).
`GetPendingJobExecutions`	Quando você invoca a API `GetPendingJobExecutions` , ela exibe uma lista de execuções de trabalhos que ainda não foram iniciadas e que podem ser iniciadas após a chamada de API. A API pode exibir até dez execuções de trabalhos pendentes no máximo. • Das dez execuções de trabalhos pendentes, as execuções `IN_PROGRESS` serão filtradas do resultado. • Das dez execuções de trabalhos pendentes, se seus trabalhos	10	15

Nome da notificação/ API	Description	Sem configuração de agendamento	Com configuração de agendamento
	estiverem no status `SCHEDULED` , eles serão filtrados do resultado.		

Limites de trabalho ativos e simultâneos

Esta seção ajudará você a aprender mais sobre trabalhos ativos e simultâneos e os limites que se aplicam a eles.

Trabalhos ativos e limite de trabalhos ativos

Quando você cria um trabalho usando o AWS IoT console ou a `CreateJob` API, o status do trabalho muda para`IN_PROGRESS`. Todos os trabalhos em andamento são trabalhos ativos e contam para o limite de trabalhos ativos. Isso inclui trabalhos que estão implementando novas execuções de trabalhos ou trabalhos que aguardam que os dispositivos concluam suas execuções de trabalho. Esse limite se aplica a trabalhos contínuos e de snapshot.

Trabalhos simultâneos e limite de simultaneidade de trabalhos

Os trabalhos em andamento que estão implementando novas execuções de trabalhos ou que estão cancelando execuções de trabalhos criados anteriormente são trabalhos simultâneos e contam para o limite de simultaneidade de trabalhos. AWS IoT Os trabalhos podem ser implementados e cancelados rapidamente a uma taxa de 1.000 dispositivos por minuto. Cada trabalho é `concurrent` e conta para o limite de simultaneidade de trabalhos somente por um curto período de tempo. Depois que as execuções do trabalho forem implementadas ou canceladas, o trabalho não é mais simultâneo e não conta para o limite de simultaneidade do trabalho. Você pode usar a simultaneidade de trabalhos para criar um grande número de trabalhos enquanto espera que os dispositivos concluam a execução do trabalho.

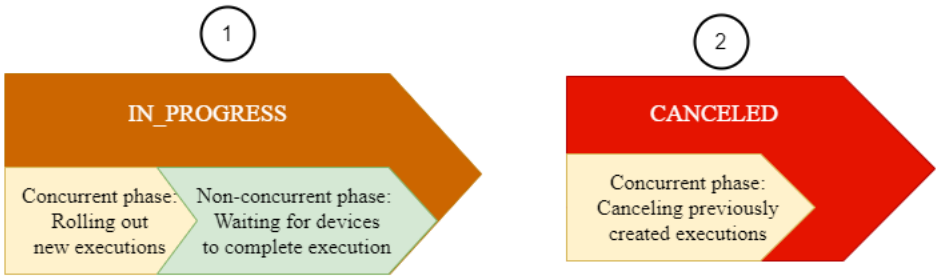
> **Note**
>
> Se um trabalho com a configuração opcional de agendamento e a distribuição do documento de trabalho programados para ocorrer durante uma janela de manutenção atingir o

`startTime` selecionado e você atingir o limite máximo de simultaneidade de trabalhos, esse trabalho programado passará para um estado de status de `CANCELED`.

Para determinar se um trabalho é simultâneo, você pode usar a `IsConcurrent` propriedade de um trabalho no AWS IoT console ou usando a `ListJob` API `DescribeJob` ou. Esse limite se aplica a trabalhos contínuos e de snapshot.

Para ver os trabalhos ativos e os limites de simultaneidade de trabalhos e outras cotas de AWS IoT trabalhos para você Conta da AWS e para solicitar um aumento de limite, consulte Pontos de extremidade e cotas de gerenciamento de AWS IoT dispositivos no. Referência geral da AWS

O diagrama a seguir mostra como a simultaneidade de trabalhos se aplica aos trabalhos em andamento e aos trabalhos que estão sendo cancelados.



Note

Novos trabalhos com `SchedulingConfig` opcional manterão um estado de status inicial `SCHEDULED` e serão atualizados para `IN_PROGRESS` ao atingirem o `startTime` selecionado. Depois que o novo trabalho com o `SchedulingConfig` opcional atingir o `startTime` selecionado e for atualizado para `IN_PROGRESS`, ele será contabilizado no limite de trabalhos ativos e no limite de simultaneidade de trabalhos. Os trabalhos com um estado de status de `SCHEDULED` contarão para o limite de trabalhos ativos, mas não contarão para o limite de simultaneidade de trabalhos.

A tabela a seguir mostra os limites que se aplicam aos trabalhos ativos e simultâneos e às fases simultâneas e não simultâneas dos estados do trabalho.

Limites de trabalho ativos e simultâneos

Status do trabalho	Fase	Limite de trabalhos ativos	Limite de simultaneidade de trabalhos
`SCHEDULED`	Fase não simultânea: o AWS IoT Jobs aguarda o agendamento `startTime` do trabalho para iniciar as notificações de execução do trabalho em seus dispositivos. Os trabalhos nessa fase contam apenas para o limite de trabalhos ativos e terão a propriedade `IsConcurrent` definida como falsa.	Aplica-se	Não se aplica
`IN_PROGRESS`	Fase simultânea: AWS IoT Jobs aceita a solicitação de criação do trabalho e começa a distribuir notificações de execução do trabalho em seus dispositivos. Os trabalhos nessa fase são simultâneos, conforme indicado pela propriedade `IsConcurrent` definida como verdadeira, e contam tanto para os trabalhos ativos quanto para os limites de simultaneidade de trabalhos.	Aplica-se	Aplica-se
	Fase não simultânea: os AWS IoT trabalhos aguardam que os dispositivos relatem os resultados de suas execuções de trabalhos. Os trabalhos nessa fase contam apenas para o limite de trabalhos ativos e terão a propriedade `IsConcurrent` definida como falsa.	Aplica-se	Não se aplica
`Canceled`	Fase simultânea: AWS IoT Jobs aceita a solicitação de cancelamento do trabalho e começa a cancelar as execuções de trabalhos criados anteriormente para	Não se aplica	Aplica-se

Status do trabalho	Fase	Limite de trabalhos ativos	Limite de simultaneidade de trabalhos
	seus dispositivos. Os trabalhos nessa fase são simultâneos e terão a proprieda de `IsConcurrent` definida como verdadeira. Uma vez que as execuções do trabalho tenham sido canceladas, o trabalho não é mais simultâneo e não conta para o limite de simultaneidade do trabalho.		

> **Note**
>
> A duração máxima de uma janela de manutenção recorrente é de 23 horas e 50 minutos.

AWS IoT Device Management comandos

> **Important**
>
> Esta documentação descreve como você pode usar o recurso de comandos no AWS IoT Device Management. Para acessar informações sobre o uso desse recurso para AWS IoT FleetWise, consulte Comandos remotos.
> Você é o único responsável por implantar comandos de forma segura e compatível com as leis aplicáveis. Para ter mais informações sobre suas responsabilidades, consulte os Termos de Serviço da AWS para Serviços da AWS IoT.

Use AWS IoT Device Management Comandos para enviar uma instrução da nuvem para um dispositivo conectado AWS IoT a. Os comandos têm como alvo um dispositivo por vez e podem ser usados para aplicativos de baixa latência e alto rendimento, como recuperar registros do lado do dispositivo ou iniciar uma alteração no estado do dispositivo.

O comando é um recurso reutilizável gerenciado pelo. AWS IoT Device Management Ele contém configurações que são aplicadas antes de serem publicadas no dispositivo. Você pode predefinir um conjunto de comandos para casos de uso específicos, como acender uma lâmpada ou destrancar a porta de um veículo.

O recurso AWS IoT Comandos permite que você:

- Crie modelos de comando reutilizáveis com cargas estáticas ou dinâmicas e execute-os em dispositivos específicos.
- Dispositivos de destino registrados como AWS IoT itens ou clientes MQTT não registrados.
- Execute vários comandos simultaneamente no mesmo dispositivo.
- Ative as notificações de eventos e acompanhe o status da execução à medida que os dispositivos processam os comandos.

Os tópicos a seguir mostram como criar comandos, enviá-los ao seu dispositivo e recuperar o status informado pelo dispositivo.

Tópicos

- Início rápido

- Conceitos e status de comandos
- Fluxo de trabalho de comandos de alto nível
- Criar e gerenciar comandos
- Iniciar e monitorar execuções de comando
- Suspender um recurso de comando

Início rápido

Conclua estas etapas para enviar seu primeiro comando:

1. Pré-requisitos - Certifique-se de estar integrado ao IoT Core, a um dispositivo conectado ao IoT Core, às permissões do IAM para operações da API Commands e a uma biblioteca de cliente MQTT instalada em seu dispositivo.
2. Crie um comando - Defina um comando com uma carga estática ou um modelo de carga que especifique as ações do dispositivo. Consulte Criar um recurso de comando.
3. Identifique seu dispositivo - Registre seu dispositivo como uma coisa de IoT ou anote o ID do cliente MQTT. Consulte Considerações sobre dispositivos de destino.
4. Configurar permissões — configure políticas do IAM para que seu dispositivo receba comandos e publique respostas.
5. Inscrever-se em tópicos - configure seu dispositivo para assinar tópicos de solicitação e resposta de comandos. Consulte Escolher o dispositivo de destino para os comandos e assinar os tópicos do MQTT.
6. Execute o comando - Inicie a execução do comando no dispositivo de destino. Consulte Iniciar uma execução de comando.

Casos de uso comuns:

- Diagnóstico remoto - recupere registros, execute diagnósticos, verifique a integridade do dispositivo
- Atualizações de configuração - Alterar configurações, atualizar parâmetros, alternar recursos
- Controle do dispositivo -Lock/unlock, power on/off, mudanças de modo

Conceitos e status de comandos

Use AWS IoT Comandos para enviar instruções da nuvem para os dispositivos conectados. Para usar esse atributo:

1. Crie um comando com uma carga contendo as configurações necessárias para execução no dispositivo.
2. Especifique o dispositivo de destino que receberá a carga e executará as ações.
3. Execute o comando no dispositivo de destino e recupere as informações de status. Para solucionar problemas, consulte CloudWatch os registros.

Para obter mais informações sobre esse fluxo de trabalho, consulte Fluxo de trabalho de comandos de alto nível.

Tópicos

- Conceitos de chaves de comandos
- Estados de comando
- Status de execução do comando

Conceitos de chaves de comandos

Os conceitos-chave a seguir ajudam você a entender o recurso Comandos. Os termos são usados de forma consistente em toda esta documentação:

- Comando - Um modelo reutilizável que define as instruções do dispositivo
- Execução - Uma instância de um comando em execução em um dispositivo
- Nome da coisa - Identificador para dispositivos registrados no registro de IoT
- ID do cliente - identificador MQTT para dispositivos não registrados
- Carga útil - Os dados da instrução enviados aos dispositivos
- Tópico - Canal MQTT para comunicação de comando

Comandos

Os comandos são instruções enviadas da nuvem para seus dispositivos de IoT como mensagens MQTT. Depois de receber a carga, os dispositivos processam as instruções e realizam as ações

correspondentes, como modificar as configurações, transmitir leituras do sensor ou carregar registros. Os dispositivos então retornam os resultados para a nuvem, permitindo monitoramento e controle remotos.

Namespace

Ao criar um comando, especifique seu namespace. Para AWS IoT Device Management comandos, use o `AWS-IoT` namespace padrão e forneça uma carga ou um payloadTemplate. Para AWS IoT FleetWise comandos, use o `AWS-IoT-FleetWise` namespace. Para obter mais informações, consulte Comandos remotos no Guia do AWS IoT FleetWise desenvolvedor.

Carga útil

Ao criar um comando, forneça uma carga estática que defina as ações que o dispositivo deve realizar. A carga pode usar qualquer formato compatível. Para garantir que os dispositivos interpretem corretamente a carga, recomendamos especificar o tipo de formato da carga útil. Os dispositivos que usam o MQTT5 protocolo podem seguir o padrão MQTT para identificar o formato. Os indicadores de formato para JSON ou CBOR estão disponíveis no tópico de solicitação de comandos.

Modelo de carga

Um modelo de carga útil define uma carga de comando com espaços reservados que geram cargas diferentes em tempo de execução com base nos valores de parâmetros fornecidos por você. Por exemplo, em vez de criar cargas separadas para diferentes valores de temperatura, crie um modelo com um espaço reservado de temperatura e especifique o valor durante a execução. Isso elimina a manutenção de várias cargas úteis semelhantes.

Dispositivo de destino

Para executar um comando, especifique um dispositivo de destino usando o nome do item (para dispositivos registrados com AWS IoT) ou o ID do cliente MQTT (para dispositivos não registrados). O ID do cliente é um identificador exclusivo definido no MQTT protocolo usado para conectar dispositivos AWS IoT a. Para obter detalhes, consulte Considerações sobre dispositivos de destino.

Tópicos de comandos

Antes de executar um comando, os dispositivos devem se inscrever no tópico de solicitação de comandos. Quando você executa um comando, a carga útil é enviada para o dispositivo neste tópico. Após a execução, os dispositivos publicam os resultados e o status no tópico de resposta dos comandos. Para obter mais informações, consulte Tópicos de comandos.

Execução de comandos

Uma execução é uma instância de um comando em execução em um dispositivo de destino. Quando você inicia uma execução, a carga é entregue ao dispositivo e uma ID de execução exclusiva é gerada. O dispositivo executa o comando e relata o progresso para AWS IoT. A lógica do lado do dispositivo determina o comportamento de execução e os relatórios de status para tópicos reservados.

Condições de valor do parâmetro

Ao criar comandos com modelos de carga útil, defina as condições de valor para validar os valores dos parâmetros antes da execução. As condições de valor garantem que os parâmetros atendam aos requisitos, evitando execuções inválidas.

Operadores suportados por CommandParameterValuetipo

Tipos numéricos (INTEGER, LONG, DOUBLE, UNSIGNEDLONG)

- `EQUALS`- O valor deve ser igual ao número especificado
- `NOT_EQUALS`- O valor não deve ser igual ao número especificado
- `GREATER_THAN`- O valor deve ser maior que o número especificado
- `GREATER_THAN_EQUALS`- O valor deve ser maior ou igual ao número especificado
- `LESS_THAN`- O valor deve ser menor que o número especificado
- `LESS_THAN_EQUALS`- O valor deve ser menor ou igual ao número especificado
- `IN_RANGE`- O valor deve estar dentro do intervalo especificado (inclusive)
- `NOT_IN_RANGE`- O valor deve estar fora do intervalo especificado (inclusive)
- `IN_SET`- O valor deve corresponder a um dos números especificados
- `NOT_IN_SET`- O valor não deve corresponder a nenhum dos números especificados

Tipo de string (STRING)

- `EQUALS`- O valor deve ser igual à string especificada
- `NOT_EQUALS`- O valor não deve ser igual à string especificada
- `IN_SET`- O valor deve corresponder a uma das cadeias de caracteres especificadas
- `NOT_IN_SET`- O valor não deve corresponder a nenhuma das strings especificadas

Tipo booleano

- As condições de valor não são suportadas

Tipo binário

- As condições de valor não são suportadas

Exemplo: comando de controle de temperatura

```
{
  "commandId": "SetTemperature",
  "namespace": "AWS-IoT",
  "payloadTemplate": "{\"temperature\":
 \"${aws:iot:commandexecution::parameter:temperature}\"}",
  "parameters": [
    {
      "name": "temperature",
      "type": "INTEGER",
      "valueConditions": [
        {
          "comparisonOperator": "IN_RANGE",
          "operand": {
            "numberRange": {
              "min": "60",
              "max": "80"
            }
          }
        }
      ]
    }
  ]
}
```

Neste exemplo, o `temperature` parâmetro deve estar entre 60 e 80 (inclusive). Solicitações de execução com valores fora desse intervalo falham na validação.

> **Note**
>
> As condições de valor são avaliadas na invocação da StartCommandExecution API. As validações com falha retornam um erro e impedem a criação da execução.

Prioridade e avaliação do valor do parâmetro

Ao iniciar as execuções de comandos com modelos de carga útil, os valores dos parâmetros são resolvidos usando a seguinte prioridade:

1. Parâmetros da solicitação de execução - Os valores fornecidos na `StartCommandExecution` solicitação têm a maior prioridade
2. Valores padrão do comando - Se um parâmetro não for fornecido na solicitação de execução, o parâmetro `defaultValue` será usado
3. Sem valor - Se nenhum for fornecido, a execução falhará como parâmetro necessário para gerar a solicitação de execução

As condições de valor são avaliadas no valor final do parâmetro derivado acima na prioridade e antes da criação da execução. Se a validação falhar, a solicitação de execução retornará um erro.

Exemplo: SetTemperature comando com **`defaultValue`**

```
{
  "parameters": [
    {
      "name": "temperature",
      "type": "INTEGER",
      "defaultValue": {"I": 72},
      "valueConditions": [
        {
          "comparisonOperator": "IN_RANGE",
          "operand": {"numberRange": {"min": "60", "max": "80"}}
        }
      ]
    }
  ]
}
```

Ao iniciar a execução:

- Se você fornecer `"temperature": {"I": 75}` na solicitação, 75 será usado
- Se você omitir o parâmetro de temperatura, o valor padrão 72 será usado
- Ambos os valores são validados em relação à condição de intervalo [60,80]

Estados de comando

Os comandos no seu Conta da AWS podem estar em um dos três estados: Disponível, Obsoleto ou Exclusão pendente.

Disponível

Após a criação bem-sucedida, um comando fica no estado Disponível e pode ser executado em dispositivos.

Preterido

Marque comandos para suspensão de uso quando não forem mais necessários. Comandos obsoletos não podem iniciar novas execuções, mas as execuções pendentes continuam sendo concluídas. Para habilitar novas execuções, restaure o comando para o estado Disponível.

Exclusão pendente

Quando você marca um comando para exclusão, ele é excluído automaticamente se for descontinuado por mais tempo do que o tempo limite máximo (padrão: 12 horas). Essa ação é permanente. Se não for preterido ou for preterido por menos do que o tempo limite, o comando entrará no estado de exclusão pendente e será removido após o tempo limite expirar.

Status de execução do comando

Quando você inicia uma execução em um dispositivo de destino, ela entra no `CREATED` status e pode fazer a transição para outros status com base nos relatórios do dispositivo. Você pode recuperar informações de status e acompanhar as execuções.

> **Note**
>
> Você pode executar vários comandos simultaneamente em um dispositivo. Use o controle de concorrência para limitar as execuções por dispositivo e evitar sobrecargas. Para obter o máximo de execuções simultâneas por dispositivo, consulte cotas de AWS IoT Device Management comandos.

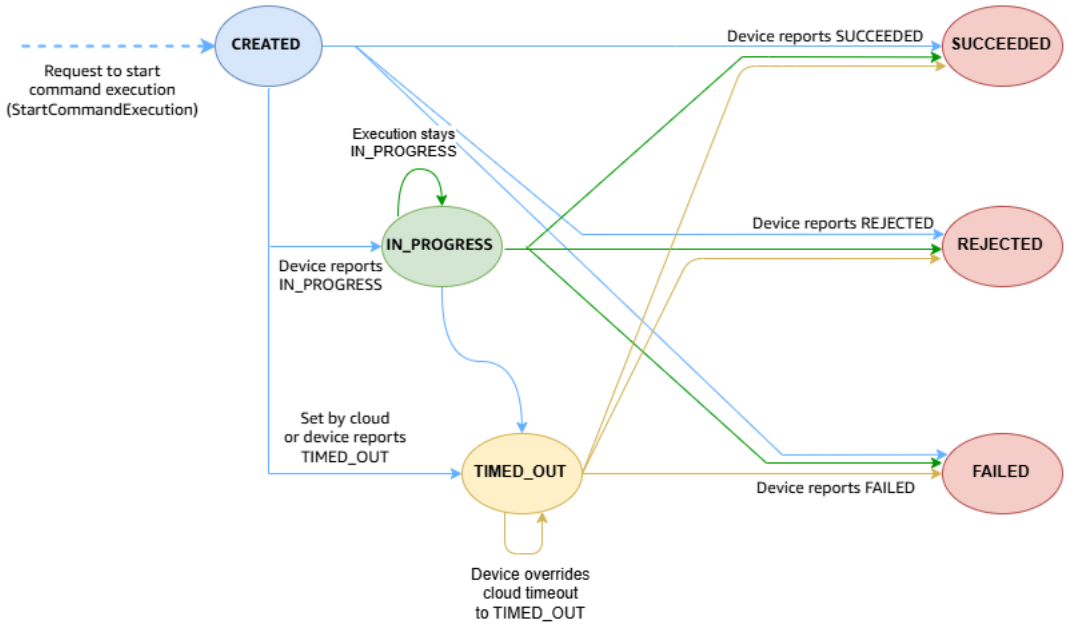
A tabela a seguir mostra os status de execução e suas transições com base no progresso da execução.

Status e origem da execução do comando

Status de execução do comando	Iniciado pelo dispositivo/ nuvem?	Execução do terminal?	Transições de status permitidas
`CREATED`	Nuvem	Não	• IN_PROGRESS • SUCCEEDED • FAILED • REJEITADA • TEMPORIZADO_LIMITE
`IN_PROGRESS`	Dispositivo	Não	• IN_PROGRESS • SUCCEEDED • FAILED • REJEITADA • TEMPORIZADO_LIMITE
`TIMED_OUT`	Dispositivo e nuvem	Não	• SUCCEEDED • FAILED • REJEITADA • TEMPORIZADO_LIMITE
`SUCCEEDED`	Dispositivo	Sim	Não aplicável
`FAILED`	Dispositivo	Sim	Não aplicável
`REJECTED`	Dispositivo	Sim	Não aplicável

Os dispositivos podem publicar atualizações de status e resultados a qualquer momento usando comandos reservados aos tópicos do MQTT. Para fornecer contexto adicional, os dispositivos podem usar `reasonCode` `reasonDescription` campos no `statusReason` objeto.

O diagrama a seguir mostra as transições de status de execução.



> **Note**
>
> Quando não AWS IoT detecta nenhuma resposta do dispositivo dentro do período de tempo limite, ele é definido `TIMED_OUT` como um status temporário, permitindo novas tentativas e alterações de estado. Se o seu dispositivo reportar explicitamente`TIMED_OUT`, isso se tornará um status de terminal sem mais transições. Para obter mais informações, consulte Execuções de comandos não terminais.

As seções a seguir descrevem as execuções terminais e não terminais e seus status.

Tópicos

- Execuções de comandos não terminais
- Execuções de comandos terminais

Execuções de comandos não terminais

Uma execução não é terminal se puder aceitar atualizações de dispositivos. As execuções não terminais são consideradas ativas. Os seguintes status não são terminais:

- CREATED

Quando você inicia uma execução no AWS IoT console ou usa a `StartCommandExecution` API, as solicitações bem-sucedidas mudam o status para`CREATED`. A partir desse status, as execuções podem passar para qualquer outro status não terminal ou terminal.

- IN_PROGRESS

  Depois de receber a carga, os dispositivos podem começar a executar instruções e realizar ações especificadas. Durante a execução, os dispositivos podem publicar respostas ao tópico de resposta dos comandos e atualizar o status para`IN_PROGRESS`. A partir de`IN_PROGRESS`, as execuções podem fazer a transição para qualquer status terminal ou não terminal, exceto. `CREATED`

  > **Note**
  >
  > A `UpdateCommandExecution` API pode ser invocada várias vezes com `IN_PROGRESS` status. Especifique detalhes adicionais de execução usando o `statusReason` objeto.

- TEMPORIZADO_LIMITE

  Tanto a nuvem quanto o dispositivo podem acionar esse status. As execuções `CREATED` ou `IN_PROGRESS` o status podem mudar `TIMED_OUT` para pelos seguintes motivos:

  - Depois de enviar o comando, um cronômetro é iniciado. Se o dispositivo não responder dentro do período especificado, o status da nuvem mudará para`TIMED_OUT`. Nesse caso, a execução não é terminal.
  - O dispositivo pode substituir o status de qualquer terminal ou relatar um tempo limite e definir o status como. `TIMED_OUT` Nesse caso, o status permanece`TIMED_OUT`, mas os campos do `StatusReason` objeto mudam com base nas informações do dispositivo. A execução se torna terminal.

  Para obter mais informações, consulte Valor do tempo limite e status de execução `TIMED_OUT`.

Execuções de comandos terminais

Uma execução se torna terminal quando não aceita mais atualizações de dispositivos. Os status a seguir são terminais. As execuções podem fazer a transição para status de terminal a partir de qualquer status que não seja terminal:`CREATED`,, `IN_PROGRESS` ou. `TIMED_OUT`

- SUCCEEDED

Se o dispositivo concluir com êxito a execução, ele poderá publicar uma resposta ao tópico de resposta do comando e atualizar o status para`SUCCEEDED`.

- FAILED

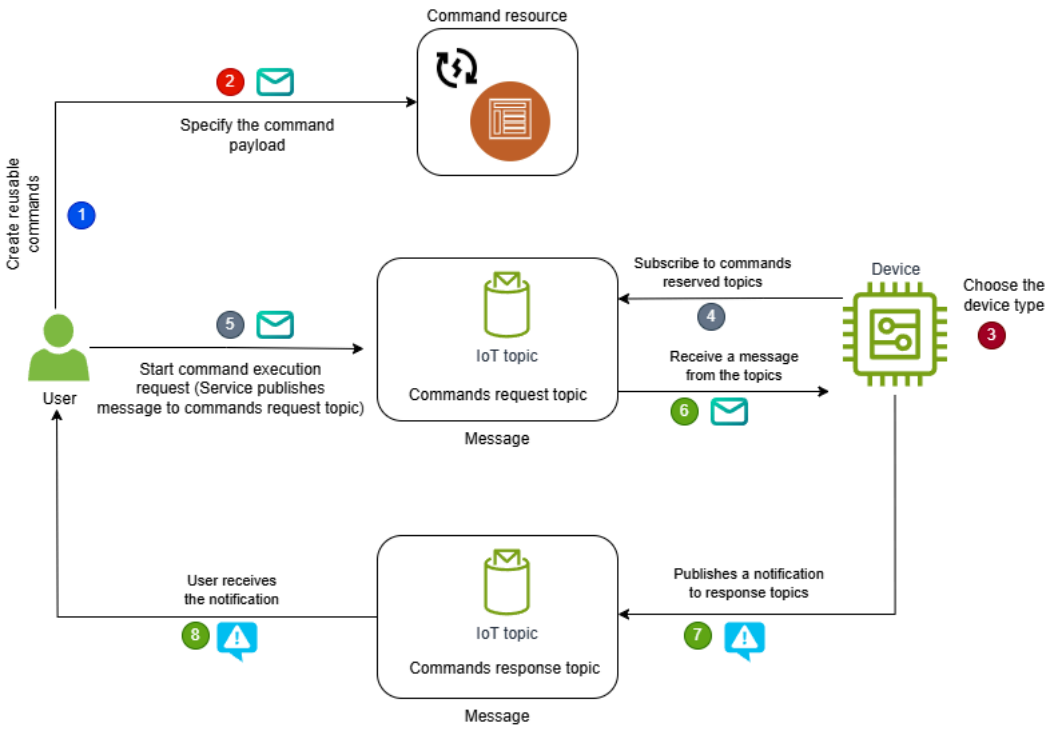
  Quando um dispositivo não consegue concluir a execução, ele pode publicar uma resposta ao tópico de resposta do comando e atualizar o status para`FAILED`. Use os `reasonDescription` campos `reasonCode` e no `statusReason` objeto, ou CloudWatch registros, para solucionar falhas.

- REJEITADA

  Quando um dispositivo recebe uma solicitação inválida ou incompatível, ele pode invocar a `UpdateCommandExecution` API com status. `REJECTED` Use os `reasonDescription` campos `reasonCode` e no `statusReason` objeto, ou CloudWatch registros, para solucionar problemas.

Fluxo de trabalho de comandos de alto nível

Esse fluxo de trabalho mostra como os dispositivos interagem com AWS IoT Device Management os comandos. Todas as solicitações da API HTTP usam credenciais Sigv4 para assinatura.



Visão geral do fluxo de trabalho

- Criar e gerenciar comandos
- Escolher o dispositivo de destino para os comandos e assinar os tópicos do MQTT
- Iniciar e monitorar as execuções de comando para o dispositivo de destino
- (Opcional) Habilitar notificações para eventos de comandos

Criar e gerenciar comandos

Para criar e gerenciar comandos para seus dispositivos, realize as etapas a seguir.

1. Criar um recurso de comando

   Crie um comando no Command Hub ou usando a `CreateCommand`API.
2. Especificar a carga útil

   Forneça uma carga em qualquer formato. Especifique o tipo de conteúdo para garantir a interpretação correta do dispositivo.

   Para comandos dinâmicos com modelos de carga útil, a carga útil final é gerada em tempo de execução usando os parâmetros fornecidos. Os modelos oferecem suporte somente ao formato JSON, mas a carga gerada pode ser enviada como JSON ou CBOR.
3. (Opcional) Gerenciar os comandos criados

   Atualize o nome de exibição e a descrição após a criação. Marque os comandos como obsoletos quando não forem mais necessários ou exclua-os permanentemente. Para modificar as informações do Payload, crie um novo Comando.

Escolher o dispositivo de destino para os comandos e assinar os tópicos do MQTT

Escolha seu dispositivo de destino e configure os tópicos do MQTT para receber comandos e publicar respostas.

1. Escolher o dispositivo de destino para o comando

   Escolha um dispositivo de destino para receber e executar o Comando. Use um nome Thing para dispositivos registrados ou uma ID de cliente para dispositivos não registrados. Para obter mais informações, consulte Considerações sobre dispositivos de destino.

2. Configurar a política AWS IoT do dispositivo

   Configure uma política do IAM concedendo permissões para receber execuções e publicar atualizações. Consulte Exemplo de política do IAM para obter exemplos de políticas.

3. Estabelecer uma conexão do MQTT

   Conecte dispositivos ao agente de mensagens e assine os tópicos de solicitação e resposta. Os dispositivos precisam de `iot:Connect` permissão. Encontre seu endpoint do plano de dados usando o comando da `DescribeEndpoint` API ou da `describe-endpoint` CLI:

   ```
   aws iot describe-endpoint --endpoint-type iot:Data-ATS
   ```

   A execução desse comando exibe o endpoint do plano de dados específico da conta, conforme mostrado abaixo.

   ```
   account-specific-prefix.iot.region.amazonaws.com
   ```

4. Inscreva-se nos tópicos de comandos

   Inscreva-se no tópico de solicitação de comandos. Quando você inicia uma execução, o agente de mensagens publica a carga neste tópico. Seu dispositivo recebe e processa o Comando.

   (Opcional) Inscreva-se nos tópicos de resposta (`acceptedourejected`) para receber a confirmação se o serviço de nuvem aceitou ou rejeitou a resposta do dispositivo.

   Neste exemplo, substitua:

   - *<device>* por `thing` ou `client` dependendo de o dispositivo que você está considerando ter sido registrado como uma coisa da IoT ou especificado como um cliente do MQTT.
   - *<DeviceID>* pelo identificador exclusivo do seu dispositivo de destino. Esse ID pode ser o exclusivo do cliente do MQTT ou o nome de uma coisa.

   > **Note**
   >
   > Se o tipo de carga não for JSON ou CBOR, o *<PayloadFormat>* campo pode não estar presente no tópico de solicitação de comandos. Para obter o formato da carga útil, recomendamos que você use MQTT5 para obter as informações de formato dos

cabeçalhos das mensagens do MQTT. Para obter mais informações, consulte Tópicos de comandos.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request/<PayloadFormat>
$aws/commands/<devices>/<DeviceID>/executions/+/response/accepted/<PayloadFormat>
$aws/commands/<devices>/<DeviceID>/executions/+/response/rejected/<PayloadFormat>
```

Iniciar e monitorar as execuções de comando para o dispositivo de destino

Depois de criar os comandos e especificar os destinos deles, você pode iniciar a execução no dispositivo de destino realizando as etapas a seguir.

1. Iniciar a execução do comando no dispositivo de destino

   Inicie a execução no Command Hub ou usando a `StartCommandExecution` API com o endpoint específico da sua conta. Use `iot:Data-ATS` para pilha dupla (IPv4/IPv6) ou `iot:Jobs` somente para. IPv4

   A API publica a carga no tópico de solicitação de comandos.

   > **Note**
   >
   > Se o dispositivo estiver off-line e usar sessões persistentes do MQTT, o Comando aguarda no agente de mensagens. Quando o dispositivo se reconecta antes do tempo limite, ele pode processar o Comando e publicar os resultados. Se o tempo limite expirar, a execução expirará e a carga será descartada.

2. Atualizar o resultado da execução do comando

   O dispositivo recebe a carga, processa o comando, executa as ações especificadas e publica os resultados no tópico de resposta dos comandos usando a API baseada em `UpdateCommandExecution` MQTT. Se estiver inscrito em tópicos aceitos e rejeitados, o dispositivo receberá a confirmação se o serviço de nuvem aceitou ou rejeitou a resposta.

   Dependendo do que você especificou no tópico da solicitação, *<devices>* podem ser coisas ou clientes, e *<DeviceID>* pode ser o nome do seu AWS IoT item ou o ID do cliente MQTT.

> **Note**
>
> Só *<PayloadFormat>* pode ser JSON ou CBOR no tópico de resposta dos comandos.

```
$aws/commands/<devices>/<DeviceID>/executions/<ExecutionId>/
response/<PayloadFormat>
```

3. (Opcional) Recuperar o resultado da execução do comando

   Recupere os resultados da execução do AWS IoT console ou usando`GetCommandExecution`. O dispositivo deve publicar os resultados no tópico de resposta dos comandos para obter as informações mais recentes. Veja detalhes adicionais, incluindo hora da última atualização, resultado e hora de conclusão.

(Opcional) Habilitar notificações para eventos de comandos

Inscreva-se nos eventos de Commands para receber notificações quando o status de execução mudar. Para obter informações detalhadas sobre eventos de execução de comandos, incluindo o formato da mensagem do evento e os atributos da carga útil, consulteEventos de execução de comandos.

1. Criar uma regra de tópico

   Inscreva-se no tópico de eventos de Comandos para receber notificações de mudança de status. Crie uma regra de tópico para rotear dados do dispositivo para outros AWS IoT serviços AWS Lambda, como Amazon SQS e AWS Step Functions, usando o AWS IoT console ou. Criação de uma AWS IoT regra

   Neste exemplo, substitua *<CommandID>* pelo identificador do comando para o qual você deseja receber notificações e *<CommandExecutionStatus>* pelo status da execução do comando.

   ```
   $aws/events/commandExecution/<CommandID>/<CommandExecutionStatus>
   ```

> **Note**
>
> Para receber notificações de todos os comandos e os status de execução deles, você pode usar caracteres curinga e assinar o tópico a seguir.

```
$aws/events/commandExecution/+/#
```

2. Receber e processar eventos de comandos

   Gerencie as notificações push de comandos e crie aplicativos usando os eventos inscritos.

O código a seguir mostra um exemplo de carga útil para as notificações de eventos de comandos que você receberá.

```
{
    "executionId": "2bd65c51-4cfd-49e4-9310-d5cbfdbc8554",
    "status":"FAILED",
    "statusReason": {
         "reasonCode": "DEVICE_TOO_BUSY",
         "reasonDescription": ""
    },
    "eventType": "COMMAND_EXECUTION",
    "commandArn":"arn:aws:iot:us-east-1:123456789012:command/0b9d9ddf-
e873-43a9-8e2c-9fe004a90086",
    "targetArn":"arn:aws:iot:us-east-1:123456789012:thing/5006c3fc-
de96-4def-8427-7eee36c6f2bd",
    "timestamp":1717708862107
}
```

Criar e gerenciar comandos

Use AWS IoT Device Management comandos para configurar ações remotas reutilizáveis ou enviar instruções imediatas aos dispositivos. Crie e gerencie comandos no AWS IoT console ou usando AWS CLI o.

Operações de criar e gerenciar comandos

- Criar um recurso de comando

- Recuperar informações sobre um comando
- Liste os comandos em seu Conta da AWS
- Atualizar um recurso de comando
- Suspender ou restaurar um recurso de comando
- Excluir um recurso de comando

Criar um recurso de comando

Forneça as seguintes informações ao criar um Comando:

- Informações gerais

  Forneça um ID de comando exclusivo para identificar o comando ao executá-lo nos dispositivos de destino. Opcionalmente, especifique um nome de exibição, uma descrição e tags para gerenciamento.

- Carga útil

  Para comandos estáticos, forneça uma carga que defina as ações do dispositivo. Especifique o tipo de formato do Payload para a interpretação correta do dispositivo.

  Para comandos dinâmicos, consulte o atributo do modelo Payload.

- Modelo de carga

  Para comandos dinâmicos, forneça um PayloadTemplate com espaços reservados e parâmetros. Opcionalmente, forneça `defaultValue` e condicione. AWS IoT Device Management Os comandos substituem os espaços reservados em tempo de execução. Os parâmetros ausentes usam seu DefaultValue. Todos os valores devem satisfazer as condições definidas.

  Os seguintes tipos de espaço reservado com distinção entre maiúsculas e minúsculas são compatíveis:

  - `${aws:iot:commandexecution::parameter:`*`parameter1`*`}`— Um espaço reservado para o valor de um parâmetro com o nome`parameter1`.
  - `${aws:iot:commandexecution::executionTimeoutSec}`— Um espaço reservado para o parâmetro de tempo limite de execução do comando fornecido em tempo de execução.

Tópicos de carga útil e comandos

Comandos reservados Os tópicos usam um formato baseado no tipo de formato Payload.

- Para `application/json` nossos tipos de `application/cbor` conteúdo, use este tópico de solicitação:

```
$aws/commands/<devices>/<DeviceID>/executions/+/request/<PayloadFormat>
```

- Para outros tipos de conteúdo ou formato não especificado, use este tópico de solicitação. O formato Payload aparece no cabeçalho da mensagem MQTT.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request
```

O tópico de resposta dos comandos usa `json` ou `cbor` formata independentemente do tipo de carga útil. *<PayloadFormat>*deve ser `json` ou`cbor`:

```
$aws/commands/<devices>/<DeviceID>/executions/<ExecutionId>/response/<PayloadFormat>
```

Criar um recurso de comando (console)

As seções a seguir descrevem as considerações sobre o formato de carga útil e a criação de comandos a partir do console.

Tópicos

- Formato de carga útil de comando estático
- Formato de carga útil de comando dinâmico
- Como criar um comando (console)

Formato de carga útil de comando estático

O Payload suporta qualquer formato de até 32 KB. Especifique o tipo de formato do Payload para uma interpretação segura e correta do dispositivo.

Especifique o tipo de formato de carga usando o `type/subtype` formato (por exemplo, `application/json` ou`application/cbor`). Padrão: `application/octet-stream`. Consulte Tipos comuns de MIME para ver os formatos compatíveis.

Formato de carga útil de comando dinâmico

O PayloadTemplate deve ser um JSON válido com pelo menos um espaço reservado, de até 32 KB.

Para o `AwsJsonSubstitution` pré-processador, o AWS IoT Device Management Commands envia notificações no formato JSON ou CBOR com base na configuração do pré-processador.

Como criar um comando (console)

Para criar um comando a partir do console, acesse o Command Hub e siga estas etapas:

1. Escolha Criar comando.
2. Especifique um ID de comando exclusivo.
3. (Opcional) Especifique o nome de exibição, a descrição e as tags.
4. Faça upload do arquivo Payload contendo as ações do dispositivo. Especifique o tipo de formato do Payload para a interpretação correta do dispositivo.
5. (Opcional) Para modelos de carga útil JSON com espaços reservados, os parâmetros são pré-preenchidos na tabela embutida para edição.
6. (Opcional) Configure o tipo de valor do parâmetro (obrigatório), o valor padrão e as condições.
7. Escolha Criar comando.

Criar um recurso de comando (CLI)

Use o comando `CreateCommand`da API ou da `create-command`CLI para criar um comando.

Tópicos

- Carga útil do comando
- Exemplo de política do IAM
- Exemplo de criação de comando estático
- Exemplo de criação de comando dinâmico

Carga útil do comando

Forneça uma carga estática ou um modelo de carga útil. As cargas estáticas são codificadas em base64. Para modelos de carga útil, a carga útil final é gerada em tempo de execução usando valores de parâmetros. Os dispositivos processam a carga e executam ações especificadas. Especifique o tipo de conteúdo do Payload para a recepção correta do dispositivo.

> **Note**
>
> As cargas não podem ser modificadas após a criação do Comando. Crie um novo comando para modificar a carga.

Exemplo de política do IAM

Antes de usar essa operação de API, garanta que sua política do IAM autorize você a realizar essa ação no dispositivo. O exemplo a seguir mostra uma política do IAM que permite ao usuário realizar a ação `CreateCommand`.

Neste exemplo, substitua:

- *`region`* por sua Região da AWS, como *`us-east-1`*.
- *`account-id`* pelo número de sua Conta da AWS , como *`123456789012`*.
- *`command-id`*com um identificador exclusivo para seu ID de AWS IoT comando, como*`LockDoor`*. Se quiser enviar mais de um comando, você poderá especificá-los na seção Recursos na política do IAM.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": "iot:CreateCommand",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:command/command-id"
    }
}
```

Exemplo de criação de comando estático

O exemplo a seguir mostra como você pode criar um comando estático. Dependendo da sua aplicação, substitua:

- *`<command-id>`* por um identificador exclusivo para o comando. Por exemplo, para trancar a porta da sua casa, você pode especificar*`LockDoor`*. Recomendamos usar o UUID. Você também pode usar caracteres alfanuméricos, “-” e “_”.

- (Opcional) *<display-name>* e *<description>*, que são campos opcionais que você pode usar para fornecer um nome amigável e uma descrição significativa para o comando, como *Lock the doors of my home*.
- `namespace`, que você pode usar para especificar o namespace do comando. Deve ser `AWS-IoT`. Para obter informações sobre como usar esse recurso para AWS IoT FleetWise, consulte Comandos remotos
- `payload` contém informações sobre a carga útil que você deseja usar ao executar o comando e seu tipo de conteúdo.

```
aws iot create-command \
    --command-id <command-id> \
    --display-name <display-name> \
    --description <description> \
    --namespace AWS-IoT \
    --payload
 '{"content":"eyAibWVzc2FnZSI6ICJIZWxsbyBJb1QiIH0=","contentType":"application/json"}'
```

A execução desse comando gera uma resposta que contém o ID e o ARN (nome do recurso da Amazon) do comando. Por exemplo, se você especificou o comando *LockDoor* durante a criação, o trecho seguinte mostra um exemplo de saída da execução dele.

```
{
    "commandId": "LockDoor",
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/LockDoor"
}
```

Exemplo de criação de comando dinâmico

O exemplo a seguir mostra como você pode criar um comando dinâmico. Dependendo da sua aplicação, substitua:

- *<command-id>* por um identificador exclusivo para o comando. Por exemplo, para definir o status da energia da luz, você pode especificar*Light_Power_Status*. Recomendamos usar o UUID. Você também pode usar caracteres alfanuméricos, “-” e “_”.
- (Opcional) *<display-name>* e *<description>*, que são campos opcionais que você pode usar para fornecer um nome amigável e uma descrição significativa para o comando, como *Turn a light ON or OFF*.

- `namespace`, que você pode usar para especificar o namespace do comando. Deve ser `AWS-IoT`. Para obter informações sobre como usar esse recurso para AWS IoT FleetWise, consulte Comandos remotos
- `payloadTemplate`contém o modelo de carga de reprodução formatado em JSON com espaços reservados.
- `preprocessor`contém a configuração do pré-processador que determina como o PayloadTemplate deve ser processado.
- `mandatoryParameter`contém os parâmetros correspondentes aos espaços reservados no PayloadTemplate, seu tipo, valores padrão e condições.

```
aws iot create-command \
    --command-id Light_Power_Status \
    --description "Turn a light ON or OFF" \
    --namespace AWS-IoT \
    --payload-
template '{"powerStatus":"${aws:iot:commandexecution::parameter:powerStatus}"}' \
    --preprocessor awsJsonSubstitution={outputFormat=JSON} \
    --mandatory-parameters "name=powerStatus, defaultValue={S=OFF},
 valueConditions=[{comparisonOperator=IN_SET, operand={strings=['ON','OFF']}}]"
```

A execução desse comando gera uma resposta que contém o ID e o ARN (nome do recurso da Amazon) do comando. Por exemplo, se você especificou o comando *`Light_Power_Status`* durante a criação, o trecho seguinte mostra um exemplo de saída da execução dele.

```
{
    "commandId": "Light_Power_Status",
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/Light_Power_Status"
}
```

Recuperar informações sobre um comando

Depois de criar um comando, você pode recuperar informações sobre ele no console da AWS IoT e usando a AWS CLI. Você pode acessar as informações a seguir.

- O ID do comando, o nome do recurso da Amazon (ARN), qualquer nome de exibição e a descrição que você especificou para o comando.

- O estado que indica se um comando está disponível para execução no dispositivo de destino, ou se ele está sendo suspenso ou foi excluído.
- A carga ou o PayloadTemplate que você forneceu.
- O pré-processador que você forneceu.
- Os parâmetros obrigatórios que você forneceu.
- A hora em que o comando foi criado e atualizado pela última vez.

Recuperar um recurso de comando (console)

Para recuperar um comando do console, acesse o Command Hub do AWS IoT console e escolha o comando que você criou para ver seus detalhes.

Além dos detalhes do comando, você pode ver o histórico dele, que fornece informações sobre as execuções no dispositivo de destino. Depois de executar esse comando no dispositivo, você encontrará informações sobre as execuções nessa guia.

Recuperar um recurso de comando (CLI)

Use a operação da API do plano de controle `GetCommand`HTTP ou o `get-command` AWS CLI comando para recuperar informações sobre um recurso de comando. Você já deve ter criado o comando usando a solicitação de API `CreateCommand` ou a CLI `create-command`.

Exemplo de política do IAM

Antes de usar essa operação de API, garanta que sua política do IAM autorize você a realizar essa ação no dispositivo. O exemplo a seguir mostra uma política do IAM que permite ao usuário realizar a ação `GetCommand`.

Neste exemplo, substitua:

- *`region`* por sua Região da AWS, como `us-east-1`.
- *`account-id`* pelo número de sua Conta da AWS , como *`123456789023`*.
- *`command-id`*com seu identificador de comando AWS IoT exclusivo, como *`LockDoor`* ou*`Light_Power_Status`*. Se quiser recuperar mais de um comando, você poderá especificá-los na seção Recurso na política do IAM.

```
{
    "Version":"2012-10-17",
```

```
    "Statement": {
        "Action": "iot:GetCommand",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:command/command-id"
    }
}
```

Recuperar um exemplo de comando (AWS CLI)

O exemplo a seguir mostra como recuperar informações sobre um comando usando o. `get-command` AWS CLI Dependendo da sua aplicação, substitua *<command-id>* pelo identificador do comando para o qual você deseja recuperar informações. Você pode acessar essas informações na resposta da CLI `create-command`.

```
aws iot get-command --command-id <command-id>
```

A execução desse comando gera uma resposta com informações sobre ele, bem como a carga útil e a hora em que ele foi criado e atualizado pela última vez. Ela também fornece informações que indicam se um comando foi suspenso ou está sendo excluído.

O código a seguir mostra um exemplo de resposta.

```
{
    "commandId": "LockDoor",
    "commandArn": "arn:aws:iot:<region>:<account>:command/LockDoor",
    "namespace": "AWS-IoT",
    "payload":{
        "content": "eyAibWVzc2FnZSI6ICJIZWxsbyBJb1QiIH0=",
        "contentType": "application/json"
    },
    "createdAt": "2024-03-23T00:50:10.095000-07:00",
    "lastUpdatedAt": "2024-03-23T00:50:10.095000-07:00",
    "deprecated": false,
    "pendingDeletion": false
}
```

Liste os comandos em seu Conta da AWS

Depois de criar comandos, você pode visualizá-los na sua conta. Na lista, você encontrará informações sobre:

- O ID dos comandos e todos os nomes de exibição que você especificou para eles.
- O nome do recurso da Amazon (ARN) dos comandos.
- O estado que indica se os comandos estão disponíveis para execução no dispositivo de destino ou se foram suspensos.

> **Note**
>
> A lista não mostra os que estão sendo excluídos da sua conta. Se os comandos estiverem com a exclusão pendente, você ainda poderá visualizar os respectivos detalhes usando o ID do comando.

- A hora em que os comandos foram criados e atualizados pela última vez.

Listar comandos na sua conta (console)

No AWS IoT console, você pode encontrar a lista de comandos que você criou e seus detalhes acessando o Command Hub.

Listar comandos em sua conta (CLI)

Para listar os comandos que você criou, use a operação de API `ListCommands` ou a CLI `list-commands`.

Exemplo de política do IAM

Antes de usar essa operação de API, garanta que sua política do IAM autorize você a realizar essa ação no dispositivo. O exemplo a seguir mostra uma política do IAM que permite ao usuário realizar a ação `ListCommands`.

Neste exemplo, substitua:

- *`region`* por sua Região da AWS, como `us-east-1`.
- *`account-id`* pelo número de sua Conta da AWS , como *`123456789012`*.

```
{
    "Version":"2012-10-17",
    "Statement": {
```

```
        "Action": "iot:ListCommands",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:command/*"
    }
}
```

Exemplo de como listar comandos em sua conta

O comando a seguir mostra como listar comandos em sua conta.

```
aws iot list-commands --namespace "AWS-IoT"
```

A execução desse comando gera uma resposta que contém uma lista dos comandos criados, a hora em que eles foram criados e quando foram atualizados pela última vez. Ele também fornece as informações de estado do comando, que indicam se ele foi suspenso ou se está disponível para execução no dispositivo de destino. Para acessar mais informações sobre os diferentes status e os respectivos motivos, consulte Status de execução do comando.

Atualizar um recurso de comando

Depois de criar um comando, você pode atualizar o nome de exibição e a descrição dele.

> **Note**
>
> Não é possível atualizar a carga útil do comando. Para atualizar essas informações ou usar uma carga útil modificada, você precisará criar outro comando.

Atualizar um recurso de comando (console)

Para atualizar um comando do console, acesse o Hub de Comando do AWS IoT console e execute as etapas a seguir.

1. Para atualizar um recurso de comando existente, escolha o comando a ser atualizado e, em Ações, escolha Editar.
2. Especifique o nome de exibição e a descrição que você deseja usar e os pares de nome-valor como tags para seu comando.
3. Escolha Editar para salvar o comando com as novas configurações.

Atualizar um recurso de comando (CLI)

Use a operação da API do plano de `UpdateCommand`controle ou a `update-command` AWS CLI para atualizar um recurso de comando. Usando essa API, você pode:

- Editar o nome de exibição e a descrição de um comando criado por você.
- Suspender um recurso de comando ou restaurar um comando que já tenha sido suspenso.

Exemplo de política do IAM

Antes de usar essa operação de API, garanta que sua política do IAM autorize você a realizar essa ação no dispositivo. O exemplo a seguir mostra uma política do IAM que permite ao usuário realizar a ação `UpdateCommand`.

Neste exemplo, substitua:

- *`region`* por sua Região da AWS, como `us-east-1`.
- *`account-id`* pelo número de sua Conta da AWS , como *`123456789012`*.
- *`command-id`* pelo seu identificador de comando exclusivo da AWS IoT , como *`LockDoor`*. Se quiser recuperar mais de um comando, você poderá especificá-los na seção Recurso na política do IAM.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": "iot:UpdateCommand",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:command/command-id"
    }
}
```

Exemplos de como atualizar informações sobre um comando (AWS CLI)

O exemplo a seguir mostra como atualizar as informações sobre um comando usando o `update-command` AWS CLI comando. Para ter informações sobre como você pode usar essa API para suspender ou restaurar um recurso de comando, consulte Atualizar um recurso de comando (CLI).

O exemplo mostra como você pode atualizar o nome de exibição e a descrição de um comando. Dependendo da sua aplicação, substitua *<command-id>* pelo identificador do comando cujas informações você deseja recuperar.

```
aws iot update-command \
    --command-id <command-id>
    --displayname <display-name> \
    --description <description>
```

A execução desse comando gera uma resposta com as informações atualizadas sobre ele e a respectiva atualização mais recente. O código a seguir mostra um exemplo de solicitação e resposta para atualizar o nome de exibição e a descrição de um comando que desliga o ar condicionado.

```
aws iot update-command \
    --command-id <LockDoor> \
    --displayname <Secondary lock door> \
    --description <Locks doors to my home>
```

A execução desse comando gera a resposta a seguir.

```
{
    "commandId": "LockDoor",
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/LockDoor",
    "displayName": "Secondary lock door",
    "description": "Locks doors to my home",
    "lastUpdatedAt": "2024-05-09T23:15:53.899000-07:00"
}
```

Suspender ou restaurar um recurso de comando

Depois de criar um comando, se não quiser mais usá-lo, você poderá marcá-lo como suspenso. Quando você suspende um comando, todas as execuções pendentes dele continuam sendo executadas no dispositivo de destino até atingirem o status terminal. Depois que um comando for suspenso, se você quiser usá-lo para enviar uma nova execução para o dispositivo de destino, deverá restaurá-lo.

> **Note**
>
> Você não pode editar um comando suspenso nem realizar novas execuções dele. Para executar novos comandos no dispositivo, você deve restaurá-lo para que o estado do comando mude para Disponível.

Para acessar informações adicionais sobre a suspensão e a restauração de um comando e considerações sobre ele, consulte Suspender um recurso de comando.

Excluir um recurso de comando

Se você não quiser mais usar um comando, poderá removê-lo permanentemente da sua conta. Se a ação de exclusão for bem-sucedida:

- Se o comando permanecer suspenso por um período maior que o tempo limite máximo de 12 horas, ele será excluído imediatamente.
- Se o comando não foi suspenso ou permanecer suspenso por um período menor que o valor do tempo limite máximo, ele entrará no estado `pending deletion`. Ele será removido automaticamente da sua conta após o tempo limite máximo de 12 horas.

> **Note**
>
> O comando poderá ser excluído mesmo se houver alguma execução pendente. O comando estará em um estado de exclusão pendente e será removido da sua conta automaticamente.

Excluir um recurso de comando (console)

Para excluir um comando do console, acesse o Hub de Comando do AWS IoT console e execute as etapas a seguir.

1. Escolha o comando a ser excluído e, em Ações, selecione Excluir.
2. Confirme que você deseja excluir o comando e, depois, selecione Excluir.

O comando será marcado para exclusão e será removido permanentemente da sua conta após 12 horas.

Excluir um recurso de comando (CLI)

Use a operação da API do plano de controle `DeleteCommand` HTTP ou o `delete-command` AWS CLI comando para excluir um recurso de comando. Se a ação de exclusão for bem-sucedida, você verá um `statusCode` HTTP de 204 ou 202, e o comando será excluído da sua conta automaticamente após o tempo limite máximo de 12 horas. O status 204 indica que o comando foi excluído.

Exemplo de política do IAM

Antes de usar essa operação de API, garanta que sua política do IAM autorize você a realizar essa ação no dispositivo. O exemplo a seguir mostra uma política do IAM que permite ao usuário realizar a ação `DeleteCommand`.

Neste exemplo, substitua:

- *`region`* por sua Região da AWS, como `us-east-1`.
- *`account-id`* pelo número de sua Conta da AWS , como *`123456789012`*.
- *`command-id`* pelo seu identificador de comando exclusivo da AWS IoT , como *`LockDoor`*. Se quiser recuperar mais de um comando, você poderá especificá-los na seção Recurso na política do IAM.

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": "iot:DeleteCommand",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:command/command-id"
    }
}
```

Exemplo de como excluir um comando (AWS CLI)

Os exemplos a seguir mostram como excluir um comando usando o `delete-command` AWS CLI comando. Dependendo da sua aplicação, substitua *`<command-id>`* pelo identificador do comando que você está excluindo.

```
aws iot delete-command --command-id <command-id>
```

Se a solicitação da API for bem-sucedida, o comando gerará um código de status 202 ou 204. Você pode usar a API `GetCommand` para verificar se o comando não existe mais na sua conta.

Iniciar e monitorar execuções de comando

Depois de criar um comando, inicie uma execução no dispositivo de destino. O dispositivo atualiza os resultados e publica o status nos tópicos reservados do MQTT. Recupere e monitore o status de execução da sua conta.

Inicie e monitore os comandos usando o AWS IoT console ou AWS CLI.

Iniciar e monitorar operações de comandos

- Iniciar uma execução de comando
- Atualizar o resultado de uma execução de comando
- Recuperar uma execução de comando
- Visualizar atualizações de comandos usando o cliente de teste do MQTT
- Liste as execuções de comandos em seu Conta da AWS
- Excluir uma execução de comando

Iniciar uma execução de comando

> **Important**
>
> Você é o único responsável por implantar comandos de forma segura e compatível com as leis aplicáveis.

Antes de iniciar uma execução, certifique-se de que:

- Você criou um Comando no AWS IoT namespace com as informações do Payload. Ao iniciar a execução, o dispositivo processa as instruções do Payload e executa ações especificadas. Consulte Criar um recurso de comando para criação de comandos.
- Seu dispositivo se inscreveu no MQTT reservou Tópicos para Comandos. Ao iniciar a execução, as informações do Payload são publicadas nesta solicitação reservada do MQTT. Tópico:

*<devices>*podem ser clientes Things ou MQTT. *<DeviceID>*é o nome da coisa ou ID do cliente. *<PayloadFormat>*Valores suportados: JSON e CBOR. Para obter mais informações, consulte Tópicos de comandos.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request/<PayloadFormat>
```

Para não JSON/CBOR*<PayloadFormat>*, use este formato de tópico de comandos:

```
$aws/commands/<devices>/<DeviceID>/executions/+/request
```

Considerações sobre dispositivos de destino

Especifique o dispositivo de destino para receber e executar o Comando. Use um nome Thing para dispositivos registrados ou uma ID de cliente para dispositivos não registrados. Depois de receber a carga, o dispositivo executa o comando e executa as ações especificadas.

AWS IoT coisa

Os dispositivos de destino podem ser coisas registradas no AWS IoT registro. As coisas simplificam a pesquisa e o gerenciamento de dispositivos.

Registre dispositivos como Coisas na página do dispositivo Connect ou usando `CreateThing`. Encontre coisas existentes no Thing Hub ou usando `DescribeThing`. Consulte Gerenciando coisas com o registro para obter detalhes do registro.

ID de cliente

Para dispositivos não registrados, use o ID do cliente.

O ID do cliente é um identificador exclusivo que você atribui aos dispositivos. Definido no protocolo MQTT, ele contém caracteres alfanuméricos, sublinhados ou traços. Cada dispositivo conectado AWS IoT precisa de um ID de cliente exclusivo.

> **Note**
>
> - Para itens registrados, o ID do cliente pode corresponder ao nome do item.
> - Ao segmentar uma ID de cliente específica, os dispositivos devem se conectar AWS IoT usando essa ID de cliente para receber a carga.

A ID do cliente é a ID do cliente MQTT que os dispositivos usam quando se conectam a. AWS IoT Core AWS IoT usa essa ID para identificar dispositivos e gerenciar conexões e assinaturas.

Considerações sobre tempo limite de execução de comandos

O tempo limite especifica a duração (em segundos) para que os dispositivos forneçam resultados de execução.

Depois de criar uma execução, um cronômetro é iniciado. Se o dispositivo ficar off-line ou não reportar os resultados dentro do tempo limite, a Execução expirará com o statusTIMED_OUT.

Padrão: 10 segundos. Máximo: 12 horas.

Valor do tempo limite e status de execução **TIMED_OUT**

Tanto a nuvem quanto o dispositivo podem relatar o tempo limite.

Depois de enviar o Comando, um cronômetro é iniciado. Se nenhuma resposta do dispositivo chegar dentro do tempo limite, a nuvem definirá o status de execução como TIMED_OUT com o código do $NO_RESPONSE_FROM_DEVICE motivo.

Isso ocorre quando:

- O dispositivo ficou off-line durante a execução.
- O dispositivo falhou ao concluir a execução dentro do tempo limite.
- O dispositivo falhou ao relatar o status dentro do tempo limite.

Nesse caso, quando o status de execução de TIMED_OUT é relatado na nuvem, a execução do comando não é terminal. Seu dispositivo pode publicar uma resposta que substitui o status de qualquer um dos status do terminal:SUCCEEDED,, FAILED ou. REJECTED A execução do comando então se torna terminal e não aceita mais atualizações.

Seu dispositivo também pode atualizar um TIMED_OUT status iniciado pela nuvem relatando que ocorreu um tempo limite quando ele estava executando o comando. Nesse caso, o status de execução do comando permanece emTIMED_OUT, mas o statusReason objeto é atualizado com base nas informações relatadas pelo dispositivo. A execução do comando então se torna terminal e nenhuma atualização adicional é aceita.

Usar sessões persistentes do MQTT

Você pode configurar sessões persistentes do MQTT para usar com o recurso de AWS IoT Device Management comandos. Esse recurso é especialmente útil quando, por exemplo, o dispositivo fica offline e você deseja garantir que o dispositivo ainda receba o comando quando voltar a ficar on-line antes do tempo limite e execute as instruções especificadas.

Por padrão, a expiração da sessão persistente do MQTT é definida como 60 minutos. Se o tempo limite de execução do comando estiver configurado para um valor que exceda essa duração, as execuções de comandos executadas por mais de 60 minutos poderão ser rejeitadas pelo agente de mensagens e falhar. Para executar comandos com mais de 60 minutos de duração, você pode solicitar um aumento no tempo de expiração da sessão persistente.

> **Note**
>
> Para garantir que você esteja usando o recurso de sessões persistentes do MQTT corretamente, defina o sinalizador Clean Start como zero. Para acessar mais informações, consulte MQTT persistent sessions.

Iniciar uma execução de comando (console)

Para começar a executar o comando no console, acesse a página do Command Hub do AWS IoT console e execute as etapas a seguir.

1. Para executar o comando que você criou, escolha Executar comando.
2. Revise as informações sobre o comando que você criou, incluindo os tópicos reservados e os parâmetros do MQTT, se aplicável.

   Para comandos dinâmicos, insira os valores dos parâmetros ou deixe-os com os padrões. Para parâmetros que não têm um valor padrão, você deve fornecer um valor a ser enviado como parte dessa execução.
3. Especifique o dispositivo de destino para receber e executar o Comando. O dispositivo pode ser especificado como uma AWS IoT coisa se tiver sido registrado AWS IoT ou usando o ID do cliente se o dispositivo ainda não tiver sido registrado. Para obter mais informações, consulte Considerações sobre dispositivos de destino.
4. (Opcional) Configure um valor de tempo limite para o comando que determine a duração pela qual você deseja que o comando seja executado antes que ele atinja o tempo limite. Se seu

comando precisar ser executado por mais de 60 minutos, talvez seja necessário aumentar o tempo de expiração das sessões persistentes do MQTT. Para obter mais informações, consulte Considerações sobre tempo limite de execução de comandos.

5. Selecione Run command.

Iniciar uma execução de comando (AWS CLI)

Use a operação da API do plano de dados HTTP `StartCommandExecution` para iniciar uma execução de comando. A solicitação e a resposta da API são correlacionadas pelo ID de execução do comando. Depois que o dispositivo concluir a execução do comando, ele poderá relatar o status e o resultado da execução para a nuvem publicando uma mensagem no tópico de resposta dos comandos. Para um código de resposta personalizado, os códigos de aplicativo que você possui podem processar a mensagem de resposta e publicar o resultado em AWS IoT.

Se seus dispositivos assinaram o tópico de solicitação de comandos, a API `StartCommandExecution` publicará a mensagem de carga útil no tópico. A carga útil pode usar qualquer formato de sua escolha. Para obter mais informações, consulte Carga útil do comando.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request/<PayloadFormat>
```

Se o formato da carga não for JSON ou CBOR, o seguinte mostra o formato do tópico de solicitação de comandos.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request
```

Exemplo de política do IAM

Antes de usar essa operação de API, garanta que sua política do IAM autorize você a realizar essa ação no dispositivo. O exemplo a seguir mostra uma política do IAM que permite ao usuário realizar a ação `StartCommandExecution`.

Neste exemplo, substitua:

- *`region`* por sua Região da AWS, como `us-east-1`.
- *`account-id`* pelo número de sua Conta da AWS , como *`123456789012`*.
- *`command-id`*com um identificador exclusivo para seu AWS IoT comando, como*`LockDoor`*. Se quiser enviar mais de um comando, você poderá especificá-los na política do IAM.

- *devices*com `thing` ou `client` dependendo se seus dispositivos foram registrados como AWS IoT itens ou especificados como clientes MQTT.
- *device-id*com o seu AWS IoT `thing-name` ou`client-id`.

```
{
  "Effect": "Allow",
  "Action": [
      "iot:StartCommandExecution"
  ],
  "Resource": [
      "arn:aws:iot:region:account-id:command/command-id",
      "arn:aws:iot:region:account-id:devices/device-id"
  ]
}
```

Para ver uma lista das chaves de condição suportadas`StartCommandExecution`, consulte Chaves de condição AWS IoT no Guia do usuário do IAM.

Acessar o endpoint do plano de dados específico da conta

Antes de executar o comando da API, você deve acessar o URL do endpoint específico da conta. Se você estiver usando endpoints de pilha dupla (IPv4e IPv6), use o. `iot:Data-ATS` O `iot:Jobs` endpoint é IPv4 apenas para. Por exemplo, se você executar este comando:

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

Ele retorna o URL do endpoint específico da conta, conforme mostrado no exemplo de resposta abaixo.

```
{
    "endpointAddress":
    "<account-specific-prefix>-ats.iot.<region>.api.com"
}
```

Iniciar um exemplo de execução de comando (AWS CLI)

O exemplo a seguir mostra como começar a executar um comando usando o `start-command-execution` AWS CLI comando.

Neste exemplo, substitua:

- *<command-arn>* pelo ARN do comando que você deseja executar. Você pode acessar essas informações na resposta do comando `create-command` da CLI. Por exemplo, se você estiver executando o comando para alterar o modo do volante, use `arn:aws:iot:`*region*`:`*account-id*`:command/`*SetComfortSteeringMode*.
- *<target-arn>* pelo ARN da coisa do dispositivo de destino, que pode ser uma coisa da IoT ou um cliente do MQTT, para o qual você deseja executar o comando. Por exemplo, se você estiver executando o comando para o dispositivo de destino `myRegisteredThing`, use `arn:aws:iot:`*region*`:`*account-id*`:thing/`*myRegisteredThing*.
- *<endpoint-url>* pelo endpoint específico da conta que você acessou em Acessar o endpoint do plano de dados específico da conta, com o prefixo `https://`. Por exemplo, .`https://`*123456789012abcd*`.jobs.iot.`*us-east-1*`.amazonaws.com`
- (Opcional) Você também pode especificar um parâmetro adicional, `executionTimeoutSeconds`, ao realizar a operação da API `StartCommandExecution`. Esse campo opcional especifica o tempo em segundos em que o dispositivo deve concluir a execução do comando. O valor padrão é de 10 segundos. Quando o status de execução do comando é `CREATED`, um cronômetro é iniciado. Se o resultado da execução do comando não for recebido antes que o cronômetro expire, o status mudará automaticamente para `TIMED_OUT`.
- ```
  aws iot-jobs-data start-command-execution \
      --command-arn <command-arn>  \
      --target-arn <target-arn> \
      --endpoint <endpoint-url> \
      --execution-timeout-seconds 900
  ```
- (Opcional) Para comandos dinâmicos, especifique os parâmetros e seus valores a serem usados para substituição. Você deve fornecer um valor para parâmetros que não tenham um DefaultValue definido na criação do comando. Se um parâmetro tiver um DefaultValue, o valor do parâmetro fornecido aqui terá precedência. Para parâmetros que têm ValueConditions definidas, o valor do parâmetro fornecido aqui deve satisfazer a condição.

  Com base no exemplo de comando `Light_Power_Status` dinâmico:
- ```
  aws iot-jobs-data start-command-execution \
      --command-arn arn:aws:iot:us-east-1:123456789012:command/Light_Power_Status  \
      --target-arn arn:aws:iot:us-east-1:123456789012:thing/exampleThing \
      --endpoint <endpoint-url> \
      --execution-timeout-seconds 900 \
  ```

```
    --parameters "powerStatus={S=ON}"
```

A execução desse comando exibe um ID de execução. Você pode usar esse ID para consultar o status, os detalhes e o histórico da execução do comando.

> **Note**
>
> Se o comando estiver suspenso, a solicitação da API `StartCommandExecution` falhará com uma exceção de validação. Para corrigir esse erro, primeiro restaure o comando usando a API `UpdateCommand` e, depois, execute a solicitação `StartCommandExecution`.

```
{
    "executionId": "07e4b780-7eca-4ffd-b772-b76358da5542"
}
```

Atualizar o resultado de uma execução de comando

Use a operação da API do plano de dados do MQTT `UpdateCommandExecution` para atualizar o status ou o resultado de uma execução de comando.

> **Note**
>
> Antes de usar essa API:
>
> - Seu dispositivo deve ter estabelecido uma conexão do MQTT e assinado os tópicos de solicitação e resposta de comandos. Para obter mais informações, consulte Fluxo de trabalho de comandos de alto nível.
> - Você já deve ter executado esse comando usando a operação da API `StartCommandExecution`.

Exemplo de política do IAM

Antes de usar essa operação da API, verifique se sua política do IAM autoriza seu dispositivo a realizar essas ações. Veja a seguir um exemplo de política que autoriza seu dispositivo a realizar a ação. Para exemplos adicionais de políticas do IAM que permitem que o usuário execute a ação do `UpdateCommandExecution` MQTT, consulteExemplos de política para conectar e publicar.

Neste exemplo, substitua:

- *Region*com o seu Região da AWS, como`us-east-1`.
- *AccountID* pelo número de sua Conta da AWS , como *123456789012*.
- *ThingName*com o nome da AWS IoT coisa para a qual você está direcionando a execução do comando, como*myRegisteredThing*.
- *commands-request-topic*e *commands-response-topic* com os nomes dos seus tópicos de solicitação e resposta de AWS IoT comandos. Para obter mais informações, consulte Fluxo de trabalho de comandos de alto nível.

Exemplo de política do IAM para o ID do cliente do MQTT

O código a seguir mostra um exemplo de política de dispositivos ao usar o ID do cliente do MQTT.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/request",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/accepted",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/rejected",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/request/json",
```

```
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/accepted/json",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/rejected/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/request",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/response/accepted",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/response/rejected",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/request/json",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/response/accepted/json",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/response/rejected/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:us-east-1:123456789012:client/${iot:ClientId}"
    }
  ]
}
```

Exemplo de política do IAM para uma coisa da IoT

O código a seguir mostra um exemplo de política de dispositivos ao usar uma coisa da AWS IoT .

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
```

```
      "Resource": "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/request",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response/accepted",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response/rejected",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/request/json",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response/accepted/json",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response/rejected/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/request",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/response/accepted",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/response/rejected",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/request/json",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/response/accepted/json",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/response/rejected/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:us-east-1:123456789012:client/${iot:ClientId}"
    }
```

```
  ]
}
```

Como usar a API **UpdateCommandExecution**

Depois que a execução do comando é recebida no tópico da solicitação, o dispositivo processa o comando. Depois, ele usa a API `UpdateCommandExecution` para atualizar o status e o resultado da execução do comando para o tópico de resposta a seguir.

```
$aws/commands/<devices>/<DeviceID>/executions/<ExecutionId>/response/<PayloadFormat>
```

Neste exemplo, *<DeviceID>* é o identificador exclusivo do dispositivo de destino, e *<execution-id>* é o identificador da execução do comando no dispositivo de destino. Eles *<PayloadFormat>* podem ser JSON ou CBOR.

> **Note**
>
> Se você não registrou seu dispositivo com AWS IoT, você pode usar o ID do cliente como seu identificador em vez de um nome de coisa.

```
$aws/commands/clients/<ClientID>/executions/<ExecutionId>/response/<PayloadFormat>
```

O dispositivo relatou atualizações no status de execução

Seus dispositivos podem usar a API para relatar qualquer uma das atualizações de status a seguir na execução do comando. Para acessar mais informações sobre esses status, consulte Status de execução do comando.

- `IN_PROGRESS`: quando o dispositivo começa a executar o comando, ele pode atualizar o status para `IN_PROGRESS`.
- `SUCCEEDED`: quando o dispositivo processa com êxito o comando e conclui sua execução, ele pode publicar uma mensagem no tópico de resposta como `SUCCEEDED`.
- `FAILED`: se o dispositivo não conseguir executar o comando, ele poderá publicar uma mensagem no tópico de resposta como `FAILED`.
- `REJECTED`: se o dispositivo falhar ao aceitar o comando, ele poderá publicar uma mensagem no tópico de resposta como `REJECTED`.

- `TIMED_OUT`: o status de execução do comando pode mudar para `TIMED_OUT` devido a qualquer um dos motivos a seguir.
  - O resultado da execução do comando não foi recebido. Isso pode acontecer porque a execução não foi concluída no período especificado ou se o dispositivo não conseguiu publicar as informações de status no tópico de resposta.
  - O dispositivo relata que ocorreu um tempo limite ao tentar executar o comando.

Para acessar mais informações sobre o status `TIMED_OUT`, consulte Valor do tempo limite e status de execução `TIMED_OUT`.

Considerações ao usar a API **`UpdateCommandExecution`**

Veja a seguir algumas considerações importantes ao usar a API `UpdateCommandExecution`.

- Seus dispositivos podem usar um `statusReason` objeto opcional para fornecer informações adicionais sobre a execução. Se seus dispositivos fornecerem esse objeto, o `reasonCode` campo do objeto será obrigatório, mas o `reasonDescription` campo é opcional.
- Quando seus dispositivos usam o `statusReason` objeto, eles `reasonCode` devem usar o padrão `[A-Z0-9_-]+` e não exceder 64 caracteres de comprimento. Se você fornecer o`reasonDescription`, certifique-se de que ele não exceda 1.024 caracteres. Ele pode usar qualquer caractere, exceto caracteres de controle, como novas linhas.
- Seus dispositivos podem usar um objeto `result` opcional para fornecer informações sobre o resultado da execução do comando, como o valor de retorno de uma chamada de função remota. Se você fornecer o `result`, ele deverá exigir pelo menos uma entrada.
- No campo `result`, você especifica as entradas como pares de chave-valor. Para cada entrada, você deve especificar as informações do tipo de dados como string, booliano ou binário. Um tipo de dados de string deve usar a chave `s`, um tipo booliano usa a chave b e um binário deve usar a chave `bin`. Certifique-se de que essas teclas estejam em minúsculas.
- Se você encontrar um erro ao executar a `UpdateCommandExecution` API, poderá ver o erro no grupo de `AWSIoTLogsV2` registros na Amazon CloudWatch. Para acessar informações sobre como habilitar o registro em log e visualizar os logs, consulte Configurar o AWS IoT registro.

Exemplo de API **`UpdateCommandExecution`**

O código a seguir mostra um exemplo de como seu dispositivo pode usar a API `UpdateCommandExecution` para relatar o status da execução, o campo `statusReason` para

fornecer informações adicionais sobre o status e o campo de resultados para fornecer informações sobre o resultado da execução, como a porcentagem da bateria do carro, nesse caso.

```
{
  "status": "IN_PROGRESS",
  "statusReason": {
    "reasonCode": "200",
    "reasonDescription": "Execution_in_progress"
  },
  "result": {
        "car_battery": {
            "s": "car battery at 50 percent"
        }
    }
}
```

Recuperar uma execução de comando

Depois de executar um comando, você pode recuperar informações sobre a execução do comando no AWS IoT console e usando o. AWS CLI Você pode acessar as informações a seguir.

> **Note**
>
> Para recuperar o status mais recente de execução do comando, seu dispositivo deve publicar as informações de status no tópico de resposta usando a API `UpdateCommandExecution` do MQTT, conforme descrito abaixo. Até que o dispositivo publique neste tópico, a API `GetCommandExecution` relatará o status como `CREATED` ou `TIMED_OUT`.

Cada execução de comando que você criar terá:

- Um ID de execução, que é um identificador exclusivo da execução do comando.
- O status da execução do comando. Quando você executa o comando no dispositivo de destino, a execução entra no estado `CREATED`. Depois, ele pode fazer a transição para outros status de execução, conforme descrito abaixo.
- O resultado da execução do comando.
- O ID de comando exclusivo e o dispositivo de destino para o qual as execuções foram criadas.
- A Data de início, que mostra a hora em que a execução do comando foi criada.

Recuperar uma execução de comando (console)

Você pode recuperar uma execução de comando do console usando qualquer um dos métodos a seguir.

- Na página Hub de comandos

  Acesse a página do Command Hub do AWS IoT console e execute essas etapas.

  1. Escolha o comando para o qual você criou uma execução no dispositivo de destino.
  2. Na página de detalhes do comando, na guia Histórico de comandos, você verá as execuções criadas. Escolha a execução cujas informações você deseja recuperar.
  3. Se seus dispositivos usaram a API `UpdateCommandExecution` para fornecer as informações do resultado, você poderá encontrá-las na guia Resultados desta página.

- Na página Hub de coisas

  Se você escolheu AWS IoT algo como dispositivo de destino ao executar o comando, você pode ver os detalhes da execução na página Thing hub.

  1. Acesse a página Thing Hub no AWS IoT console e escolha a coisa para a qual você criou a execução do comando.
  2. Na página de detalhes da coisa, no histórico de comandos, você verá as execuções criadas. Escolha a execução cujas informações você deseja recuperar.
  3. Se seus dispositivos usaram a API `UpdateCommandExecution` para fornecer as informações do resultado, você poderá encontrar essas informações na guia Resultados desta página.

Recuperar uma execução de comando (CLI)

Use a operação da API HTTP do plano de `GetCommandExecution` AWS IoT Core controle para recuperar informações sobre a execução de um comando. Você já deve ter executado esse comando usando a operação da API `StartCommandExecution`.

Exemplo de política do IAM

Antes de usar essa operação de API, garanta que sua política do IAM autorize você a realizar essa ação no dispositivo. O exemplo a seguir mostra uma política do IAM que permite ao usuário realizar a ação `GetCommandExecution`.

Neste exemplo, substitua:

- *region* por sua Região da AWS, como `us-east-1`.
- *account-id* pelo número de sua Conta da AWS , como *123456789012*.
- *command-id*com seu identificador de AWS IoT comando exclusivo, como*LockDoor*.
- *devices*com `thing` ou `client` dependendo se seus dispositivos foram registrados como AWS IoT itens ou especificados como clientes MQTT.
- *device-id*com o seu AWS IoT `thing-name` ou`client-id`.

```
{
  "Effect": "Allow",
  "Action": [
      "iot:GetCommandExecution"
  ],
  "Resource": [
      "arn:aws:iot:region:account-id:command/command-id",
      "arn:aws:iot:region:account-id:devices/device-id"
  ]
}
```

Exemplo de como recuperar uma execução de comando

O exemplo a seguir mostra como recuperar informações sobre um comando que foi executado usando o `start-command-execution` AWS CLI comando. O exemplo a seguir mostra como recuperar informações sobre um comando executado para desativar o modo do volante.

Neste exemplo, substitua:

- *<execution-id>* pelo identificador da execução do comando cujas informações você deseja recuperar.
- *<target-arn>* pelo nome do recurso da Amazon (ARN) do dispositivo para o qual você planeja a execução. Você pode acessar essas informações na resposta do comando `start-command-execution` da CLI.
- Opcionalmente, se seus dispositivos usaram a API `UpdateCommandExection` para fornecer o resultado da execução, você poderá especificar se deseja incluir o resultado da execução do comando na resposta da API `GetCommandExecution` usando a API `GetCommandExecution`.

```
aws iot get-command-execution
    --execution-id <execution-id> \
    --target-arn <target-arn> \
    --include-result
```

A execução desse comando gera uma resposta que contém informações sobre o ARN da execução do comando, o status da execução e a hora em que ela foi iniciada e concluída. Ela também fornece um objeto `statusReason` que contém informações adicionais sobre o status. Para acessar mais informações sobre os diferentes status e os respectivos motivos, consulte Status de execução do comando.

O código a seguir mostra um exemplo de resposta da solicitação da API.

> **Note**
>
> O campo `completedAt` na resposta de execução corresponde ao momento em que o dispositivo relata o status terminal para a nuvem. No caso do status `TIMED_OUT`, esse campo será definido somente quando o dispositivo informar um tempo limite. Quando o status `TIMED_OUT` é definido pela nuvem, o status `TIMED_OUT` não é atualizado. Para acessar mais informações sobre o comportamento do tempo limite, consulte Considerações sobre tempo limite de execução de comandos.

```
{
    "executionId": "07e4b780-7eca-4ffd-b772-b76358da5542",
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/LockDoor",
    "targetArn": "arn:aws:iot:us-east-1:123456789012:thing/myRegisteredThing",
    "status": "SUCCEEDED",
    "statusReason": {
        "reasonCode": "DEVICE_SUCCESSFULLY_EXECUTED",
        "reasonDescription": "SUCCESS"
    },
    "result": {
        "sn": { "s": "ABC-001" },
        "digital": { "b": true }
    },
    "createdAt": "2024-03-23T00:50:10.095000-07:00",
    "completedAt": "2024-03-23T00:50:10.095000-07:00"
}
```

Visualizar atualizações de comandos usando o cliente de teste do MQTT

É possível usar o cliente de teste do MQTT para visualizar a troca de mensagens pelo MQTT ao usar o recurso de comandos. Depois que seu dispositivo estabelecer uma conexão MQTT com AWS IoT, você poderá criar um comando, especificar a carga útil e executá-la no dispositivo. Quando você executa o comando, se seu dispositivo se inscreveu no tópico de solicitação reservada do MQTT para comandos, ele vê a mensagem de carga publicada nesse tópico.

O dispositivo então recebe as instruções de carga útil e executa as operações especificadas no AWS IoT dispositivo. Em seguida, ele usa a `UpdateCommandExecution` API para publicar o resultado da execução do comando e as informações de status nos tópicos de resposta reservados do MQTT para comandos. AWS IoT Device Management escuta atualizações sobre os tópicos de resposta, armazena as informações atualizadas e publica registros AWS CloudTrail na Amazon. CloudWatch Em seguida, você pode recuperar as informações mais recentes sobre a execução do comando no console ou usando a `GetCommandExecution` API.

As etapas a seguir mostram como usar o cliente de teste do MQTT para observar mensagens.

1. Abra o cliente de teste MQTT no AWS IoT console.
2. Na guia Inscrever-se, insira o tópico a seguir e escolha Inscrever-se, onde *<thingId>* está o nome do dispositivo com o qual você se registrou AWS IoT.

   > **Note**
   >
   > Você pode encontrar o nome da coisa para o seu dispositivo na página Thing Hub do AWS IoT console. Se você não registrou seu dispositivo como uma coisa, você pode registrar o dispositivo ao se conectar a AWS IoT partir da página Connect device.

   ```
   $aws/commands/things/<thingId>/executions/+/request
   ```

3. (Opcional) Na guia Assinar, você também pode inserir os tópicos a seguir e escolher Assinar.

   ```
   $aws/commands/things/+/executions/+/response/accepted/json
   $aws/commands/things/+/executions/+/response/rejected/json
   ```

4. Depois de iniciar uma execução de comando, a carga útil de mensagem será enviada ao dispositivo usando o tópico de solicitação assinado pelo dispositivo, ou seja, `$aws/commands/`

`things/<thingId>/executions/+/request`. No cliente de teste do MQTT, você deve ver a carga útil que contém as instruções para que o dispositivo processe o comando.

5. Depois que o dispositivo começar a executar o comando, ele poderá publicar atualizações de status no tópico de resposta reservada do MQTT a seguir.

   ```
   $aws/commands/<devices>/<device-id>/executions/<executionId>/response/json
   ```

   Por exemplo, pense em um comando que você executou para ligar o ar condicionado do seu carro a fim de reduzir a temperatura até o valor desejado. O JSON a seguir mostra um exemplo de mensagem que o veículo publicou no tópico de resposta que mostra que ele não executou o comando.

   ```
   {
     "deviceId": "My_Car",
     "executionId": "07e4b780-7eca-4ffd-b772-b76358da5542",
     "status": "FAILED",
     "statusReason": {
       "reasonCode": "CAR_LOW_ON_BATTERY",
       "reasonDescription": "Car battery is lower than 5 percent"
     }
   }
   ```

   Nesse caso, você pode carregar a bateria do carro e executar o comando novamente.

Liste as execuções de comandos em seu Conta da AWS

Depois de executar um comando, você pode recuperar informações sobre a execução do comando no AWS IoT console e usando o. AWS CLI Você pode acessar as informações a seguir.

- Um ID de execução, que é um identificador exclusivo da execução do comando.
- O status da execução do comando. Quando você executa o comando no dispositivo de destino, a execução entra no estado `CREATED`. Depois, ele pode fazer a transição para outros status de execução, conforme descrito abaixo.
- O ID de comando exclusivo e o dispositivo de destino para o qual as execuções foram criadas.
- A Data de início, que mostra a hora em que a execução do comando foi criada.

Listar as execuções de comandos na sua conta (console)

Você pode recuperar todas as execuções de comandos do console usando qualquer um dos métodos a seguir.

- Na página Hub de comandos

  Acesse a página do Command Hub do AWS IoT console e execute essas etapas.

  1. Escolha o comando para o qual você criou uma execução no dispositivo de destino.
  2. Na página de detalhes do comando, acesse a guia Histórico de comandos. Você verá uma lista das execuções criadas.
- Na página Hub de coisas

  Se você escolheu AWS IoT algo como dispositivo de destino ao executar o comando e criou várias execuções de comando para um único dispositivo, você pode ver as execuções do dispositivo na página Thing hub.

  1. Acesse a página Thing Hub no AWS IoT console e escolha a coisa para a qual você criou as execuções.
  2. Na página de detalhes da coisa, no histórico de comandos, você verá a lista de execuções criadas para o dispositivo.

Listar as execuções de comandos em sua conta (CLI)

Use a operação da API HTTP do plano de `ListCommandExecutions` AWS IoT Core controle para listar todas as execuções de comandos em sua conta.

Exemplo de política do IAM

Antes de usar essa operação de API, garanta que sua política do IAM autorize você a realizar essa ação no dispositivo. O exemplo a seguir mostra uma política do IAM que permite ao usuário realizar a ação `ListCommandExecutions`.

Neste exemplo, substitua:

- *`region`* por sua Região da AWS, como `us-east-1`.
- *`account-id`* pelo número de sua Conta da AWS , como *`123456789012`*.
- *`command-id`*com seu identificador de AWS IoT comando exclusivo, como*`LockDoor`*.

```
{
  "Effect": "Allow",
  "Action": "iot:ListCommandExecutions",
  "Resource": *
}
```

Exemplo de como listar as execuções de comandos

O exemplo a seguir mostra como listar as execuções de comandos na Conta da AWS.

Ao executar o comando, você deve especificar se deseja filtrar a lista para exibir somente execuções de comandos que foram criadas para um dispositivo específico usando o `targetArn` ou execuções para um comando especificado usando o `commandArn`.

Neste exemplo, substitua:

- *<target-arn>* pelo nome do recurso da Amazon (ARN) do dispositivo no qual pretende realizar a execução, como `arn:aws:iot:`*us-east-1*`:`*123456789012*`:thing/`*b8e4157c98f332cffb37627f*.
- *<target-arn>* pelo nome do recurso da Amazon (ARN) do dispositivo no qual você planeja a execução, como `arn:aws:iot:`*us-east-1*`:`*123456789012*`:thing/`*b8e4157c98f332cffb37627f*.
- *<after>* pelo tempo após o qual você deseja listar as execuções criadas, por exemplo, `2024-11-01T03:00`.

```
aws iot list-command-executions \
--target-arn <target-arn> \
--started-time-filter '{after=<after>}' \
--sort-order "ASCENDING"
```

A execução desse comando gera uma resposta com uma lista das execuções de comando criadas, a hora em que elas começaram a ser realizadas e quando foram concluídas. Ela também fornece informações de status e o objeto `statusReason` que contém informações adicionais sobre o status.

```
{
    "commandExecutions": [
        {
            "commandArn": "arn:aws:iot:us-east-1:123456789012:command/TestMe002",
            "executionId": "b2b654ca-1a71-427f-9669-e74ae9d92d24",
```

```
            "targetArn": "arn:aws:iot:us-east-1:123456789012:thing/
b8e4157c98f332cffb37627f",
            "status": "TIMED_OUT",
            "createdAt": "2024-11-24T14:39:25.791000-08:00",
            "startedAt": "2024-11-24T14:39:25.791000-08:00"
        },
        {
            "commandArn": "arn:aws:iot:us-east-1:123456789012:command/TestMe002",
            "executionId": "34bf015f-ef0f-4453-acd0-9cca2d42a48f",
            "targetArn": "arn:aws:iot:us-east-1:123456789012:thing/
b8e4157c98f332cffb37627f",
            "status": "IN_PROGRESS",
            "createdAt": "2024-11-24T14:05:36.021000-08:00",
            "startedAt": "2024-11-24T14:05:36.021000-08:00"
        }
    ]
}
```


Para acessar mais informações sobre os diferentes status e os respectivos motivos, consulte Status de execução do comando.

Excluir uma execução de comando

Se você não quiser mais usar uma execução de comando, poderá removê-la permanentemente da sua conta.

> **Note**
>
> - A execução de um comando só poderá ser excluída se tiver entrado em um status terminal, como `SUCCEEDED`, `FAILED` ou `REJECTED`.
> - Essa operação só pode ser executada usando a AWS IoT Core API ou AWS CLI o. Essa opção não está disponível no console.

Exemplo de política do IAM

Antes de usar essa operação da API, verifique se sua política do IAM autoriza seu dispositivo a realizar essas ações. Veja a seguir um exemplo de política que autoriza seu dispositivo a realizar a ação.

Neste exemplo, substitua:

- *Region*com o seu Região da AWS, como`us-east-1`.
- *AccountID* pelo número de sua Conta da AWS , como *123456789012*.
- *CommandID* pelo identificador do comando cuja execução você deseja excluir.
- *devices*com `thing` ou `client` dependendo se seus dispositivos foram registrados como AWS IoT itens ou especificados como clientes MQTT.
- *device-id*com o seu AWS IoT `thing-name` ou`client-id`.

```
{
  "Effect": "Allow",
  "Action": [
      "iot:DeleteCommandExecution"
  ],
  "Resource": [
      "arn:aws:iot:region:account-id:command/command-id",
      "arn:aws:iot:region:account-id:devices/device-id"
  ]
}
```

Exemplo de como excluir uma execução de comando

O exemplo a seguir mostra como excluir um comando usando o `delete-command` AWS CLI comando. Dependendo da sua aplicação, substitua *<execution-id>* pelo identificador da execução do comando que você está excluindo e *<target-arn>* pelo ARN do dispositivo de destino.

```
aws iot delete-command-execution \
--execution-id <execution-id> \
--target-arn <target-arn>
```

Se a solicitação da API for bem-sucedida, a execução do comando gerará um código de status de 200. Você pode usar a API `GetCommandExecution` para verificar se a execução do comando não existe mais na sua conta.

Suspender um recurso de comando

Desative os comandos para indicar que estão desatualizados e não devem ser usados. Por exemplo, os comandos obsoletos não são mais mantidos ativamente ou ao criar comandos mais novos com o mesmo ID, mas com cargas diferentes.

Considerações importantes

Considerações importantes ao descontinuar comandos:

- A descontinuação de um comando não o exclui. Você pode recuperar o comando usando seu ID e restaurá-lo para reutilização.
- A tentativa de iniciar novas execuções em comandos obsoletos gera um erro, impedindo o uso de comandos desatualizados.
- Para executar um comando obsoleto, primeiro restaure-o. Após a restauração, o comando fica disponível para uso e execução regulares nos dispositivos de destino.
- Se você descontinuar um comando enquanto as execuções estiverem em andamento, elas continuarão em execução até serem concluídas. Você ainda pode recuperar o status de execução.

Suspender um recurso de comando (console)

Para descontinuar um comando do console, acesse o Command Hub do AWS IoT console e execute as etapas a seguir.

1. Escolha o comando que você deseja descontinuar e, em Ações, escolha Descontinuar.
2. Confirme que você deseja suspender o comando e, depois, selecione Suspender.

Suspender um recurso de comando (CLI)

Marque os comandos como obsoletos usando a CLI. `update-command` Você deve descontinuar um comando antes de excluí-lo. Para usar um comando obsoleto, restaure-o primeiro.

```
aws iot update-command \
    --command-id <command-id> \
    --deprecated
```

Por exemplo, se você suspendeu o comando *ACSwitch* atualizado no exemplo acima, o código a seguir mostrará um exemplo de saída da execução.

```
{
    "commandId": "turnOffAc",
    "deprecated": true,
    "lastUpdatedAt": "2024-05-09T23:16:51.370000-07:00"
```

```
}
```

Conferir a hora e o status da suspensão

Use a `GetCommand` API para determinar se um comando está obsoleto e quando foi descontinuado pela última vez.

```
aws iot get-command --command-id <turnOffAC>
```

Esse comando gera uma resposta contendo informações do comando, incluindo registros de data e hora de criação e descontinuação do último campo atualizado. Isso ajuda a determinar a vida útil do comando e se ele deve ser excluído ou reutilizado. Veja a seguir um exemplo de resposta para o *`turnOffAc`* comando:

```
{
    "commandId": "turnOffAC",
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/turnOffAC",
    "namespace": "AWS-IoT",
    "payload": {
        "content": "testPayload.json",
        "contentType": "application/json"
    },
    "createdAt": "2024-03-23T00:50:10.095000-07:00",
    "lastUpdatedAt": "2024-05-09T23:16:51.370000-07:00",
    "deprecated": false
}
```

Restaurar um recurso de comando

Para usar ou enviar o `ACSwitch` comando para seu dispositivo, restaure-o primeiro.

Para restaurar um comando do console, acesse o Hub de Comando do AWS IoT console, escolha o comando que você deseja restaurar e, em Ações, escolha Restaurar.

Para restaurar um comando usando a AWS IoT Core API ou a AWS CLI, use a operação da `UpdateCommand` API ou a `update-command` CLI. O código a seguir mostra um exemplo de solicitação e resposta.

```
aws iot update-command \
    --command-id <command-id>
```

```
    --no-deprecated
```

O código a seguir mostra um exemplo de saída.

```
{
    "commandId": "ACSwitch",
    "deprecated": false,
    "lastUpdatedAt": "2024-05-09T23:17:21.954000-07:00"
}
```

AWS IoT tunelamento seguro

Quando os dispositivos são implantados atrás de firewalls restritos em sites remotos, você precisa de uma maneira de obter acesso a esses dispositivos para solução de problemas, atualizações de configuração e outras tarefas operacionais. Use o tunelamento seguro para estabelecer comunicação bidirecional com dispositivos remotos por meio de uma conexão segura gerenciada por. AWS IoT O encapsulamento seguro não requer atualizações para a regra de firewall de entrada existente, portanto, é possível manter o mesmo nível de segurança fornecido pelas regras de firewall em um site remoto.

Por exemplo, um dispositivo de sensor localizado em uma fábrica que está a algumas centenas de milhas de distância está tendo problemas para medir a temperatura da fábrica. É possível usar o encapsulamento seguro para abrir e iniciar rapidamente uma sessão nesse dispositivo de sensor. Depois de identificar o problema (por exemplo, um arquivo de configuração incorreto), será possível redefinir o arquivo e reiniciar o dispositivo de sensor por meio da mesma sessão. Em comparação com uma solução de problemas mais tradicional (por exemplo, enviar um técnico à fábrica para investigar o dispositivo de sensor), o encapsulamento seguro diminui a resposta a incidentes e o tempo de recuperação e os custos operacionais.

O que é encapsulamento seguro?

Use o encapsulamento seguro para acessar dispositivos implantados por trás de firewalls com restrição de portas em locais remotos. Você pode se conectar ao dispositivo de destino usando seu laptop ou computador desktop como dispositivo de origem usando o Nuvem AWS. A origem e o destino se comunicam usando um proxy local de código aberto que é executado em cada dispositivo. O proxy local se comunica com o Nuvem AWS usando uma porta aberta permitida pelo firewall, normalmente 443. Os dados transmitidos pelo túnel são criptografados usando o Transport Layer Security (TLS).

Tópicos

- Conceitos de encapsulamento seguro
- Como funciona o encapsulamento seguro
- Ciclo de vida do túnel seguro

Conceitos de encapsulamento seguro

Os termos a seguir são usados pelo encapsulamento seguro ao estabelecer comunicação com dispositivos remotos. Para obter mais informações sobre encapsulamento seguro, consulte Como funciona o encapsulamento seguro.

Token de acesso para cliente (CAT)

: Um par de tokens gerados pelo encapsulamento seguro quando um túnel é criado. O CAT é usado pelos dispositivos de origem e de destino para se conectar ao serviço de encapsulamento seguro. O CAT só pode ser usado uma vez para se conectar ao túnel. Para se reconectar ao túnel, gire os tokens de acesso do cliente usando a operação de RotateTunnelAccessTokenAPI ou o comando CLI rotate-tunnel-access-token.

Token de cliente

: Um valor exclusivo gerado pelo cliente que o tunelamento AWS IoT seguro pode usar para todas as conexões de repetição subsequentes com o mesmo túnel. Esse campo é opcional. Se o token do cliente não for fornecido, o token de acesso do cliente (CAT) só poderá ser usado uma vez para o mesmo túnel. Tentativas de conexão subsequentes usando o mesmo CAT serão rejeitadas. Para obter mais informações sobre o uso de tokens de cliente, consulte a implementação da referência de proxy local em GitHub.

Aplicativo de destino

: O aplicativo que é executado no dispositivo de destino. Por exemplo, o aplicativo de destino pode ser um daemon SSH para estabelecer uma sessão SSH usando o encapsulamento seguro.

Dispositivo de destino

: O dispositivo remoto que você deseja acessar.

Atendente do dispositivo

: Um aplicativo de IoT que se conecta ao gateway do AWS IoT dispositivo e escuta novas notificações de túnel pelo MQTT. Para obter mais informações, consulte Snippet de atendente de IoT.

Proxy local

: Um proxy de software que é executado nos dispositivos de origem e de destino e retransmite um fluxo de dados entre o serviço de encapsulamento seguro e o aplicativo do dispositivo. O

proxy local pode ser executado no modo de origem ou no modo de destino. Para obter mais informações, consulte Proxy local.

Dispositivo de origem

O dispositivo que um operador usa para iniciar uma sessão no dispositivo de destino, geralmente um laptop ou computador desktop.

Túnel

Um caminho lógico AWS IoT que permite a comunicação bidirecional entre um dispositivo de origem e um dispositivo de destino.

Como funciona o encapsulamento seguro

Veja a seguir como o encapsulamento seguro estabelece uma conexão entre o dispositivo de origem e o de destino. Para obter informações sobre os diferentes termos, como token de acesso do cliente (CAT), consulte Conceitos de encapsulamento seguro.

1. Abrir um túnel

   Para abrir um túnel para iniciar uma sessão com seu dispositivo de destino remoto, você pode usar o Console de gerenciamento da AWS comando AWS CLI open-tunnel ou a OpenTunnel API.

2. Baixe o par de tokens de acesso do cliente

   Depois de abrir um túnel, você pode baixar o token de acesso do cliente (CAT) para sua origem e destino e salvá-lo em seu dispositivo de origem. Você deve recuperar o CAT e salvá-lo agora antes de iniciar o proxy local.

3. Inicie o proxy local no modo de destino

   O atendente de IoT que foi instalado e está sendo executado no seu dispositivo de destino será inscrito no tópico reservado do MQTT `$aws/things/thing-name/tunnels/notify` e receberá o CAT. Aqui *`thing-name`* está o nome da AWS IoT coisa que você cria para o seu destino. Para obter mais informações, consulte Proteger tópicos de túneis.

   O atendente de IoT então usa o CAT para iniciar o proxy local no modo de destino e configurar uma conexão no lado de destino do túnel. Para obter mais informações, consulte Snippet de atendente de IoT.

4. Inicie o proxy local no modo de origem

Depois que o túnel for aberto, AWS IoT Device Management fornece o CAT para a fonte que você pode baixar no dispositivo de origem. Você pode usar o CAT para iniciar o proxy local no modo de origem, que então conecta o lado de origem do túnel. Para mais informações sobre proxy local, consulte Proxy local.

5. Abra uma sessão SSH

   Como os dois lados do túnel estão conectados, você pode iniciar uma sessão SSH usando o proxy local no lado de origem.

Para obter mais informações sobre como usar o Console de gerenciamento da AWS para abrir um túnel e iniciar uma sessão SSH, consulteAbra um túnel e inicie a sessão SSH no dispositivo remoto.

O vídeo a seguir descreve como o encapsulamento seguro funciona e orienta você no processo de configuração de uma sessão SSH em um dispositivo Raspberry Pi.

Ciclo de vida do túnel seguro

Os túneis podem ter o status de `OPEN` ou `CLOSED`. As conexões com o túnel podem ter o status de `CONNECTED` ou `DISCONNECTED`. Veja a seguir como os diferentes status de túnel e conexão funcionam.

1. Ao abrir um túnel, ele terá o status de `OPEN`. O status da conexão de origem e de destino do túnel é definido como `DISCONNECTED`.
2. Quando um dispositivo (de origem ou de destino) se conecta ao túnel, o status da conexão correspondente muda para `CONNECTED`.
3. Quando um dispositivo se desconecta do túnel enquanto o status do túnel permanece `OPEN`, o status da conexão correspondente volta a ser `DISCONNECTED`. Um dispositivo pode se conectar e se desconectar de um túnel repetidamente enquanto o túnel permanece como `OPEN`.

> **Note**
>
> Os tokens de acesso do cliente (CAT) só podem ser usados uma vez para conectar-se a um túnel. Para se reconectar ao túnel, gire os tokens de acesso do cliente usando a operação de RotateTunnelAccessTokenAPI ou o comando CLI rotate-tunnel-access-token. Para ver exemplos, consulte Resolvendo problemas de conectividade de tunelamento AWS IoT seguro por meio da rotação dos tokens de acesso do cliente.

4. Quando você faz uma chamada para `CloseTunnel` ou o túnel permanece `OPEN` por mais tempo que o valor `MaxLifetimeTimeout`, o status do túnel se torna `CLOSED`. É possível configurar `MaxLifetimeTimeout` ao fazer a chamada `OpenTunnel`. `MaxLifetimeTimeout` assumirá o padrão de 12 horas se você não especificar um valor.

   > **Note**
   >
   > Um túnel não pode ser reaberto quando está no status `CLOSED`.

5. Você pode fazer a chamada para `DescribeTunnel` e `ListTunnels` para visualizar metadados do túnel enquanto o túnel estiver visível. O túnel pode ficar visível no AWS IoT console por pelo menos três horas antes de ser excluído.

AWS IoT tutoriais de tunelamento seguro

AWS IoT o tunelamento seguro ajuda os clientes a estabelecer comunicação bidirecional com dispositivos remotos protegidos por um firewall por meio de uma conexão segura gerenciada pela. AWS IoT

Para demonstrar o tunelamento AWS IoT seguro, use nossa demonstração de tunelamento AWS IoT seguro em. GitHub

Os tutoriais a seguir ajudarão você a aprender como começar e usar o encapsulamento seguro. Você aprenderá a:

1. Criar um encapsulamento seguro usando os métodos de configuração rápida e manual para acessar o dispositivo remoto.
2. Configure o proxy local ao usar o método de configuração manual e conecte-se ao encapsulamento para acessar o dispositivo de destino.
3. Faça SSH no dispositivo remoto a partir de um navegador sem precisar configurar o proxy local.
4. Converta um túnel criado usando o AWS CLI ou usando o método de configuração manual para usar o método de configuração rápida.

Tutoriais nesta seção

Os tutoriais desta seção se concentram na criação de um túnel usando a Console de gerenciamento da AWS e a Referência da AWS IoT API. No AWS IoT console, você pode criar um túnel a partir

da página do hub de túneis ou da página de detalhes de algo que você criou. Para obter mais informações, consulte Métodos de criação de túneis no AWS IoT console.

Este tutorial contém as seguintes seções:

- Abra um túnel e use o SSH baseado em navegador para acessar o dispositivo remoto

  Este tutorial mostra como abrir um túnel a partir da página Hub de encapsulamento usando o método de configuração rápida. Você também aprenderá a usar o SSH baseado em navegador para acessar o dispositivo remoto usando uma interface de linha de comando contextualizada no console. AWS IoT

- Abra um túnel usando a configuração manual e conecte-se ao dispositivo remoto

  Este tutorial mostra como abrir um túnel a partir da página Hub de encapsulamento usando o método de configuração manual. Você também aprenderá como configurar e iniciar o proxy local a partir de um terminal em seu dispositivo de origem e conectar-se ao encapsulamento.

- Abra um túnel para dispositivo remoto e use SSH baseado em navegador

  Este tutorial mostra como abrir um túnel a partir da página de detalhes de um item que você criou. Você aprenderá a criar um novo encapsulamento e usar um encapsulamento existente. O encapsulamento existente corresponde ao túnel aberto mais recente que foi criado para o dispositivo. Você também pode usar o SSH baseado em navegador para acessar o dispositivo remoto.

AWS IoT tutoriais de tunelamento seguro

- Abra um túnel e inicie a sessão SSH no dispositivo remoto
- Abra um túnel para dispositivo remoto e use SSH baseado em navegador

Abra um túnel e inicie a sessão SSH no dispositivo remoto

Nesses tutoriais, você aprenderá como acessar remotamente um dispositivo protegido por um firewall. Você não pode iniciar uma sessão SSH direto no dispositivo porque o firewall bloqueia todo o tráfego de entrada. Os tutoriais mostram como você pode abrir um encapsulamento e depois usá-lo para iniciar uma sessão SSH em um dispositivo remoto.

Pré-requisitos para o tutorial

Os pré-requisitos para executar o tutorial podem variar dependendo se você usa os métodos de configuração manual ou rápida para abrir um túnel e acessar o dispositivo remoto.

> **Note**
>
> Em ambos os métodos de configuração, você deve permitir o tráfego de saída na porta 443.

- Para obter informações sobre os pré-requisitos do tutorial do método de configuração rápida, consulte Pré-requisitos para o método de configuração rápida.
- Para obter informações sobre os pré-requisitos do tutorial do método de configuração manual, consulte Pré-requisitos para o método de configuração manual. Se você usar esse método de configuração, deverá configurar o proxy local no dispositivo de origem. Para baixar o código-fonte do proxy local, consulte Implementação de referência do proxy local em GitHub.

Métodos de configuração de túnel

Nesses tutoriais, você aprenderá sobre os métodos de configuração manual e rápida para abrir um túnel e conectar-se ao dispositivo remoto. A tabela a seguir mostra a diferença entre os métodos de configuração. Depois de criar o túnel, você pode usar uma interface de linha de comando no navegador para SSH no dispositivo remoto. Se você perder os tokens ou o túnel for desconectado, poderá enviar novos tokens de acesso para se reconectar ao túnel.

Métodos de configuração rápida e manual

Critérios	Configuração rápida	Configuração manual
Criação de túnel	Crie um novo túnel com configurações padrão editáveis. Para acessar seu dispositivo remoto, você só pode usar o SSH como serviço de destino.	Crie um túnel especificando manualmente as configurações do túnel. Você pode usar esse método para se conectar ao dispositivo remoto usando serviços diferentes do SSH.
Tokens de acesso	O token de acesso de destino será entregue automaticamente ao seu dispositivo no tópico reservado do	Você não precisa baixar ou gerenciar manualmente o token em seu dispositivo de origem. O token de acesso de destino

Critérios	Configuração rápida	Configuração manual
	MQTT, se um nome de item for especificado ao criar o túnel. Você não precisa baixar ou gerenciar o token em seu dispositivo de origem.	é entregue automaticamente ao dispositivo remoto no tópico MQTT reservado, se um nome de item for especificado ao criar o túnel.
Proxy local	Um proxy local baseado na web é configurado automaticamente para você interagir com o dispositivo. Não é necessário configurar manualmente o proxy local.	Você terá que configurar e iniciar manualmente o proxy local. Para configurar o proxy local, você pode usar o AWS IoT Device Client ou baixar a implementação de referência do proxy local em GitHub.

Métodos de criação de túneis no AWS IoT console

Os tutoriais desta seção mostram como criar um túnel usando a Console de gerenciamento da AWS e a OpenTunnelAPI. Se você configurar o destino ao criar um túnel, o tunelamento AWS IoT seguro entregará o token de acesso do cliente de destino ao dispositivo remoto por meio do MQTT e ao tópico reservado do MQTT,). `$aws/things/RemoteDeviceA/tunnels/notify` Após o recebimento da mensagem MQTT, o atendente de IoT no dispositivo remoto inicia o proxy local no modo de destino. Para obter mais informações, consulte Tópicos reservados.

> **Note**
>
> É possível omitir a configuração de destino se quiser entregar o token de acesso do cliente de destino ao dispositivo remoto por meio de outro método. Para obter mais informações, consulte Como configurar um dispositivo remoto e usar o atendente de IoT.

No AWS IoT console, você pode criar um túnel usando um dos métodos a seguir. Para obter informações sobre tutoriais que ajudarão você a aprender a criar um túnel usando esses métodos, consulte Tutoriais nesta seção.

- Hub de túneis

  Ao criar o túnel, você poderá especificar se deseja usar os métodos de configuração rápida ou manual para criar o túnel e fornecer os detalhes opcionais da configuração do túnel. Os detalhes da configuração também incluem o nome do dispositivo de destino e o serviço que você

deseja usar para se conectar ao dispositivo. Depois de criar um túnel, você pode usar o SSH no navegador ou abrir um terminal fora do AWS IoT console para acessar seu dispositivo remoto.

- Página de detalhes do item

  Ao criar o túnel, você também poderá especificar se deseja usar o túnel aberto mais recente ou criar um novo túnel para o dispositivo, além de escolher os métodos de configuração e fornecer quaisquer detalhes opcionais de configuração do túnel. Não é possível editar os detalhes da configuração de um túnel existente. Você pode usar o método de configuração rápida para alternar os tokens de acesso e o SSH no dispositivo remoto dentro do navegador. Para abrir um túnel usando esse método, você deve ter criado uma coisa de IoT (por exemplo,`RemoteDeviceA`) no AWS IoT registro. Para obter mais informações, consulte Registrar um dispositivo no AWS IoT registro.

Tutoriais nesta seção

- Abra um túnel e use o SSH baseado em navegador para acessar o dispositivo remoto
- Abra um túnel usando a configuração manual e conecte-se ao dispositivo remoto

Abra um túnel e use o SSH baseado em navegador para acessar o dispositivo remoto

Você pode usar a configuração rápida ou o método de configuração manual para criar um túnel. Este tutorial mostra como abrir um túnel usando o método de configuração rápida e usar o SSH baseado em navegador para se conectar ao dispositivo remoto. Para obter um exemplo que mostra como abrir um túnel usando o método de configuração manual, consulte Abra um túnel usando a configuração manual e conecte-se ao dispositivo remoto.

Usando o método de configuração rápida, você pode criar um novo túnel com configurações padrão que podem ser editadas. Um proxy local baseado na web é configurado para você e o token de acesso é entregue automaticamente ao seu dispositivo de destino remoto usando o MQTT. Depois de criar um túnel, você pode começar a interagir com seu dispositivo remoto usando uma interface de linha de comando dentro do console.

Com o método de configuração rápida, você deve usar o SSH como serviço de destino para acessar o dispositivo remoto. Para obter informações sobre os diferentes métodos de configuração, consulte Métodos de configuração de túnel.

Pré-requisitos para o método de configuração rápida

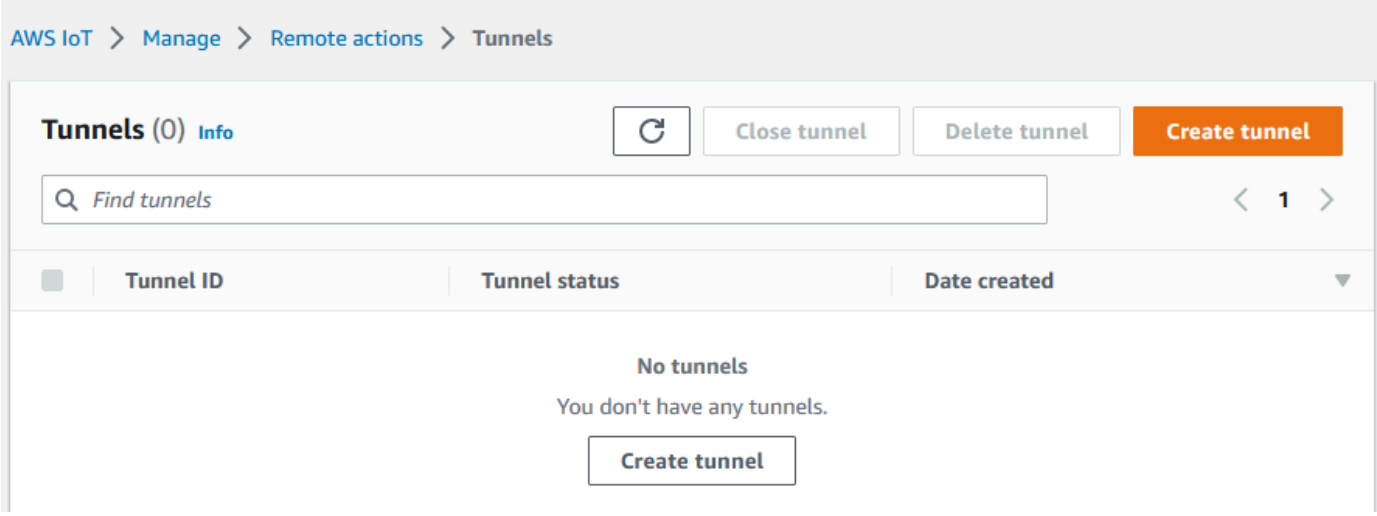
- Os firewalls que protegem o dispositivo remoto devem permitir o tráfego de saída na porta 443. O túnel que você criar usará essa porta para se conectar ao dispositivo remoto.
- Você tem um agente de dispositivo IoT (consulteSnippet de atendente de IoT) em execução no dispositivo remoto que se conecta ao gateway do AWS IoT dispositivo e está configurado com uma assinatura de tópico do MQTT. Para obter mais informações, consulte conectar um dispositivo ao gateway do AWS IoT dispositivo.
- É necessário ter um daemon SSH em execução no dispositivo remoto.

Abrir um túnel

Você pode abrir um túnel seguro usando a Console de gerenciamento da AWS, a Referência da AWS IoT API ou AWS CLI a. Opcionalmente, você pode configurar um nome de destino, mas isso não é necessário para este tutorial. Se você configurar o destino, o encapsulamento seguro entregará automaticamente o token de acesso ao dispositivo remoto usando o MQTT. Para obter mais informações, consulte Métodos de criação de túneis no AWS IoT console.

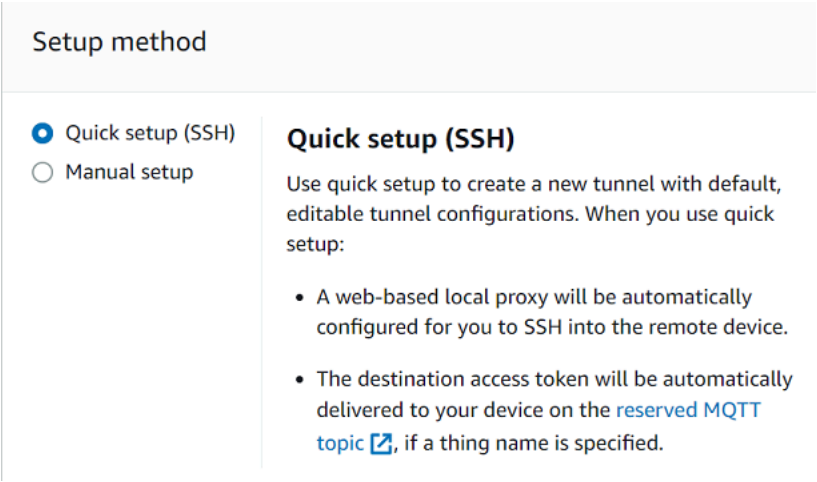
Abrir um túnel usando o console

1. Vá para o hub de túneis do AWS IoT console e selecione Criar túnel.



2. Para este tutorial, selecione Configuração rápida como método de criação de túneis e, em seguida, selecione Próximo.

> **Note**
>
> Se você criar um túnel seguro a partir da página de detalhes de algo que você criou, você pode escolher entre criar um novo túnel ou usar um existente. Para obter mais informações, consulte Abra um túnel para dispositivo remoto e use SSH baseado em navegador.

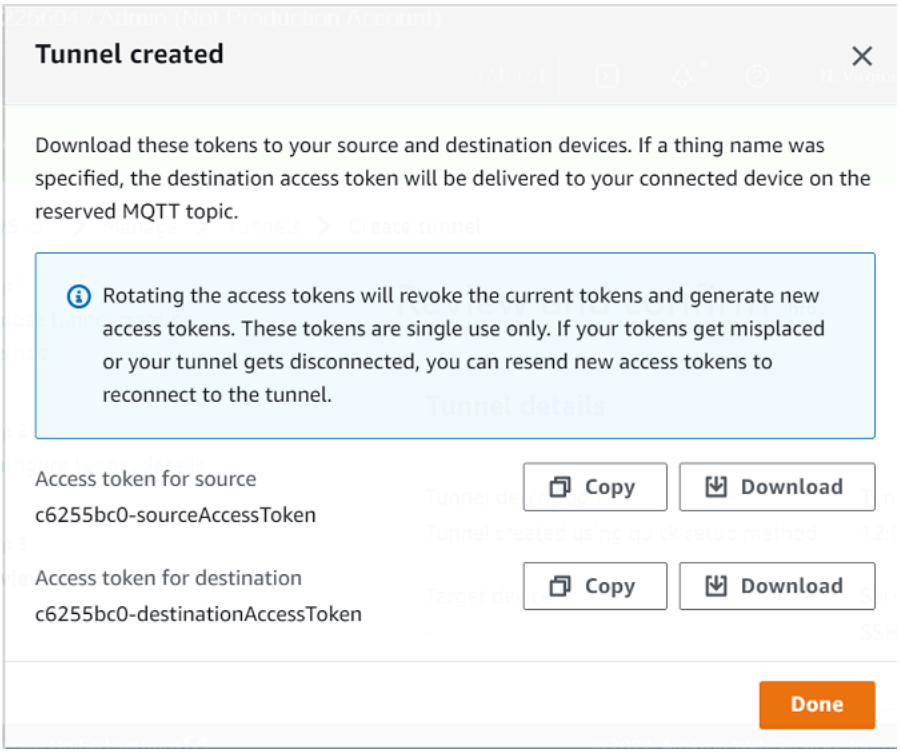


3. Revise e confirme os detalhes da configuração do túnel. Para criar um túnel, escolha Confirmar e criar. Se você quiser editar esses detalhes, escolha Anterior para voltar à página anterior e, em seguida, confirme e crie o túnel.

> **Note**
>
> Ao usar a configuração rápida, o nome do serviço não pode ser editado. Você deve usar o SSH como o Serviço.

4. Para criar o túnel, selecione Concluído.

   Neste tutorial, não é necessário fazer download dos tokens de acesso de origem ou de destino. Esses tokens só podem ser usados uma vez para se conectar ao túnel. Se o túnel for desconectado, você poderá gerar e enviar novos tokens para o dispositivo remoto para se reconectar ao túnel. Para obter mais informações, consulte Reenviar tokens de acesso ao túnel.



Abrir um túnel usando a API

Para abrir um novo túnel, você pode usar a operação OpenTunnelda API.

> **Note**
>
> Você pode criar um túnel usando o método de configuração rápida somente a partir do AWS IoT console. Quando você usa a AWS IoT API de referência da API ou a AWS CLI, ela usa o método de configuração manual. Você pode abrir o túnel existente que você criou e, em seguida, alterar o método de configuração do túnel para usar a configuração rápida. Para obter mais informações, consulte Abra um túnel existente e use o SSH baseado em navegador.

Mais adiante, veja um exemplo de como executar essa operação da API. Opcionalmente, se você quiser especificar o nome do item e o serviço de destino, use o parâmetro `DestinationConfig`. Para obter um exemplo que mostra como usar esse parâmetro, consulte Abra um novo túnel para o dispositivo remoto.

```
aws iotsecuretunneling open-tunnel
```

A execução desse comando cria um novo túnel e fornece os tokens de acesso de origem e destino.

```
{
    "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
    "tunnelArn": "arn:aws:iot:us-
east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
    "sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
    "destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

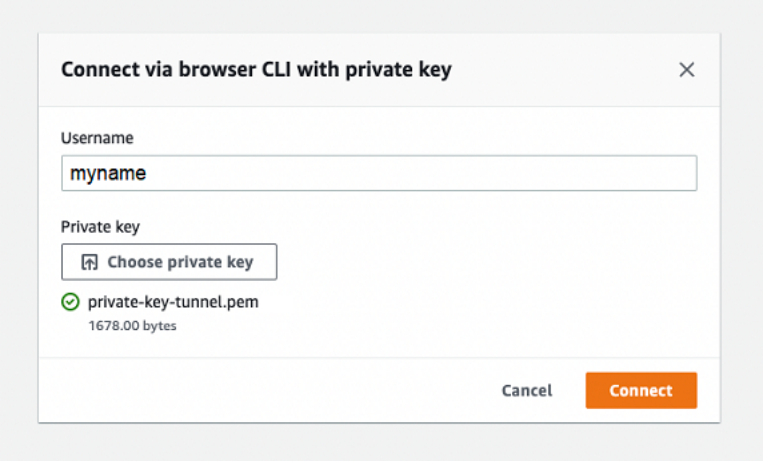
Usando o SSH baseado em navegador

Depois de criar um túnel usando o método de configuração rápida e seu dispositivo de destino se conectar ao túnel, você poderá acessar o dispositivo remoto usando um SSH baseado em navegador. Usando o SSH baseado em navegador, você pode se comunicar diretamente com o dispositivo remoto inserindo comandos em uma interface de linha de comando contextualizada dentro do console. Este atributo facilita a interação com o dispositivo remoto porque você não precisa abrir um terminal fora do console ou configurar o proxy local.

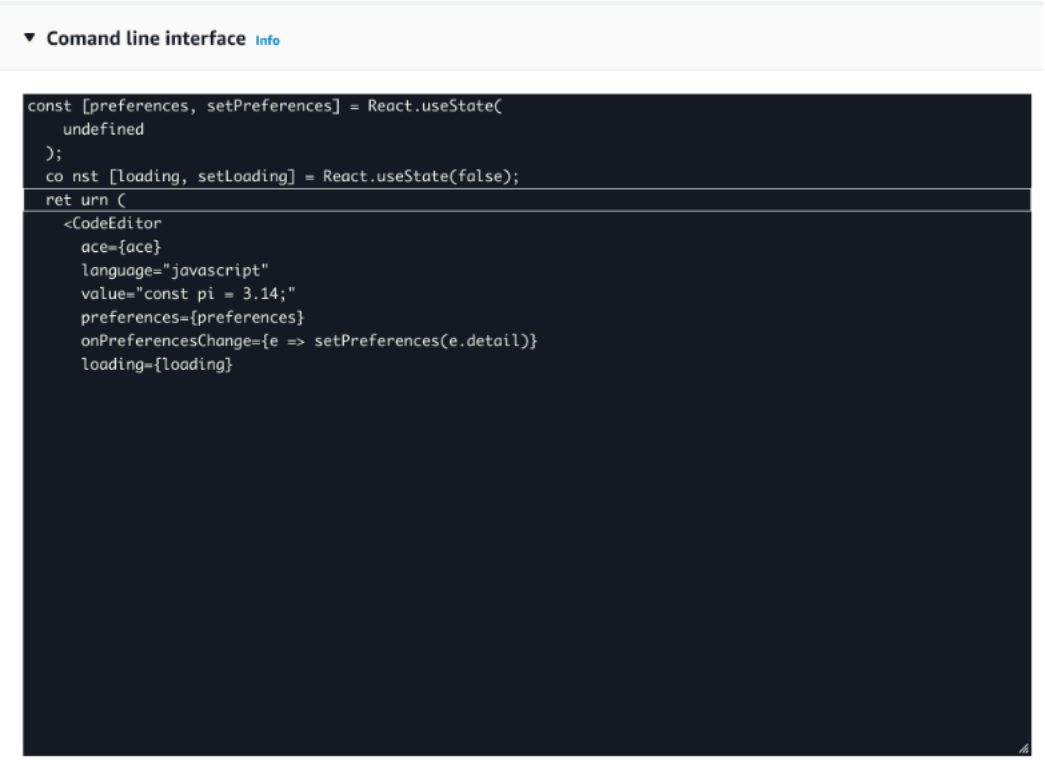
Como usar o SSH baseado em navegador

1. Acesse o Hub túneis do AWS IoT console escolha o túnel que você criou para visualizar seus detalhes.
2. Expanda a seção Secure Shell (SSH) e selecione Conectar.
3. Escolha se você deseja se autenticar na conexão SSH fornecendo seu nome de usuário e senha ou, para uma autenticação mais segura, você pode usar a chave privada do seu dispositivo. Se você estiver se autenticando usando a chave privada, observe que somente as chaves RSA formatadas em PEM (256 e 512) funcionam com o console SSH. AWS IoT Secure Tunneling

   - Para se conectar usando seu nome de usuário e senha, selecione Usar senha. Em seguida, você pode inserir seu nome de usuário e senha e começar a usar a CLI do navegador.
   - Para se conectar usando a chave privada do seu dispositivo de destino, selecione Usar chave privada. Especifique seu nome de usuário e faça o upload do arquivo de chave privada do dispositivo e, em seguida, escolha Conectar para começar a usar a CLI no navegador.



Depois de se autenticar na conexão SSH, você pode começar rapidamente a inserir comandos e interagir com o dispositivo usando a CLI do navegador, pois o proxy local já foi configurado para você.



Se a CLI do navegador permanecer aberta após a duração do túnel, ela poderá expirar, fazendo com que a interface da linha de comando seja desconectada. Você pode duplicar o túnel e iniciar outra sessão para interagir com o dispositivo remoto dentro do próprio console.

Solução de problemas ao usar o SSH baseado em navegador

A seguir, mostramos como solucionar alguns problemas que você pode encontrar ao usar o SSH baseado em navegador.

- Você vê um erro em vez da interface de linha de comando

  Talvez você esteja vendo o erro porque seu dispositivo de destino foi desconectado. Você pode escolher Gerar novos tokens de acesso para gerar novos tokens de acesso e enviar os tokens para seu dispositivo remoto usando o MQTT. Os novos tokens podem ser usados para se reconectar ao túnel. A reconexão ao túnel limpa o histórico e atualiza a sessão da linha de comando.
- Você vê um erro de túnel desconectado ao autenticar usando chave privada

  Talvez você esteja vendo o erro porque sua chave privada pode não ter sido aceita pelo dispositivo de destino. Para solucionar esse erro, verifique o arquivo de chave privada que você carregou para autenticação. Se você ainda vir um erro, verifique os logs do seu dispositivo. Você também pode tentar se reconectar ao túnel enviando novos tokens de acesso ao seu dispositivo remoto.
- Seu túnel foi fechado ao usar a sessão

  Se seu túnel foi fechado porque permaneceu aberto por mais do que a duração especificada, sua sessão de linha de comando pode ser desconectada. Um túnel não pode ser reaberto depois de fechado. Para se reconectar, você deve abrir outro túnel para o dispositivo.

  Você pode duplicar um túnel para criar um novo túnel com as mesmas configurações do túnel fechado. Você pode duplicar um túnel fechado a partir do AWS IoT console. Para duplicar o túnel, escolha o túnel que foi fechado para ver seus detalhes e, em seguida, escolha Duplicar túnel. Especifique a duração do túnel que você deseja usar e, em seguida, crie o novo túnel.

Liberar

- Fechar túnel

  Recomendamos que você feche o túnel depois de terminar de usá-lo. Um túnel também pode ficar fechado se permanecer aberto por mais tempo do que a duração especificada do túnel. Um túnel não pode ser reaberto depois de fechado. Você ainda pode duplicar um túnel escolhendo o túnel fechado e, em seguida, escolhendo Duplicar túnel. Especifique a duração do túnel que você deseja usar e, em seguida, crie o novo túnel.

- Para fechar um túnel individual ou vários túneis a partir do AWS IoT console, acesse o Hub de túneis, escolha os túneis que você deseja fechar e escolha Fechar túnel.
- Para fechar um túnel individual ou vários túneis usando a AWS IoT API de referência da API, use a CloseTunnelAPI.

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
```

- Excluir túnel

  Você pode excluir um túnel permanentemente do seu Conta da AWS.

  > **Warning**
  >
  > Uma ação de exclusão é permanente e não pode ser desfeita.

  - Para excluir um túnel individual ou vários túneis a partir do AWS IoT console, acesse o Hub de túneis, escolha os túneis que você deseja excluir e escolha Excluir túnel.
  - Para excluir um túnel individual ou vários túneis usando a AWS IoT API de referência da API, use a CloseTunnelAPI. Ao usar a API, defina o sinalizador `delete` como `true`.

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    --delete true
```

Abra um túnel usando a configuração manual e conecte-se ao dispositivo remoto

Ao abrir um túnel, você pode escolher a configuração rápida ou o método de configuração manual para abrir um túnel no dispositivo remoto. Este tutorial mostra como abrir um túnel usando o método de configuração manual e configurar e iniciar o proxy local para se conectar ao dispositivo remoto.

Ao usar o método de configuração manual, você deve especificar manualmente as configurações do túnel ao criar o túnel. Depois de criar o túnel, você pode usar o SSH no navegador ou abrir um terminal fora do AWS IoT console. Este tutorial mostra como usar o terminal fora do console para acessar o dispositivo remoto. Você também aprenderá a configurar o proxy local e depois se conectar ao proxy local para interagir com o dispositivo remoto. Para se conectar ao proxy local, você deve baixar o token de acesso de origem ao criar o túnel.

Com este método de configuração, você pode usar outros serviços além do SSH, como FTP, para conectar-se ao dispositivo remoto. Para obter informações sobre os diferentes métodos de configuração, consulte Métodos de configuração de túnel.

Pré-requisitos para o método de configuração manual

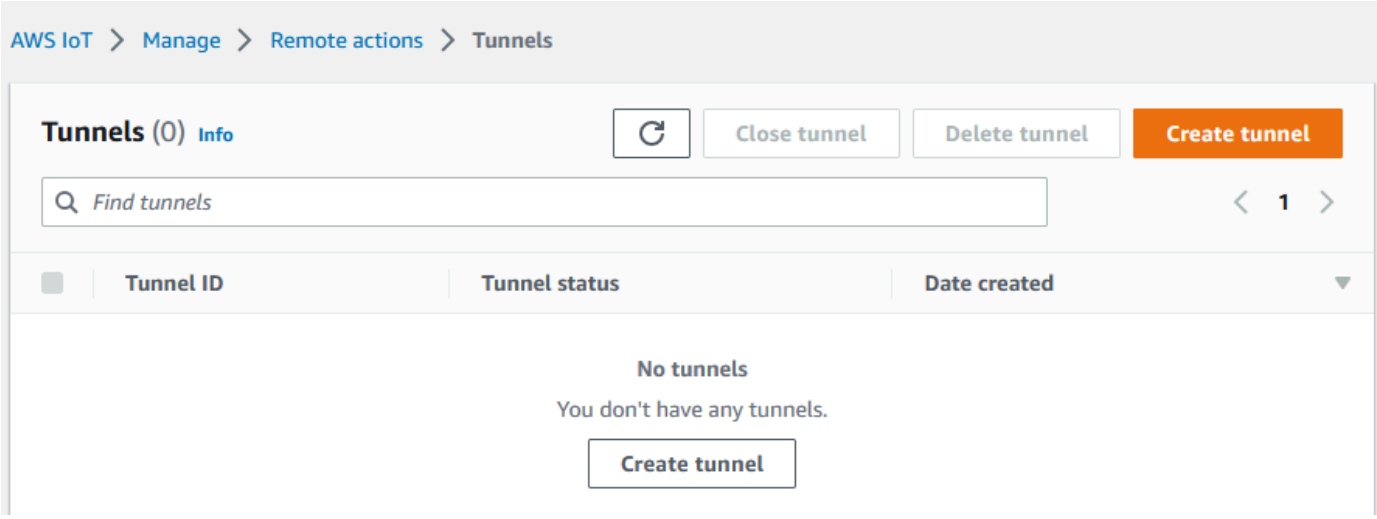
- Os firewalls que protegem o dispositivo remoto devem permitir o tráfego de saída na porta 443. O túnel que você criar usará essa porta para se conectar ao dispositivo remoto.
- Você tem um agente de dispositivo IoT (consulteSnippet de atendente de IoT) em execução no dispositivo remoto que se conecta ao gateway do AWS IoT dispositivo e está configurado com uma assinatura de tópico do MQTT. Para obter mais informações, consulte conectar um dispositivo ao gateway do AWS IoT dispositivo.
- É necessário ter um daemon SSH em execução no dispositivo remoto.
- Você baixou o código-fonte do proxy local GitHube o construiu para a plataforma de sua escolha. O arquivo executável do proxy local criado aparecerá como `localproxy` neste tutorial.

Abrir um túnel

Você pode abrir um túnel seguro usando a Console de gerenciamento da AWS, a Referência da AWS IoT API ou AWS CLI a. Opcionalmente, você pode configurar um nome de destino, mas isso não é necessário para este tutorial. Se você configurar o destino, o encapsulamento seguro entregará automaticamente o token de acesso ao dispositivo remoto usando o MQTT. Para obter mais informações, consulte Métodos de criação de túneis no AWS IoT console.

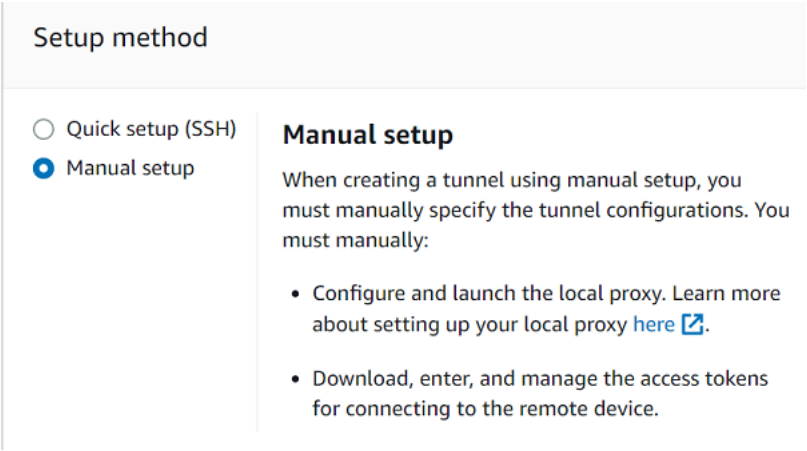
Abrir um túnel no console

1. Vá para o hub de túneis do AWS IoT console e selecione Criar túnel.



2. Para este tutorial, selecione Configuração manual como método de criação de túneis e, em seguida, selecione Próximo. Para obter informações sobre como usar o método de configuração rápida para criar um túnel, consulte Abra um túnel e use o SSH baseado em navegador para acessar o dispositivo remoto.

> **Note**
>
> Se você criar um túnel seguro na página de detalhes de um item, poderá escolher se deseja criar um novo túnel ou usar um existente. Para obter mais informações, consulte Abra um túnel para dispositivo remoto e use SSH baseado em navegador.



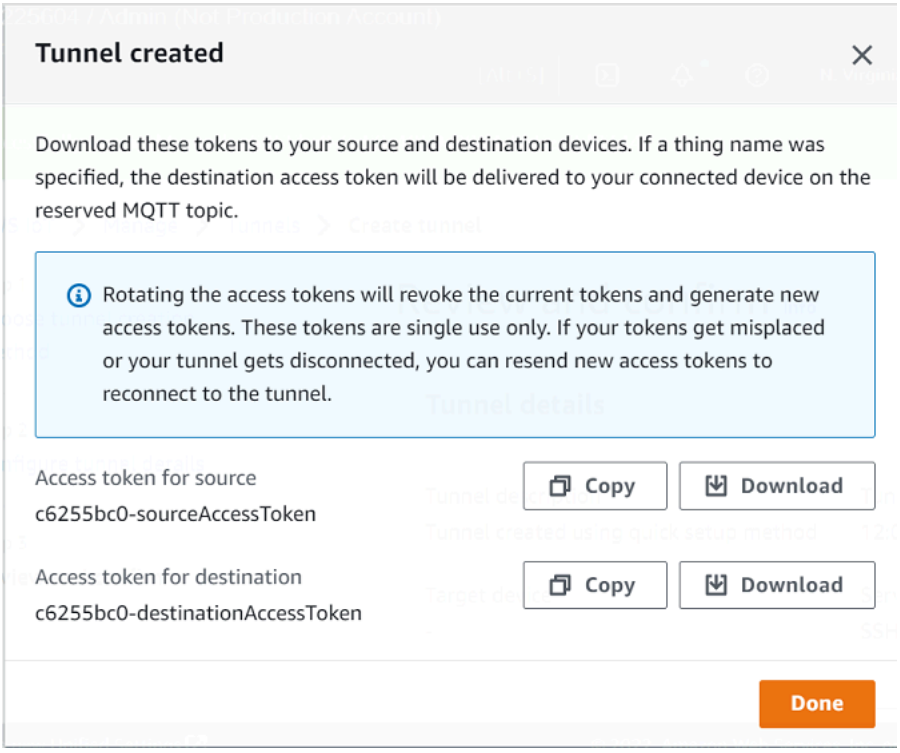
3. (Opcional) Insira as definições de configuração do seu túnel. Você também pode pular essa etapa e prosseguir para a próxima etapa para criar um túnel.

Insira uma descrição do túnel, a duração do tempo limite do túnel e as tags de recursos como pares de valores-chave para ajudá-lo a identificar seu recurso. Neste tutorial, você pode ignorar a configuração de destino.

> **Note**
>
> Você não será cobrado com base no tempo pelo qual você mantém um túnel aberto. Você só incorre em cobranças ao criar um novo túnel. Para obter informações sobre preços, consulte Encapsulamento seguro em AWS IoT Device Management preços.

4. Faça o download dos tokens de acesso do cliente e escolha Concluído. Os tokens não estarão disponíveis para download depois que você escolher Concluído.

   Esses tokens só podem ser usados uma vez para se conectar ao túnel. Se você perder os tokens ou o túnel for desconectado, você poderá gerar e enviar novos tokens ao seu dispositivo remoto para reconectar-se ao túnel.



Abrir um túnel usando a API

Para abrir um novo túnel, você pode usar a operação OpenTunnelda API. Você também pode especificar configurações adicionais usando a API, como a duração do túnel e a configuração de destino.

```
aws iotsecuretunneling open-tunnel \
    --region us-east-1 \
    --endpoint https://api.us-east-1.tunneling.iot.amazonaws.com
```

A execução desse comando cria um novo túnel e fornece os tokens de acesso de origem e destino.

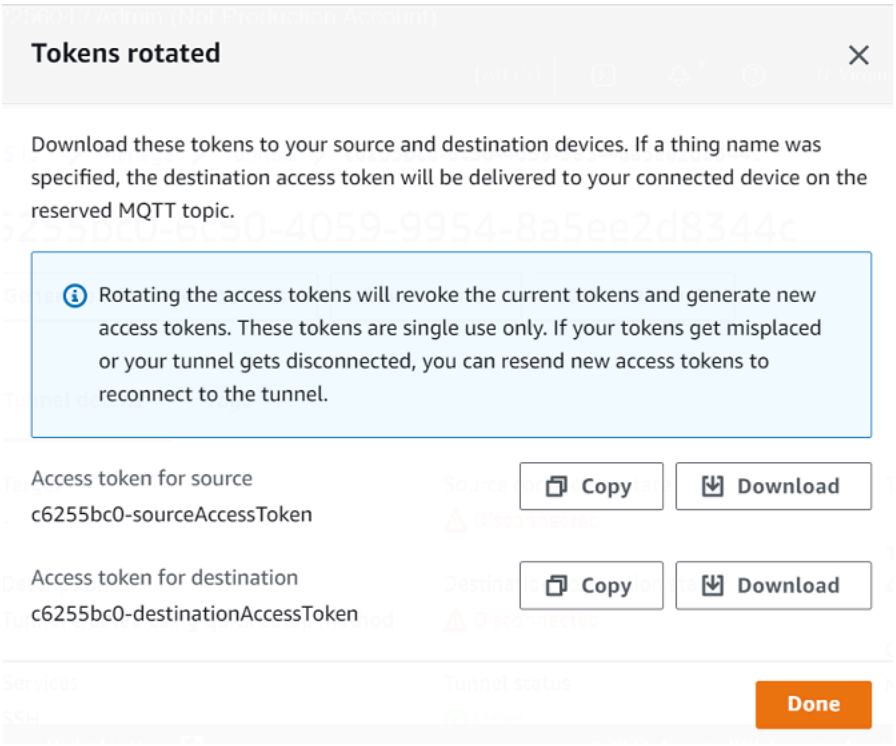
```
{
    "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
    "tunnelArn": "arn:aws:iot:us-
east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
    "sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
    "destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

Reenviar tokens de acesso ao túnel

Os tokens que você obteve ao criar um túnel só podem ser usados uma vez para se conectar ao túnel. Se você perder o token de acesso ou o túnel for desconectado, você poderá reenviar novos tokens de acesso ao dispositivo remoto usando o MQTT sem custo adicional. AWS IoT o tunelamento seguro revogará os tokens atuais e retornará novos tokens de acesso para se reconectar ao túnel.

Para alternar os tokens a partir do console

1. Acesse o hub de túneis do AWS IoT console e escolha o túnel que você criou.
2. Na página de detalhes do túnel, selecione Gerar novos tokens de acesso e, em seguida, selecione Próximo.
3. Baixe os novos tokens de acesso para seu túnel e selecione Concluído. Esses tokens só podem ser usados uma vez. Se você perder esses tokens ou o túnel for desconectado, você poderá reenviar novos tokens de acesso.



Para alternar os tokens de acesso usando a API

Para girar os tokens de acesso ao túnel, você pode usar a operação da [RotateTunnelAccessToken](#)API para revogar os tokens atuais e retornar novos tokens de acesso para se reconectar ao túnel. Por exemplo, o comando a seguir alterna os tokens de acesso para o dispositivo de destino, *RemoteThing1*.

```
aws iotsecuretunneling rotate-tunnel-access-token \
    --tunnel-id <tunnel-id> \
    --client-mode DESTINATION \
    --destination-config thingName=<RemoteThing1>,services=SSH \
    --region <region>
```

A execução desse comando gera o novo token de acesso, como mostrado no exemplo a seguir. O token é então entregue ao dispositivo usando o MQTT para se conectar ao túnel, se o atendente do dispositivo estiver configurado corretamente.

```
{
    "destinationAccessToken": "destination-access-token",
    "tunnelArn": "arn:aws:iot:region:account-id:tunnel/tunnel-id"
}
```

Para exemplos que mostram como e quando alternar os tokens de acesso, consulte Resolvendo problemas de conectividade de tunelamento AWS IoT seguro por meio da rotação dos tokens de acesso do cliente.

Configurar e iniciar o proxy local

Para se conectar ao dispositivo remoto, abra um terminal em seu laptop e configure e inicie o proxy local. O proxy local transmite dados enviados pelo aplicativo em execução no dispositivo de origem usando o tunelamento seguro em uma conexão segura. WebSocket Você pode baixar a fonte do proxy local em GitHub.

Depois de configurar o proxy local, copie o token de acesso do cliente de origem e use-o para iniciar o proxy local no modo de origem. A seguir, segue um exemplo de comando para iniciar o proxy local. No comando a seguir, o proxy local está configurado para receber novas conexões na porta 5555. Neste comando:

- `-r`especifica o Região da AWS, que deve ser a mesma região em que seu túnel foi criado.
- `-s` especifica a porta à qual o proxy deve se conectar.

- `-t` especifica o texto do token do cliente.

```
./localproxy -r us-east-1 -s 5555 -t source-client-access-token
```

A execução desse comando iniciará o proxy local no modo de origem. Se você receber o seguinte erro após executar o comando, configure o caminho da CA. Para obter informações, consulte Proxy local de tunelamento seguro ativado. GitHub

```
Could not perform SSL handshake with proxy server: certificate verify failed
```

Veja a seguir um exemplo de saída da execução do proxy local no modo `source`.

```
...
...

Starting proxy in source mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-east-1.amazonaws.com:443
Resolved proxy  server IP: 10.10.0.11
Connected successfully with proxy server
```

```
Performing SSL handshake with proxy server
Successfully completed SSL handshake with proxy server
HTTP/1.1 101 Switching Protocols

...

Connection: upgrade
channel-id: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
upgrade: websocket

...

Web socket session ID: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-east-1.amazonaws.com:443
Setting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
Resolved bind IP: 127.0.0.1
Listening for new connection on port 5555
```

Iniciar uma sessão SSH

Abra outro terminal e use o comando a seguir para iniciar uma nova sessão SSH conectando-se ao proxy local na porta 5555.

```
ssh username@localhost -p 5555
```

Talvez seja solicitada uma senha para a sessão SSH. Quando finalizar a sessão SSH, digite **`exit`** para fechar a sessão.

Liberar

- Fechar túnel

  Recomendamos que você feche o túnel depois de terminar de usá-lo. Um túnel também pode ficar fechado se permanecer aberto por mais tempo do que a duração especificada do túnel. Um túnel não pode ser reaberto depois de fechado. Você ainda pode duplicar um túnel abrindo o túnel

fechado e selecionando Duplicar túnel. Especifique a duração do túnel que você deseja usar e, em seguida, crie o novo túnel.

- Para fechar um túnel individual ou vários túneis a partir do AWS IoT console, acesse o Hub de túneis, escolha os túneis que você deseja fechar e escolha Fechar túnel.
- Para fechar um túnel individual ou vários túneis usando a AWS IoT API de referência da API, use a operação da CloseTunnelAPI.

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
```

- Excluir túnel

  Você pode excluir um túnel permanentemente do seu Conta da AWS.

  > **Warning**
  >
  > Uma ação de exclusão é permanente e não pode ser desfeita.

  - Para excluir um túnel individual ou vários túneis a partir do AWS IoT console, acesse o Hub de túneis, escolha os túneis que você deseja excluir e escolha Excluir túnel.
  - Para excluir um túnel individual ou vários túneis usando a AWS IoT API de referência da API, use a operação da CloseTunnelAPI. Ao usar a API, defina o sinalizador `delete` como `true`.

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    --delete true
```

Abra um túnel para dispositivo remoto e use SSH baseado em navegador

No AWS IoT console, você pode criar um túnel a partir do hub de túneis ou da página de detalhes de uma coisa de IoT que você criou. Ao criar um túnel a partir do hub Túneis, você pode especificar se deseja criar um túnel usando a configuração rápida ou a configuração manual. Para obter um tutorial de exemplo, consulte Abra um túnel e inicie a sessão SSH no dispositivo remoto.

Ao criar um túnel a partir da página de detalhes do objeto do AWS IoT console, você também pode especificar se deseja criar um novo túnel ou abrir um túnel existente para essa coisa, conforme

ilustrado neste tutorial. Se você escolher um túnel existente, poderá acessar o túnel aberto mais recente que você criou para esse dispositivo. Você pode então usar a interface de linha de comando no terminal para conectar-se via SSH ao dispositivo.

Pré-requisitos

- Os firewalls que protegem o dispositivo remoto devem permitir o tráfego de saída na porta 443. O túnel que você criar usará essa porta para se conectar ao dispositivo remoto.
- Você criou uma coisa de IoT (por exemplo,`RemoteDevice1`) no AWS IoT registro. Esse item corresponde à representação do seu dispositivo remoto na nuvem. Para obter mais informações, consulte Registrar um dispositivo no AWS IoT registro.
- Você tem um agente de dispositivo IoT (consulteSnippet de atendente de IoT) em execução no dispositivo remoto que se conecta ao gateway do AWS IoT dispositivo e está configurado com uma assinatura de tópico do MQTT. Para obter mais informações, consulte conectar um dispositivo ao gateway do AWS IoT dispositivo.
- É necessário ter um daemon SSH em execução no dispositivo remoto.

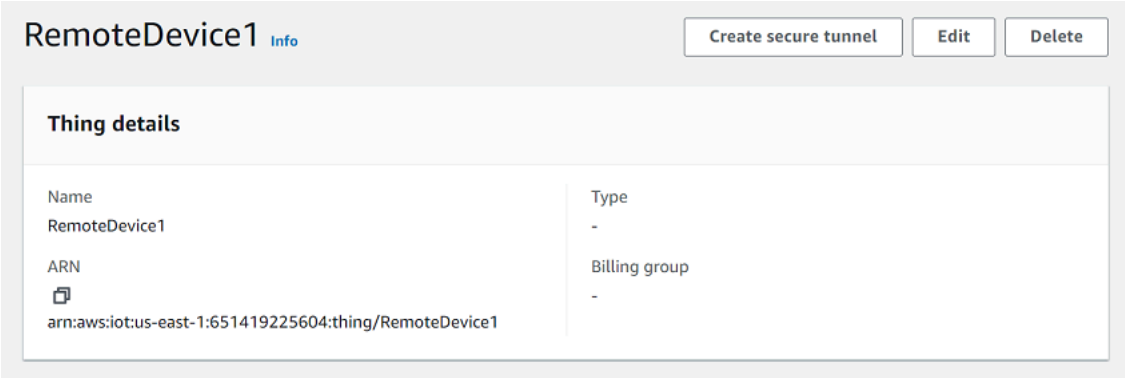
Abra um novo túnel para o dispositivo remoto

Digamos que você queira abrir um túnel em seu dispositivo remoto, `RemoteDevice1`. Primeiro, crie um item da IoT com o nome `RemoteDevice1` no AWS IoT registro. Em seguida, você pode criar um túnel usando a Console de gerenciamento da AWS AWS IoT API de referência da API ou AWS CLI a.

Ao configurar um destino ao criar um túnel, o serviço de encapsulamento seguro entrega o token de acesso do cliente de destino ao dispositivo remoto por meio do MQTT e do tópico MQTT reservado (`$aws/things/RemoteDeviceA/tunnels/notify`). Para obter mais informações, consulte Métodos de criação de túneis no AWS IoT console.

Para criar um túnel para um dispositivo remoto a partir do console

1. Escolha o item, `RemoteDevice1`, para ver seus detalhes e, em seguida, escolha Criar túnel seguro.



2. Escolha se deseja criar um novo túnel ou abrir um túnel existente. Para criar um novo túnel, escolha Criar novo túnel. Você pode então escolher se deseja usar a configuração manual ou o método de configuração rápida para criar o túnel. Para obter mais informações, consulte Abra um túnel usando a configuração manual e conecte-se ao dispositivo remoto e Abra um túnel e use o SSH baseado em navegador para acessar o dispositivo remoto.

Criar um túnel para um dispositivo remoto usando a API

Para abrir um novo túnel, você pode usar a operação OpenTunnelda API. O código a seguir mostra um exemplo de execução deste comando.

```
aws iotsecuretunneling open-tunnel \
    --region us-east-1 \
    --endpoint https://api.us-east-1.tunneling.iot.amazonaws.com
    --cli-input-json file://input.json
```

O exemplo a seguir mostra o conteúdo do arquivo `input.json`. Você pode usar o `destinationConfig` parâmetro para especificar o nome do dispositivo de destino (por exemplo, *`RemoteDevice1`*) e o serviço que você deseja usar para acessar o dispositivo de destino, como *`SSH`*. Se preferir, você também poderá especificar parâmetros adicionais, como descrição do túnel e tags.

Conteúdo de input.json

```
{
   "description": "Tunnel to remote device1",
   "destinationConfig": {
      "services": [ "SSH" ],
      "thingName": "RemoteDevice1"
   }
```

```
}
```

A execução desse comando cria um novo túnel e fornece os tokens de acesso de origem e destino.

```
{
    "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
    "tunnelArn": "arn:aws:iot:us-
east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
    "sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
    "destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

Abra um túnel existente e use o SSH baseado em navegador

Digamos que você tenha criado o túnel para seu dispositivo remoto usando o método de configuração manual ou usando a AWS IoT API de referência da API. `RemoteDevice1` Em seguida, você pode abrir o túnel existente para o dispositivo e escolher Configuração rápida para usar o atributo SSH baseado em navegador. As configurações de um túnel existente não podem ser editadas, então você não pode usar o método de configuração manual.

Para usar o atributo SSH baseado em navegador, você não precisará baixar o token de acesso de origem nem configurar o proxy local. Um proxy local baseado na web será configurado automaticamente para que você possa começar a interagir com seu dispositivo remoto.

Para usar o método de configuração rápida e o SSH baseado em navegador

1. Vá para a página de detalhes do item que você criou, `RemoteDevice1`, e crie um túnel seguro.
2. Escolha Usar túnel existente para abrir o túnel aberto mais recente que você criou para o dispositivo remoto. As configurações de um túnel não podem ser editadas, então você não pode usar o método de configuração manual. Para usar o método de configuração rápida, selecione Configuração rápida.
3. Continue revisando e confirmando os detalhes da configuração do túnel e criando o túnel. As configurações do túnel não podem ser editadas.

   Quando você cria o túnel, o tunelamento seguro usa a operação da RotateTunnelAccessTokenAPI para revogar os tokens de acesso originais e gerar novos tokens de acesso. Se seu dispositivo remoto usar MQTT, esses tokens serão automaticamente entregues ao dispositivo remoto no tópico do MQTT no qual ele está inscrito. Você também pode optar por baixar esses tokens manualmente para o dispositivo de origem.

Depois de criar o túnel, você pode usar o SSH baseado em navegador para interagir com o dispositivo remoto diretamente do console usando a interface de linha de comando no contexto. Para usar essa interface de linha de comando, escolha o túnel para o item que você criou e, na página de detalhes, expanda a seção Interface de linha de comando. Como o proxy local já foi configurado para você, você pode começar a inserir comandos para começar rapidamente a acessar e interagir com seu dispositivo remoto, `RemoteDevice1`.

Para obter mais informações sobre o método de configuração rápida e o uso do SSH baseado em navegador, consulte Abra um túnel e use o SSH baseado em navegador para acessar o dispositivo remoto.

Liberar

- Fechar túnel

  Recomendamos que você feche o túnel depois de terminar de usá-lo. Um túnel também pode ficar fechado se permanecer aberto por mais tempo do que a duração especificada do túnel. Um túnel não pode ser reaberto depois de fechado. Você ainda pode duplicar um túnel abrindo o túnel fechado e selecionando Duplicar túnel. Especifique a duração do túnel que você deseja usar e, em seguida, crie o novo túnel.

  - Para fechar um túnel individual ou vários túneis a partir do AWS IoT console, acesse o Hub de túneis, escolha os túneis que você deseja fechar e escolha Fechar túnel.
  - Para fechar um túnel individual ou vários túneis usando a AWS IoT API de referência da API, use a operação da CloseTunnelAPI.

    ```
    aws iotsecuretunneling close-tunnel \
        --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    ```

- Excluir túnel

  Você pode excluir um túnel permanentemente do seu Conta da AWS.

  > **Warning**
  >
  > Uma ação de exclusão é permanente e não pode ser desfeita.

  - Para excluir um túnel individual ou vários túneis a partir do AWS IoT console, acesse o Hub de túneis, escolha os túneis que você deseja excluir e escolha Excluir túnel.

- Para excluir um túnel individual ou vários túneis usando a AWS IoT API de referência da API, use a operação da CloseTunnelAPI. Ao usar a API, defina o sinalizador `delete` como `true`.

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    --delete true
```

Proxy local

O proxy local transmite dados enviados pelo aplicativo em execução no dispositivo de origem usando o tunelamento seguro em uma conexão segura. WebSocket Você pode baixar a fonte do proxy local em GitHub.

O proxy local pode ser executado em dois modos: `source` ou `destination`. No modo de origem, o proxy local é executado no mesmo dispositivo ou na mesma rede do aplicativo cliente que inicia a conexão TCP. No modo de destino, o proxy local é executado no dispositivo remoto com o aplicativo de destino. Um único túnel pode suportar até três fluxos de dados por vez usando a multiplexação de túneis. Para cada fluxo de dados, o encapsulamento seguro usa várias conexões TCP, o que reduz a possibilidade de um tempo limite. Para obter mais informações, consulte Multiplexar fluxos de dados e usar conexões TCP simultâneas em um túnel seguro.

Como usar o proxy local

Você pode executar o proxy local nos dispositivos de origem e destino para transmitir dados para os endpoints de encapsulamento seguro. Se seus dispositivos estiverem em uma rede que usa um proxy da Web, o proxy da Web poderá interceptar as conexões antes de encaminhá-las para a Internet. Nesse caso, você precisará configurar seu proxy local para usar o proxy da web. Para obter mais informações, consulte Configurar o proxy local para dispositivos que usam proxy da web.

Fluxo de trabalho do proxy local

As etapas a seguir mostram como o proxy local é executado nos dispositivos de origem e destino.

1. Conecte o proxy local para proteger o encapsulamento

   O proxy local primeiro estabelece uma conexão com o serviço de encapsulamento seguro. Ao iniciar o proxy local, use os seguintes argumentos:

   - O `-r` argumento para especificar o local Região da AWS em que o túnel é aberto.

- O argumento `-t` para transmitir o token de acesso do cliente de origem ou de destino retornado do `OpenTunnel`.

> **Note**
>
> Dois proxies locais que usam o mesmo valor de token de acesso do cliente não podem ser conectados ao mesmo tempo.

2. Executar ações de origem ou de destino

   Depois que a WebSocket conexão é estabelecida, o proxy local executa ações no modo de origem ou no modo de destino, dependendo de sua configuração.

   Por padrão, o proxy local tenta se reconectar ao tunelamento seguro se ocorrer algum erro input/output (E/S) ou se a WebSocket conexão for fechada inesperadamente. Isso faz com que a conexão TCP seja fechada. Se ocorrerem erros de soquete TCP, o proxy local enviará uma mensagem pelo túnel a fim de notificar o outro lado para fechar sua conexão TCP. Por padrão, o proxy local sempre usa comunicação SSL.

3. Iniciar o proxy local

   Depois de usar o túnel, é seguro encerrar o processo de proxy local. Recomendamos que você feche explicitamente o túnel chamando `CloseTunnel`. Os clientes de túnel ativos podem não ser fechados logo após a chamada `CloseTunnel`.

Para obter mais informações sobre como usar o Console de gerenciamento da AWS para abrir um túnel e iniciar uma sessão SSH, consulteAbra um túnel e inicie a sessão SSH no dispositivo remoto.

Melhores práticas de proxy local

Ao executar o proxy local, siga estas melhores práticas:

- Evite o uso do `-t` argumento de proxy local para transmitir em um token de acesso. Recomendamos que você use a `AWSIOT_TUNNEL_ACCESS_TOKEN` variável de ambiente para definir o token de acesso para o proxy local.
- Execute o executável do proxy local com menos privilégios no sistema operacional ou ambiente.
  - Evite executar o proxy local como administrador no Windows.
  - Evite executar o proxy local como raiz no Linux e no macOS.

- Considere executar o proxy local em hosts, contêineres, sandboxes, prisão chroot separados ou em um ambiente virtualizado.
- Crie o proxy local com sinalizadores de segurança relevantes, dependendo da cadeia de ferramentas.
- Em dispositivos com várias interfaces de rede, use o argumento `-b` para vincular o soquete TCP à interface de rede usada para se comunicar com o aplicativo de destino.

Exemplo de comando e saída

Veja a seguir um exemplo de comando executado e a saída correspondente. O exemplo mostra como o proxy local pode ser configurado nos modos `source` e `destination`. O proxy local atualiza o protocolo HTTPS WebSockets para estabelecer uma conexão de longa duração e, em seguida, começa a transmitir dados por meio da conexão aos terminais do dispositivo de tunelamento seguro.

Antes de executar esses comandos:

Você deve ter aberto um túnel e obtido os tokens de acesso do cliente para a origem e o destino. Você também deve ter criado o proxy local conforme descrito anteriormente. Para criar o proxy local, abra o código-fonte do proxy local no GitHub repositório e siga as instruções para criar e instalar o proxy local.

> **Note**
>
> Os comandos a seguir usados nos exemplos usam o sinalizador `verbosity` para ilustrar uma visão geral das diferentes etapas descritas anteriormente após a execução do proxy local. Recomendamos que você use esse sinalizador apenas para fins de teste.

Rodar o proxy local no modo de origem

Os comandos a seguir exibem como executar o proxy local no modo de origem.

Linux/macOS

No Linux ou no macOS, execute os seguintes comandos no terminal para configurar e iniciar o proxy local na sua fonte.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
```

```
./localproxy -s 5555 -v 5 -r us-west-2
```

Onde:

- `-s` é a porta de escuta de origem, que inicia o proxy local no modo de origem.
- `-v` é a verbosidade da saída, que pode ser um valor entre zero e seis.
- `-r` é a região do endpoint em que o túnel é aberto.

Para obter mais informações sobre os parâmetros, consulte Opções definidas usando argumentos de linha de comando.

Windows

No Windows, você configura o proxy local da mesma forma que você faz para Linux ou macOS, mas a forma como você define as variáveis de ambiente é diferente das outras plataformas. Execute os comandos a seguir na janela `cmd` para configurar e iniciar o proxy local em sua origem.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
.\localproxy -s 5555 -v 5 -r us-west-2
```

Onde:

- `-s` é a porta de escuta de origem, que inicia o proxy local no modo de origem.
- `-v` é a verbosidade da saída, que pode ser um valor entre zero e seis.
- `-r` é a região do endpoint em que o túnel é aberto.

Para obter mais informações sobre os parâmetros, consulte Opções definidas usando argumentos de linha de comando.

> **Note**
>
> Ao usar a versão mais recente do proxy local no modo de origem, você deve incluir o AWS CLI parâmetro `--destination-client-type V1` no dispositivo de origem para compatibilidade com versões anteriores. Isso se aplica ao se conectar a qualquer um desses modos de destino:
>
> - AWS IoT Cliente do dispositivo

- AWS IoT Componente de tunelamento seguro ou componente de tunelamento AWS IoT Greengrass Version 2 seguro
- Qualquer código de demonstração do AWS IoT Secure Tunneling escrito antes de 2022
- Versões 1.X do proxy local

Esse parâmetro garante a comunicação adequada entre o proxy de origem atualizado e os clientes de destino mais antigos. Para obter mais informações sobre as versões de proxy local, consulte AWS IoT Secure Tunneling on. GitHub

Veja a seguir um exemplo de saída da execução do proxy local no modo `source`.

```
...
...

Starting proxy in source mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Resolved proxy  server IP: 10.10.0.11
Connected successfully with proxy server
Performing SSL handshake with proxy server
Successfully completed SSL handshake with proxy server
HTTP/1.1 101 Switching Protocols

...

Connection: upgrade
channel-id: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
upgrade: websocket

...

Web socket session ID: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Setting up web socket pings for every 5000 milliseconds
Scheduled next read:

...
```

```
Starting web socket read loop continue reading...
Resolved bind IP: 127.0.0.1
Listening for new connection on port 5555
```

Como executar o proxy local no modo de destino

Os comandos a seguir exibem como executar o proxy local no modo de destino.

Linux/macOS

No Linux ou no macOS, execute os seguintes comandos no terminal para configurar e iniciar o proxy local no seu destino.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
./localproxy -d 22 -v 5 -r us-west-2
```

Em que:

- `-d` é o aplicativo de destino que inicia o proxy local no modo de destino.
- `-v` é a verbosidade da saída, que pode ser um valor entre zero e seis.
- `-r` é a região do endpoint em que o túnel é aberto.

Para obter mais informações sobre os parâmetros, consulte Opções definidas usando argumentos de linha de comando.

Windows

No Windows, você configura o proxy local da mesma forma que você faz para Linux ou macOS, mas a forma como você define as variáveis de ambiente é diferente das outras plataformas. Execute os comandos a seguir na janela `cmd` para configurar e iniciar o proxy local em sua origem.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
.\localproxy -d 22 -v 5 -r us-west-2
```

Em que:

- `-d` é o aplicativo de destino que inicia o proxy local no modo de destino.
- `-v` é a verbosidade da saída, que pode ser um valor entre zero e seis.

- `-r` é a região do endpoint em que o túnel é aberto.

Para obter mais informações sobre os parâmetros, consulte Opções definidas usando argumentos de linha de comando.

> **Note**
>
> Ao usar a versão mais recente do proxy local no modo de destino, você deve incluir o AWS CLI parâmetro `--destination-client-type V1` no dispositivo de destino para compatibilidade com versões anteriores. Isso se aplica ao se conectar a um destes modos de origem:
>
> - Tunelamento seguro baseado em navegador a partir do console. AWS
> - Versões 1.X do proxy local
>
> Esse parâmetro garante a comunicação adequada entre o proxy de destino atualizado e os clientes de destino mais antigos. Para obter mais informações sobre as versões de proxy local, consulte AWS IoT Secure Tunneling on. GitHub

Veja a seguir um exemplo de saída da execução do proxy local no modo `destination`.

```
...
...

Starting proxy in destination mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Resolved proxy  server IP: 10.10.0.11
Connected successfully with proxy server
Performing SSL handshake with proxy server
Successfully completed SSL handshake with proxy server
HTTP/1.1 101 Switching Protocols

...

Connection: upgrade
channel-id: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
upgrade: websocket
```

```
...

Web socket session ID: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Setting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
```

Configurar o proxy local para dispositivos que usam proxy da web

Você pode usar o proxy local em AWS IoT dispositivos para se comunicar com o AWS IoT tunelamento APIs seguro. O proxy local transmite dados enviados pelo aplicativo do dispositivo usando tunelamento seguro por meio de uma conexão segura. WebSocket O proxy local pode funcionar no modo `source` ou `destination`. No modo `source`, ele é executado no mesmo dispositivo ou rede que inicia a conexão TCP. No modo `destination`, o proxy local é executado no dispositivo remoto com o aplicativo de destino. Para obter mais informações, consulte Proxy local.

O proxy local precisa se conectar diretamente à Internet para usar o AWS IoT tunelamento seguro. Para uma conexão TCP de longa duração com tunelamento seguro, o proxy local atualiza a solicitação HTTPS para estabelecer uma WebSockets conexão com um dos terminais de conexão do dispositivo de tunelamento seguro.

Se seus dispositivos estiverem em uma rede que usa um proxy da Web, o proxy da Web poderá interceptar as conexões antes de encaminhá-las para a Internet. Para estabelecer uma conexão de longa duração com os endpoints de conexão do dispositivo de encapsulamento seguro, configure seu proxy local para usar o proxy da Web conforme descrito na especificação do websocket.

> **Note**
>
> O AWS IoT Device Client não é compatível com dispositivos que usam um proxy da web. Para trabalhar com o proxy da web, você precisará usar um proxy local e configurá-lo para funcionar com um proxy da web, conforme descrito abaixo.

As etapas a seguir mostram como o proxy local funciona com um proxy da web.

1. O proxy local envia uma solicitação `CONNECT` HTTP ao proxy da web que contém o endereço remoto do serviço de encapsulamento seguro, junto com as informações de autenticação do proxy da web.
2. O proxy da web então criará uma conexão de longa duração com os endpoints remotos de encapsulamento seguro.
3. A conexão TCP foi estabelecida e o proxy local agora funcionará nos modos de origem e destino para transmissão de dados.

Para concluir este procedimento, execute as seguintes etapas.

- Crie o proxy local
- Configure seu proxy da web
- Configurar e iniciar o proxy local

Crie o proxy local

Abra o código-fonte do proxy local no GitHub repositório e siga as instruções para criar e instalar o proxy local.

Configure seu proxy da web

O proxy local depende do mecanismo de encapsulamento HTTP descrito pela especificação HTTP/1.1.. Para estar em conformidade com as especificações, seu proxy da web deve permitir que os dispositivos usem o método `CONNECT`.

A forma como você configura seu proxy da web depende do proxy da web que você está usando e da versão do proxy da web. Para ter certeza de que você configurou o proxy da web corretamente, verifique a documentação do proxy da web.

Para configurar seu proxy da web, primeiro identifique sua URL de proxy da web e confirme se seu proxy da web é compatível com o encapsulamento HTTP. O URL do proxy da web será usado posteriormente quando você configurar e iniciar o proxy local.

1. Identifique a URL de seu proxy da web

   Sua URL de proxy da web estará no seguinte formato.

```
protocol://web_proxy_host_domain:web_proxy_port
```

AWS IoT o tunelamento seguro oferece suporte somente à autenticação básica para proxy da web. Para usar a autenticação básica, você deve especificar **`username`** e **`password`** como parte da URL do proxy da web. A URL de proxy da web estará no seguinte formato.

```
protocol://username:password@web_proxy_host_domain:web_proxy_port
```

- *`protocol`*pode ser `http` ou`https`. Recomendamos usar o `https`.
- *`web_proxy_host_domain`*é o endereço IP do seu proxy da web ou um nome DNS que é resolvido para o endereço IP do seu proxy da web.
- *`web_proxy_port`*é a porta na qual o proxy da web está escutando.
- O proxy da web usa isso **`username`** e **`password`** para autenticar a solicitação.

2. Teste o URL do proxy web

   Para confirmar se seu proxy da web oferece suporte ao encapsulamento TCP, use um comando `curl` e certifique-se de obter uma resposta `2xx` ou uma `3xx`.

   Por exemplo, se o URL do proxy da web for `https://server.com:1235`, use um `proxy-insecure` sinalizador com o comando `curl` porque o proxy da web pode depender de um certificado autoassinado.

   ```
   export HTTPS_PROXY=https://server.com:1235
   curl -I https://aws.amazon.com --proxy-insecure
   ```

   Se o URL do seu proxy da web tiver uma porta `http` (por exemplo, `http://server.com:1234`), você não precisará usar o sinalizador `proxy-insecure`.

   ```
   export HTTPS_PROXY=http://server.com:1234
   curl -I https://aws.amazon.com
   ```

Configurar e iniciar o proxy local

Para configurar o proxy local para usar um proxy da web, você deve configurar a `HTTPS_PROXY` variável de ambiente com os nomes de domínio DNS ou os endereços IP e números de porta que seu proxy da web usa.

Depois de configurar o proxy local, você pode usar o proxy local conforme explicado neste documento README.

> **Note**
>
> A declaração da variável de ambiente diferencia maiúsculas e minúsculas. Recomendamos definir cada variável uma vez usando todas as letras maiúsculas ou minúsculas. Os exemplos a seguir mostram o nome da variável de ambiente com todas as letras maiúsculas. Se a mesma variável for especificada usando letras maiúsculas e minúsculas, a variável especificada usando letras minúsculas terá precedência.

Os comandos a seguir mostram como configurar o proxy local que está sendo executado no seu destino para usar o proxy da web e iniciar o proxy local.

- `AWSIOT_TUNNEL_ACCESS_TOKEN`: essa variável contém o token de acesso do cliente (CAT) para o destino.
- `HTTPS_PROXY`: essa variável contém o URL do proxy da web ou o endereço IP para configurar o proxy local.

Os comandos mostrados nos exemplos a seguir dependem do sistema operacional que você usa e se o proxy da web está escutando em uma porta HTTP ou HTTPS.

Proxy da Web escutando em uma porta HTTP

Se o proxy da Web estiver escutando em uma porta HTTP, você poderá fornecer a URL ou o endereço IP do proxy da Web para a variável `HTTPS_PROXY`.

Linux/macOS

No Linux ou macOS, execute os seguintes comandos no terminal para configurar e iniciar o proxy local em seu destino para usar um proxy da web escutando uma porta HTTP.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
```

```
export HTTPS_PROXY=http:proxy.example.com:1234
./localproxy -r us-east-1 -d 22
```

Se você precisar se autenticar com o proxy, deverá especificar um **username** e **password** como parte da variável HTTPS_PROXY.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http://username:password@proxy.example.com:1234
./localproxy -r us-east-1 -d 22
```

Windows

No Windows, você configura o proxy local da mesma forma que você faz para Linux ou macOS, mas a forma como você define as variáveis de ambiente é diferente das outras plataformas. Execute os comandos a seguir na janela cmd para configurar e iniciar o proxy local em seu destino para usar um proxy da web escutando uma porta HTTP.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://proxy.example.com:1234
.\localproxy -r us-east-1 -d 22
```

Se você precisar se autenticar com o proxy, deverá especificar um **username** e **password** como parte da variável HTTPS_PROXY.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://username:password@10.15.20.25:1234
.\localproxy -r us-east-1 -d 22
```

Proxy da web escutando em uma porta HTTPS

Execute os comandos a seguir se o proxy da web estiver escutando em uma porta HTTPS.

> **Note**
>
> Se você estiver usando um certificado autoassinado para o proxy da web ou se estiver executando o proxy local em um sistema operacional que não tenha suporte nativo ao OpenSSL e configurações padrão, você precisará configurar seus certificados de proxy da web conforme descrito na seção Configuração do certificado no repositório. GitHub

Os comandos a seguir serão semelhantes à forma como você configurou seu proxy da web para um proxy HTTP, com a exceção de que você também especificará o caminho para os arquivos de certificado que você instalou conforme descrito anteriormente.

Linux/macOS

No Linux ou macOS, execute os seguintes comandos no terminal para configurar e iniciar o proxy local em seu destino para usar um proxy da web escutando uma porta HTTPS.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:proxy.example.com:1234
./localproxy -r us-east-1 -d 22 -c /path/to/certs
```

Se você precisar se autenticar com o proxy, deverá especificar um **`username`** e **`password`** como parte da variável HTTPS_PROXY.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http://username:password@proxy.example.com:1234
./localproxy -r us-east-1 -d 22 -c /path/to/certs
```

Windows

No Windows, execute os seguintes comandos na janela cmd para configurar e iniciar o proxy local em execução no seu destino para usar um proxy da web escutando uma porta HTTP.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://proxy.example.com:1234
.\localproxy -r us-east-1 -d 22 -c \path\to\certs
```

Se você precisar se autenticar com o proxy, deverá especificar um **`username`** e **`password`** como parte da variável HTTPS_PROXY.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://username:password@10.15.20.25:1234
.\localproxy -r us-east-1 -d 22 -c \path\to\certs
```

Exemplo de comando e saída

Veja a seguir um exemplo de um comando executado em um sistema operacional Linux e a saída correspondente. O exemplo mostra um proxy da web que está escutando em uma porta

HTTP e como o proxy local pode ser configurado para usar o proxy da web nos modos `source` e `destination`. Antes de poder executar esses comandos, você já deve ter aberto um túnel e obtido os tokens de acesso do cliente para a origem e o destino. Você também deve ter criado o proxy local e configurado seu proxy da web conforme descrito anteriormente.

Veja a seguir uma visão geral das etapas após você iniciar o proxy local. O proxy local:

- Identifica o URL do proxy da web para que ele possa usar o URL para se conectar ao servidor proxy.
- Estabelece uma conexão TCP com o proxy da web.
- Envia uma solicitação `CONNECT` HTTP para o proxy da web e aguarda a `HTTP/1.1 200` resposta, o que indica que a conexão foi estabelecida.
- Atualiza o protocolo HTTPS WebSockets para estabelecer uma conexão de longa duração.
- Inicia a transmissão de dados por meio da conexão com os endpoints do dispositivo de encapsulamento seguro.

> **Note**
>
> Os comandos a seguir usados nos exemplos usam o sinalizador `verbosity` para ilustrar uma visão geral das diferentes etapas descritas anteriormente após a execução do proxy local. Recomendamos que você use esse sinalizador apenas para fins de teste.

Rodar o proxy local no modo de origem

Os comandos a seguir mostram como executar o proxy local no modo de origem.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:username:password@10.15.10.25:1234
./localproxy -s 5555 -v 5 -r us-west-2
```

Veja a seguir um exemplo de saída da execução do proxy local no modo `source`.

```
...

Parsed basic auth credentials for the URL
Found Web proxy information in the environment variables, will use it to connect via
 the proxy.
```

```
...

Starting proxy in source mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Resolved Web proxy IP: 10.10.0.11
Connected successfully with Web Proxy
Successfully sent HTTP CONNECT to the Web proxy
Full response from the Web proxy:
HTTP/1.1 200 Connection established
TCP tunnel established successfully
Connected successfully with proxy server
Successfully completed SSL handshake with proxy server
Web socket session ID: 0a109afffee745f5-00001341-000b8138-cc6c878d80e8adb0-f186064b
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Seting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
Resolved bind IP: 127.0.0.1
Listening for new connection on port 5555
```

Como executar o proxy local no modo de destino

Os comandos a seguir mostram como executar o proxy local no modo de destino.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:username:password@10.15.10.25:1234
./localproxy -d 22 -v 5 -r us-west-2
```

Veja a seguir um exemplo de saída da execução do proxy local no modo `destination`.

```
...

Parsed basic auth credentials for the URL
Found Web proxy information in the environment variables, will use it to connect via
 the proxy.
```

```
...

Starting proxy in destination mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Resolved Web proxy IP: 10.10.0.1
Connected successfully with Web Proxy
Successfully sent HTTP CONNECT to the Web proxy
Full response from the Web proxy:
HTTP/1.1 200 Connection established
TCP tunnel established successfully
Connected successfully with proxy server
Successfully completed SSL handshake with proxy server
Web socket session ID: 06717bfffed3fd05-00001355-000b8315-da3109a85da804dd-24c3d10d
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Seting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
```

Multiplexar fluxos de dados e usar conexões TCP simultâneas em um túnel seguro

Você pode usar vários fluxos de dados por túnel usando o atributo de multiplexação de encapsulamento seguro. Com a multiplexação, você pode solucionar problemas de dispositivos usando vários fluxos de dados. Você também pode reduzir sua carga operacional eliminando a necessidade de criar, implantar e iniciar vários proxies locais ou abrir vários túneis no mesmo dispositivo. Por exemplo, a multiplexação pode ser usada no caso de um navegador da Web que exija o envio de vários fluxos de dados HTTP e SSH.

Para cada fluxo de dados, o tunelamento AWS IoT seguro suporta conexões TCP simultâneas. O uso de conexões simultâneas reduz o potencial de tempo limite no caso de várias solicitações do cliente. Por exemplo, ele pode reduzir o tempo de carregamento ao acessar remotamente um servidor web local ao dispositivo de destino.

As seções a seguir explicam mais sobre multiplexação e uso de conexões TCP simultâneas e seus diferentes casos de uso.

Tópicos

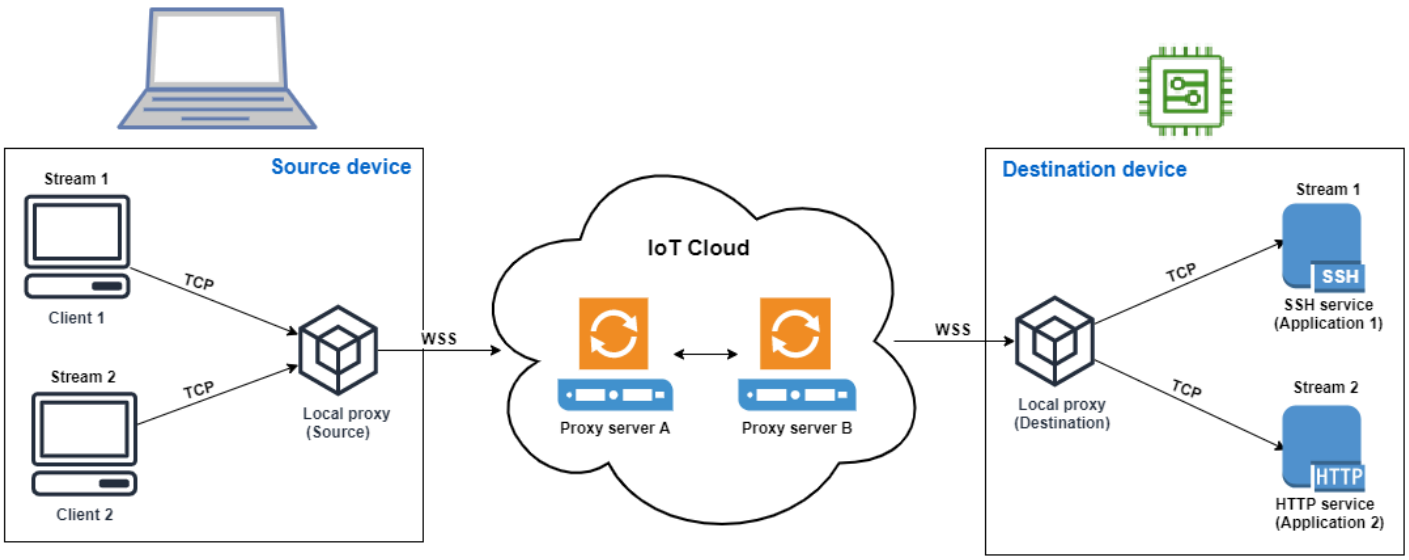
- Multiplexação de vários fluxos de dados em um túnel seguro
- Como usar conexões TCP simultâneas em um túnel seguro

Multiplexação de vários fluxos de dados em um túnel seguro

Você pode usar o atributo de multiplexação para dispositivos que usam várias conexões ou portas. A multiplexação também pode ser usada quando você precisa de várias conexões com um dispositivo remoto para solucionar qualquer problema. Por exemplo, ela pode ser usada no caso de um navegador da web que exija o envio de vários fluxos de dados HTTP e SSH. Os dados do aplicativo de ambos os fluxos são enviados ao dispositivo simultaneamente pelo túnel multiplexado.

Exemplo de caso de uso

Digamos que você precise se conectar a um aplicativo web no dispositivo para alterar alguns parâmetros de rede e, ao mesmo tempo, emitir comandos shell por meio do terminal para verificar se o dispositivo está funcionando corretamente com os novos parâmetros de rede. Nesse cenário, talvez seja necessário conectar-se ao dispositivo por meio de HTTP e SSH e transferir dois fluxos de dados paralelos para acessar simultaneamente o aplicativo web e o terminal. Com o atributo de multiplexação, esses dois fluxos independentes podem ser transferidos pelo mesmo túnel ao mesmo tempo.



Como configurar um túnel multiplexado

O procedimento a seguir explica como configurar um túnel multiplexado para solucionar problemas de dispositivos usando aplicativos que exigem conexões com várias portas. Você configurará um túnel com dois fluxos multiplexados: um fluxo HTTP e um fluxo SSH.

1. (Opcional) Criar arquivos de configuração

   Como alternativa, você pode configurar o dispositivo de origem e de destino com arquivos de configuração. Use arquivos de configuração se seus mapeamentos de portas provavelmente mudarem com frequência. Você pode pular essa etapa se preferir especificar explicitamente o mapeamento de portas usando a CLI ou se não precisar iniciar o proxy local nas portas de escuta designadas. Para obter mais informações sobre como usar arquivos de configuração, consulte Opções definidas via --config em. GitHub

   1. No dispositivo de origem, na pasta em que o proxy local será executado, crie uma pasta de configuração chamada `Config`. Dentro desta pasta, crie um arquivo chamado `SSHSource.ini` com o seguinte conteúdo:

      ```
      HTTP1 = 5555
      SSH1 = 3333
      ```

   2. No dispositivo de origem, na pasta em que o proxy local será executado, crie uma pasta de configuração chamada `Config`. Dentro desta pasta, crie um arquivo chamado `SSHDestination.ini` com o seguinte conteúdo:

      ```
      HTTP1 = 80
      SSH1 = 22
      ```

2. Abrir um túnel

   Abra um túnel usando a `OpenTunnel` operação da API ou o `open-tunnel` comando CLI. Configure o destino especificando `SSH1` e `HTTP1` como os serviços e o nome da AWS IoT coisa que corresponde ao seu dispositivo remoto. Seus aplicativos SSH e HTTP estão sendo executados nesse dispositivo remoto. Você já deve ter criado a coisa de IoT no AWS IoT registro. Para obter mais informações, consulte Gerenciar objetos com o Registro.

   ```
   aws iotsecuretunneling open-tunnel \
    --destination-config thingName=RemoteDevice1,services=HTTP1,SSH1
   ```

A execução desse comando gera os tokens de acesso de origem e destino que você usará para executar o proxy local.

```
{
 "tunnelId": "b2de92a3-b8ff-46c0-b0f2-afa28b00cecd",
 "tunnelArn": "arn:aws:iot:us-west-2:431600097591:tunnel/b2de92a3-b8ff-46c0-b0f2-
afa28b00cecd",
 "sourceAccessToken": source_client_access_token,
 "destinationAccessToken": destination_client_access_token
}
```

3. Configurar e iniciar o proxy local

Antes de executar o proxy local, configure o AWS IoT Device Client ou baixe o código-fonte do proxy local GitHube crie-o para a plataforma de sua escolha. Em seguida, você pode iniciar o proxy local de destino e de origem para se conectar ao túnel seguro. Para obter mais informações sobre como configurar e usar o proxy local, consulte Como usar o proxy local.

> **Note**
>
> No dispositivo de origem, se você não usar nenhum arquivo de configuração ou especificar o mapeamento de portas usando a CLI, ainda poderá usar o mesmo comando para executar o proxy local. O proxy local no modo de origem selecionará automaticamente as portas disponíveis para uso e os mapeamentos para você.

Start local proxy using configuration files

Execute os comandos a seguir para executar o proxy local nos modos de origem e destino usando arquivos de configuração.

```
// ----------------- Start the destination local proxy ----------------------
./localproxy -r us-east-1 -m dst -t destination_client_access_token

// ----------------- Start the source local proxy ---------------------------
// You also run the same command below if you want the local proxy to
// choose the mappings for you instead of using configuration files.
./localproxy -r us-east-1 -m src -t source_client_access_token
```

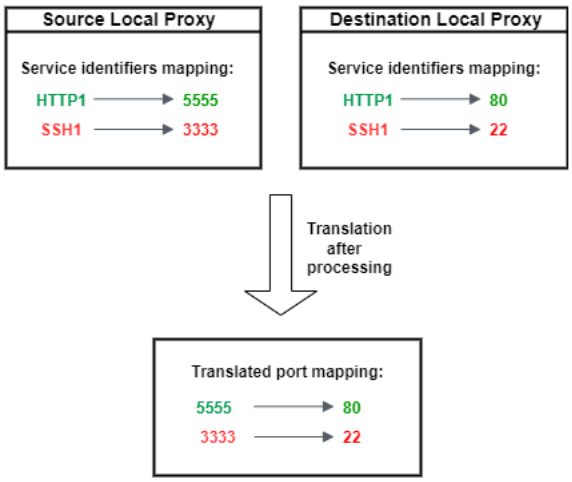
Start local proxy using CLI port mapping

Execute os comandos a seguir para executar o proxy local nos modos de origem e destino especificando explicitamente os mapeamentos de portas usando a CLI.

```
// ----------------- Start the destination local proxy
 ----------------------------------
./localproxy -r us-east-1 -d HTTP1=80,SSH1=22 -t destination_client_access_token

// ----------------- Start the source local proxy
 ---------------------------------------
./localproxy -r us-east-1 -s HTTP1=5555,SSH1=33 -t source_client_access_token
```

Os dados do aplicativo da conexão SSH e HTTP agora podem ser transferidos simultaneamente pelo túnel multiplexado. Conforme visto no mapa abaixo, o identificador de serviço atua como um formato legível para traduzir o mapeamento da porta entre o dispositivo de origem e o de destino. Com essa configuração, o tunelamento seguro encaminha qualquer tráfego HTTP de entrada da porta do *5555* dispositivo de origem para a porta do dispositivo de destino e qualquer tráfego SSH de entrada de porta *3333* para porta *80* no dispositivo de destino. *22*



Como usar conexões TCP simultâneas em um túnel seguro

AWS IoT o tunelamento seguro suporta mais de uma conexão TCP simultaneamente para cada fluxo de dados. Você pode usar esse recurso quando precisar de conexões simultâneas com um dispositivo remoto. O uso de conexões TCP simultâneas reduz o potencial de tempo limite no caso de múltiplas solicitações do cliente. Por exemplo, ao acessar um servidor web com vários

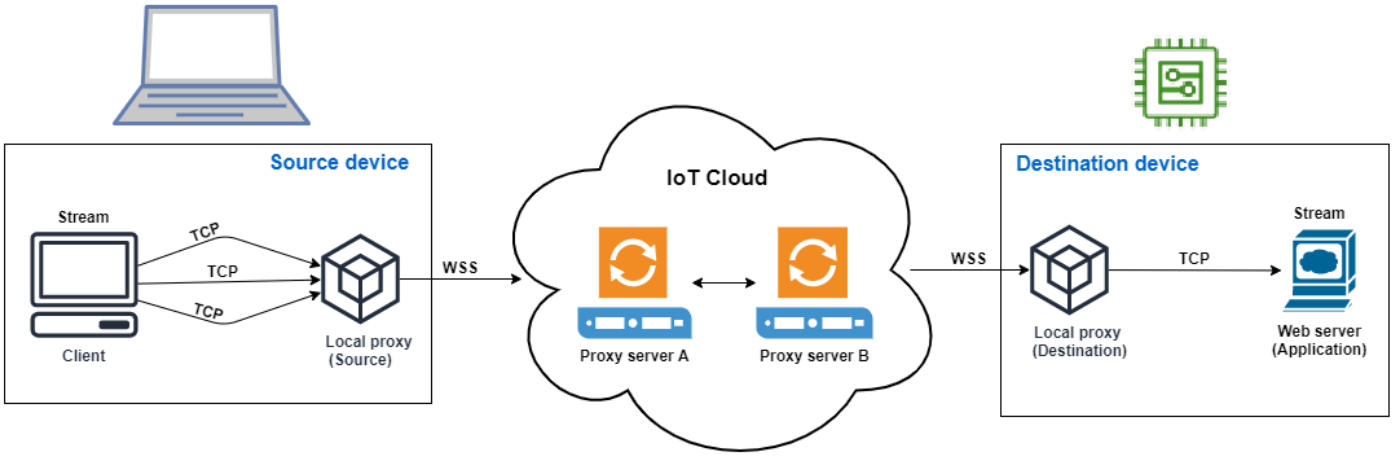
componentes em execução, conexões TCP simultâneas podem reduzir o tempo necessário para carregar o site.

> **Note**
>
> As conexões TCP simultâneas têm um limite de largura de banda de 800 kilobytes por segundo para cada uma. Conta da AWS AWS IoT o tunelamento seguro pode configurar esse limite para você, dependendo do número de solicitações recebidas.

Exemplo de caso de uso

Digamos que você precise acessar remotamente um servidor da web que esteja localizado no dispositivo de destino e tenha vários componentes em execução nele. Com uma única conexão TCP, ao tentar acessar o servidor web, o carregamento sequencial pode aumentar o tempo necessário para carregar os recursos no site. As conexões TCP simultâneas podem reduzir o tempo de carregamento atendendo aos requisitos de recursos do site, reduzindo assim o tempo de acesso. O diagrama a seguir mostra como as conexões TCP simultâneas são compatíveis com o fluxo de dados para o aplicativo do servidor da web em execução no dispositivo remoto.

> **Note**
>
> Se você quiser acessar vários aplicativos em execução no dispositivo remoto usando o túnel, você pode usar a multiplexação de túneis. Para obter mais informações, consulte Multiplexação de vários fluxos de dados em um túnel seguro.



Como usar conexões TCP simultâneas

O procedimento a seguir explica como usar conexões TCP simultâneas para acessar o navegador da Web no dispositivo remoto. Quando há várias solicitações do cliente, o tunelamento AWS IoT seguro configura automaticamente conexões TCP simultâneas para lidar com as solicitações, reduzindo assim o tempo de carregamento.

1. Abrir um túnel

   Abra um túnel usando a `OpenTunnel` operação da API ou o `open-tunnel` comando CLI. Configure o destino especificando `HTTP` como o serviço e o nome do AWS IoT item que corresponde ao seu dispositivo remoto. Seu aplicativo de servidor da web está sendo executado neste dispositivo remoto. Você já deve ter criado a coisa de IoT no AWS IoT registro. Para obter mais informações, consulte Gerenciar objetos com o Registro.

   ```
   aws iotsecuretunneling open-tunnel \
    --destination-config thingName=RemoteDevice1,services=HTTP
   ```

   A execução desse comando gera os tokens de acesso de origem e destino que você usará para executar o proxy local.

   ```
   {
    "tunnelId": "b2de92a3-b8ff-46c0-b0f2-afa28b00cecd",
    "tunnelArn": "arn:aws:iot:us-west-2:431600097591:tunnel/b2de92a3-b8ff-46c0-b0f2-
   afa28b00cecd",
    "sourceAccessToken": source_client_access_token,
    "destinationAccessToken": destination_client_access_token
   }
   ```

2. Configurar e iniciar o proxy local

   Antes de executar o proxy local, baixe o código-fonte do proxy local GitHube crie-o para a plataforma de sua escolha. Você pode então iniciar o proxy local de destino e de origem para se conectar ao túnel seguro e começar a usar o aplicativo de servidor web remoto.

> **Note**
>
> Para que o tunelamento AWS IoT seguro use conexões TCP simultâneas, você deve atualizar para a versão mais recente do proxy local. Esse atributo não estará disponível se você configurar o proxy local usando o AWS IoT Device Client.

```
// Start the destination local proxy
./localproxy -r us-east-1 -d HTTP=80 -t destination_client_access_token

// Start the source local proxy
./localproxy -r us-east-1 -s HTTP=5555 -t source_client_access_token
```

Para obter mais informações sobre como configurar e usar o proxy local, consulte Como usar o proxy local.

Agora você pode usar o túnel para acessar o aplicativo do servidor web. AWS IoT o tunelamento seguro configurará e manipulará automaticamente as conexões TCP simultâneas quando houver várias solicitações do cliente.

Como configurar um dispositivo remoto e usar o atendente de IoT

O atendente da IoT é usado para receber a mensagem MQTT que inclui o token de acesso do cliente e iniciar um proxy local no dispositivo remoto. Você deverá instalar e executar o atendente IoT no dispositivo remoto se desejar que o encapsulamento seguro entregue o token de acesso do cliente usando MQTT. O atendente da IoT deve assinar o seguinte tópico MQTT reservado da IoT:

> **Note**
>
> Se você quiser entregar o token de acesso do cliente de destino ao dispositivo remoto por meio de métodos diferentes da assinatura do tópico reservado do MQTT, talvez seja necessário um receptor do token de acesso do cliente (CAT) de destino e um proxy local. O receptor CAT deve funcionar com o mecanismo de entrega de token de acesso do cliente escolhido e ser capaz de iniciar um proxy local no modo de destino.

Snippet de atendente de IoT

O atendente de IoT deve se inscrever no seguinte tópico reservado do IoT MQTT para poder receber a mensagem do MQTT e iniciar o proxy local:

`$aws/things/thing-name/tunnels/notify`

Onde `thing-name` está o nome da AWS IoT coisa associada ao dispositivo remoto.

Veja seguir um exemplo de carga da mensagem MQTT:

```
{
    "clientAccessToken": "destination-client-access-token",
    "clientMode": "destination",
    "region": "aws-region",
    "services": ["destination-service"]
}
```

Depois de receber uma mensagem MQTT, o atendente da IoT deve iniciar um proxy local no dispositivo remoto com os parâmetros apropriados.

O código Java a seguir demonstra como usar o AWS IoT Device SDK e a biblioteca Java para criar um agente ProcessBuilderde IoT simples para trabalhar com tunelamento seguro.

```
// Find the IoT device endpoint for your Conta da AWS
final String endpoint = iotClient.describeEndpoint(new
 DescribeEndpointRequest().withEndpointType("iot:Data-ATS")).getEndpointAddress();

// Instantiate the IoT Agent with your AWS credentials
final String thingName = "RemoteDeviceA";
final String tunnelNotificationTopic = String.format("$aws/things/%s/tunnels/notify",
 thingName);
final AWSIotMqttClient mqttClient = new AWSIotMqttClient(endpoint, thingName,
                 "your_aws_access_key", "your_aws_secret_key");

try {
    mqttClient.connect();
    final TunnelNotificationListener listener = new
 TunnelNotificationListener(tunnelNotificationTopic);
    mqttClient.subscribe(listener, true);
}
finally {
    mqttClient.disconnect();
```

```
}

private static class TunnelNotificationListener extends AWSIotTopic {
    public TunnelNotificationListener(String topic) {
        super(topic);
    }

    @Override
    public void onMessage(AWSIotMessage message) {
        try {
            // Deserialize the MQTT message
            final JSONObject json = new JSONObject(message.getStringPayload());

            final String accessToken = json.getString("clientAccessToken");
            final String region = json.getString("region");

            final String clientMode = json.getString("clientMode");
            if (!clientMode.equals("destination")) {
                throw new RuntimeException("Client mode " + clientMode + " in the MQTT
 message is not expected");
            }

            final JSONArray servicesArray = json.getJSONArray("services");
            if (servicesArray.length() > 1) {
                throw new RuntimeException("Services in the MQTT message has more than
 1 service");
            }
            final String service = servicesArray.get(0).toString();
            if (!service.equals("SSH")) {
                throw new RuntimeException("Service " + service + " is not supported");
            }

            // Start the destination local proxy in a separate process to connect to
 the SSH Daemon listening port 22
            final ProcessBuilder pb = new ProcessBuilder("localproxy",
                        "-t", accessToken,
                        "-r", region,
                        "-d", "localhost:22");
            pb.start();
        }
        catch (Exception e) {
            log.error("Failed to start the local proxy", e);
        }
    }
```

```
}
```

Como controlar o acesso aos túneis

O encapsulamento seguro fornece ações, recursos e chaves de contexto de condição específicos do serviço para uso em políticas de permissões do IAM.

Pré-requisitos de acesso ao túnel

- Saiba como proteger AWS recursos usando políticas do IAM.
- Saiba como criar e avaliar as condições do IAM.
- Saiba como proteger AWS recursos usando tags de recursos.

Políticas de acesso ao túnel

Você deve usar as políticas a seguir para autorizar as permissões de uso da API de encapsulamento seguro. Para obter mais informações sobre AWS IoT segurança, consulteGerenciamento de identidade e acesso para AWS IoT.

IoT: OpenTunnel

A `iot:OpenTunnel` ação de política concede permissão a uma entidade principal para chamar OpenTunnel.

No `Resource` elemento da declaração de política do IAM:

- Especifique o ARN do túnel curinga:

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:tunnel/*`
- Especifique um ARN de item para gerenciar a `OpenTunnel` permissão para itens específicos de IoT:

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:thing/`*`thing-name`*

Por exemplo, a instrução da política a seguir permite abrir um túnel para o item da IoT chamado `TestDevice`.

```
{
```

```
    "Effect": "Allow",
    "Action": "iot:OpenTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*",
        "arn:aws:iot:aws-region:aws-account-id:thing/TestDevice"
    ]
}
```

A `iot:OpenTunnel` ação da política oferece suporte às seguintes chaves de condição:

- `iot:ThingGroupArn`
- `iot:TunnelDestinationService`
- `aws:RequestTag/tag-key`
- `aws:SecureTransport`
- `aws:TagKeys`

A instrução de política a seguir permite abrir um túnel para o item caso ele pertença a um grupo de itens com um nome que comece com `TestGroup` e o serviço de destino configurado no túnel seja SSH.

```
{
    "Effect": "Allow",
    "Action": "iot:OpenTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "ForAnyValue:StringLike": {
            "iot:ThingGroupArn": [
                "arn:aws:iot:aws-region:aws-account-id:thinggroup/TestGroup*"
            ]
        },
        "ForAllValues:StringEquals": {
            "iot:TunnelDestinationService": [
                "SSH"
            ]
        }
    }
}
```

Também é possível usar tags de recurso para controlar a permissão para abrir túneis. Por exemplo, a instrução de política a seguir permite que um túnel seja aberto caso a chave de tag `Owner` esteja presente com um valor de `Admin` e nenhuma outra tag for especificada. Para obter informações gerais sobre o uso de tags, consulte Marcando seus recursos AWS IoT.

```
{
    "Effect": "Allow",
    "Action": "iot:OpenTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/Owner": "Admin"
        },
        "ForAllValues:StringEquals": {
            "aws:TagKeys": "Owner"
        }
    }
}
```

IoT: RotateTunnelAccessToken

A `iot:RotateTunnelAccessToken` ação de política concede permissão a uma entidade principal para chamar RotateTunnelAccessToken.

No `Resource` elemento da declaração de política do IAM:

- Especifique um ARN de túnel totalmente qualificado:

  `arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

  Você também pode usar o ARN do túnel curinga:

  `arn:aws:iot:aws-region:aws-account-id:tunnel/*`

- Especifique um ARN de item para gerenciar a `RotateTunnelAccessToken` permissão para itens específicos de IoT:

  `arn:aws:iot:aws-region:aws-account-id:thing/thing-name`

Por exemplo, a instrução de política a seguir permite que você alterne o token de acesso de origem de um túnel ou o token de acesso de destino de um cliente para o item de IoT chamado `TestDevice`.

```
{
    "Effect": "Allow",
    "Action": "iot:RotateTunnelAccessToken",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*",
        "arn:aws:iot:aws-region:aws-account-id:thing/TestDevice"
    ]
}
```

A `iot:RotateTunnelAccessToken` ação da política oferece suporte às seguintes chaves de condição:

- `iot:ThingGroupArn`
- `iot:TunnelDestinationService`
- `iot:ClientMode`
- `aws:SecureTransport`

A instrução de política a seguir permite alternar o token de acesso de destino para o item se o item pertencer a um grupo de objetos com um nome que começa com `TestGroup`, o serviço de destino configurado no túnel for SSH e o cliente estiver no modo `DESTINATION`.

```
{
    "Effect": "Allow",
    "Action": "iot:RotateTunnelAccessToken",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "ForAnyValue:StringLike": {
            "iot:ThingGroupArn": [
                "arn:aws:iot:aws-region:aws-account-id:thinggroup/TestGroup*"
            ]
        },
        "ForAllValues:StringEquals": {
            "iot:TunnelDestinationService": [
                "SSH"
```

```
            ],
            "iot:ClientMode": "DESTINATION"
        }
    }
}
```

IoT: DescribeTunnel

A `iot:DescribeTunnel` ação de política concede permissão a uma entidade principal para chamar DescribeTunnel.

No `Resource` elemento da declaração de política do IAM, especifique um ARN de túnel totalmente qualificado:

`arn:aws:iot:`*`aws-region`*`: `*`aws-account-id`*`:tunnel/`*`tunnel-id`*

Você também pode usar o ARN curinga:

`arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:tunnel/*`

A `iot:DescribeTunnel` ação da política oferece suporte às seguintes chaves de condição:

- `aws:ResourceTag/`*`tag-key`*
- `aws:SecureTransport`

A instrução de política a seguir permite que você chame `DescribeTunnel` caso o túnel solicitado esteja marcado com a chave `Owner` com um valor de `Admin`.

```
{
    "Effect": "Allow",
    "Action": "iot:DescribeTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/Owner": "Admin"
        }
    }
}
```

IoT: ListTunnels

A `iot:ListTunnels` ação de política concede permissão a uma entidade principal para chamar ListTunnels.

No `Resource` elemento da declaração de política do IAM:

- Especifique o ARN do túnel curinga:

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:tunnel/*`

- Especifique um ARN de item para gerenciar a `ListTunnels` permissão para itens de IoT selecionados:

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:thing/`*`thing-name`*

A `iot:ListTunnels` ação de política oferece suporte à seguinte chave de condição `aws:SecureTransport`.

A instrução de política a seguir permite listar túneis para o item chamado `TestDevice`.

```
{
    "Effect": "Allow",
    "Action": "iot:ListTunnels",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*",
        "arn:aws:iot:aws-region:aws-account-id:thing/TestDevice"
    ]
}
```

IoT: ListTagsForResource

A `iot:ListTagsForResource` ação de política concede permissão a uma entidade principal para chamar `ListTagsForResource`.

No `Resource` elemento da declaração de política do IAM, especifique um ARN de túnel totalmente qualificado:

`arn:aws:iot:`*`aws-region`*`: `*`aws-account-id`*`:tunnel/`*`tunnel-id`*

Você também pode usar o ARN do túnel curinga:

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

A `iot:ListTagsForResource` ação de política oferece suporte à seguinte chave de condição `aws:SecureTransport`.

IoT: CloseTunnel

A `iot:CloseTunnel` ação de política concede permissão a uma entidade principal para chamar CloseTunnel.

No `Resource` elemento da declaração de política do IAM, especifique um ARN de túnel totalmente qualificado:

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

Você também pode usar o ARN do túnel curinga:

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

A `iot:CloseTunnel` ação da política oferece suporte às seguintes chaves de condição:

- `iot:Delete`
- `aws:ResourceTag/tag-key`
- `aws:SecureTransport`

A instrução de política a seguir permite chamar `CloseTunnel` se o parâmetro `Delete` da solicitação for `false` e a solicitação estiver marcada com a chave `Owner` com um valor de `QATeam`.

```
{
    "Effect": "Allow",
    "Action": "iot:CloseTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "Bool": {
            "iot:Delete": "false"
        },
        "StringEquals": {
            "aws:ResourceTag/Owner": "QATeam"
        }
    }
```

```
}
```

IoT: TagResource

A `iot:TagResource` ação de política concede permissão a uma entidade principal para chamar `TagResource`.

No `Resource` elemento da declaração de política do IAM, especifique um ARN de túnel totalmente qualificado:

`arn:aws:iot:`*`aws-region`*`: `*`aws-account-id`*`:tunnel/`*`tunnel-id`*

Você também pode usar o ARN do túnel curinga:

`arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:tunnel/*`

A `iot:TagResource` ação de política oferece suporte à seguinte chave de condição `aws:SecureTransport`.

IoT: UntagResource

A `iot:UntagResource` ação de política concede permissão a uma entidade principal para chamar `UntagResource`.

No `Resource` elemento da declaração de política do IAM, especifique um ARN de túnel totalmente qualificado:

`arn:aws:iot:`*`aws-region`*`: `*`aws-account-id`*`:tunnel/`*`tunnel-id`*

Você também pode usar o ARN do túnel curinga:

`arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:tunnel/*`

A `iot:UntagResource` ação de política oferece suporte à seguinte chave de condição `aws:SecureTransport`.

Resolvendo problemas de conectividade de tunelamento AWS IoT seguro por meio da rotação dos tokens de acesso do cliente

Ao usar o tunelamento AWS IoT seguro, você pode ter problemas de conectividade mesmo se o túnel estiver aberto. As seções a seguir mostram alguns possíveis problemas e como você pode

resolvê-los alternando os tokens de acesso do cliente. Para alternar o token de acesso do cliente (CAT), use a RotateTunnelAccessTokenAPI ou o. rotate-tunnel-access-token AWS CLI Dependendo se você encontrar um erro ao usar o cliente no modo de origem ou destino, você pode alternar o CAT no modo de origem ou destino, ou em ambos.

> **Note**
>
> - Se você não tiver certeza se o CAT precisa ser alternado na origem ou no destino, você pode alternar o CAT na origem e no destino configurando como `ClientMode` ALL ao usar a `RotateTunnelAccessToken` API.
> - Alternar o CAT não prolonga a duração do túnel. Por exemplo, digamos que a duração do túnel seja de 12 horas e o túnel já esteja aberto há 4 horas. Quando você alterna os tokens de acesso, os novos tokens gerados só podem ser usados pelas 8 horas restantes.

Tópicos

- Erro de token de acesso do cliente inválido
- Erro de incompatibilidade do token do cliente
- Problemas de conectividade com o dispositivo remoto

Erro de token de acesso do cliente inválido

Ao usar o tunelamento AWS IoT seguro, você pode encontrar um erro de conexão ao usar o mesmo token de acesso do cliente (CAT) para se reconectar ao mesmo túnel. Nesse caso, o proxy local não consegue se conectar ao servidor proxy de encapsulamento seguro. Se você usar um cliente no modo de origem, você poderá ver a seguinte mensagem de erro:

```
Invalid access token: The access token was previously used and cannot be used again
```

O erro ocorre porque o token de acesso do cliente (CAT) só pode ser usado uma vez pelo proxy local e, em seguida, torna-se inválido. Para resolver esse erro, alterne o token de acesso do cliente no modo `SOURCE` para gerar um novo CAT para a origem. Para obter um exemplo que mostra como alternar o CAT de origem, consulte Exemplo de CAT da fonte alternada.

Erro de incompatibilidade do token do cliente

> **Note**
>
> Não é recomendável usar tokens de cliente para reutilizar o CAT. Em vez disso, recomendamos que você use a `RotateTunnelAccessToken` API para alternar os tokens de acesso do cliente para se reconectar ao túnel.

Se você estiver usando tokens de cliente, poderá reutilizar o CAT para se reconectar ao túnel. Para reutilizar o CAT, você deve fornecer o token do cliente com o CAT na primeira vez em que se conectar ao encapsulamento seguro. O encapsulamento seguro armazena o token do cliente, portanto, para tentativas de conexão subsequentes usando o mesmo token, o token do cliente também deve ser fornecido. Para obter mais informações sobre o uso de tokens de cliente, consulte a implementação da referência de proxy local em GitHub.

Se você usar um cliente no modo de origem, poderá ver a seguinte mensagem de erro:

```
Invalid client token: The provided client token does not match the client token
    that was previously set.
```

O erro ocorre porque o token do cliente fornecido não corresponde ao token do cliente fornecido com o CAT ao acessar o túnel. Para resolver esse erro, alterne o CAT no modo `SOURCE` para gerar um novo CAT para a origem. Por exemplo:

Exemplo de CAT da fonte alternada

Veja a seguir um exemplo de como executar a `RotateTunnelAccessToken` API no modo `SOURCE` para gerar um novo CAT para a fonte:

```
aws iotsecuretunneling rotate-tunnel-access-token \
    --region <region> \
    --tunnel-id <tunnel-id> \
    --client-mode SOURCE
```

A execução desse comando gera um novo token de acesso à fonte e retorna o ARN do seu túnel.

```
{
    "sourceAccessToken": "<source-access-token>",
```

```
    "tunnelArn": "arn:aws:iot:<region>:<account-id>:tunnel/<tunnel-id>"
}
```

Agora você pode usar o novo token de origem para conectar o proxy local no modo de origem.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=<source-access-token>
./localproxy -r <region> -s <port>
```

Veja a seguir um exemplo de saída da execução do proxy local:

```
...

[info]    Starting proxy in source mode
...
[info]    Successfully established websocket connection with proxy server ...
[info]    Listening for new connection on port <port>
...
```

Problemas de conectividade com o dispositivo remoto

Ao usar o tunelamento AWS IoT seguro, o dispositivo pode ser desconectado inesperadamente, mesmo se o túnel estiver aberto. Para identificar se um dispositivo ainda está conectado ao túnel, você pode usar a DescribeTunnelAPI ou o AWS CLI describe-tunnel.

Um dispositivo pode ser desconectado por vários motivos. Para resolver o problema de conectividade, você pode alternar o CAT no destino se o dispositivo estiver desconectado devido aos seguintes motivos possíveis:

- O CAT no destino tornou-se inválido.
- O token não foi entregue ao dispositivo pelo tópico reservado do MQTT para encapsulamento seguro:

  `$aws/things/<thing-name>/tunnels/notify`

O exemplo a seguir mostra como resolver o problema:

Exemplo de CAT alternar destino

Considere um dispositivo remoto, *<RemoteThing1>*. Para abrir um túnel para esse item, você pode usar o seguinte comando:

```
aws iotsecuretunneling open-tunnel \
    --region <region> \
    --destination-config thingName=<RemoteThing1>,services=SSH
```

A execução desse comando gera os detalhes do túnel e o CAT para sua origem e destino.

```
{
    "sourceAccessToken": "<source-access-token>",
    "destinationAccessToken": "<destination-access-token>",
    "tunnelId": "<tunnel-id>",
    "tunnelArn": "arn:aws:iot:<region>:<account-id>:tunnel/tunnel-id"
}
```

No entanto, quando você usa a DescribeTunnelAPI, a saída indica que o dispositivo foi desconectado, conforme ilustrado abaixo:

```
aws iotsecuretunneling describe-tunnel \
    --tunnel-id <tunnel-id> \
    --region <region>
```

A execução desse comando mostra que o dispositivo ainda não está conectado.

```
{
    "tunnel": {
        ...
        "destinationConnectionState": {
            "status": "DISCONNECTED"
        },
        ...
    }
}
```

Para resolver esse erro, execute a `RotateTunnelAccessToken` API com o cliente no modo `DESTINATION` e as configurações do destino. A execução desse comando revoga o token de acesso antigo, gera um novo token e o reenvia para o tópico do MQTT:

`$aws/things/<thing-name>/tunnels/notify`

```
aws iotsecuretunneling rotate-tunnel-access-token \
```

```
    --tunnel-id <tunnel-id> \
    --client-mode DESTINATION \
    --destination-config thingName=<RemoteThing1>,services=SSH \
    --region <region>
```

A execução desse comando gera o novo token de acesso, como mostrado abaixo. O token é então entregue ao dispositivo para se conectar ao túnel, se o atendente do dispositivo estiver configurado corretamente.

```
{
    "destinationAccessToken": "destination-access-token",
    "tunnelArn": "arn:aws:iot:region:account-id:tunnel/tunnel-id"
}
```

Provisionamento de dispositivos

AWS fornece várias maneiras diferentes de provisionar um dispositivo e instalar certificados de cliente exclusivos nele. Esta seção descreve todas as maneiras e como selecionar a melhor delas para sua solução de IoT. Essas opções são descritas em detalhes no whitepaper intitulado Fabricação e provisionamento de dispositivos com certificados X.509 no AWS IoT Core.

Selecione a opção que melhor se adapta à sua situação

- Você pode instalar certificados em dispositivos de IoT antes de eles serem entregues

  Se você puder instalar com segurança certificados de cliente exclusivos em seus dispositivos de IoT antes de serem entregues para uso pelo usuário final, convém usar o just-in-timeprovisionamento (JITP) ou o registro (JITR). just-in-time

  Usando o JITP e o JITR, a autoridade de certificação (CA) usada para assinar o certificado do dispositivo é registrada AWS IoT e reconhecida AWS IoT quando o dispositivo se conecta pela primeira vez. O dispositivo é provisionado AWS IoT em sua primeira conexão usando os detalhes de seu modelo de provisionamento.

  Para obter mais informações sobre o provisionamento de item único, JITP, JITR e em massa de dispositivos com certificados exclusivos, consulte the section called “Provisionamento de dispositivos com certificados de dispositivo”.
- Usuários finais ou instaladores podem usar um aplicativo para instalar certificados em seus dispositivos de IoT

  Se não estiver a seu alcance instalar com segurança certificados de cliente exclusivos em seu dispositivo de IoT antes de ele ser entregue ao usuário final, mas o usuário final ou um instalador puderem usar um aplicativo para registrar os dispositivos e instalar os certificados exclusivos, use o processo de provisionamento por usuário confiável.

  Usar um usuário confiável, como um usuário final ou um instalador com uma conta conhecida, pode simplificar o processo de fabricação do dispositivo. Em vez de um certificado de cliente exclusivo, os dispositivos têm um certificado temporário que permite que o dispositivo se conecte AWS IoT por apenas 5 minutos. Durante essa janela de 5 minutos, o usuário confiável obtém um certificado de cliente exclusivo com uma vida útil mais longa, que é instalado no dispositivo. A vida útil limitada do certificado de reivindicação minimiza o risco de um certificado comprometido.

Para obter mais informações, consulte the section called “Provisionamento por usuário confiável”.

- Usuários finais NÃO PODEM usar um aplicativo para instalar certificados em seus dispositivos de IoT

  Se nenhuma das opções anteriores funcionar em sua solução de IoT, o processo de provisionamento por reivindicação é uma opção. Com esse processo, seus dispositivos de IoT têm um certificado de reivindicação que é compartilhado por outros dispositivos da frota. Na primeira vez que um dispositivo se conecta a um certificado de solicitação, AWS IoT registra o dispositivo usando seu modelo de provisionamento e emite ao dispositivo seu certificado de cliente exclusivo para acesso posterior. AWS IoT

  Essa opção permite o provisionamento automático de um dispositivo quando ele se conecta AWS IoT, mas pode apresentar um risco maior no caso de um certificado de solicitação comprometido. Se um certificado de reivindicação for comprometido, você poderá desativá-lo. A desativação do certificado de reivindicação impede que todos os dispositivos com esse certificado sejam registrados no futuro. No entanto, a desativação do certificado de reivindicação não bloqueará dispositivos que já foram provisionados.

  Para obter mais informações, consulte the section called “Provisionamento por reivindicação”.

Dispositivos de provisionamento em AWS IoT

Ao provisionar um dispositivo com AWS IoT, você deve criar recursos para que seus dispositivos AWS IoT possam se comunicar com segurança. Outros recursos podem ser criados para ajudar a gerenciar a frota de dispositivos. Os seguintes recursos podem ser criados durante o processo de provisionamento:

- Uma coisa da IoT.

  Coisas de IoT são entradas no registro do AWS IoT dispositivo. Cada coisa tem um nome exclusivo e um conjunto de atributos, e está associada a um dispositivo físico. As coisas podem ser definidas usando um tipo de coisa ou agrupadas em grupos de coisas. Para obter mais informações, consulte Gerenciando dispositivos com AWS IoT.

  Embora não seja necessário, criar objetos possibilita gerenciar a frota de dispositivos de forma mais eficaz, podendo pesquisar dispositivos por tipo, grupo e atributos de objetos. Para obter mais informações, consulte Indexação de frotas.

> **Note**
>
> Para indexar os dados de status de conectividade do objeto, provisione o objeto e configure-o para que o nome dela corresponda ao do ID do cliente usado na solicitação de conexão.

- Um certificado X.509.

  Os dispositivos usam certificados X.509 para realizar autenticação mútua com. AWS IoT Você pode registrar um certificado existente ou AWS IoT gerar e registrar um novo certificado para você. Associe um certificado a um dispositivo anexando-o à coisa que representa o dispositivo. Também é necessário copiar o certificado e a chave privada associada no dispositivo. Os dispositivos apresentam o certificado ao se conectar AWS IoT a. Para obter mais informações, consulte Autenticação.

- Uma política da IoT.

  As políticas da IoT definem as operações que um dispositivo pode executar na AWS IoT. As políticas da IoT são anexadas aos certificados do dispositivo. Quando um dispositivo apresenta o certificado para AWS IoT, ele recebe as permissões especificadas na política. Para obter mais informações, consulte Autorização. Cada dispositivo precisa de um certificado para se comunicar com a AWS IoT.

AWS IoT oferece suporte ao provisionamento automatizado de frotas usando modelos de provisionamento. Os modelos de provisionamento descrevem os recursos AWS IoT necessários para provisionar seu dispositivo. Os modelos contêm variáveis que permitem usar um modelo para provisionar vários dispositivos. Ao provisionar um dispositivo, especifique valores para as variáveis específicas do dispositivo usando um dicionário ou mapa. Para provisionar outro dispositivo, especifique novos valores no dicionário.

É possível usar o provisionamento automatizado independentemente de os dispositivos terem certificados exclusivos (e a chave privada associada) ou não.

Provisionamento de frota APIs

Há várias categorias de APIs uso no provisionamento de frotas:

- Essas funções do plano de controle criam e gerenciam os modelos de provisionamento de frota e configuram políticas de usuário confiáveis.
  - CreateProvisioningTemplate
  - CreateProvisioningTemplateVersion
  - DeleteProvisioningTemplate
  - DeleteProvisioningTemplateVersion
  - DescribeProvisioningTemplate
  - DescribeProvisioningTemplateVersion
  - ListProvisioningTemplates
  - ListProvisioningTemplateVersions
  - UpdateProvisioningTemplate
- Usuários confiáveis podem usar essa função de plano de controle para gerar uma reivindicação de integração temporária. Essa reivindicação temporária é passada para o dispositivo durante a configuração da rede Wi-Fi ou método semelhante.
  - CreateProvisioningClaim
- A API MQTT usada durante o processo de provisionamento por dispositivos com um certificado de reivindicação de provisionamento incorporado em um dispositivo ou passado para ele por um usuário confiável.
  - the section called “CreateCertificateFromCsr”
  - the section called “CreateKeysAndCertificate”
  - the section called “RegisterThing”

Provisionar dispositivos que não têm certificados de dispositivo usando o provisionamento de frotas

Ao usar o provisionamento de AWS IoT frota, AWS IoT pode gerar e entregar com segurança certificados de dispositivos e chaves privadas aos seus dispositivos quando eles se conectam AWS IoT pela primeira vez. AWS IoT fornece certificados de cliente assinados pela autoridade de certificação (CA) raiz da Amazon.

Há duas maneiras de usar o provisionamento de frotas:

- Provisionamento por reivindicação

- Provisionamento por usuário confiável

Provisionamento por reivindicação

Os dispositivos podem ser fabricados com um certificado de alegação de provisionamento e uma chave privada (que são credenciais de finalidade especial) incorporados neles. Se esses certificados estiverem registrados AWS IoT, o serviço poderá trocá-los por certificados de dispositivo exclusivos que o dispositivo possa usar para operações regulares. Esse processo inclui as seguintes etapas:

Antes de entregar o dispositivo

1. Chame `CreateProvisioningTemplate` para criar um modelo de provisionamento. Essa API retorna um ARN de modelo. Para obter mais informações, consulte API MQTT de provisionamento de dispositivos.

   Você também pode criar um modelo de aprovisionamento de frota no AWS IoT console.

   a. No painel de navegação, escolha o menu suspenso Conectar muitos dispositivos. Depois, selecione Conectar muitos dispositivos.

   b. Escolha Criar modelo de provisionamento.

   c. Escolha o cenário de provisionamento que melhor se adapta aos seus processos de instalação. Em seguida, escolha Próximo.

   d. Preencha o fluxo de trabalho do modelo.

2. Crie certificados e chaves privadas associadas a serem usados como certificados de reivindicação de provisionamento.

3. Registre esses certificados AWS IoT e associe uma política de IoT que restrinja o uso dos certificados. O exemplo de política da IoT a seguir restringe o uso do certificado associado a essa política para dispositivos de provisionamento.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Receive"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/$aws/certificates/create/
*",
    "arn:aws:iot:us-east-1:123456789012:topic/$aws/provisioning-
templates/templateName/provision/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/certificates/
create/*",
    "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/provisioning-
templates/templateName/provision/*"
            ]
        }
    ]
}
```

4. Dê permissão ao AWS IoT serviço para criar ou atualizar recursos de IoT, como itens e certificados em sua conta, ao provisionar dispositivos. Faça isso anexando a política `AWSIoTThingsRegistration` gerenciada a uma função do IAM (chamada de função de provisionamento) que confia no responsável pelo serviço. AWS IoT
5. Produza o dispositivo com o certificado de reivindicação de provisionamento incorporado de forma segura nele.

O dispositivo agora está pronto para ser entregue no local onde será instalado para uso.

> **Important**
>
> As chaves privadas de alegação de provisionamento devem ser protegidas o tempo todo, inclusive no dispositivo. Recomendamos que você use AWS IoT CloudWatch métricas e

registros para monitorar as indicações de uso indevido. Se você detectar uso indevido, desative o certificado de reivindicação de provisionamento para que ele não possa ser usado para o provisionamento de dispositivos.

Como inicializar o dispositivo para uso

1. O dispositivo usa o AWS IoT Device SDKs, Mobile SDKs e AWS IoT Device Client para se conectar e se autenticar AWS IoT usando o certificado de solicitação de aprovisionamento que está instalado no dispositivo.

   > **Note**
   >
   > Por segurança, o `certificateOwnershipToken` devolvido pelo `CreateCertificateFromCsr` e `CreateKeysAndCertificate` expira após uma hora. `RegisterThing` deve ser chamado antes que `certificateOwnershipToken` expire. Se o certificado criado por `CreateCertificateFromCsr` ou `CreateKeysAndCertificate` não tiver sido ativado e não tiver sido anexado a uma política ou a uma coisa quando o token expirar, o certificado será excluído. Se o token expirar, o dispositivo poderá chamar `CreateCertificateFromCsr` ou `CreateKeysAndCertificate` novamente para gerar um novo certificado.

2. O dispositivo obtém um certificado permanente e uma chave privada usando uma destas opções. O dispositivo usará o certificado e a chave para todas as autenticações futuras com AWS IoT.

   a. Ligue `CreateKeysAndCertificate`para criar um novo certificado e uma nova chave privada usando a autoridade de AWS certificação.

      Ou

   b. Chame `CreateCertificateFromCsr` para gerar um certificado de uma solicitação de assinatura de certificado que mantém sua chave privada segura.

3. No dispositivo, chame `RegisterThing` para registrar o dispositivo com a AWS IoT e criar recursos de nuvem.

   O serviço de Provisionamento de frotas usa um modelo de provisionamento para definir e criar recursos de nuvem como coisas da IoT. O modelo pode especificar atributos e grupos aos quais

a coisa pertence. Os grupos de coisas devem existir antes que uma nova possa ser adicionada a eles.

4. Depois de salvar o certificado permanente no dispositivo, o dispositivo deverá se desconectar da sessão iniciada com o certificado de reivindicação de provisionamento e reconectar usando o certificado permanente.

O dispositivo agora está pronto para se comunicar normalmente com AWS IoT.

Provisionamento por usuário confiável

Em muitos casos, um dispositivo se conecta AWS IoT pela primeira vez quando um usuário confiável, como um usuário final ou técnico de instalação, usa um aplicativo móvel para configurar o dispositivo em seu local de implantação.

> **Important**
>
> É necessário gerenciar o acesso e a permissão do usuário confiável para realizar esse procedimento. Uma maneira de fazer isso é fornecer e manter uma conta para o usuário confiável que os autentica e concede acesso às operações da API e aos recursos da AWS IoT necessários para realizar este procedimento.

Antes de entregar o dispositivo

1. Ligue `CreateProvisioningTemplate`para criar um modelo de provisionamento e retornar seu e. *`templateArn templateName`*
2. Crie um perfil do IAM que será utilizado por um usuário confiável para iniciar o processo de provisionamento. O modelo de provisionamento permite que somente esse usuário provisione um dispositivo. Por exemplo:

```
{
    "Effect": "Allow",
    "Action": [
        "iot:CreateProvisioningClaim"
    ],
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:provisioningtemplate/templateName"
    ]
```

```
}
```

3. Dê permissão ao AWS IoT serviço para criar ou atualizar recursos de IoT, como itens e certificados em sua conta ao provisionar dispositivos. Você faz isso anexando a política `AWSIoTThingsRegistration` gerenciada a uma função do IAM (chamada de função de provisionamento) que confia no responsável pelo serviço. AWS IoT
4. Forneça os meios para identificar seus usuários confiáveis, por exemplo, fornecendo a eles uma conta que possa autenticá-los e autorizar suas interações com as operações de AWS API necessárias para registrar seus dispositivos.

Como inicializar o dispositivo para uso

1. Um usuário confiável entra no aplicativo para dispositivos móveis ou no web service de provisionamento.
2. O aplicativo móvel ou o aplicativo web usa o perfil do IAM e chama `CreateProvisioningClaim` para obter um certificado de reivindicação de provisionamento temporário da AWS IoT.

   > **Note**
   >
   > Por segurança, o certificado de reivindicação de provisionamento temporário retornado pelo `CreateProvisioningClaim` expira após cinco minutos. As etapas a seguir devem retornar com êxito um certificado válido antes que o certificado de reivindicação de provisionamento temporário expire. Os certificados de reivindicação de provisionamento temporário não são exibidos na lista de certificados da sua conta.

3. O aplicativo móvel ou aplicativo web fornece o certificado de reivindicação de provisionamento temporário para o dispositivo juntamente com todas as informações de configuração necessárias, como credenciais de Wi-Fi.
4. O dispositivo usa o certificado de reivindicação de provisionamento temporário para conectar-se à AWS IoT usando o AWS IoT Device SDKs, Mobile SDKs e AWS IoT Device Client.
5. O dispositivo obtém um certificado permanente e uma chave privada usando uma dessas opções em até cinco minutos após a conexão AWS IoT com o certificado de solicitação de aprovisionamento temporário. O dispositivo usará o certificado e a chave com os quais essas opções retornarão para todas as autenticações futuras AWS IoT.

a. Ligue `CreateKeysAndCertificate`para criar um novo certificado e uma nova chave privada usando a autoridade de AWS certificação.

   Ou

b. Chame `CreateCertificateFromCsr` para gerar um certificado de uma solicitação de assinatura de certificado que mantém sua chave privada segura.

> **Note**
>
> Lembre-se `CreateKeysAndCertificate`ou `CreateCertificateFromCsr`deve devolver um certificado válido dentro de cinco minutos após a conexão AWS IoT com o certificado de solicitação de aprovisionamento temporário.

6. O dispositivo liga `RegisterThing`para registrar o dispositivo AWS IoT e criar recursos na nuvem.

   O serviço de Provisionamento de frotas usa um modelo de provisionamento para definir e criar recursos de nuvem como coisas da IoT. O modelo pode especificar atributos e grupos aos quais a coisa pertence. Os grupos de coisas devem existir antes que uma nova possa ser adicionada a eles.

7. Depois de salvar o certificado permanente no dispositivo, o dispositivo deve se desconectar da sessão iniciada com o certificado de reivindicação de provisionamento temporário e reconectar usando o certificado permanente.

O dispositivo agora está pronto para se comunicar normalmente com AWS IoT.

Usando ganchos de pré-provisionamento com a CLI AWS

O procedimento a seguir cria um modelo de provisionamento com hooks de pré-provisionamento. A função do Lambda usada aqui é um exemplo que pode ser modificado.

Como criar e aplicar um hook de pré-provisionamento a um modelo de provisionamento

1. Crie uma função do Lambda que tenha uma entrada e uma saída definidas. As funções do Lambda são altamente personalizáveis. `allowProvisioning` e `parameterOverrides` são necessários para criar hooks de pré-provisionamento. Para obter mais informações sobre a

criação de funções Lambda, consulte Usando AWS Lambda com a interface de linha de AWS comando.

Veja a seguir um exemplo de uma saída de função do Lambda:

```
{
  "allowProvisioning": True,
  "parameterOverrides": {
    "incomingKey0": "incomingValue0",
    "incomingKey1": "incomingValue1"
  }
}
```

2. AWS IoT usa políticas baseadas em recursos para chamar o Lambda, então você deve dar AWS IoT permissão para chamar sua função do Lambda.

> **Important**
>
> Certifique-se de incluir `source-arn` ou `source-account` nas chaves de contexto de condição global das políticas anexadas à sua ação do Lambda, para evitar a manipulação de permissões. Para obter mais informações sobre isso, consulte Prevenção contra o ataque do “substituto confuso” em todos os serviços.

Veja a seguir um exemplo que usa add-permission para conceder à IoT permissão para o Lambda.

```
aws lambda add-permission \
    --function-name myLambdaFunction \
    --statement-id iot-permission \
    --action lambda:InvokeFunction \
    --principal iot.amazonaws.com
```

3. Adicione um gancho de pré-provisionamento a um modelo usando o comando ou. create-provisioning-templateupdate-provisioning-template

O exemplo de CLI a seguir usa o create-provisioning-templatepara criar um modelo de provisionamento com ganchos de pré-provisionamento:

```
aws iot create-provisioning-template \
```

```
    --template-name myTemplate \
    --provisioning-role-arn arn:aws:iam:us-east-1:1234564789012:role/myRole \
    --template-body file://template.json \
    --pre-provisioning-hook file://hooks.json
```

A saída desse comando é semelhante à seguinte:

```
{
    "templateArn": "arn:aws:iot:us-east-1:1234564789012:provisioningtemplate/
myTemplate",
    "defaultVersionId": 1,
    "templateName": myTemplate
}
```

Também é possível carregar um parâmetro de um arquivo em vez de digitar tudo como um valor de parâmetro da linha de comando para economizar tempo. Para obter mais informações, consulte Carregar parâmetros da AWS CLI a partir de um arquivo. Veja a seguir o parâmetro `template` no formato JSON expandido:

```
{
    "Parameters" : {
        "DeviceLocation": {
            "Type": "String"
        }
    },
    "Mappings": {
        "LocationTable": {
            "Seattle": {
                "LocationUrl": "https://example.aws"
            }
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "AttributePayload" : {
                    "version" : "v1",
                    "serialNumber" : "serialNumber"
                },
                "ThingName" : {"Fn::Join":["",["ThingPrefix_",
{"Ref":"SerialNumber"}]]},
```

```
                "ThingTypeName" : {"Fn::Join":["",["ThingTypePrefix_",
{"Ref":"SerialNumber"}]]},
                "ThingGroups" : ["widgets", "WA"],
                "BillingGroup": "BillingGroup"
            },
            "OverrideSettings" : {
                "AttributePayload" : "MERGE",
                "ThingTypeName" : "REPLACE",
                "ThingGroups" : "DO_NOTHING"
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateId": {"Ref": "AWS::IoT::Certificate::Id"},
                "Status" : "Active",
                "ThingPrincipalType" : "EXCLUSIVE_THING"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : {
                    "Version": "2012-10-17",
                    "Statement": [{
                        "Effect": "Allow",
                        "Action":["iot:Publish"],
                        "Resource": ["arn:aws:iot:us-east-1:504350838278:topic/foo/
bar"]
                    }]
                }
            }
        }
    },
    "DeviceConfiguration": {
        "FallbackUrl": "https://www.example.com/test-site",
        "LocationUrl": {
            "Fn::FindInMap": ["LocationTable",{"Ref": "DeviceLocation"},
 "LocationUrl"]}
    }
}
```

Veja a seguir o parâmetro `pre-provisioning-hook` no formato JSON expandido:

```
{
     "targetArn" : "arn:aws:lambda:us-
east-1:765219403047:function:pre_provisioning_test",
     "payloadVersion" : "2020-04-01"
}
```

Provisionamento de dispositivos com certificados de dispositivo

AWS IoT fornece três maneiras de provisionar dispositivos quando eles já têm um certificado de dispositivo (e uma chave privada associada):

- Provisionamento de uma única coisa com um modelo de provisionamento. Essa é uma boa opção se você precisa provisionar somente um dispositivo por vez.
- Just-in-time provisionamento (JITP) com um modelo que provisiona um dispositivo quando ele se conecta pela primeira vez a. AWS IoT Essa é uma boa opção se você precisa registrar uma grande quantidade de dispositivos, mas não tem informações sobre eles para montar uma lista de provisionamento em massa.
- Registro em massa. Essa opção permite que você especifique uma lista de valores de um modelo de provisionamento de uma única coisa que são armazenados em um arquivo em um bucket do S3. Essa abordagem funciona bem se você tem uma grande quantidade de dispositivos conhecidos cujas características desejadas podem ser montadas por você em uma lista.

Tópicos

- Provisionamento de uma única coisa
- Just-in-time provisionamento
- Registro em massa

Provisionamento de uma única coisa

Para provisionar algo, use a RegisterThingAPI ou o comando `register-thing` CLI. O comando `register-thing` da CLI usa os seguintes argumentos:

--template-body

O modelo provisionado.

--parameters

Uma lista de pares de nome-valor para os parâmetros usados no modelo de provisionamento, no formato JSON (por exemplo, `{"ThingName" : "MyProvisionedThing", "CSR" : "csr-text"}`).

Consulte Modelos de provisionamento.

RegisterThingou `register-thing` retorna o ARNs para os recursos e o texto do certificado que ele criou:

```
{
    "certificatePem": "certificate-text",
    "resourceArns": {
    "PolicyLogicalName": "arn:aws:iot:us-
west-2:123456789012:policy/2A6577675B7CD1823E271C7AAD8184F44630FFD7",
    "certificate": "arn:aws:iot:us-west-2:123456789012:cert/
cd82bb924d4c6ccbb14986dcb4f40f30d892cc6b3ce7ad5008ed6542eea2b049",
    "thing": "arn:aws:iot:us-west-2:123456789012:thing/MyProvisionedThing"
    }
}
```

Se um parâmetro for omitido do dicionário, o valor padrão será usado. Se nenhum valor padrão estiver especificado, o parâmetro não será substituído por um valor.

Just-in-time provisionamento

Você pode usar o just-in-time provisionamento (JITP) para provisionar seus dispositivos quando eles tentarem se conectar pela primeira vez. AWS IoT Para provisionar o dispositivo, você deve habilitar o registro automático e associar um modelo de provisionamento ao certificado da CA usado para assinar o certificado do dispositivo. Os sucessos e erros de provisionamento são registrados como na Amazon. Métricas de provisionamento de dispositivos CloudWatch

Tópicos

- Visão geral de JITP
- Registrar CA usando o modelo de provisionamento
- Registrar CA usando o nome do modelo de provisionamento

Visão geral de JITP

Quando um dispositivo tenta se conectar AWS IoT usando um certificado assinado por um certificado CA registrado, AWS IoT carrega o modelo do certificado CA e o usa para fazer a chamada RegisterThing. O fluxo de trabalho de JITP registra primeiro um certificado com um valor de status de `PENDING_ACTIVATION`. Quando o fluxo de provisionamento do dispositivo for concluído, o status do certificado será alterado para `ACTIVE`.

AWS IoT define os seguintes parâmetros que você pode declarar e referenciar nos modelos de provisionamento:

- `AWS::IoT::Certificate::Country`
- `AWS::IoT::Certificate::Organization`
- `AWS::IoT::Certificate::OrganizationalUnit`
- `AWS::IoT::Certificate::DistinguishedNameQualifier`
- `AWS::IoT::Certificate::StateName`
- `AWS::IoT::Certificate::CommonName`
- `AWS::IoT::Certificate::SerialNumber`
- `AWS::IoT::Certificate::Id`

Os valores para esses parâmetros de modelo de provisionamento são limitados ao que o JITP pode extrair do campo de assunto do certificado do dispositivo que está sendo provisionado. O certificado deve conter valores para todos os parâmetros no corpo do modelo. O parâmetro `AWS::IoT::Certificate::Id` se refere a um ID gerado internamente, e não a um ID contido no certificado. Você pode obter o valor desse ID usando a `principal()` função dentro de uma AWS IoT regra.

> **Note**
>
> Você pode AWS IoT Core just-in-time provisionar dispositivos usando o recurso de provisionamento (JITP) sem precisar enviar toda a cadeia de confiança na primeira conexão de um dispositivo para o. AWS IoT Core A apresentação do certificado da CA é opcional, mas é necessário que o dispositivo envie a extensão Server Name Indication (SNI) ao se conectar ao AWS IoT Core.

Exemplo de corpo do modelo

O arquivo JSON a seguir é um exemplo de corpo do modelo de JITP completo.

```
{
   "Parameters":{
      "AWS::IoT::Certificate::CommonName":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::SerialNumber":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Country":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Id":{
         "Type":"String"
      }
   },
   "Resources":{
      "thing":{
         "Type":"AWS::IoT::Thing",
         "Properties":{
            "ThingName":{
               "Ref":"AWS::IoT::Certificate::CommonName"
            },
            "AttributePayload":{
               "version":"v1",
               "serialNumber":{
                  "Ref":"AWS::IoT::Certificate::SerialNumber"
               }
            },
            "ThingTypeName":"lightBulb-versionA",
            "ThingGroups":[
               "v1-lightbulbs",
               {
                  "Ref":"AWS::IoT::Certificate::Country"
               }
            ]
         },
         "OverrideSettings":{
            "AttributePayload":"MERGE",
            "ThingTypeName":"REPLACE",
            "ThingGroups":"DO_NOTHING"
```

```
         }
      },
      "certificate":{
         "Type":"AWS::IoT::Certificate",
         "Properties":{
            "CertificateId":{
               "Ref":"AWS::IoT::Certificate::Id"
            },
            "Status":"ACTIVE"
         }
      },
      "policy":{
         "Type":"AWS::IoT::Policy",
         "Properties":{
            "PolicyDocument":"{ \"Version\": \"2012-10-17\",        \"Statement
\": [{ \"Effect\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\":
 [\"arn:aws:iot:us-east-1:123456789012:topic/foo/bar\"] }] }"
         }
      }
   }
}
```

Este modelo de exemplo declara valores para os parâmetros de provisionamento `AWS::IoT::Certificate::CommonName`, `AWS::IoT::Certificate::SerialNumber`, `AWS::IoT::Certificate::Country` e `AWS::IoT::Certificate::Id` que são extraídos do certificado e usados na seção `Resources`. Em seguida, o fluxo de trabalho de JITP usa esse modelo para executar as seguintes ações:

- Registrar um certificado e definir seu status como PENDING_ACTIVE.
- Criar um recurso de coisa.
- Criar um recurso de política.
- Anexar a política ao certificado.
- Anexe o certificado ao objeto da .
- Atualizar o status do certificado como ACTIVE.

O provisionamento do dispositivo falhará se o certificado não tiver todas as propriedades mencionadas na seção `Parameters` do `templateBody`. Por exemplo, se `AWS::IoT::Certificate::Country` estiver incluído no modelo, mas o certificado não tiver uma propriedade `Country`, o provisionamento do dispositivo falhará.

Você também pode usar CloudTrail para solucionar problemas com seu modelo JITP. Para obter informações sobre as métricas registradas na Amazon CloudWatch, consulteMétricas de provisionamento de dispositivos. Para obter mais informações sobre modelos de provisionamento, consulte Modelos de provisionamento.

> **Note**
>
> Durante o processo de provisionamento, o just-in-time provisionamento (JITP) chama outras operações de API do plano de controle. AWS IoT Essas chamadas podem exceder as Cotas de controle de utilização da AWS IoT definidas para a conta e resultar em chamadas limitadas. Entre em contato com o Suporte ao cliente da AWS para aumentar as cotas do controle de utilização, se necessário.

Registrar CA usando o modelo de provisionamento

Para registrar uma CA usando um modelo de provisionamento completo, siga estas etapas:

1. Salve o modelo de provisionamento e as informações do ARN da função, como no exemplo a seguir, como um arquivo JSON:

```
{
     "templateBody" : "{\r\n    \"Parameters\" : {\r\n
 \"AWS::IoT::Certificate::CommonName\": {\r\n            \"Type\": \"String\"\r
\n        },\r\n        \"AWS::IoT::Certificate::SerialNumber\": {\r\n
  \"Type\": \"String\"\r\n        },\r\n        \"AWS::IoT::Certificate::Country
\": {\r\n            \"Type\": \"String\"\r\n        },\r\n
 \"AWS::IoT::Certificate::Id\": {\r\n            \"Type\": \"String\"\r
\n        }\r\n    },\r\n    \"Resources\": {\r\n        \"thing\": {\r
\n            \"Type\": \"AWS::IoT::Thing\",\r\n            \"Properties
\": {\r\n                \"ThingName\": {\r\n                    \"Ref\":
 \"AWS::IoT::Certificate::CommonName\"\r\n                },\r\n
    \"AttributePayload\": {\r\n                    \"version\": \"v1\",\r\n
                \"serialNumber\": {\r\n                        \"Ref\":
 \"AWS::IoT::Certificate::SerialNumber\"\r\n                    }\r\n
    },\r\n                \"ThingTypeName\": \"lightBulb-versionA\",\r\n
       \"ThingGroups\": [\r\n                    \"v1-lightbulbs\",\r\n
          {\r\n                        \"Ref\": \"AWS::IoT::Certificate::Country
\"\r\n                    }\r\n                ]\r\n            },\r\n
  \"OverrideSettings\": {\r\n                \"AttributePayload\": \"MERGE\",\r\n
               \"ThingTypeName\": \"REPLACE\",\r\n                \"ThingGroups
```

```
\": \"DO_NOTHING\"\r\n            }\r\n        },\r\n        \"certificate\": {\r
\n            \"Type\": \"AWS::IoT::Certificate\",\r\n            \"Properties
\": {\r\n                \"CertificateId\": {\r\n                    \"Ref\":
 \"AWS::IoT::Certificate::Id\"\r\n                },\r\n                \"Status\":
 \"ACTIVE\"\r\n            },\r\n            \"OverrideSettings\": {\r\n
     \"Status\": \"DO_NOTHING\"\r\n            }\r\n        },\r\n        \"policy
\": {\r\n            \"Type\": \"AWS::IoT::Policy\",\r\n            \"Properties
\": {\r\n                \"PolicyDocument\": \"{ \\\"Version\\\": \\\"2012-10-17
     \\\", \\\"Statement\\\": [{ \\\"Effect\\\": \\\"Allow\\\", \\\"Action\\\":[\\
\"iot:Publish\\\"], \\\"Resource\\\": [\\\"arn:aws:iot:us-east-1:123456789012:topic
\/foo\/bar\\\"] }] }\"\r\n            }\r\n        }\r\n    }\r\n}",
     "roleArn" : "arn:aws:iam::123456789012:role/JITPRole"
}
```

Neste exemplo, o valor do campo `templateBody` deve ser um objeto JSON especificado como uma string de escape e pode usar somente os valores da lista anterior. É possível usar uma variedade de ferramentas para criar a saída JSON necessária, como `json.dumps` (Python) ou `JSON.stringify` (Node). O valor do campo `roleARN` deve ser o ARN de uma função que tenha a `AWSIoTThingsRegistration` anexada a ele. Além disso, o modelo pode usar um `PolicyName` existente em vez da `PolicyDocument` em linha no exemplo.

2. Registre um certificado CA com a operação Register CACertificate API ou o `register-ca-certificate`comando CLI. Você precisa especificar o diretório do modelo de provisionamento e as informações do ARN da função que você salvou na etapa anterior:

   O seguinte exemplo mostra como registrar um certificado CA no modo `DEFAULT` usando a AWS CLI:

   ```
   aws iot register-ca-certificate --ca-certificate file://your-ca-cert --
   verification-cert file://your-verification-cert
                  --set-as-active --allow-auto-registration --registration-config
    file://your-template
   ```

   O seguinte exemplo mostra como registrar um certificado CA no modo `SNI_ONLY` usando a AWS CLI:

   ```
   aws iot register-ca-certificate --ca-certificate file://your-ca-cert --certificate-
   mode SNI_ONLY
                   --set-as-active --allow-auto-registration --registration-config
    file://your-template
   ```

Para obter mais informações, consulte Registrar certificados CA.

3. (Opcional) Atualize as configurações de um certificado CA usando a operação Update CACertificate API ou o comando `update-ca-certificate`CLI.

   O seguinte exemplo mostra como atualizar um certificado CA usando a AWS CLI:

```
aws iot update-ca-certificate --certificate-id caCertificateId
                --new-auto-registration-status ENABLE --registration-config
 file://your-template
```

Registrar CA usando o nome do modelo de provisionamento

Para registrar uma CA usando o nome de um modelo de provisionamento, siga estas etapas:

1. Salve o corpo do modelo de provisionamento como um arquivo JSON. Você pode encontrar um exemplo de um corpo de modelo no corpo de modelo de exemplo.
2. Para criar um modelo de aprovisionamento, use a CreateProvisioningTemplateAPI ou o comando CLI: `create-provisioning-template`

```
aws iot create-provisioning-template --template-name your-template-name \
        --template-body file://your-template-body.json --type JITP \
        --provisioning-role-arn arn:aws:iam::123456789012:role/test
```

> **Note**
>
> Para just-in-time provisionamento (JITP), você deve especificar o tipo de modelo a ser `JITP` ao criar o modelo de provisionamento. Para obter mais informações sobre o tipo de modelo, consulte CreateProvisioningTemplatea Referência AWS da API.

3. Para registrar a CA com o nome do modelo, use a CACertificate API de registro ou o comando da `register-ca-certificate`CLI:

```
aws iot register-ca-certificate --ca-certificate file://your-ca-cert --
verification-cert file://your-verification-cert \
        --set-as-active --allow-auto-registration --registration-config
 templateName=your-template-name
```

Registro em massa

É possível usar o comando `start-thing-registration-task` para registrar coisas em massa. Esse comando usa um modelo de provisionamento, um nome de bucket do S3, um nome de chave, e o ARN de uma função que permite acesso ao arquivo no bucket do S3. O arquivo no bucket do S3 contém os valores usados para substituir os parâmetros no modelo. O arquivo deve ser um JSON delimitado por nova linha. Cada linha contém todos os valores dos parâmetros para o registro de um único dispositivo. Por exemplo:

```
{"ThingName": "foo", "SerialNumber": "123", "CSR": "csr1"}
{"ThingName": "bar", "SerialNumber": "456", "CSR": "csr2"}
```

As seguintes operações de APIs relacionadas ao registro em massa podem ser úteis:

- ListThingRegistrationTasks: lista as tarefas atuais de provisionamento em massa.
- DescribeThingRegistrationTask: fornece informações sobre uma tarefa específica de registro de itens em massa.
- StopThingRegistrationTask: interrompe uma tarefa de registro em massa.
- ListThingRegistrationTaskReports: usado para verificar os resultados e as falhas de uma tarefa de registro de itens em massa.

> **Note**
>
> - Somente uma tarefa de operação de registro de coisas em massa pode ser executada de cada vez (por conta).
> - As operações de registro em massa chamam outras operações de API do plano de AWS IoT controle. Essas chamadas podem exceder as Cotas de controle de utilização da AWS IoT na conta e causar erros de limitação. Entre em contato com o Suporte ao AWS Cliente para aumentar suas cotas AWS IoT de limitação, se necessário.

Modelos de provisionamento

Um modelo de provisionamento é um documento JSON que usa parâmetros para descrever os recursos que seu dispositivo deve usar para interagir. AWS IoT Um modelo de provisionamento contém duas seções: `Parameters` e `Resources`. Há dois tipos de modelos de provisionamento em.

AWS IoT Um é usado para just-in-time provisionamento (JITP) e registro em massa, e o segundo é usado para provisionamento de frotas.

Tópicos

- Seção de parâmetros
- Seção de recursos
- Exemplo de modelo para registro em massa
- Exemplo de modelo para just-in-time provisionamento (JITP)
- Provisionamento de frotas

Seção de parâmetros

A seção `Parameters` declara os parâmetros usados na seção `Resources`. Cada parâmetro declara um nome, um tipo e um valor padrão opcional. O valor padrão é usado quando o dicionário passado com o modelo não contém um valor para o parâmetro. A seção `Parameters` de um documento de modelo é semelhante à seguinte:

```
{
    "Parameters" : {
        "ThingName" : {
            "Type" : "String"
        },
        "SerialNumber" : {
            "Type" : "String"
        },
        "Location" : {
            "Type" : "String",
            "Default" : "WA"
        },
        "CSR" : {
            "Type" : "String"
        }
    }
}
```

Esse trecho de código do corpo de modelo declara quatro parâmetros: `ThingName`, `SerialNumber`, `Location` e `CSR`. Todos esses parâmetros são do tipo `String`. O parâmetro `Location` declara um valor padrão de `"WA"`.

Seção de recursos

A `Resources` seção do corpo do modelo declara os recursos necessários para que seu dispositivo se comunique com AWS IoT: uma coisa, um certificado e uma ou mais políticas de IoT. Cada recurso especifica um nome lógico, um tipo e um conjunto de propriedades.

Um nome lógico permite que você faça referência a um recurso em outro lugar no modelo.

O tipo especifica o tipo de recurso que você está declarando. Os tipos válidos são:

- `AWS::IoT::Thing`
- `AWS::IoT::Certificate`
- `AWS::IoT::Policy`

As propriedades que você especifica dependem do tipo de recurso que você está declarando.

Recursos de coisas

Os recursos de coisas são declarados usando as seguintes propriedades:

- `ThingName`: String.
- `AttributePayload`: opcional. Uma lista de pares nome-valor.
- `ThingTypeName`: opcional. String para um tipo de coisa associado à coisa.
- `ThingGroups`: opcional. Uma lista de grupos aos quais a coisa pertence.
- `BillingGroup`: opcional. String para o nome de um grupo de faturamento associado.
- `PackageVersions`: opcional. String para um pacote associado e nomes de versão.

Recursos de certificados

É possível especificar certificados de uma das seguintes maneiras:

- Uma solicitação de assinatura de certificado (CSR).
- Um ID de certificado de um certificado de dispositivo existente. (Somente o certificado IDs pode ser usado com um modelo de aprovisionamento de frota.)
- Um certificado de dispositivo criado com um certificado da CA registrado na AWS IoT. Se houver mais de um certificado CA registrado com o mesmo campo de assunto, você também deverá passar o certificado CA usado para assinar o certificado do dispositivo.

> **Note**
>
> Ao declarar um certificado em um modelo, use somente um desses métodos. Por exemplo, se você usar uma CSR, não será possível especificar também um ID de certificado ou um certificado de dispositivo. Para obter mais informações, consulte Certificados do cliente X.509.

Para obter mais informações, consulte Visão geral do certificado X.509.

Os recursos de certificados são declarados usando as seguintes propriedades:

- `CertificateSigningRequest`: String.
- `CertificateId`: String.
- `CertificatePem`: String.
- `CACertificatePem`: String.
- `Status`: opcional. String que pode ser `ACTIVE` ou `INACTIVE`. Padronizada como ACTIVE.
- `ThingPrincipalType`: opcional. String que especifica o tipo de relacionamento entre a coisa e a entidade principal (o certificado).
  - `EXCLUSIVE_THING`: estabelece um relacionamento exclusivo. A entidade principal só pode ser vinculada a essa coisa específica e a nenhuma outra.
  - `NON_EXCLUSIVE_THING`: anexa a entidade principal especificada às coisas. É possível anexar várias coisas à entidade principal. Esse é o valor padrão caso não seja especificado.

> **Note**
>
> Você também pode provisionar dispositivos sem certificados de dispositivos. Para obter mais informações, consulte Provisionar dispositivos que não têm certificados de dispositivo usando o provisionamento de frotas.

Exemplos:

- Certificado especificado com um CSR:

```
{
    "certificate" : {
```

```
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "CertificateSigningRequest": {"Ref" : "CSR"},
            "Status" : "ACTIVE"
        }
    }
}
```

- Certificado especificado com um ID de certificado existente:

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "CertificateId": {"Ref" : "CertificateId"}
        }
    }
}
```

- Certificado especificado com um certificado .pem existente .pem e certificado .pem da CA:

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "CACertificatePem": {"Ref" : "CACertificatePem"},
            "CertificatePem": {"Ref" : "CertificatePem"}
        }
    }
}
```

- Vincule exclusivamente uma coisa a uma entidade principal:

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "ThingPrincipalType" : "EXCLUSIVE_THING"
        }
    }
}
```

Recursos de políticas

Os recursos de políticas são declarados com uma das seguintes propriedades:

- `PolicyName`: opcional. String. Padroniza para um hash do documento de política. O `PolicyName` só pode referenciar políticas da AWS IoT , mas não políticas do IAM. Se você estiver usando uma política da AWS IoT existente, insira o nome da política para a propriedade `PolicyName`. Não inclua a propriedade `PolicyDocument`.
- `PolicyDocument`: opcional. Um objeto JSON especificado como uma string de escape. Se `PolicyDocument` não for fornecido, a política já deverá estar criada.

> **Note**
>
> Se uma seção `Policy` estiver presente, `PolicyName` ou `PolicyDocument`, mas não ambos, deve ser especificado.

Configurações de substituição

Se um modelo especificar um recurso que já existe, a seção `OverrideSettings` permitirá que você especifique a ação a ser executada:

`DO_NOTHING`

Deixe o recurso como está.

`REPLACE`

Substitui o recurso pelo recurso especificado no modelo.

`FAIL`

Faz com que a solicitação falhe com um `ResourceConflictsException`.

`MERGE`

Válido apenas para as propriedades `ThingGroups` e `AttributePayload` de uma `thing`. Mescla os atributos ou associações de grupo existentes da coisa com os especificados no modelo.

Ao declarar um recurso de coisa, você pode especificar `OverrideSettings` para as seguintes propriedades:

- `ATTRIBUTE_PAYLOAD`
- `THING_TYPE_NAME`
- `THING_GROUPS`

Ao declarar um recurso de certificado, você pode especificar `OverrideSettings` para a propriedade `Status`.

`OverrideSettings` não estão disponíveis para recursos de política.

Exemplo de recurso

O trecho de código do modelo a seguir declara uma coisa, um certificado e uma política:

```
{
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "ThingName" : {"Ref" : "ThingName"},
                "AttributePayload" : { "version" : "v1", "serialNumber" :  {"Ref" :
 "SerialNumber"}},
                "ThingTypeName" :  "lightBulb-versionA",
                "ThingGroups" : ["v1-lightbulbs", {"Ref" : "Location"}]
            },
            "OverrideSettings" : {
                "AttributePayload" : "MERGE",
                "ThingTypeName" : "REPLACE",
                "ThingGroups" : "DO_NOTHING"
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateSigningRequest": {"Ref" : "CSR"},
                "Status" : "ACTIVE"
            }
        },
        "policy" : {
```

```
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : "{ \"Version\": \"2012-10-17\",
 \"Statement\": [{ \"Effect\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\":
 [\"arn:aws:iot:us-east-1:123456789012:topic/foo/bar\"] }] }"
            }
        }
    }
}
```

A coisa é declarada com:

- O nome lógico `"thing"`.
- O tipo `AWS::IoT::Thing`.
- Um conjunto de propriedades de coisas.

  As propriedades da coisa incluem o nome, um conjunto de atributos, um nome opcional de tipo de coisa e uma lista opcional de grupos de coisas aos quais a coisa pertence.

Os parâmetros são referenciados por `{"Ref":"`*`parameter-name`*`"}`. Quando o modelo é avaliado, os parâmetros são substituídos pelo valor do parâmetro do dicionário passado com o modelo.

O certificado é declarado com:

- O nome lógico `"certificate"`.
- O tipo `AWS::IoT::Certificate`.
- Um conjunto de propriedades.

  As propriedades incluem a CSR do certificado e a configuração do status como `ACTIVE`. O texto da CSR é passado como um parâmetro no dicionário passado com o modelo.

A política é declarada com:

- O nome lógico `"policy"`.
- O tipo `AWS::IoT::Policy`.
- O nome de uma política existente ou um documento de política.

Exemplo de modelo para registro em massa

O arquivo JSON a seguir é um exemplo de modelo de provisionamento completo que especifica o certificado com uma CSR:

(O valor do campo `PolicyDocument` deve ser um objeto JSON especificado como uma string de escape.)

```
{
    "Parameters" : {
        "ThingName" : {
            "Type" : "String"
        },
        "SerialNumber" : {
            "Type" : "String"
        },
        "Location" : {
            "Type" : "String",
            "Default" : "WA"
        },
        "CSR" : {
            "Type" : "String"
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "ThingName" : {"Ref" : "ThingName"},
                "AttributePayload" : { "version" : "v1", "serialNumber" :  {"Ref" :
 "SerialNumber"}},
                "ThingTypeName" :  "lightBulb-versionA",
                "ThingGroups" : ["v1-lightbulbs", {"Ref" : "Location"}]
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateSigningRequest": {"Ref" : "CSR"},
                "Status" : "ACTIVE",
                "ThingPrincipalType" : "EXCLUSIVE_THING"
            }
        },
```

```
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : "{ \"Version\": \"2012-10-17\",
 \"Statement\": [{ \"Effect\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\":
 [\"arn:aws:iot:us-east-1:123456789012:topic/foo/bar\"] }] }"
            }
        }
    }
}
```

Exemplo de modelo para just-in-time provisionamento (JITP)

O arquivo JSON a seguir é um exemplo de modelo de provisionamento completo que especifica um certificado existente com um ID de certificado:

```
{
   "Parameters":{
      "AWS::IoT::Certificate::CommonName":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::SerialNumber":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Country":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Id":{
         "Type":"String"
      }
   },
   "Resources":{
      "thing":{
         "Type":"AWS::IoT::Thing",
         "Properties":{
            "ThingName":{
               "Ref":"AWS::IoT::Certificate::CommonName"
            },
            "AttributePayload":{
               "version":"v1",
               "serialNumber":{
                  "Ref":"AWS::IoT::Certificate::SerialNumber"
               }
```

```
            },
            "ThingTypeName":"lightBulb-versionA",
            "ThingGroups":[
               "v1-lightbulbs",
               {
                  "Ref":"AWS::IoT::Certificate::Country"
               }
            ]
         },
         "OverrideSettings":{
            "AttributePayload":"MERGE",
            "ThingTypeName":"REPLACE",
            "ThingGroups":"DO_NOTHING"
         }
      },
      "certificate":{
         "Type":"AWS::IoT::Certificate",
         "Properties":{
            "CertificateId":{
               "Ref":"AWS::IoT::Certificate::Id"
            },
            "Status":"ACTIVE",
            "ThingPrincipalType" : "EXCLUSIVE_THING"
         }
      },
      "policy":{
         "Type":"AWS::IoT::Policy",
         "Properties":{
            "PolicyDocument":"{ \"Version\": \"2012-10-17\",        \"Statement
\": [{ \"Effect\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\":
 [\"arn:aws:iot:us-east-1:123456789012:topic/foo/bar\"] }] }"
         }
      }
   }
}
```

> **Important**
>
> Você deve usar `CertificateId` em um modelo usado para o provisionamento JIT.

Para obter mais informações sobre o tipo de modelo de aprovisionamento, consulte a referência `CreateProvisioningTemplate`da AWS API.

Para obter mais informações sobre como usar esse modelo para just-in-time provisionamento, consulte: Provisionamento J. ust-in-time

Provisionamento de frotas

Os modelos de provisionamento de frota são usados AWS IoT para configurar a nuvem e o dispositivo. Esses modelos usam os mesmos parâmetros e recursos que o JITP e os modelos de registro em massa. Para obter mais informações, consulte Modelos de provisionamento.
Os modelos de provisionamento de frotas podem conter uma seção `Mapping` e uma seção `DeviceConfiguration`. É possível usar funções intrínsecas dentro de um modelo de provisionamento de frotas para gerar uma configuração específica de dispositivo. Os modelos de aprovisionamento de frotas são recursos nomeados e identificados por ARNs (por exemplo,`arn:aws:iot:us-west-2:1234568788:provisioningtemplate/`*`templateName`*).

Mapeamentos

A seção opcional `Mappings` corresponde uma chave a um conjunto de valores nomeados correspondente. Por exemplo, se você quiser definir valores com base em uma AWS região, você pode criar um mapeamento que usa o Região da AWS nome como chave e contém os valores que você deseja especificar para cada região específica. Você usa a função intrínseca `Fn::FindInMap` para recuperar valores em um mapa.

Não é possível incluir parâmetros, pseudoparâmetros ou chamar funções intrínsecas na seção `Mappings`.

Configuração do dispositivo

A seção de configuração do dispositivo contém os dados arbitrários que você deseja enviar para os dispositivos ao provisionar. Por exemplo:

```
{
    "DeviceConfiguration": {
        "Foo":"Bar"
    }
}
```

Se você estiver enviando mensagens para seus dispositivos usando o formato de carga útil do JavaScript Object Notation (JSON), AWS IoT Core formate esses dados como JSON. Se você estiver usando o formato de carga útil Concise Binary Object Representation (CBOR), o AWS IoT Core formatará esses dados como CBOR. A seção `DeviceConfiguration` não é compatível com objetos JSON aninhados.

Funções intrínsecas

As funções intrínsecas são usadas em qualquer seção do modelo de provisionamento, exceto a seção `Mappings`.

`Fn::Join`

Anexa um conjunto de valores em um único valor, separados pelo delimitador especificado. Se um delimitador é uma string vazia, os valores são concatenados sem delimitador.

> **Important**
>
> `Fn::Join` não é compatível com the section called “Recursos de políticas”.

`Fn::Select`

Retorna um único objeto de uma lista de objetos por índice.

> **Important**
>
> `Fn::Select` não verifica valores `null` ou se o índice está fora dos limites da matriz. Ambas as condições resultam em um erro de provisionamento, portanto, escolha um valor de índice e garanta que a lista contenha valores não nulos.

`Fn::FindInMap`

Retorna o valor correspondente às chaves em um mapa de dois níveis que é declarado na seção `Mappings`.

`Fn::Split`

Divide uma string em uma lista de valores de string para que seja possível selecionar um elemento na lista de strings. Especifique um delimitador que determine onde a string é dividida

(por exemplo, uma vírgula). Depois de dividir uma string, use `Fn::Select` para selecionar um elemento.

Por exemplo, se uma string de sub-rede delimitada por vírgula IDs for importada para seu modelo de pilha, você poderá dividir a string em cada vírgula. Na lista de sub-rede IDs, use `Fn::Select` para especificar uma ID de sub-rede para um recurso.

`Fn::Sub`

Substitui variáveis em uma string de entrada por valores especificados por você. É possível usar essa função para criar comandos ou saídas que incluem valores que não estão disponíveis até que você crie ou atualize uma pilha.

Exemplo de modelo de provisionamento por frota

```
{
    "Parameters" : {
        "ThingName" : {
            "Type" : "String"
        },
        "SerialNumber": {
            "Type": "String"
        },
        "DeviceLocation": {
            "Type": "String"
        }
    },
    "Mappings": {
        "LocationTable": {
            "Seattle": {
                "LocationUrl": "https://example.aws"
            }
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "AttributePayload" : {
                    "version" : "v1",
                    "serialNumber" : "serialNumber"
                },
```

```
                "ThingName" : {"Ref" : "ThingName"},
                "ThingTypeName" : {"Fn::Join":["",["ThingPrefix_",
{"Ref":"SerialNumber"}]]},
                "ThingGroups" : ["v1-lightbulbs", "WA"],
                "BillingGroup": "LightBulbBillingGroup"
            },
            "OverrideSettings" : {
                "AttributePayload" : "MERGE",
                "ThingTypeName" : "REPLACE",
                "ThingGroups" : "DO_NOTHING"
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateId": {"Ref": "AWS::IoT::Certificate::Id"},
                "Status" : "Active",
                "ThingPrincipalType" : "EXCLUSIVE_THING"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : {
                    "Version": "2012-10-17",
                    "Statement": [{
                        "Effect": "Allow",
                        "Action":["iot:Publish"],
                        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/foo/
bar"]
                    }]
                }
            }
        }
    },
    "DeviceConfiguration": {
        "FallbackUrl": "https://www.example.com/test-site",
        "LocationUrl": {
            "Fn::FindInMap": ["LocationTable",{"Ref": "DeviceLocation"},
 "LocationUrl"]}
        }
}
```

> **Note**
>
> Um modelo de provisionamento existente pode ser atualizado para adicionar um hook de pré-provisionamento.

Ganchos de pré-provisionamento

AWS recomenda o uso de funções de gancho de pré-provisionamento ao criar modelos de provisionamento para permitir mais controle sobre quais e quantos dispositivos sua conta incorpora. Os hooks de pré-provisionamento são funções do Lambda que validam parâmetros passados do dispositivo antes de permitir que ele seja provisionado. Essa função do Lambda deve existir na sua conta antes de provisionar um dispositivo, já que ela é chamada sempre que um dispositivo envia uma solicitação por meio de the section called “RegisterThing”.

> **Important**
>
> Certifique-se de incluir `source-arn` ou `source-account` nas chaves de contexto de condição global das políticas anexadas à sua ação do Lambda, para evitar a manipulação de permissões. Para obter mais informações sobre isso, consulte Prevenção contra o ataque do “substituto confuso” em todos os serviços.

Para dispositivos a serem provisionados, a função do Lambda deve aceitar o objeto de entrada e retornar o objeto de saída descrito nesta seção. O provisionamento prosseguirá somente se a função do Lambda retornar um objeto com `"allowProvisioning": True`.

Entrada de hook de pré-provisão

AWS IoT envia esse objeto para a função Lambda quando um dispositivo se registra com. AWS IoT

```
{
    "claimCertificateId" : "string",
    "certificateId" : "string",
    "certificatePem" : "string",
    "templateArn" : "arn:aws:iot:us-east-1:1234567890:provisioningtemplate/MyTemplate",
    "clientId" : "221a6d10-9c7f-42f1-9153-e52e6fc869c1",
    "parameters" : {
```

```
        "string" : "string",
        ...
    }
}
```

O objeto `parameters` passado para a função do Lambda contém as propriedades no argumento `parameters` passado na carga da solicitação dethe section called “RegisterThing”.

Valor de retorno do hook de pré-provisão

A função do Lambda deve retornar uma resposta que indique se autorizou a solicitação de provisionamento e os valores de quaisquer propriedades a serem substituídos.

Veja a seguir um exemplo de uma resposta bem-sucedida da função de pré-provisionamento.

```
{
    "allowProvisioning": true,
    "parameterOverrides" : {
        "Key": "newCustomValue",
        ...
    }
}
```

Os valores de `"parameterOverrides"` serão adicionados ao parâmetro `"parameters"` na carga da solicitação de the section called “RegisterThing”.

> **Note**
>
> - Se a função Lambda falhar, a solicitação de provisionamento falhará `ACCESS_DENIED` e um erro será registrado no Logs. CloudWatch
> - Se a função do Lambda não retornar `"allowProvisioning": "true"` na resposta, a solicitação de provisionamento falhará com `ACCESS_DENIED`.
> - A função do Lambda deve concluir a execução e retornar em até 5 segundos, caso contrário, a solicitação de provisionamento falhará.

Exemplo de hook de pré-provisionamento do Lambda

Python

Um exemplo de hook de pré-provisionamento do Lambda em Python.

```
import json

def pre_provisioning_hook(event, context):
    print(event)

    return {
        'allowProvisioning': True,
        'parameterOverrides': {
            'DeviceLocation': 'Seattle'
        }
    }
```

Java

Um exemplo de hook de pré-provisionamento do Lambda em Java.

Classe de handler:

```
package example;

import java.util.Map;
import java.util.HashMap;
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class PreProvisioningHook implements
 RequestHandler<PreProvisioningHookRequest, PreProvisioningHookResponse> {

    public PreProvisioningHookResponse handleRequest(PreProvisioningHookRequest
 object, Context context) {
        Map<String, String> parameterOverrides = new HashMap<String, String>();
        parameterOverrides.put("DeviceLocation", "Seattle");

        PreProvisioningHookResponse response = PreProvisioningHookResponse.builder()
                .allowProvisioning(true)
                .parameterOverrides(parameterOverrides)
                .build();
```

```
        return response;
    }

}
```

Classe de solicitação:

```
package example;

import java.util.Map;
import lombok.Builder;
import lombok.Data;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;

@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class PreProvisioningHookRequest {
    private String claimCertificateId;
    private String certificateId;
    private String certificatePem;
    private String templateArn;
    private String clientId;
    private Map<String, String> parameters;
}
```

Classe de resposta:

```
package example;

import java.util.Map;
import lombok.Builder;
import lombok.Data;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;


@Data
@Builder
@AllArgsConstructor
```

```
@NoArgsConstructor
public class PreProvisioningHookResponse {
    private boolean allowProvisioning;
    private Map<String, String> parameterOverrides;
}
```

JavaScript

Um exemplo de um gancho de pré-provisionamento no Lambda. JavaScript

```
exports.handler = function(event, context, callback) {
    console.log(JSON.stringify(event, null, 2));
    var reply = {
        allowProvisioning: true,
        parameterOverrides: {
            DeviceLocation: 'Seattle'
        }
     };
     callback(null, reply);
}
```

Assinatura de certificado autogerenciada usando provedor AWS IoT Core de certificados

Você pode criar um provedor de AWS IoT Core certificados para assinar solicitações de assinatura de certificados (CSRs) no aprovisionamento de AWS IoT frotas. Um provedor de certificados faz referência a uma função do Lambda e à API MQTT para provisionamento de frotas `CreateCertificateFromCsr`. A função do Lambda aceita uma CSR e retorna um certificado de cliente assinado.

Quando você não tem um provedor de certificados com o seu Conta da AWS, a API CreateCertificateFromCsr MQTT é chamada no provisionamento da frota para gerar o certificado a partir de uma CSR. Depois de criar um provedor de certificado, o comportamento da API do CreateCertificateFromCsr MQTT mudará e todas as chamadas para essa API do MQTT invocarão o provedor do certificado para emitir o certificado.

Com o provedor de AWS IoT Core certificados, você pode implementar soluções que utilizam autoridades de certificação privadas (CAs) CA Privada da AWS, como outras de confiança CAs pública ou sua própria infraestrutura de chave pública (PKI) para assinar a CSR. Além disso, você

pode usar o provedor de certificados para personalizar os campos do certificado do seu cliente, como períodos de validade, algoritmos de assinatura, emissores e extensões.

> **Important**
>
> Em seguida, é possível criar apenas um provedor de certificados por Conta da AWS. A mudança de comportamento de assinatura se aplica a toda a frota que chama a API CreateCertificateFromCsr MQTT até que você exclua o provedor de certificados do seu Conta da AWS.

Neste tópico:

- Como a assinatura autogerenciada de certificados funciona no provisionamento de frotas
- Entrada da função do Lambda do provedor de certificados
- Valor de retorno da função do Lambda do provedor de certificados
- Exemplo de função do Lambda
- Assinatura de certificado autogerenciada para provisionamento de frota
- AWS CLI comandos para provedor de certificados

Como a assinatura autogerenciada de certificados funciona no provisionamento de frotas

Principais conceitos

Os conceitos a seguir fornecem detalhes que podem ajudar você a entender como a assinatura autogerenciada de certificados funciona no provisionamento de AWS IoT frotas. Para obter mais informações, consulte Provisionar itens que não têm certificados de dispositivo usando o provisionamento de frota.

AWS IoT aprovisionamento de frotas

Com o provisionamento de AWS IoT frota (abreviação de provisionamento de frota), AWS IoT Core gera e entrega com segurança certificados de dispositivos aos seus dispositivos quando eles se conectam pela primeira vez. AWS IoT Core Você pode usar o aprovisionamento de frotas para conectar dispositivos que não têm certificados de dispositivo ao AWS IoT Core.

Solicitação de assinatura de certificado (CSR)

No processo de provisionamento da frota, um dispositivo faz uma solicitação AWS IoT Core por meio do MQTT de provisionamento da frota. APIs Essa solicitação inclui uma solicitação de assinatura de certificado (CSR), que será assinada para produzir um certificado de cliente.

AWS assinatura gerenciada de certificados no provisionamento de frotas

AWS gerenciado é a configuração padrão para assinatura de certificados no aprovisionamento de frotas. Com a assinatura de certificado AWS gerenciada, AWS IoT Core assinará CSRs usando a sua própria CAs.

Assinatura de certificado autogerenciada em provisionamento de frota

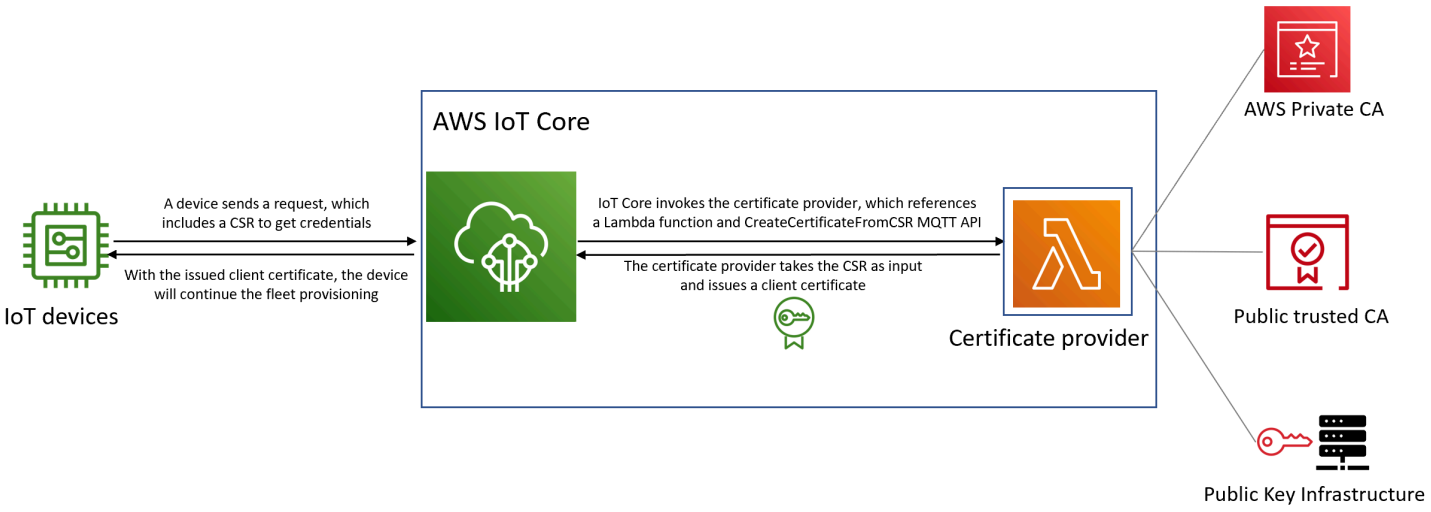
Autogerenciada é outra opção para assinatura de certificado no provisionamento de frota. Com a assinatura de certificado autogerenciada, você cria um provedor AWS IoT Core de certificados para assinar CSRs. Você pode usar a assinatura de certificado autogerenciada para assinar CSRs com uma CA gerada pela CA AWS privada, outra CA publicamente confiável ou sua própria infraestrutura de chave pública (PKI).

AWS IoT Core provedor de certificados

AWS IoT Core provedor de certificados (abreviação de provedor de certificados) é um recurso gerenciado pelo cliente usado para assinatura autogerenciada de certificados no provisionamento de frotas.

Diagrama

O diagrama a seguir é uma ilustração simplificada de como a assinatura de autocertificado funciona no provisionamento de AWS IoT frotas.



- Quando um novo dispositivo de IoT é fabricado ou introduzido na frota, ele precisa de certificados de cliente para se autenticar. AWS IoT Core
- Como parte do processo de provisionamento da frota, o dispositivo solicita certificados do cliente AWS IoT Core por meio do MQTT de provisionamento da frota. APIs Essa solicitação inclui uma solicitação de assinatura de certificado (CSR).
- AWS IoT Core invoca o provedor do certificado e passa o CSR como entrada para o provedor.
- O provedor do certificado usa a CSR como entrada e emite um certificado do cliente.

  Para assinatura AWS gerenciada de certificados, AWS IoT Core assina a CSR usando sua própria CA e emite um certificado de cliente.
- Com o certificado de cliente emitido, o dispositivo continuará o provisionamento da frota e estabelecerá uma conexão segura com o AWS IoT Core.

Entrada da função do Lambda do provedor de certificados

AWS IoT Core envia o objeto a seguir para a função Lambda quando um dispositivo se registra nela. O valor de `certificateSigningRequest` é a CSR no formato Privacy-Enhanced Mail (PEM) fornecido na solicitação. `CreateCertificateFromCsr` `principalId`É o ID do principal usado para se conectar AWS IoT Core ao fazer a `CreateCertificateFromCsr` solicitação. `clientId`é o ID do cliente definido para a conexão MQTT.

```
{
 "certificateSigningRequest": "string",
 "principalId": "string",
```

```
 "clientId": "string"
}
```

Valor de retorno da função do Lambda do provedor de certificados

A função do Lambda deve retornar uma resposta que contenha o valor `certificatePem`. Veja a seguir um exemplo de uma resposta bem-sucedida. AWS IoT Core usará o valor de retorno (`certificatePem`) para criar o certificado.

```
{
 "certificatePem": "string"
}
```

Se o cadastro for bem-sucedido, `CreateCertificateFromCsr` retornará o mesmo `certificatePem` na resposta `CreateCertificateFromCsr`. Para obter mais informações, consulte o exemplo de carga útil de resposta de CreateCertificateFromCsr.

Exemplo de função do Lambda

Antes de criar um provedor de certificados, é necessário criar uma função do Lambda para assinar um CSR. A seguir está um exemplo de função do Lambda em Python. Essa função chama CA Privada da AWS para assinar a CSR de entrada, usando uma CA privada e o algoritmo de assinatura `SHA256WITHRSA`. O certificado de cliente devolvido será válido por um ano. Para obter mais informações sobre CA Privada da AWS e como criar uma CA privada, consulte O que é CA AWS privada? e Criação de uma CA privada.

```
import os
import time
import uuid
import boto3

def lambda_handler(event, context):
    ca_arn = os.environ['CA_ARN']
    csr = (event['certificateSigningRequest']).encode('utf-8')

    acmpca = boto3.client('acm-pca')
    cert_arn = acmpca.issue_certificate(
        CertificateAuthorityArn=ca_arn,
        Csr=csr,
        Validity={"Type": "DAYS", "Value": 365},
```

```
        SigningAlgorithm='SHA256WITHRSA',
        IdempotencyToken=str(uuid.uuid4())
    )['CertificateArn']

    # Wait for certificate to be issued
    time.sleep(1)
    cert_pem = acmpca.get_certificate(
        CertificateAuthorityArn=ca_arn,
        CertificateArn=cert_arn
    )['Certificate']

    return {
        'certificatePem': cert_pem
    }
```

Important

- Os certificados retornados pela função do Lambda devem ter o mesmo nome de assunto e chave pública da Solicitação de Assinatura de Certificado (CSR).
- A função do Lambda deve terminar de ser executada em 5 segundos.
- A função Lambda deve estar na mesma região do Conta da AWS recurso do provedor de certificados.
- O responsável pelo AWS IoT serviço deve receber a permissão de invocação para a função Lambda. Para evitar problemas de substituto confuso, recomendamos definir as permissões de invocação `sourceArn` e `sourceAccount`. Para obter mais informações, consulte Prevenção de confused deputy entre serviços.

O seguinte exemplo de política baseada em recursos para o Lambda concede a AWS IoT a permissão para invocar a função do Lambda:

```
{
 "Version":"2012-10-17",
 "Id": "InvokePermission",
 "Statement": [
  {
   "Sid": "LambdaAllowIotProvider",
   "Effect": "Allow",
   "Principal": {
```

```
    "Service": "iot.amazonaws.com"
   },
   "Action": "lambda:InvokeFunction",
   "Resource": "arn:aws:lambda:us-east-1:123456789012:function:my-function",
   "Condition": {
    "StringEquals": {
     "AWS:SourceAccount": "123456789012"
    },
    "ArnLike": {
    "AWS:SourceArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider/my-
certificate-provider"
    }
   }
  }
 ]
}
```

Assinatura de certificado autogerenciada para provisionamento de frota

Você pode escolher a assinatura de certificado autogerenciada para provisionamento de frota usando AWS CLI ou Console de gerenciamento da AWS.

AWS CLI

Para escolher a assinatura de certificado autogerenciada, você deve criar um provedor de AWS IoT Core certificados para fazer login CSRs no aprovisionamento de frotas. AWS IoT Core invoca o provedor do certificado, que usa uma CSR como entrada e retorna um certificado do cliente. Para criar um provedor de certificados, use a operação da API `CreateCertificateProvider` ou o comando da CLI `create-certificate-provider`.

> **Note**
>
> Depois de criar um provedor de certificados, o comportamento da API `CreateCertificateFromCsr` para provisionamento de frotas mudará, de modo que todas as chamadas para `CreateCertificateFromCsr` invocarão o provedor de certificados para criar os certificados. Pode levar alguns minutos para que esse comportamento mude após a criação de um provedor de certificados.

```
aws iot create-certificate-provider \
```

```
                --certificateProviderName my-certificate-provider \
                --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-
function-1 \
                --accountDefaultForOperations CreateCertificateFromCsr
```

O exemplo a seguir mostra uma saída de exemplo para esse comando:

```
{
 "certificateProviderName": "my-certificate-provider",
 "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-
certificate-provider"
}
```

Para obter mais informações, consulte `CreateCertificateProvider` na Referência de APIs de AWS IoT.

Console de gerenciamento da AWS

Para escolher a assinatura de certificado autogerenciada usando Console de gerenciamento da AWS, siga as etapas:

1. Acesse o console do AWS IoT.
2. Na navegação esquerda, em Segurança, escolha Assinatura de certificado.
3. Na página Assinatura do certificado, em Detalhes da assinatura do certificado, escolha Editar método de assinatura do certificado.
4. Na página Editar método de assinatura do certificado, em Método de assinatura do certificado, escolha Autogerenciado.
5. Na seção Configurações autogerenciadas, insira um nome para o provedor de certificados e crie ou escolha uma função do Lambda.
6. Escolha Atualizar assinatura do certificado.

AWS CLI comandos para provedor de certificados

Criar provedor de certificados

Para criar um provedor de certificados, use a operação da API `CreateCertificateProvider` ou o comando da CLI `create-certificate-provider`.

> **Note**
>
> Depois de criar um provedor de certificados, o comportamento da API `CreateCertificateFromCsr` para provisionamento de frotas mudará, de modo que todas as chamadas para `CreateCertificateFromCsr` invocarão o provedor de certificados para criar os certificados. Pode levar alguns minutos para que esse comportamento mude após a criação de um provedor de certificados.

```
aws iot create-certificate-provider \
                --certificateProviderName my-certificate-provider \
                --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-function-1 \
                --accountDefaultForOperations CreateCertificateFromCsr
```

O exemplo a seguir mostra uma saída de exemplo para esse comando:

```
{
 "certificateProviderName": "my-certificate-provider",
 "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-certificate-provider"
}
```

Para obter mais informações, consulte `CreateCertificateProvider` Referência de APIs de AWS IoT.

Atualizar provedor de certificados

Para atualizar um provedor de certificados, use a operação da API `UpdateCertificateProvider` ou o comando da CLI `update-certificate-provider`.

```
aws iot update-certificate-provider \
                --certificateProviderName my-certificate-provider \
                --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-function-2 \
                --accountDefaultForOperations CreateCertificateFromCsr
```

O exemplo a seguir mostra uma saída de exemplo para esse comando:

```
{
```

```
 "certificateProviderName": "my-certificate-provider",
 "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-
certificate-provider"
}
```

Para obter mais informações, consulte UpdateCertificateProvider na Referência de APIs de AWS IoT.

Descrever provedor de certificados

Para descrever um provedor de certificados, use a operação da API `DescribeCertificateProvider` ou o comando da CLI `describe-certificate-provider`.

```
aws iot describe-certificate-provider --certificateProviderName my-certificate-provider
```

O exemplo a seguir mostra uma saída de exemplo para esse comando:

```
{
 "certificateProviderName": "my-certificate-provider",
 "lambdaFunctionArn": "arn:aws:lambda:us-east-1:123456789012:function:my-function",
 "accountDefaultForOperations": [
  "CreateCertificateFromCsr"
 ],
 "creationDate": "2022-11-03T00:15",
 "lastModifiedDate": "2022-11-18T00:15"
}
```

Para obter mais informações, consulte DescribeCertificateProvider Referência de APIs de AWS IoT.

Excluir provedor de certificados

Para excluir um provedor de certificados, use a operação da API `DeleteCertificateProvider` ou o comando da CLI `delete-certificate-provider`. Se você excluir o recurso do provedor de certificados, o comportamento do `CreateCertificateFromCsr` será retomado e AWS IoT criará certificados AWS IoT assinados por um CSR.

```
aws iot delete-certificate-provider --certificateProviderName my-certificate-provider
```

Esse comando não retorna nenhuma saída.

Para obter mais informações, consulte `DeleteCertificateProvider` Referência de APIs de AWS IoT.

Listar provedor de certificados

Para listar os provedores de certificados em seu Conta da AWS, use a operação de `ListCertificateProviders` API ou o `list-certificate-providers` comando CLI.

```
aws iot list-certificate-providers
```

O exemplo a seguir mostra uma saída de exemplo para esse comando:

```
{
 "certificateProviders": [
  {
   "certificateProviderName": "my-certificate-provider",
   "certificateProviderArn": "arn:aws:iot:us-
east-1:123456789012:certificateprovider:my-certificate-provider"
  }
 ]
}
```

Para obter mais informações, consulte `ListCertificateProvider` na Referência de APIs de AWS IoT.

Criação de políticas e perfis do IAM para um usuário instalando um dispositivo

> **Note**
>
> Esses procedimentos devem ser usados somente quando orientados pelo AWS IoT console. Para acessar essa página a partir do console, abra Criar um novo modelo de provisionamento.

Por que isso não pode ser feito no AWS IoT console?

Para uma experiência mais segura, as ações do IAM são executadas no console do IAM. Os procedimentos nesta seção recomendam que você examine as etapas para criar os perfis e políticas do IAM necessárias para usar o modelo de provisionamento.

Criação de uma política do IAM para o usuário que instalará um dispositivo

Esse procedimento descreve como criar uma política do IAM que autoriza um usuário a instalar um dispositivo usando um modelo de provisionamento.

Ao realizar esse procedimento, você alternará entre o console do IAM e o AWS IoT console. Recomendamos que os dois consoles estejam abertos simultaneamente enquanto você conclui o procedimento.

Para criar uma política do IAM para o usuário que instalará um dispositivo

1. Abra Hub de políticas no console do IAM.
2. Escolha Criar política.
3. Na página Criar política, escolha a guia JSON.
4. Vá para a página no AWS IoT console em que você escolheu Configurar política e função do usuário.
5. Em Exemplo de política de provisionamento, escolha Copiar.
6. Volte para o console do IAM.
7. No editor JSON, cole a política que você copiou do AWS IoT console. Essa política é específica para o modelo que você está criando no AWS IoT console.
8. Para continuar, escolha Próximo: Tags.
9. Na página Adicionar tags (opcional), escolha Adicionar tag para cada tag que você deseja adicionar a essa política. Você poderá ignorar essa etapa se não tiver tags para adicionar.
10. Para continuar, escolha Próximo: Revisar.
11. Na página Review Policy (Revisar política), faça o seguinte:

    a. Em Nome*, insira um nome para a política que ajude você a lembrar a finalidade dela.

       Anote o nome que você está dando a essa política, já que ele será usado no próximo procedimento.

b. Você pode optar por inserir uma descrição opcional da política que está criando.

c. Revise o restante da política e as tags.

12. Escolha Criar política para concluir a criação da nova política.

Depois de criar a nova política, continue no the section called “Criação de um perfil do IAM para o usuário que instalará um dispositivo” para criar a função do usuário à qual você vai anexá-la.

Criação de um perfil do IAM para o usuário que instalará um dispositivo

Essas etapas descrevem como criar um perfil do IAM que autentica o usuário que instalará um dispositivo usando um modelo de provisionamento.

Para criar uma política do IAM para o usuário que instalará um dispositivo

1. Abra Hub de Funções no console do IAM.
2. Selecione Criar perfil.
3. Em Selecionar entidade confiável, escolha o tipo de entidade confiável a que você deseja dar acesso ao modelo que você está criando.
4. Escolha ou insira a identificação da entidade confiável à qual você deseja conceder acesso e escolha Próximo.
5. Na página Adicionar permissões, na caixa de pesquisa Políticas de permissão, insira o nome da política criada no procedimento anterior.
6. Para a lista de políticas, selecione a política criada no procedimento anterior e selecione Próximo.
7. Na seção Nomear, revisar e criar, faça o seguinte:

   a. Em Nome do perfil, insira um nome de função que ajude você a se lembrar da finalidade da função.

   b. Em Descrição, você pode optar por inserir uma descrição opcional da função. Isso não é necessário para continuar.

   c. Revise os valores na Etapa 1 e Etapa 2.

   d. Em Adicionar tags (opcional), você pode optar por adicionar tags a essa função. Isso não é necessário para continuar.

   e. Verifique se as informações nessa página estão completas e corretas e escolha Criar função.

Depois de criar a nova função, retorne ao AWS IoT console para continuar criando o modelo.

Atualização de uma política existente para autorizar um novo modelo

As etapas a seguir descrevem como adicionar um novo modelo a uma política do IAM que autoriza um usuário a instalar um dispositivo usando um modelo de provisionamento.

Para adicionar um novo modelo a uma política do IAM existente

1. Abra Hub de políticas no console do IAM.
2. Na caixa de pesquisa, insira o nome da política que será atualizada.
3. Na lista abaixo da caixa de pesquisa, encontre a política que você deseja atualizar e escolha o nome dela.
4. Em Resumo da política, escolha a guia JSON, se esse painel ainda não estiver visível.
5. Para modificar o documento da política, escolha Editar política.
6. No editor, escolha a guia JSON, se esse painel ainda não estiver visível.
7. No documento da política, encontre a declaração de política que contém a ação `iot:CreateProvisioningClaim`.

   Se o documento de política não contiver uma declaração de política com a ação `iot:CreateProvisioningClaim`, copie o trecho de declaração a seguir e cole-o como uma entrada adicional na matriz `Statement` do documento de política.

   > **Note**
   >
   > Esse trecho deve ser colocado antes do caractere `]` de fechamento na matriz `Statement`. Talvez seja necessário adicionar uma vírgula antes ou depois desse trecho para corrigir erros de sintaxe.

```
{
    "Effect": "Allow",
    "Action": [
        "iot:CreateProvisioningClaim"
    ],
    "Resource": [
        "--PUT YOUR NEW TEMPLATE ARN HERE--"
    ]
```

```
}
```

8. Vá para a página no AWS IoT console em que você escolheu Modificar permissões de função de usuário.
9. Procure o ARN do recurso do modelo e escolha Copiar.
10. Volte para o console do IAM.
11. Cole o Amazon Resource Name (ARN) copiado no topo da lista de ARNs modelos na `Statement` matriz para que seja a primeira entrada.

    Se esse for o único ARN na matriz, remova a vírgula no final do valor que você acabou de colar.
12. Revise a declaração de política atualizada e corrija os erros indicados pelo editor.
13. Para salvar o documento de política atualizado, escolha Revisar política.
14. Selecione Revisar política e, em seguida, escolha, Salvar alterações.
15. Volte para o AWS IoT console.

API MQTT de provisionamento de dispositivos

O serviço de Provisionamento de frotas dá suporte às seguintes operações da API MQTT:

- the section called “CreateCertificateFromCsr”
- the section called “CreateKeysAndCertificate”
- the section called “RegisterThing”

Essa API suporta buffers de resposta no formato Concise Binary Object Representation (CBOR) e JavaScript Object Notation (JSON), dependendo do tópico. *`payload-format`* Para clareza, os exemplos de resposta e de solicitação nesta seção são mostrados no formato JSON.

`payload-format`	Tipo de dados do formato de resposta
cbor	Representação Concisa de Objetos Binários (CBOR)
json	JavaScript Notação de objeto (JSON)

> **Important**
>
> Antes de publicar um tópico de mensagem de solicitação, assine os tópicos de resposta para receber a resposta. As mensagens usadas por esta API usam o protocolo MQTT de publicação/assinatura para fornecer uma interação de solicitação e resposta.
> Se você não assinar os tópicos de resposta antes de publicar uma solicitação, talvez não receba os resultados dessa solicitação.
> O IoT Core Fleet Provisioning exibe os resultados da API de provisionamento de dispositivos do MQTT por meio da mesma conexão do MQTT usada para publicar a solicitação da API.

CreateCertificateFromCsr

Cria um certificado a partir de uma solicitação de assinatura de certificado (CSR). AWS IoT fornece certificados de cliente assinados pela autoridade de certificação (CA) raiz da Amazon. O novo certificado tem um status `PENDING_ACTIVATION`. Quando você chama `RegisterThing` para provisionar uma coisa com esse certificado, o status do certificado muda para `ACTIVE` ou `INACTIVE` conforme descrito no modelo.

Para obter mais informações sobre como criar um certificado de cliente usando seu certificado de Autoridade de Certificação e uma solicitação de assinatura de certificado, consulte Criar um certificado de cliente usando o certificado CA.

> **Note**
>
> Por segurança, o `certificateOwnershipToken` devolvido pelo `CreateCertificateFromCsr` expira após uma hora. `RegisterThing` deve ser chamado antes que `certificateOwnershipToken` expire. Se o certificado criado por `CreateCertificateFromCsr` não tiver sido ativado nem anexado a uma política ou a um objeto quando o token expirar, o certificado será excluído. Se o token expirar, o dispositivo poderá chamar `CreateCertificateFromCsr` novamente para gerar um novo certificado.

CreateCertificateFromCsr pedido

Publique uma mensagem com o tópico `$aws/certificates/create-from-csr/`*`payload-format`*.

`payload-format`

O formato da carga útil da mensagem como `cbor` ou `json`.

CreateCertificateFromCsr carga útil de solicitação

```
{
    "certificateSigningRequest": "string"
}
```

`certificateSigningRequest`

A CSR, no formato PEM.

CreateCertificateFromCsr resposta

Assine `$aws/certificates/create-from-csr/`*`payload-format`*`/accepted`.

`payload-format`

O formato da carga útil da mensagem como `cbor` ou `json`.

CreateCertificateFromCsr carga útil de resposta

```
{
    "certificateOwnershipToken": "string",
    "certificateId": "string",
    "certificatePem": "string"
}
```

`certificateOwnershipToken`

O token para provar a propriedade do certificado durante o provisionamento.

`certificateId`

O ID do certificado. As operações de gerenciamento de certificado usam apenas um certificateId.

`certificatePem`

Os dados do certificado, no formato PEM.

CreateCertificateFromCsr erro

Para receber respostas de erro, assine `$aws/certificates/create-from-csr/`*`payload-format`*`/rejected`.

`payload-format`

O formato da carga útil da mensagem como `cbor` ou `json`.

CreateCertificateFromCsr carga útil de erro

```
{
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
}
```

`statusCode`

O código do status.

`errorCode`

O código do erro.

`errorMessage`

A mensagem de erro.

CreateKeysAndCertificate

Cria novas chaves e um certificado. AWS IoT fornece certificados de cliente assinados pela autoridade de certificação (CA) raiz da Amazon. O novo certificado tem um status `PENDING_ACTIVATION`. Quando você chama `RegisterThing` para provisionar uma coisa com esse certificado, o status do certificado muda para `ACTIVE` ou `INACTIVE` conforme descrito no modelo.

> **Note**
>
> Por segurança, o `certificateOwnershipToken` devolvido pelo `CreateKeysAndCertificate` expira após uma hora. `RegisterThing` deve ser

chamado antes que `certificateOwnershipToken` expire. Se o certificado criado por `CreateKeysAndCertificate` não tiver sido ativado nem anexado a uma política ou a um objeto quando o token expirar, o certificado será excluído. Se o token expirar, o dispositivo poderá chamar `CreateKeysAndCertificate` novamente para gerar um novo certificado.

CreateKeysAndCertificate pedido

Publique uma mensagem em `$aws/certificates/create/`*`payload-format`* com uma carga de mensagem vazia.

`payload-format`

O formato da carga útil da mensagem como `cbor` ou `json`.

CreateKeysAndCertificate resposta

Assine `$aws/certificates/create/`*`payload-format`*`/accepted`.

`payload-format`

O formato da carga útil da mensagem como `cbor` ou `json`.

CreateKeysAndCertificate resposta

```
{
    "certificateId": "string",
    "certificatePem": "string",
    "privateKey": "string",
    "certificateOwnershipToken": "string"
}
```

`certificateId`

O ID do certificado.

`certificatePem`

Os dados do certificado, no formato PEM.

`privateKey`

A chave privada.

`certificateOwnershipToken`

O token para provar a propriedade do certificado durante o provisionamento.

CreateKeysAndCertificate erro

Para receber respostas de erro, assine `$aws/certificates/create/`*`payload-format`*`/rejected`.

`payload-format`

O formato da carga útil da mensagem como `cbor` ou `json`.

CreateKeysAndCertificate carga útil de erro

```
{
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
}
```

`statusCode`

O código do status.

`errorCode`

O código do erro.

`errorMessage`

A mensagem de erro.

RegisterThing

Provisiona uma coisa usando um modelo predefinido.

RegisterThing pedido

Publique uma mensagem em `$aws/provisioning-templates/`*`templateName`*`/provision/`*`payload-format`*.

`payload-format`

O formato da carga útil da mensagem como `cbor` ou `json`.

`templateName`

O nome do modelo provisionado.

RegisterThing carga útil de solicitação

```
{
    "certificateOwnershipToken": "string",
    "parameters": {
        "string": "string",
        ...
    }
}
```

`certificateOwnershipToken`

O token para provar a propriedade do certificado. AWS IoT gera o token quando você cria um certificado por MQTT.

`parameters`

Opcional. Pares de chave-valor do dispositivo que são usados pelos hooks de pré-provisionamento para avaliar a solicitação de registro.

RegisterThing resposta

Assine `$aws/provisioning-templates/`*`templateName`*`/provision/`*`payload-format`*`/accepted`.

`payload-format`

O formato da carga útil da mensagem como `cbor` ou `json`.

`templateName`

O nome do modelo provisionado.

RegisterThing carga útil de resposta

```
{
    "deviceConfiguration": {
        "string": "string",
        ...
    },
    "thingName": "string"
}
```

`deviceConfiguration`

A configuração do dispositivo definida no modelo.

`thingName`

O nome da coisa da IoT criada durante o provisionamento.

RegisterThing resposta de erro

Para receber respostas de erro, assine `$aws/provisioning-templates/`*`templateName`*`/provision/`*`payload-format`*`/rejected`.

`payload-format`

O formato da carga útil da mensagem como `cbor` ou `json`.

`templateName`

O nome do modelo provisionado.

RegisterThing carga útil de resposta de erro

```
{
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
```

```
}
```

`statusCode`

O código do status.

`errorCode`

O código do erro.

`errorMessage`

A mensagem de erro.

Indexação de frotas

Você pode usar a indexação de frotas para indexar, pesquisar e agregar os dados de seus dispositivos das seguintes fontes: registro da AWS IoT, sombra do dispositivo da AWS IoT, conectividade da AWS IoT, Catálogo de pacotes de software do AWS IoT Device Management e violações do AWS IoT Device Defender. É possível consultar um grupo de dispositivos e agregar estatísticas sobre registros de dispositivos baseados em diferentes combinações de atributos do dispositivo, incluindo estado, conectividade e violações do dispositivo. Com a indexação de frotas, é possível organizar, investigar e solucionar problemas em sua frota de dispositivos.

A indexação de frotas oferece os seguintes recursos.

Gerenciamento de atualizações de índice

É possível configurar um índice de frota para indexar atualizações dos seus grupos de objetos, registros de objetos, sombras de dispositivos, conectividade de dispositivos e violações de dispositivos. Quando você ativa a indexação de frotas, o AWS IoT cria um índice das suas objetos ou grupos de objetos. `AWS_Things` é o índice criado para todas as suas objetos. `AWS_ThingGroups` é o índice que contém todos seus grupos de objetos. Quando a indexação de frotas estiver ativa, você poderá executar consultas em seu índice. Por exemplo, você pode encontrar todos os dispositivos portáteis com mais de 70% de duração da bateria. AWS IoT atualiza o índice continuamente com seus dados mais recentes. Para acessar mais informações, consulte Gerenciar a indexação de frotas.

Consultar o status de conectividade de um dispositivo específico

Essa API fornece acesso de baixa latência e alto throughput às informações mais recentes de conectividade específicas do dispositivo. Para acessar mais informações, consulte Status de conectividade de dispositivos.

Pesquisa em várias fontes dos dados

É possível criar uma string de consulta com base em uma linguagem de consulta e usá-la para pesquisar em fontes de dados. Você também precisa configurar as fontes de dados na configuração de indexação da frota para que a configuração de indexação contenha as fontes de dados nas quais você deseja pesquisar. A string de consulta descreve o objeto que você quer encontrar. Você pode

criar consultas usando campos AWS gerenciados, campos personalizados e quaisquer atributos de suas fontes de dados indexadas. Para acessar mais informações sobre fontes de dados compatíveis com a indexação de frotas, consulte Gerenciamento da indexação de objetos.

Consulta de dados agregados

É possível pesquisar seus dispositivos em busca de dados agregados e retornar estatísticas, percentil, cardinalidade ou uma lista de objetos com consultas de pesquisa sobre campos específicos. Você pode executar agregações em campos AWS gerenciados ou em qualquer atributo configurado como campos personalizados nas configurações de indexação da frota. Para acessar mais informações sobre consulta de agregação, consulte Consultar dados agregados.

Monitoramento de dados agregados e criação de alarmes usando as métricas da frota

Você pode usar métricas de frota para enviar dados agregados CloudWatch automaticamente, analisar tendências e criar alarmes para monitorar o estado agregado de sua frota com base em limites predefinidos. Para acessar mais informações sobre métricas de frota, consulte Métricas de frota.

Gerenciamento da indexação de frotas

A indexação de frotas gerencia dois tipos de índices: indexação de objetos e indexação de grupos de objetos.

Indexação de objetos

O índice criado para todas as suas objetos é chamado de `AWS_Things`. A indexação de coisas comporta as seguintes fontes de dados: dados de registro da AWS IoT, dados da sombra do dispositivo da AWS IoT, dados de conectividade da AWS IoT e dados de violações do AWS IoT Device Defender. Ao adicionar essas fontes de dados à sua configuração de indexação de frotas, é possível pesquisar objetos, consultar dados agregados e criar grupos dinâmicos de objetos e métricas de frota com base em suas consultas de pesquisa.

Registro -AWS IoT fornece um registro que ajuda você a gerenciar coisas. É possível adicionar os dados do registro à configuração de indexação da frota para pesquisar dispositivos com base nos

nomes, descrições e outros atributos do registro das objetos. Para acessar mais informações sobre o registro, consulte Como gerenciar objetos com o registro.

Sombra- o serviço de sombra do dispositivo do AWS IoT oferece sombras que ajudam você a armazenar os dados de estado de um dispositivo. A indexação de objetos é compatível tanto com sombras clássicas sem nome quanto sombras nomeadas. Para indexar sombras nomeadas, ative as configurações de sombra nomeada e especifique os nomes de sombra na configuração de indexação de objetos. Por padrão, você pode adicionar até 10 nomes de sombra por Conta da AWS. Para saber como aumentar o limite de número de nomes de sombras, consulte AWS IoT Device Management Cotas na Referência geral da AWS .

Para adicionar sombras nomeadas para indexação:

- Se você usa o console do AWS IoT, ative a Indexação de objetos, selecione Adicionar sombras nomeadas e adicione os nomes das sombras por meio da Seleção de sombras nomeadas.
- Se você usar o AWS Command Line Interface (AWS CLI), `namedShadowIndexingMode` defina como ser `ON` e especifique os nomes das sombras em `IndexingFilter`. Para ver exemplos de comandos da CLI, consulte Gerenciamento da indexação de objetos.

> **Important**
>
> 20 de julho de 2022 é a versão de Disponibilidade Geral (GA) da integração de indexação de frotas de gerenciamento de AWS IoT dispositivos com sombras AWS IoT Core nomeadas e AWS IoT Device Defender detecção de violações. Com esta versão do GA, é possível indexar sombras nomeadas específicas especificando nomes das sombras. Caso tenha adicionado suas sombras nomeadas para indexação durante o período de pré-visualização pública desse atributo, de 30 de novembro de 2021 a 19 de julho de 2022, recomendamos que você reconfigure suas definições de indexação de frotas e escolha nomes de sombra específicos para reduzir o custo de indexação e otimizar o desempenho.

Para acessar mais informações sobre as sombras, consulte Serviço de sombra do dispositivo do AWS IoT.

Conectividade - os dados de conectividade do dispositivo ajudam você a identificar o status da conexão dos dispositivos. Esses dados de conectividade são orientados por eventos do ciclo de vida. Quando um cliente se conecta ou desconecta, AWS IoT publica eventos do ciclo de vida com mensagens para tópicos do MQTT. Uma mensagem de conexão ou desconexão pode ser uma lista

de elementos JSON fornecendo detalhes sobre o status da conexão. Para acessar mais informações sobre a conectividade de dispositivos, consulte Eventos de ciclo de vida.

Violações do Device Defender - os dados de AWS IoT Device Defender violações ajudam a identificar comportamentos anômalos do dispositivo em relação aos comportamentos normais que você define em um Perfil de Segurança. Um perfil de segurança contém um conjunto de comportamentos esperados. Cada comportamento utiliza uma métrica que especifica o comportamento normal dos dispositivos. Para acessar mais informações sobre violações do Device Defender, consulte Detectar AWS IoT Device Defender.

Para acessar mais informações, consulte Gerenciamento da indexação de objetos.

Indexação de grupo de objetos

`AWS_ThingGroups` é o índice que contém todos os seus grupos de objetos. Você pode usar esse índice para pesquisar grupos com base no nome do grupo, na descrição, nos atributos e em todos os nomes de grupo pai.

Para acessar mais informações, consulte Gerenciamento da indexação de grupos de objetos.

Campos gerenciados

Os campos gerenciados contêm dados associados a coisas, grupos de coisas, sombras de dispositivos, conectividade de dispositivos e violações do Device Defender. AWS IoT define o tipo de dados nos campos gerenciados. Você especifica os valores de cada campo gerenciado ao criar AWS IoT algo. Por exemplo, nomes de objetos, grupos de objetos e descrições de objetos são todos os campos gerenciados. A indexação de frotas indexa campos gerenciados com base no modo de indexação especificado por você. Os campos gerenciados não podem ser alterados e não aparecem em `customFields`. Para acessar mais informações, consulte Campos personalizados.

A seguir está uma lista dos campos gerenciados para indexação de objetos:

- Campos gerenciados para o registro

```
"managedFields" : [
  {name:thingId, type:String},
  {name:thingName, type:String},
  {name:registry.version, type:Number},
  {name:registry.thingTypeName, type:String},
  {name:registry.thingGroupNames, type:String},
```

```
]
```

- Campos gerenciados para sombras clássicas sem nome

```
"managedFields" : [
  {name:shadow.version, type:Number},
  {name:shadow.hasDelta, type:Boolean}
]
```

- Campos gerenciados para sombras nomeadas

```
"managedFields" : [
  {name:shadow.name.shadowName.version, type:Number},
  {name:shadow.name.shadowName.hasDelta, type:Boolean}
]
```

- Campos gerenciados para conectividade de objetos

```
"managedFields" : [
  {name:connectivity.timestamp, type:Number},
  {name:connectivity.version, type:Number},
  {name:connectivity.connected, type:Boolean},
  {name:connectivity.disconnectReason, type:String}
]
```

- Campos gerenciados para o Device Defender

```
"managedFields" : [
  {name:deviceDefender.violationCount, type:Number},
  {name:deviceDefender.securityprofile.behaviorname.metricName, type:String},
  {name:deviceDefender.securityprofile.behaviorname.lastViolationTime, type:Number},
  {name:deviceDefender.securityprofile.behaviorname.lastViolationValue, type:String},
  {name:deviceDefender.securityprofile.behaviorname.inViolation, type:Boolean}
]
```

- Campos gerenciados para grupos de objetos

```
"managedFields" : [
  {name:description, type:String},
  {name:parentGroupNames, type:String},
  {name:thingGroupId, type:String},
  {name:thingGroupName, type:String},
  {name:version, type:Number},
```

```
]
```

A tabela a seguir lista os campos gerenciados que não podem ser pesquisados.

Fonte de dados	Campo gerenciado que não pode ser pesquisado
Registro	`registry.version`
Sombra nomeada	`shadow.version`
Sombra nomeada	`shadow.name.*.version`
Device Defender	`deviceDefender.version`
Grupos de objetos	`version`

Campos personalizados

Você pode agregar atributos de objetos, dados de sombra do dispositivo e dados de violações do Device Defender criando campos personalizados para indexá-los. O atributo `customFields` é uma lista de pares de nomes de campos e tipos de dados. É possível realizar consultas de agregação com base no tipo de dados. O modo de indexação selecionado afeta os campos que podem ser especificados em `customFields`. Por exemplo, ao especificar o modo de indexação `REGISTRY`, não será possível especificar um campo personalizado de uma sombra de um objeto. Você pode usar o comando update-indexing-configurationCLI para criar ou atualizar os campos personalizados (veja um exemplo de comando em Atualizar exemplos de configuração de indexação).

- Nomes de campos personalizados

Os nomes de campo personalizados para atributos de objetos e grupos de objetos iniciam com `attributes.`, seguidos pelo nome do atributo. Se a indexação de sombra sem nome estiver ativada, as objetos podem ter nomes de campos personalizados que iniciem com `shadow.desired` ou `shadow.reported`, seguidos pelo nome do valor de dados da sombra sem nome. Se a indexação de sombra nomeada estiver ativada, as objetos podem ter nomes de campos personalizados que iniciem com `shadow.name.*.desired.` ou `shadow.name.*.reported.`, seguidos pelo valor de dados da sombra nomeada. Se a indexação de violações do Device

Defender estiver ativada, as objetos poderão ter nomes de campos personalizados que iniciem com`deviceDefender.`, seguidos pelo valor dos dados de violações do Device Defender.

O nome do valor de dados ou atributo que acompanha o prefixo só pode possuir caracteres alfanuméricos, - (hífen) e _ (sublinhado). Ele não pode ter espaços.

Caso haja uma inconsistência de tipo entre um campo personalizado na configuração e o valor sendo indexado, a indexação de frotas ignorará o valor inconsistente para consultas de agregação. CloudWatch Os registros são úteis para solucionar problemas de consulta de agregação. Para obter mais informações, consulte Solução de problemas de consultas de agregação para o serviço de indexação de frota.

- Nomes de campos personalizados

Os tipos de campos personalizados têm os seguintes valores compatíveis: `Number`, `String` e `Boolean`.

Gerenciar a indexação de objetos

O índice criado para todas as suas objetos é `AWS_Things`. É possível controlar o que indexar a partir das seguintes fontes de dados: dados de registro da AWS IoT, dados da sombra do dispositivo da AWS IoT, dados de conectividade da AWS IoT e dados de violações do AWS IoT Device Defender.

Neste tópico:

- Habilitar a indexação de objetos
- Descrever um índice de objetos
- Consultar um índice de objetos
- Restrições e limitações
- Autorização

Habilitar a indexação de objetos

Você usa o comando update-indexing-configurationCLI ou a operação da UpdateIndexingConfigurationAPI para criar o `AWS_Things` índice e controlar sua configuração. Ao usar o parâmetro `--thing-indexing-configuration` (`thingIndexingConfiguration`),

você controla o tipo de dados (por exemplo, dados de registro, sombra, conectividade do dispositivo e violações do Device Defender) que é indexado.

O parâmetro `--thing-indexing-configuration` leva uma string com a seguinte estrutura:

```
{
  "thingIndexingMode": "OFF"|"REGISTRY"|"REGISTRY_AND_SHADOW",
  "thingConnectivityIndexingMode": "OFF"|"STATUS",
  "deviceDefenderIndexingMode": "OFF"|"VIOLATIONS",
  "namedShadowIndexingMode": "OFF"|"ON",
  "managedFields": [
    {
      "name": "string",
      "type": "Number"|"String"|"Boolean"
    },
    ...
  ],
  "customFields": [
    {
      "name": "string",
      "type": "Number"|"String"|"Boolean"
    },
    ...
  ],
  "filter": {
     "namedShadowNames": [ "string" ],
     "geoLocations": [
        {
            "name": "String",
            "order": "LonLat|LatLon"
        }
    ]
  }
}
```

Modos de indexação de objetos

Você pode especificar diferentes modos de indexação em sua configuração de indexação, dependendo de quais fontes de dados você quer indexar e nas quais deseja pesquisar dispositivos:

- `thingIndexingMode`: controla se o registro ou a sombra estão indexados. Quando `thingIndexingMode` é definido como `OFF`, a indexação de itens é desativada.

- `thingConnectivityIndexingMode`: especifica se dados de conectividade são indexados.

- `deviceDefenderIndexingMode`: especifica se os dados de violações do Device Defender são indexados.
- `namedShadowIndexingMode`: especifica se os dados de sombra nomeada são indexados. Para selecionar sombras nomeadas para adicionar à sua configuração de indexação de frotas, defina `namedShadowIndexingMode` como `ON` e especifique os nomes de sombras nomeadas em `filter`.

A tabela abaixo mostra os valores válidos para cada modo de indexação e a fonte de dados indexada para cada valor.

Atributo	Valores válidos	Registro	Shadow	Conectividade	Violações de DD	Sombra nomeada
`thingIndexingMode`	DESL.					
	REGISTRY	✓				
	REGISTRY_AND_SHADOW	✓	✓			
`thingConnectivityIndexingMode`	Não especificado.					
	DESL.					
	STATUS			✓		
`deviceDefenderIndexingMode`	Não especificado.					
	DESL.					
	VIOLAÇÕES				✓	

Atributo	Valores válidos	Registro	Shadow	Conectividade	Violações de DD	Sombra nomeada
`namedShadowIndexingMode`	Não especificado.					
	DESL.					
	ON					✓

Campos gerenciados e campos personalizados

Campos gerenciados

Os campos gerenciados contêm dados associados a coisas, grupos de coisas, sombras de dispositivos, conectividade de dispositivos e violações do Device Defender. AWS IoT define o tipo de dados nos campos gerenciados. Você especifica os valores de cada campo gerenciado ao criar um objeto do AWS IoT . Por exemplo, nomes de objetos, grupos de objetos e descrições de objetos são todos os campos gerenciados. A indexação de frotas indexa campos gerenciados com base no modo de indexação especificado por você. Os campos gerenciados não podem ser alterados e não aparecem em `customFields`.

Campos personalizados

Você pode agregar atributos, dados de sombra do dispositivo e dados de violações do Device Defender criando campos personalizados para indexá-los. O atributo `customFields` é uma lista de pares de nomes de campos e tipos de dados. É possível realizar consultas de agregação com base no tipo de dados. O modo de indexação selecionado afeta os campos que podem ser especificados em `customFields`. Por exemplo, ao especificar o modo de indexação `REGISTRY`, não será possível especificar um campo personalizado de uma sombra de um objeto. Você pode usar o comando update-indexing-configurationCLI para criar ou atualizar os campos personalizados (veja um exemplo de comando em Atualizar exemplos de configuração de indexação). Para acessar mais informações, consulte Campos personalizados.

Filtro de indexação

O filtro de indexação fornece seleções adicionais para sombras nomeadas e dados de localização geográfica.

`namedShadowNames`

Para adicionar sombras nomeadas à sua configuração de indexação de frotas, defina `namedShadowIndexingMode` como `ON` e especifique os nomes de sombras nomeadas no filtro `namedShadowNames`.

Exemplo

```
"filter": {
     "namedShadowNames": [ "namedShadow1", "namedShadow2" ]
}
```

`geoLocations`

Para adicionar dados de localização geográfica à sua configuração de indexação de frota:

- Se seus dados de localização geográfica estiverem armazenados em uma sombra clássica (sem nome), defina `thingIndexingMode` como REGISTRY_AND_SHADOW e especifique seus dados de localização geográfica no filtro `geoLocations`.

  O exemplo de filtro abaixo especifica um objeto geoLocation em uma sombra clássica (sem nome):

  ```
  "filter": {
       "geoLocations": [
          {
              "name": "shadow.reported.location",
              "order": "LonLat"
          }
       ]
    }
  ```

- Se seus dados de localização geográfica estiverem armazenados em uma sombra nomeada, defina `namedShadowIndexingMode` como ON, adicione o nome da sombra no filtro `namedShadowNames` e especifique seus dados de localização geográfica no filtro `geoLocations`.

  O exemplo de filtro abaixo especifica um objeto geoLocation em uma sombra nomeada (`nameShadow1`):

  ```
  "filter": {
       "namedShadowNames": [ "namedShadow1" ],
       "geoLocations": [
  ```

```
        {
            "name": "shadow.name.namedShadow1.reported.location",
            "order": "LonLat"
        }
    ]
}
```

Para obter mais informações, consulte IndexingFiltera Referência da AWS IoTAPI.

Atualizar exemplos de configuração de indexação

Para atualizar sua configuração de indexação, use o comando AWS IoT update-indexing-configuration CLI. O exemplo a seguir mostra como usar update-indexing-configuration.

Sintaxe curta:

```
aws iot update-indexing-configuration --thing-indexing-configuration \
'thingIndexingMode=REGISTRY_AND_SHADOW, deviceDefenderIndexingMode=VIOLATIONS,
namedShadowIndexingMode=ON,filter={namedShadowNames=[thing1shadow]},
 thingConnectivityIndexingMode=STATUS,
customFields=[{name=attributes.version,type=Number},
{name=shadow.name.thing1shadow.desired.DefaultDesired, type=String},
 {name=shadow.desired.power, type=Boolean},
{name=deviceDefender.securityProfile1.NUMBER_VALUE_BEHAVIOR.lastViolationValue.number,
 type=Number}]'
```

Sintaxe do JSON:

```
aws iot update-indexing-configuration --cli-input-json \ '{
          "thingIndexingConfiguration": { "thingIndexingMode": "REGISTRY_AND_SHADOW",
          "thingConnectivityIndexingMode": "STATUS",
          "deviceDefenderIndexingMode": "VIOLATIONS",
          "namedShadowIndexingMode": "ON",
          "filter": { "namedShadowNames": ["thing1shadow"]},
          "customFields": [ { "name": "shadow.desired.power", "type": "Boolean" },
          {"name": "attributes.version", "type": "Number"},
          {"name": "shadow.name.thing1shadow.desired.DefaultDesired", "type":
 "String"},
          {"name":
 "deviceDefender.securityProfile1.NUMBER_VALUE_BEHAVIOR.lastViolationValue.number",
 "type": Number} ] } }'
```

Esse comando não retorna nenhuma saída.

Para verificar o status do índice do objeto, execute o comando `describe-index` da CLI:

```
aws iot describe-index --index-name "AWS_Things"
```

A saída do comando `describe-index` é semelhante a:

```
{
    "indexName": "AWS_Things",
    "indexStatus": "ACTIVE",
    "schema": "MULTI_INDEXING_MODE"
}
```

> **Note**
>
> A indexação da frota pode levar alguns minutos para atualizar o índice da frota. Recomendamos aguardar até que `indexStatus` exiba ATIVO antes de usá-lo. Os seus valores no campo do esquema podem ser diferentes, dependendo das fontes de dados que configurou. Para acessar mais informações, consulte Descrever um índice de objetos.

Para obter seus detalhes de configuração de indexação de objetos, execute o comando `get-indexing-configuration` da CLI:

```
aws iot get-indexing-configuration
```

A saída do comando `get-indexing-configuration` é semelhante a:

```
{
    "thingIndexingConfiguration": {
        "thingIndexingMode": "REGISTRY_AND_SHADOW",
        "thingConnectivityIndexingMode": "STATUS",
        "deviceDefenderIndexingMode": "VIOLATIONS",
        "namedShadowIndexingMode": "ON",
        "managedFields": [
            {
                "name": "connectivity.disconnectReason",
                "type": "String"
            },
            {
```

```
            "name": "registry.version",
            "type": "Number"
        },
        {
            "name": "thingName",
            "type": "String"
        },
        {
            "name": "deviceDefender.violationCount",
            "type": "Number"
        },
        {
            "name": "shadow.hasDelta",
            "type": "Boolean"
        },
        {
            "name": "shadow.name.*.version",
            "type": "Number"
        },
        {
            "name": "shadow.version",
            "type": "Number"
        },
        {
            "name": "connectivity.version",
            "type": "Number"
        },
        {
            "name": "connectivity.timestamp",
            "type": "Number"
        },
        {
            "name": "shadow.name.*.hasDelta",
            "type": "Boolean"
        },
        {
            "name": "registry.thingTypeName",
            "type": "String"
        },
        {
            "name": "thingId",
            "type": "String"
        },
        {
```

```
                "name": "connectivity.connected",
                "type": "Boolean"
            },
            {
                "name": "registry.thingGroupNames",
                "type": "String"
            }
        ],
        "customFields": [
            {
                "name": "shadow.name.thing1shadow.desired.DefaultDesired",
                "type": "String"
            },

            {
                "name":
 "deviceDefender.securityProfile1.NUMBER_VALUE_BEHAVIOR.lastViolationValue.number",
                "type": "Number"
            },
            {
                "name": "shadow.desired.power",
                "type": "Boolean"
            },
            {
                "name": "attributes.version",
                "type": "Number"
            }
        ],
        "filter": {
              "namedShadowNames": [
                  "thing1shadow"
              ]
          }
      },
    "thingGroupIndexingConfiguration": {
        "thingGroupIndexingMode": "OFF"
    }
}
```

Para atualizar os campos personalizados, execute o comando `update-indexing-configuration`. Um exemplo é este:

```
aws iot update-indexing-configuration --thing-indexing-configuration
```

```
 'thingIndexingMode=REGISTRY_AND_SHADOW,customFields=[{name=attributes.version,type=Number},
{name=attributes.color,type=String},{name=shadow.desired.power,type=Boolean},
{name=shadow.desired.intensity,type=Number}]'
```

Esse comando adicionou `shadow.desired.intensity` à configuração de indexação.

> **Note**
>
> Atualizar a configuração de indexação de campos personalizados substitui todos os campos personalizados existentes. Especifique todos os campos personalizados ao chamar update-indexing-configuration.

Depois que o índice é reconstruído, você pode usar a consulta de agregação nos campos recém-adicionados, dados de registro de pesquisa, dados de sombra e dados de status de conectividade de objeto.

Ao alterar o modo de indexação, verifique se todos os campos personalizados são válidos utilizando o novo modo de indexação. Por exemplo, se você começar usando o modo `REGISTRY_AND_SHADOW` com um campo personalizado chamado `shadow.desired.temperature`, deverá excluir o campo personalizado `shadow.desired.temperature` antes de alterar o modo de indexação para `REGISTRY`. Se a configuração de indexação contiver campos personalizados que não são indexados pelo modo de indexação, a atualização falhará.

Descrever um índice de objetos

O comando a seguir mostra como usar o comando describe-index da CLI para recuperar o status atual do índice de objetos.

```
aws iot describe-index --index-name "AWS_Things"
```

A resposta do comando pode ser semelhante a:

```
{
    "indexName": "AWS_Things",
    "indexStatus": "BUILDING",
    "schema": "REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS"
```

```
}
```

A primeira vez que você faz a indexação de frotas, AWS IoT cria seu índice. Não é possível consultar o índice se `indexStatus` estiver no estado `BUILDING`. O `schema` do índice de objetos indica qual tipo de dados (`REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS`) é indexado.

A alteração da configuração do índice faz com que o índice seja recompilado. Durante esse processo, o `indexStatus` é `REBUILDING`. É possível executar consultas em dados no índice de objetos enquanto ele está sendo reconstruído. Por exemplo, se você alterar a configuração do índice de `REGISTRY` para `REGISTRY_AND_SHADOW` enquanto o índice estiver sendo recompilado, poderá consultar dados do registro, incluindo as últimas atualizações. No entanto, você não pode consultar os dados de sombra até que a reconstrução seja concluída. O tempo necessário para compilar ou recompilar o índice depende da quantidade de dados.

Você pode observar valores diferentes no campo do esquema conforme as fontes de dados configuradas. A tabela a seguir exibe os diferentes valores de esquema e as descrições correspondentes:

Schema	Description
DESL.	Nenhuma fonte de dados está configurada ou indexada.
REGISTRY	Os dados de registro são indexados.
REGISTRY_AND_SHADOW	Os dados de registro e dados de sombra (clássica) sem nome são indexados.
REGISTRY_AND_CONNECTIVITY	Os dados do registro e os dados de conectividade são indexados.
REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS	Os dados de registro, dados de sombra (clássica) sem nome e dados de conectividade são indexados.
MULTI_INDEXING_MODE	Os dados de sombra nomeada ou de violações do Device Defender são indexados, além dos dados de registro, sombra (clássica) sem nome ou conectividade.

Consultar um índice de objetos

Use o comando search-index da CLI para consultar dados no índice.

```
aws iot search-index --index-name "AWS_Things" --query-string
          "thingName:mything*"
```


```
{
    "things":[{
         "thingName":"mything1",
         "thingGroupNames":[
            "mygroup1"
         ],
         "thingId":"a4b9f759-b0f2-4857-8a4b-967745ed9f4e",
         "attributes":{
            "attribute1":"abc"
         },
         "connectivity": {
            "connected":false,
            "timestamp":1556649874716,
            "disconnectReason": "CONNECTION_LOST"
         }
    },
    {
        "thingName":"mything2",
        "thingTypeName":"MyThingType",
        "thingGroupNames":[
            "mygroup1",
            "mygroup2"
        ],
        "thingId":"01014ef9-e97e-44c6-985a-d0b06924f2af",
        "attributes":{
            "model":"1.2",
            "country":"usa"
        },
        "shadow":{
            "desired":{
                "location":"new york",
                "myvalues":[3, 4, 5]
            },
            "reported":{
                "location":"new york",
                "myvalues":[1, 2, 3],
```

```
                "stats":{
                    "battery":78
                }
            },
            "metadata":{
                 "desired":{
                       "location":{
                            "timestamp":123456789
                        },
                       "myvalues":{
                            "timestamp":123456789
                       }
                 },
                 "reported":{
                       "location":{
                            "timestamp":34535454
                        },
                       "myvalues":{
                            "timestamp":34535454
                       },
                       "stats":{
                            "battery":{
                                  "timestamp":34535454
                            }
                       }
                 }
            },
            "version":10,
            "timestamp":34535454
        },
        "connectivity": {
            "connected":true,
            "timestamp":1556649855046
        }
    }],
    "nextToken":"AQFCuvk7zZ3D9pOYMbFCeHbdZ+h=G"
}
```

Na resposta JSON, "`connectivity`" (conforme habilitado pela configuração `thingConnectivityIndexingMode=STATUS`) fornece um valor booliano, um registro de data/hora e um disconnectReason que indicam se o dispositivo está conectado ao AWS IoT Core. O dispositivo "`mything1`" desconectado (`false`) no horário POSIX 1556649874716 devido a

`CONNECTION_LOST`. Para acessar mais informações sobre motivos de desconexão, consulte Eventos de ciclo de vida.

```
"connectivity": {
    "connected":false,
    "timestamp":1556649874716,
    "disconnectReason": "CONNECTION_LOST"
}
```

O dispositivo `"mything2"` conectado (`true`) no horário POSIX `1556649855046`:

```
"connectivity": {
    "connected":true,
    "timestamp":1556649855046
}
```

Os registros de data/hora são expressos em milissegundos desde o epoch, portanto, `1556649855046` representa 18:44:15.046 na terça-feira, 30 de abril de 2019 (UTC).

> **Important**
>
> Se um dispositivo foi desconectado por aproximadamente uma hora, o valor de `"timestamp"` e o valor de `"disconnectReason"` do status de conectividade poderá ser ausente.

Restrições e limitações

Estas são as restrições e limitações para `AWS_Things`.

Campos de sombra com tipos complexos

Um campo de sombra será indexado somente se o valor do campo for um tipo simples, como um objeto JSON que não contém uma matriz ou uma matriz que consiste inteiramente em tipos simples. Tipo simples é uma string, um número ou um dos literais `true` ou `false`. Por exemplo, dado estado de sombra a seguir, o valor do campo `"palette"` não é indexado porque é uma matriz que contém itens de tipos complexos. O valor de campo `"colors"` é indexado, pois cada valor na matriz é uma string.

```
{
```

```
    "state": {
        "reported": {
            "switched": "ON",
            "colors": [ "RED", "GREEN", "BLUE" ],
            "palette": [
                {
                    "name": "RED",
                    "intensity": 124
                },
                {
                    "name": "GREEN",
                    "intensity": 68
                },
                {
                    "name": "BLUE",
                    "intensity": 201
                }
            ]
        }
    }
}
```

Nomes de campo de sombra aninhados

Os nomes de campos de sombra aninhados são armazenados como uma string delimitada por ponto (.). Por exemplo, considerando um documento de sombra:

```
{
  "state": {
    "desired": {
      "one": {
        "two": {
          "three": "v2"
        }
      }
    }
  }
}
```

O nome do campo `three` é armazenado como `desired.one.two.three`. Se você também tiver um documento de sombra, ele é armazenado da seguinte maneira:

```
{
```

```
  "state": {
    "desired": {
      "one.two.three": "v2"
    }
  }
}
```

Ambos correspondem a uma consulta para `shadow.desired.one.two.three:v2`. Como prática recomendada, não use pontos em nomes de campos de sombra.

Metadados de sombra

Um campo de uma seção de metadados de sombra é indexado, mas somente se o campo correspondente na seção `"state"` da sombra for indexado. (No exemplo anterior, o campo `"palette"` na seção de metadados da sombra também não é indexado.)

Dispositivos não registrados

A indexação da frota indexa o status de conectividade de um dispositivo cuja conexão `clientId` é a mesma de `thingName` de um objeto registrada no Registro.

Sombras não registradas

Se você costuma UpdateThingShadowcriar uma sombra usando um nome de coisa que não foi registrado na sua AWS IoT conta, os campos nessa sombra não são indexados. Isso se aplica tanto à sombra clássica sem nome quanto à sombra nomeada.

Valores numéricos

Se qualquer registro ou dado de sombra for reconhecido pelo serviço como um valor numérico, ele será indexado dessa maneira. Você pode realizar consultas envolvendo intervalos e operadores de comparação de valores numéricos, (por exemplo `"attribute.foo<5"` ou `"shadow.reported.foo:[75 TO 80]"`). Para ser reconhecido como numérico, o valor dos dados deve ser um número JSON do tipo literal e válido. O valor pode ser um número inteiro no intervalo -2^53...2^53-1, um ponto flutuante de precisão dupla com notação exponencial opcional ou parte de uma matriz que contém apenas esses valores.

Valores nulos

Valores nulos não são indexados.

Valores máximos

O número máximo de campos personalizados para consultas de agregação é cinco.

O número máximo de percentis solicitados para consultas de agregação é 100.

Autorização

Você pode especificar o índice de coisas como um Amazon Resource Name (ARN) em uma ação AWS IoT política, da seguinte forma.

Ação	Recurso
`iot:SearchIndex`	O ARN de um índice (por exemplo, `arn:aws:iot: your-aws-region your-aws-account :index/AWS_Things`).
`iot:DescribeIndex`	O ARN de um índice (por exemplo, `arn:aws:iot: your-aws-region :index/AWS_Things`).

> **Note**
>
> Se você tiver permissões para consultar o índice de frota, você poderá acessar os dados das objetos em toda a frota.

Gerenciamento da indexação de grupos de objetos

`AWS_ThingGroups` é o índice que contém todos os seus grupos de objetos. Você pode usar esse índice para pesquisar grupos com base no nome do grupo, na descrição, nos atributos e em todos os nomes de grupo pai.

Habilitar a indexação de grupos de objetos

Você pode usar a `thing-group-indexing-configuration` configuração na UpdateIndexingConfigurationAPI para criar o `AWS_ThingGroups` índice e controlar sua configuração. Você pode usar a GetIndexingConfigurationAPI para recuperar a configuração de indexação atual.

Use o comando update-indexing-configuration da CLI para atualizar as configurações da indexação de grupos de objetos:

```
aws iot update-indexing-configuration --thing-group-indexing-configuration
 thingGroupIndexingMode=ON
```

Você também pode atualizar configurações para a indexação de objetos e grupos de objetos em um único comando, como mostrado a seguir:

```
aws iot update-indexing-configuration --thing-indexing-configuration
 thingIndexingMode=REGISTRY --thing-group-indexing-configuration
 thingGroupIndexingMode=ON
```

Estes são valores válidos para `thingGroupIndexingMode`.

DESL.

Sem indexing/delete índice.

ON

Criar ou configurar o índice `AWS_ThingGroups`.

Para recuperar as configurações atuais de indexação de objetos e grupos de objetos, execute o comando get-indexing-configuration da CLI:

```
aws iot get-indexing-configuration
```

A resposta do comando é semelhante a:

```
{
   "thingGroupIndexingConfiguration": {
        "thingGroupIndexingMode": "ON"
    }
}
```

Descrever índices de grupos

Para recuperar o status atual do índice `AWS_ThingGroups`, use o comando describe-index da CLI:

```
aws iot describe-index --index-name "AWS_ThingGroups"
```

A resposta do comando é semelhante a:

```
{
   "indexStatus": "ACTIVE",
   "indexName": "AWS_ThingGroups",
   "schema": "THING_GROUPS"
}
```

AWS IoT cria seu índice na primeira vez que você indexa. Você não pode consultar o índice se o `indexStatus` é `BUILDING`.

Consultar um índice de grupos de objetos

Para consultar dados no índice, use o comando search-index da CLI:

```
aws iot search-index --index-name "AWS_ThingGroups" --query-string
 "thingGroupName:mythinggroup*"
```

Autorização

Você pode especificar o índice de grupos de coisas como um ARN de recurso em uma ação de AWS IoT política, da seguinte maneira.

Ação	Recurso
`iot:SearchIndex`	O ARN de um índice (por exemplo, `arn:aws:iot:` *`your-aws-region`* `:index/AWS_ThingGroups`).
`iot:DescribeIndex`	O ARN de um índice (por exemplo, `arn:aws:iot:` *`your-aws-region`* `:index/AWS_ThingGroups`).

Consultas do status de conectividade do dispositivo

AWS IoT A indexação de frota oferece suporte à consulta de conectividade de dispositivos individuais, permitindo que você recupere com eficiência o status da conectividade e os metadados relacionados para dispositivos específicos. Esse recurso complementa os recursos existentes de indexação e consulta em toda a frota.

Como funciona

O suporte à consulta de conectividade de dispositivos pode ser usado para recuperação otimizada do status de conectividade de um único dispositivo. Essa API fornece acesso de baixa latência e alto throughput às informações mais recentes de conectividade específicas do dispositivo. Depois de habilitar a indexação de conectividade, você terá acesso a essa API de consulta, que será cobrada como consultas padrão. Para acessar mais informações, consulte Preços do AWS IoT Device Management.

Recursos

Com o suporte à consulta de conectividade de dispositivos, você pode:

1. Consultar o estado atual de conectividade (conectado ou desconectado) de determinado dispositivo usando o respectivo `thingName`.
2. Recuperar metadados adicionais de conectividade, incluindo:
   a. Motivo da desconexão
   b. Carimbos de data/hora do evento de conexão ou desconexão mais recente.

> **Note**
>
> A indexação da frota indexa o status de conectividade de um dispositivo cuja conexão `clientId` é a mesma de `thingName` de um objeto registrada no Registro.

Benefícios

1. Baixa latência: reflete o estado mais recente de conectividade do dispositivo e oferece baixa latência para refletir as alterações do estado de conexão do IoT Core. O IoT Core determina que um dispositivo está desconectado assim que recebe uma solicitação de desconexão dele ou no caso de um dispositivo se desconectar sem enviar uma solicitação de desconexão. O IoT Core aguardará 1,5x do tempo de keep-alive configurado antes de decidir desconectar o cliente. A API de status da conectividade exibirá essas alterações normalmente em menos de um segundo após o IoT Core determinar a mudança de estado conectado de um dispositivo.
2. Alto throughput: aceita 350 transações por segundo (TPS) por padrão e pode ser ajustável para um valor maior mediante solicitação.

3. Retenção de dados: armazena dados de eventos indefinidamente quando o ConnectivityIndexing modo Fleet Indexing (FI) está ativado e o item não é excluído. Se você desabilitar a indexação de conectividade, os registros não serão retidos.

> **Note**
>
> Se a indexação do status de conectividade tiver sido habilitada antes do lançamento dessa API, o Fleet Indexing começará a rastrear as alterações do status de conectividade após o lançamento da API e exibirá o status atualizado com base nessas alterações.

Pré-requisitos

Para usar o suporte à consulta de conectividade do dispositivo:

1. Configurar uma AWS conta
2. Integre e registre dispositivos AWS IoT Core na sua região preferida
3. Habilite o Fleet Indexing com indexação de conectividade.

> **Note**
>
> Nenhuma configuração adicional será necessária se você já tiver a indexação de conectividade habilitada.

Para receber instruções detalhadas de configuração, consulte o Guia do desenvolvedor da AWS IoT.

Exemplos

```
aws iot get-thing-connectivity-data --thing-name myThingName
```

```
{
   "connected": true,
   "disconnectReason": "NONE",
   "thingName": "myThingName",
```

```
   "timestamp": "2024-12-19T10:00:00.000000-08:00"
}
```

- `thingName`: o nome do dispositivo conforme indicado na solicitação. Isso também corresponde ao clientId usado para se conectar ao AWS IoT Core.
- `disconnectReason`: motivo da desconexão. Será NONE para um dispositivo conectado.
- `connected`: o valor booliano verdadeiro indicando que esse dispositivo está conectado no momento.
- `timestamp`: o carimbo de data/hora que representa a desconexão mais recente do dispositivo em milissegundos.

```
aws iot get-thing-connectivity-data --thing-name myThingName
```

```
{
   "connected": false,
   "disconnectReason": "CLIENT_INITIATED_DISCONNECT",
   "thingName": "myThingName",
   "timestamp": "2024-12-19T10:30:00.000000-08:00"
}
```

- `thingName`: o nome do dispositivo conforme indicado na solicitação. Isso também corresponde ao clientId usado para se conectar ao AWS IoT Core.
- `disconnectReason`: O motivo da desconexão é CLIENT_INITIATED_DISCONNECT indicando que o cliente indicou que ele se desconectaria. AWS IoT Core
- `connected`: o valor booliano falso indicando que esse dispositivo está desconectado no momento.
- `timestamp`: o carimbo de data/hora que representa a desconexão mais recente do dispositivo em milissegundos.

```
aws iot get-thing-connectivity-data --thing-name neverConnectedThing
```

```
{
   "connected": false,
```

```
   "disconnectReason": "UNKNOWN",
   "thingName": "neverConnectedThing"
}
```

- `thingName`: o nome do dispositivo conforme indicado na solicitação. Isso também corresponde ao clientId usado para se conectar ao AWS IoT Core.
- `disconnectReason`: motivo da desconexão. Será “UNKNOWN” para um dispositivo que nunca foi conectado ou para o qual o Fleet Indexing não tenha o último motivo de desconexão armazenado.
- `connected`: o valor booliano falso indicando que esse dispositivo está desconectado no momento.
- `timestamp`: o carimbo de data/hora não é exibido para um dispositivo que nunca foi conectado ou para o qual o Fleet Indexing não tem o último carimbo de data/hora armazenado.

Consulta de dados agregados

AWS IoT fornece quatro APIs (`GetStatistics`,`GetCardinality`,`GetPercentiles`, e`GetBucketsAggregation`) que permitem pesquisar dados agregados em sua frota de dispositivos.

> **Note**
>
> Para problemas com valores ausentes ou inesperados para a agregação APIs, leia o guia de solução de problemas de indexação de frotas.

GetStatistics

A GetStatisticsAPI e o comando get-statistics CLI retornam a contagem, a média, a soma, o mínimo, o máximo, a soma dos quadrados, a variância e o desvio padrão para o campo agregado especificado.

O comando da CLI get-statistics usa os seguintes parâmetros:

`index-name`

O nome do índice a ser pesquisado. O valor padrão é `AWS_Things`.

`query-string`

A consulta usada para pesquisar o índice. Você pode especificar `"*"` para obter a contagem de todas as coisas indexadas em seu Conta da AWS.

`aggregationField`

(Opcional) O campo a ser agregado. Esse campo deve ser um campo gerenciado ou personalizado definido ao chamar update-indexing-configuration. Se você não especificar um campo de agregação, `registry.version` será usado como o campo de agregação.

`query-version`

A versão da consulta a ser usada. O valor padrão é `2017-09-30`.

O tipo de campo de agregação pode afetar as estatísticas retornadas.

GetStatistics com valores de string

Se você agregar em um campo de string, chamar `GetStatistics` retornará uma contagem de dispositivos que têm atributos que correspondem à consulta. Por exemplo:

```
aws iot get-statistics --aggregation-field 'attributes.stringAttribute'
            --query-string '*'
```

Esse comando retorna o número de dispositivos que contêm um atributo chamado `stringAttribute`:

```
{
  "statistics": {
    "count": 3
  }
}
```

GetStatistics com valores booleanos

Quando você chama `GetStatistics` com um campo de agregação booliano:

- AVERAGE é a porcentagem de dispositivos que correspondem à consulta.
- MINIMUM é 0 ou 1 conforme as seguintes regras:

  - Se todos os valores do campo de agregação forem `false`, MINIMUM será 0.
  - Se todos os valores do campo de agregação forem `true`, MINIMUM será 1.
  - Se os valores do campo de agregação forem uma mistura de `false` e `true`, MINIMUM será 0.
- MAXIMUM é 0 ou 1 conforme as seguintes regras:
  - Se todos os valores do campo de agregação forem `false`, MAXIMUM será 0.
  - Se todos os valores do campo de agregação forem `true`, MAXIMUM será 1.
  - Se os valores do campo de agregação forem uma mistura de `false` e `true`, MAXIMUM será 1.
- SUM é a soma do equivalente inteiro dos valores boolianos.
- COUNT é a contagem de objetos que correspondem aos critérios da string de consulta e contêm um valor de campo de agregação válido.

GetStatistics com valores numéricos

Quando você chama `GetStatistics` e especifica um campo de agregação do tipo `Number`, `GetStatistics` retorna os seguintes valores:

contagem

A contagem de objetos que correspondem aos critérios da string de consulta e contêm um valor de campo de agregação válido.

média

A média dos valores numéricos que correspondem à consulta.

soma

A soma dos valores numéricos que correspondem à consulta.

mínimo

O menor dos valores numéricos que correspondem à consulta.

máximo

O maior dos valores numéricos que correspondem à consulta.

sumOfSquares

A soma dos quadrados dos valores numéricos que correspondem à consulta.

variância

A variação dos valores numéricos que correspondem à consulta. A variância de um conjunto de valores é a média dos quadrados das diferenças de cada valor em relação ao valor médio do conjunto.

stdDeviation

O desvio padrão dos valores numéricos que correspondem à consulta. O desvio padrão de um conjunto de valores é uma medida de como os valores estão distribuídos.

O exemplo a seguir mostra como chamar get-statistics com um campo personalizado numérico.

```
aws iot get-statistics --aggregation-field 'attributes.numericAttribute2'
            --query-string '*'
```

```
{
  "statistics": {
    "count": 3,
    "average": 33.333333333333336,
    "sum": 100.0,
    "minimum": -125.0,
    "maximum": 150.0,
    "sumOfSquares": 43750.0,
    "variance": 13472.22222222222,
    "stdDeviation": 116.06990230986766
  }
}
```

Para campos de agregação numéricos, se os valores de campo excederem o valor duplo máximo, os valores estatísticos estarão vazios.

GetCardinality

A GetCardinalityAPI e o comando get-cardinality CLI retornam a contagem aproximada de valores exclusivos que correspondem à consulta. Por exemplo, você pode querer encontrar o número de dispositivos com níveis de bateria inferiores a 50%:

```
aws iot get-cardinality --index-name AWS_Things --query-string "batterylevel
          > 50" --aggregation-field "shadow.reported.batterylevel"
```

Este comando retorna o número de itens com níveis de bateria superiores a 50%:

```
{
    "cardinality": 100
}
```

`cardinality` é sempre retornado por get-cardinality mesmo se não houver campos correspondentes. Por exemplo:

```
aws iot get-cardinality --query-string "thingName:Non-existent*"
          --aggregation-field "attributes.customField_STR"
```

```
{
    "cardinality": 0
}
```

O comando da CLI get-cardinality usa os seguintes parâmetros:

`index-name`

O nome do índice a ser pesquisado. O valor padrão é `AWS_Things`.

`query-string`

A consulta usada para pesquisar o índice. Você pode especificar `"*"` para obter a contagem de todas as coisas indexadas em seu Conta da AWS.

`aggregationField`

O campo a ser agregado.

`query-version`

A versão da consulta a ser usada. O valor padrão é `2017-09-30`.

GetPercentiles

A GetPercentilesAPI e o comando get-percentiles CLI agrupam os valores agregados que correspondem à consulta em agrupamentos de percentis. Os agrupamentos de percentil padrão são: 1, 5, 25, 50, 75, 95, 99, embora você possa especificar o seu próprio quando chamar

`GetPercentiles`. Esta função retorna um valor para cada grupo de percentis especificado (ou os agrupamentos de percentil padrão). O grupo de percentis “1” contém o valor de campo agregado que ocorre em aproximadamente um por cento dos valores que correspondem à consulta. O grupo de percentil “5” contém o valor de campo agregado que ocorre em aproximadamente cinco por cento dos valores que correspondem à consulta, e assim por diante. O resultado é uma aproximação. Quanto mais valores correspondem à consulta, mais precisos os valores do percentil.

O exemplo a seguir mostra como chamar o comando get-percentiles da CLI.

```
aws iot get-percentiles --query-string "thingName:*" --aggregation-field
          "attributes.customField_NUM" --percents 10 20 30 40 50 60 70 80 90 99
```


```
{
    "percentiles": [
        {
            "value": 3.0,
            "percent": 80.0
        },
        {
            "value": 2.5999999999999996,
            "percent": 70.0
        },
        {
            "value": 3.0,
            "percent": 90.0
        },
        {
            "value": 2.0,
            "percent": 50.0
        },
        {
            "value": 2.0,
            "percent": 60.0
        },
        {
            "value": 1.0,
            "percent": 10.0
        },
        {
            "value": 2.0,
            "percent": 40.0
        },
```

```
        {
            "value": 1.0,
            "percent": 20.0
        },
        {
            "value": 1.4,
            "percent": 30.0
        },
        {
            "value": 3.0,
            "percent": 99.0
        }
    ]
}
```

O comando a seguir mostra a saída retornada de get-percentiles quando não há documentos correspondentes.

```
aws iot get-percentiles --query-string "thingName:Non-existent*"
          --aggregation-field "attributes.customField_NUM"
```

```
{
    "percentiles": []
}
```

O comando da CLI get-percentile usa os seguintes parâmetros:

`index-name`

O nome do índice a ser pesquisado. O valor padrão é `AWS_Things`.

`query-string`

A consulta usada para pesquisar o índice. Você pode especificar `"*"` para obter a contagem de todas as coisas indexadas em seu Conta da AWS.

`aggregationField`

O campo a ser agregado, que deve ser do tipo `Number`.

`query-version`

A versão da consulta a ser usada. O valor padrão é `2017-09-30`.

`percents`

(Opcional) você pode usar esse parâmetro para especificar agrupamentos de percentil personalizados.

GetBucketsAggregation

A GetBucketsAggregationAPI e o comando get-buckets-aggregation CLI retornam uma lista de buckets e o número total de itens que se encaixam nos critérios da sequência de caracteres de consulta.

O exemplo a seguir mostra como chamar o comando get-buckets-aggregation da CLI.

```
aws iot get-buckets-aggregation --query-string '*' --index-name AWS_Things --
aggregation-field 'shadow.reported.batterylevelpercent' --buckets-aggregation-type
 'termsAggregation={maxBuckets=5}'
```

O comando retorna os seguintes:

```
{
    "totalCount": 20,
    "buckets": [
        {
            "keyValue": "100",
            "count": 12
        },
        {
            "keyValue": "90",
            "count": 5
        },
        {
            "keyValue": "75",
            "count": 3
        }
    ]
}
```

O comando da CLI get-buckets-aggregation usa os seguintes parâmetros:

`index-name`

O nome do índice a ser pesquisado. O valor padrão é `AWS_Things`.

`query-string`

A consulta usada para pesquisar o índice. Você pode especificar "*" para obter a contagem de todas as coisas indexadas em seu Conta da AWS.

`aggregation-field`

O campo a ser agregado.

`buckets-aggregation-type`

O controle básico da forma de resposta e do tipo de agregação do bucket a ser executado.

Autorização

Você pode especificar o índice de grupos de coisas como um ARN de recurso em uma ação de AWS IoT política, da seguinte maneira.

Ação	Recurso
`iot:GetStatistics`	O ARN de um índice (por exemplo, `arn:aws:iot:` *`your-aws-region`* `:index/AWS_Things` ou `arn:aws:iot:` *`your-aws-region`* `:index/AWS_ThingGroups`).

Sintaxe de consulta

Na indexação de frotas, uma sintaxe de consulta é utilizada para especificar consultas.

Recursos compatíveis

A sintaxe de consulta dá suporte aos seguintes atributos:

- Termos e frases
- Pesquisar campos
- Pesquisa de prefixo
- Pesquisa de intervalo

- Operadores boolianos `AND`, `OR`, `NOT` e `-`. O hífen é usado para excluir algo dos resultados de pesquisa (por exemplo, `thingName:(tv* AND -plasma)`).
- Agrupamento
- Agrupamento de campos
- Escape de caracteres especiais (tal como \)
- O principal uso de curingas é limitado a 1 termo por consulta. Por exemplo, não é possível procurar `thingName:*my` e `thingGroupNames:*echo` na mesma consulta. As consultas que incluem um caractere curinga inicial têm um comprimento máximo de consulta de cem caracteres.

> **Note**
>
> Contas com um limite de API de consulta do Fleet Indexing superior a 30 solicitações por segundo estão sujeitas à limitação ao usar os principais termos de consulta curinga. Esse requests-per-second limite de 30 se aplica no nível da conta em todas as consultas APIs de indexação de frotas.

Atributos não compatíveis

A sintaxe de consulta é compatível com os seguintes atributos:

- Expressões regulares
- Aumento
- Classificação
- Pesquisas difusas
- Pesquisa de proximidade
- Classificação
- Agregação
- Caracteres especiais: `` ` ``, `@`, `#`, `%`, `\`, `/`, `'`, `;`, e `,`. Observe que `,` só tem suporte em consultas geográficas.

Observações

Alguns comentários sobre a linguagem de consulta:

- O operador padrão é AND. Uma consulta de `"thingName:abc thingType:xyz"` é equivalente a `"thingName:abc AND thingType:xyz"`.
- Se um campo não for especificado, AWS IoT pesquisa o termo em todos os campos Registry, Device Shadow e Device Defender.
- Todos os nomes de campos diferenciam maiúsculas de minúsculas.
- A pesquisa não diferencia maiúsculas de minúsculas. As palavras são separadas por caracteres de espaço em branco, conforme definido pelo `Character.isWhitespace(int)` de Java.
- A indexação de dados de sombra do dispositivo (sombras sem nome e sombras nomeadas) inclui as seções relatadas, desejadas, delta e metadados.
- As versões da sombra do dispositivo e registro não são pesquisáveis, mas estão presentes na resposta.
- O número máximo de termos em uma consulta é doze.
- O caractere especial `,` só tem suporte em consultas geográficas.

Exemplo de consultas de objetos

Especifique consultas em uma string de consulta usando uma sintaxe de consulta. As consultas são passadas para a API `SearchIndex`. A tabela a seguir lista alguns exemplos de sequências de consulta.

String de consulta	Resultado
`abc`	Consulta “abc” em qualquer registro, sombra (sombra clássica sem nome e sombra nomeada) ou campo de violações do Device Defender.
`thingName:myThingName`	Consultas sobre algo com o nome "myThingName”.
`thingName:my*`	Consulto objetos com nomes que começam com "my".
`thingName:*my`	Consultas de coisas com nomes que terminam com "my". O uso de curingas à esquerda é limitado a um termo por consulta. As consultas que incluem um caractere curinga à esquerda têm um comprimento máximo de consulta de cem caracteres.

String de consulta	Resultado
`thingName:*my*`	Consultas de coisas que contêm a substring “my”. O uso de curingas à esquerda é limitado a um termo por consulta. As consultas que incluem um caractere curinga à esquerda têm um comprimento máximo de consulta de cem caracteres.
`thingName:ab?`	Consulto objetos com nomes que possuam "ab" e um caractere adicional (por exemplo: "aba", "abb", "abc" e assim por diante).
`thingTypeName:aa`	Consulto objetos que estejam associados ao tipo "aa".
`thingGroupNames:a`	Consulta objetos com um grupo de objetos principal ou um grupo de cobrança chamado "a".
`thingGroupNames:a*`	Consulta objetos com um nome de grupo principal ou nome de grupo de cobrança que corresponda ao padrão "a*".
`attributes.myAttribute:75`	Consulto objetos com um atributo chamado "myAttribute" que tem o valor 75.
`attributes.myAttribute: [75 TO 80]`	Consulto objetos com um atributo chamado "myAttribute" cujo valor esteja dentro de um intervalo numérico (75 a 80, inclusive).
`attributes.myAttribute: {75 TO 80]`	Consulto objetos com um atributo chamado "myAttrib ute" cujo valor esteja dentro do intervalo numérico (>75 e <=80).
`attributes.serialNumber: ["abcd" TO "abcf"]`	Consulto objetos com um atributo chamado "serialNu mber" cujo valor esteja dentro de um intervalo de sequência alfanumérica. Essa consulta retornará objetos com um atributo "serialNumber" com valores "abcd", "abce" ou "abcf".

String de consulta	Resultado
`attributes.myAttribute:i*t`	Consulto objetos com um atributo chamado "myAttribute" em que o valor seja 'i', seguido por qualquer número de caracteres, seguido por 't'.
`attributes.attr1:abc AND attributes.attr2<5 NOT attributes.attr3>10`	Consulto objetos que combinam termos usando expressões boolianas. Essa consulta retorno objetos que tenham um atributo nomeado "attr1" com um valor "abc", um atributo chamado "attr2" que seja menor que 5 e um atributo chamado "attr3" que não seja maior que 10.
`shadow.hasDelta:true`	Consulto objetos com uma sombra sem nome que possui um elemento delta.
`NOT attributes.model:legacy`	Consulta de objetos em que o atributo denominado "modelo" não é "legado".
`shadow.reported.stats.battery:{70 TO 100} (v2 OR v3) NOT attributes.model:legacy`	Consulto objetos com o seguinte: • O atributo `stats.battery` de sombra do objeto tem um valor entre 70 e 100. • O texto "v2" ou "v3" ocorre em um nome de objeto, nome do tipo ou valores do atributo. • O atributo `model` do objeto não é definido como "legado".
`shadow.reported.myvalues:2`	Consulto objetos onde a matriz `myvalues` na seção relatada da sombra contém um valor igual a 2.
`shadow.reported.location:* NOT shadow.desired.stats.battery:*`	Consulto objetos com o seguinte: • O atributo `location` existe na seção `reported` da sombra. • O atributo `stats.battery` não existe na seção `desired` da sombra.

String de consulta	Resultado
`shadow.name.<shadowName>.hasDelta:true`	Consulto objetos que tenham uma sombra com o nome fornecido e também um elemento delta.
`shadow.name.<shadowName>.desired.filament:*`	Consulto objetos que tenham uma sombra com o nome fornecido e também uma propriedade de filamento desejada.
`shadow.name.<shadowName>.reported.location:*`	Consulto objetos que tenham uma sombra com o nome fornecido e onde o atributo `location` exista na seção relatada da sombra nomeada.
`connectivity.connected:true`	Consulta todos os dispositivos conectados.
`connectivity.connected:false`	Consulta todos os dispositivos desconectados.
`connectivity.connected:true AND connectivity.timestamp : [1557651600000 TO 1557867600000]`	Consulta todos os dispositivos conectados com um registro de data/hora de conexão >= 1557651600000 e <= 1557867600000. Os registros de data/hora são expressos em milissegundos desde o epoch.
`connectivity.connected:false AND connectivity.timestamp : [1557651600000 TO 1557867600000]`	Consulta todos os dispositivos desconectados com um registro de data/hora de desconexão >= 1557651600000 e <= 1557867600000. Os registros de data/hora são expressos em milissegundos desde o epoch.
`connectivity.connected:true AND connectivity.timestamp > 1557651600000`	Consulta todos os dispositivos conectados com um registro de data/hora de conexão > 1557651600000. Os registros de data/hora são expressos em milissegundos desde o epoch.
`connectivity.connected:*`	Consulta todos os dispositivos com informações de conectividade presentes.
`connectivity.disconnectReason:*`	Consulta todos os dispositivos com disconnectReason de conectividade presente.

String de consulta	Resultado
`connectivity.disconnectReason:CLIENT_INITIATED_DISCONNECT`	Consulta todos os dispositivos desconectados devido a CLIENT_INITIATED_DISCONNECT.
`deviceDefender.violationCount:[0 TO 100]`	Consulto objetos com valor de contagem de violações do Device Defender que estejam dentro do intervalo numérico (0 a 100, inclusive).
`deviceDefender.<device-SecurityProfile>.disconnectBehavior.inViolation:true`	Consulto objetos que estejam em violação do comportamento `disconnectBehavior`, conforme definido no perfil de segurança `device-SecurityProfile`. Observe que inViolation:false não é uma consulta válida.
`deviceDefender.<device-SecurityProfile>.disconnectBehavior.lastViolationValue.number>2`	Consultas sobre itens que violam o comportamento `disconnectBehavior` definido no dispositivo do perfil de segurança, SecurityProfile com um valor do último evento de violação maior que 2.
`deviceDefender.<device-SecurityProfile>.disconnectBehavior.lastViolationTime>1634227200000`	Consultas sobre itens que violam o comportamento `disconnectBehavior` definido no dispositivo do perfil de segurança, SecurityProfile com um último evento de violação após um período especificado.
`shadow.name.gps-tracker.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km`	Consultas sobre coisas que estão dentro da distância radial de 15,5 km das coordenadas de 47.6204,-122.3491. Essa sequência de caracteres de consulta se aplica a quando seus dados de localização são armazenados em uma sombra nomeada.
`shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km`	Consultas sobre coisas que estão dentro da distância radial de 15,5 km das coordenadas de 47.6204,-122.3491. Essa sequência de caracteres de consulta se aplica a quando seus dados de localização são armazenados em uma sombra clássica.

Exemplo de consultas de grupos de objetos

As consultas são especificadas em uma string de consulta usando uma sintaxe de consulta e passadas para a API `SearchIndex`. A tabela a seguir lista alguns exemplos de sequências de consulta.

String de consulta	Resultado
`abc`	Consulta "abc" em qualquer campo.
`thingGroupName:myGroupThingName`	Consultas para um grupo de coisas com o nome "myGroupThingNome”.
`thingGroupName:my*`	Consulta grupos de objetos com nomes que começam com "my".
`thingGroupName:ab?`	Consulta grupos de objetos com nomes que têm "ab" e um caractere adicional (por exemplo: "aba", "abb", "abc" e assim por diante).
`attributes.myAttribute:75`	Consulta grupos de objetos com um atributo chamado "myAttribute" que tem o valor 75.
`attributes.myAttribute:[75 TO 80]`	Consulta grupos de objetos com um atributo chamado "myAttribute" cujo valor está dentro de um intervalo numérico (75 a 80, inclusive).
`attributes.myAttribute:[75 TO 80]`	Consulta grupos de objetos com um atributo chamado "myAttribute" cujo valor está dentro do intervalo numérico (>75 e <=80).
`attributes.myAttribute:["abcd" TO "abcf"]`	Consulta grupos de objetos com um atributo chamado "myAttribute" cujo valor está dentro de um intervalo de sequência alfanumérica. Essa consulta retornará grupos de objetos com um atributo "serialNumber" com valores "abcd", "abce" ou "abcf".

String de consulta	Resultado
`attributes.myAttribute:i*t`	Consulta grupos de objetos com um atributo chamado "myAttribute" cujo valor é 'i', seguido por qualquer número de caracteres, seguido por 't'.
`attributes.attr1:abc AND attributes.attr2<5 NOT attributes.attr3>10`	Consulta grupos de objetos que combinam termos usando expressões boolianas. Essa consulta retorna grupos de objetos que tenham um atributo chamado "attr1" com um valor "abc", um atributo chamado "attr2" que seja menor que 5 e um atributo chamado "attr3" que não seja maior que 10.
`NOT attributes.myAttribute:cde`	Consulta grupos de objetos onde o atributo chamado "myAttribute" não é "cde".
`parentGroupNames:(myParentThingGroupName)`	Consultas para grupos de coisas cujo nome do grupo principal corresponda a "myParentThingGroupName”.
`parentGroupNames:(myParentThingGroupName OR myRootThingGroupName)`	Consultas para grupos de coisas cujo nome do grupo principal corresponda a "myParentThingGroupName" ou "myRootThingGroupName”.
`parentGroupNames:(myParentThingGroupNa*)`	Consultas para grupos de coisas cujo nome do grupo principal começa com "myParentThingGroupNa”.

Indexação de dados de localização

Você pode usar a indexação de frotas da AWS IoT para indexar os últimos dados de localização enviados dos seus dispositivos e pesquisar dispositivos usando consultas geográficas. Esse atributo resolve casos de uso de monitoramento e gerenciamento de dispositivos, como rastreamento de localização e pesquisa de proximidade. A indexação de localização funciona de maneira semelhante a outros recursos de indexação de frotas e com configurações adicionais para especificar na indexação de objetos.

Os casos de uso comuns incluem: pesquisar e agregar dispositivos localizados dentro dos limites geográficos desejados, obter informações específicas da localização usando termos de consulta relacionados aos metadados e ao estado do dispositivo de fontes de dados indexadas, fornecer

uma visão granular, como a filtragem de resultados para uma área geográfica específica, reduzir os atrasos de renderização nos mapas de monitoramento da frota e rastrear a localização do último dispositivo relatado, identificar dispositivos que estão fora dos limites desejados e gerar alarmes usando métricas de frota. Para começar com indexação de localização e geoconsultas, consulte ???.

Formatos de dados compatíveis

AWS IoT a indexação de frotas suporta os seguintes formatos de dados de localização:

1. Representação de texto bem conhecida de sistemas de referência de coordenadas

   Uma string que segue o formato Informações geográficas: representação de texto bem conhecida de sistemas de referência de coordenadas. Um exemplo pode ser `"POINT(long lat)"`.
2. Uma string que representa as coordenadas

   Uma string com o formato `"latitude, longitude"` ou `"longitude, latitude"`. Se você usar `"longitude, latitude"`, também deverá especificar `order` em `geoLocations`. Um exemplo pode ser `"41.12,-71.34"`.
3. Um objeto com chaves lat(latitude), lon(longitude)

   Esse formato se aplica à sombra clássica e à sombra nomeada. Chaves compatíveis: `lat`, `latitude`, `lon`, `long`, `longitude`. Um exemplo pode ser `{"lat": 41.12, "lon": -71.34}`.
4. Uma matriz que representa as coordenadas

   Uma matriz com o formato `[lat,lon]` ou`[lon,lat]`. Se você usar o formato `[lon,lat]`, que é o mesmo das coordenadas em GeoJSON (aplicável à sombra clássica e à sombra nomeada), você também deve especificar `order` em `geoLocations`.

   Um exemplo pode ser:

```
{
  "location": {
    "coordinates": [
      **Longitude**,
      **Latitude**
    ],
    "type": "Point",
    "properties": {
```

```
      "country": "United States",
      "city": "New York",
      "postalCode": "*****",
      "horizontalAccuracy": 20,
      "horizontalConfidenceLevel": 0.67,
      "state": "New York",
      "timestamp": "2023-01-04T20:59:13.024Z"
    }
  }
}
```

Como indexar dados de localização

As etapas a seguir mostram como atualizar a configuração de indexação dos dados de localização e usar consultas geográficas para pesquisar dispositivos.

1. Saiba onde seus dados de localização estão armazenados

   Atualmente, a indexação de frotas oferece suporte à indexação de dados de localização armazenados em sombras clássicas ou sombras nomeadas.

2. Usar formatos de dados de localização compatíveis

   Verifique se o formato dos dados de localização segue um dos formatos de dados compatíveis.

3. Atualizar configuração de indexação

   Se necessário, habilite a configuração de indexação de itens (registro). Você também deve ativar a indexação na sombra clássica ou na sombra nomeada que contenha seus dados de localização. Ao atualizar sua indexação de itens, você deve incluir seus dados de localização na configuração de indexação.

4. Criar e executar consultas geográficas

   Dependendo dos seus casos de uso, crie geoconsultas e execute-as para pesquisar dispositivos. A consulta geográfica que você compõe deve seguir a Sintaxe da consulta. É possível encontrar alguns exemplos em ???.

Atualizar a configuração da indexação de objeto

Para indexar dados de localização, atualize a configuração de indexação e inclua seus dados de localização. Dependendo de onde seus dados de localização estão, siga as etapas para atualizar sua configuração de indexação:

Dados de localização armazenados em sombras clássicas

Se seus dados de localização estiverem armazenados em uma sombra clássica, você deverá definir `thingIndexingMode` como `REGISTRY_AND_SHADOW` e especificar seus dados de localização nos campos `geoLocations` (`name` e `order`) em `filter`.

No exemplo de configuração de indexação a seguir, você especifica o caminho dos dados de localização `shadow.reported.coordinates` como `name` e `LonLat` como `order`.

```
{
 "thingIndexingMode": "REGISTRY_AND_SHADOW",
 "filter": {
  "geoLocations": [
   {
    "name": "shadow.reported.coordinates",
    "order": "LonLat"
   }
  ]
 }
}
```

- `thingIndexingMode`

  O modo de indexação controla se o Registro ou a sombra é indexado. Quando `thingIndexingMode` é definido como `OFF`, a indexação de itens é desativada.

  Para indexar dados de localização armazenados em uma sombra clássica, você deve definir `thingIndexingMode` como `REGISTRY_AND_SHADOW`. Para obter mais informações, consulte ???.
- `filter`

  O filtro de indexação fornece seleções adicionais para sombras nomeadas e dados de localização geográfica. Para obter mais informações, consulte ???.
- `geoLocations`

A lista de destinos de localização geográfica que você seleciona para indexar. O número máximo padrão de destinos de localização geográfica para indexação é 1. Para aumentar o limite, consulte Cotas de AWS IoT Device Management.

- `name`

  O nome do campo de destino de localização geográfica. Um exemplo de valor de `name` pode ser o caminho dos dados de localização de sua sombra: `shadow.reported.coordinates`.

- `order`

  A ordem do campo de destino de localização geográfica. Valores válidos: `LatLon` e `LonLat`. `LatLon` significa latitude e longitude. `LonLat` significa longitude e latitude. Esse campo é opcional. O valor padrão é `LatLon`.

Dados de localização armazenados em sombras nomeadas

Se seus dados de localização estiverem armazenados em uma sombra nomeada, defina `namedShadowIndexingMode` como `ON`, adicione seus nomes de sombra nomeados ao campo `namedShadowNames` em `filter` e especifique seu caminho de dados de localização no campo `geoLocations` em `filter`.

No exemplo de configuração de indexação a seguir, você especifica o caminho dos dados de localização `shadow.name.namedShadow1.reported.coordinates` como `name` e `LonLat` como `order`.

```
{
 "thingIndexingMode": "REGISTRY",
 "namedShadowIndexingMode": "ON",
 "filter": {
  "namedShadowNames": [
   "namedShadow1"
  ],
  "geoLocations": [
   {
    "name": "shadow.name.namedShadow1.reported.coordinates",
    "order": "LonLat"
   }
  ]
 }
}
```

- `thingIndexingMode`

  O modo de indexação controla se o Registro ou a sombra é indexado. Quando `thingIndexingMode` é definido como `OFF`, a indexação de itens é desativada.

  Para indexar dados de localização armazenados em uma sombra nomeada, você deve definir `thingIndexingMode` como `REGISTRY` (ou `REGISTRY_AND_SHADOW`). Para obter mais informações, consulte ???.

- `filter`

  O filtro de indexação fornece seleções adicionais para sombras nomeadas e dados de localização geográfica. Para obter mais informações, consulte ???.

- `geoLocations`

  A lista de destinos de localização geográfica que você seleciona para indexar. O número máximo padrão de destinos de localização geográfica para indexação é 1. Para aumentar o limite, consulte Cotas de AWS IoT Device Management.

- `name`

  O nome do campo de destino de localização geográfica. Um exemplo de valor de `name` pode ser o caminho dos dados de localização de sua sombra: `shadow.name.namedShadow1.reported.coordinates`.

- `order`

  A ordem do campo de destino de localização geográfica. Valores válidos: `LatLon` e `LonLat`. `LatLon` significa latitude e longitude. `LonLat` significa longitude e latitude. Esse campo é opcional. O valor padrão é `LatLon`.

Exemplo de consultas geográficas

Depois de concluir a configuração de indexação dos dados de localização, execute consultas geográficas para pesquisar dispositivos. Você também pode combinar suas consultas geográficas com outras cadeias de caracteres de consulta. Para obter mais informações, consulte ??? e ???.

Consulta de exemplo 1

Este exemplo pressupõe que os dados de localização estão armazenados em uma sombra nomeada `gps-tracker`. A saída desse comando é a lista de dispositivos que estão dentro da distância radial de 15,5 km do ponto central com coordenadas (47,6204,-122,3491).

```
aws iot search-index --query-string \
"shadow.name.gps-tracker.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

Consulta de exemplo 2

Este exemplo pressupõe que os dados de localização estão armazenados em uma sombra clássica. A saída desse comando é a lista de dispositivos que estão dentro da distância radial de 15,5 km do ponto central com coordenadas (47,6204,-122,3491).

```
aws iot search-index --query-string \
"shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

Consulta de exemplo 3

Este exemplo pressupõe que os dados de localização estão armazenados em uma sombra clássica. A saída desse comando é a lista de dispositivos que não estão conectados e estão fora da distância radial de 15,5 km do ponto central com coordenadas (47,6204,-122,3491).

```
aws iot search-index --query-string \
"connectivity.connected:false AND (NOT
 shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km)"
```

Tutorial de inicialização

Este tutorial demonstra como usar a indexação de frotas para indexar seus dados de localização. Para simplificar, você cria um objeto para representar seu dispositivo e armazena os dados de localização em uma sombra nomeada, atualiza a configuração de indexação do objeto para indexação de localização e executa exemplos de consultas geográficas para pesquisar dispositivos dentro de um limite radial.

Este tutorial leva cerca de 15 minutos para ser concluído.

Neste tópico:

- Pré-requisitos

- Criar objeto e sombra
- Atualizar a configuração da indexação de objeto
- Executar consulta geográfica

Pré-requisitos

- Instale a versão mais recente da AWS CLI.
- Familiarize-se com Indexação de localização e consultas geográficas, Gerenciar indexação de objeto e Sintaxe de consulta.

Criar objeto e sombra

Você cria um objeto para representar seu dispositivo e uma sombra nomeada para armazenar seus dados de localização (coordenadas 47,61564,-122,33584).

1. Execute o comando apresentado a seguir para criar o que representa a bicicleta com o nome Bike-1. Para obter mais informações sobre como criar algo usando AWS CLI, consulte create-thing from Reference. AWS CLI

   ```
   aws iot create-thing --thing-name "Bike-1" \
   --attribute-payload '{"attributes": {"model":"OEM-2302-12", "battery":"35",
    "acqDate":"06/09/23"}}'
   ```

   A saída desse comando pode ser semelhante à seguinte:

   ```
   {
       "thingName": "Bike-1",
       "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/Bike-1",
       "thingId": "df9cf01d-b0c8-48fe-a2e2-e16cff6b23df"
   }
   ```

2. Execute o comando a seguir para criar uma sombra nomeada para armazenar os dados de localização da Bike-1 (coordenadas 47.61564, -122.33584). Para obter mais informações sobre como criar uma sombra nomeada usando AWS CLI, consulte update-thing-shadowem AWS CLIReferência.

   ```
   aws iot-data update-thing-shadow \
   --thing-name Bike-1 \
   ```

```
--shadow-name Bike1-shadow \
--cli-binary-format raw-in-base64-out \
--payload '{"state":{"reported":{"coordinates":{"lat": 47.6153, "lon": -122.3333}}}}'
 \
"output.txt" \
```

Esse comando não retorna nenhuma saída. Para ver a sombra nomeada que você criou, você pode executar o comando list-named-shadows-for-thing CLI.

```
aws iot-data list-named-shadows-for-thing --thing-name Bike-1
```

A saída desse comando pode ser semelhante à seguinte:

```
{
    "results": [
        "Bike1-shadow"
    ],
    "timestamp": 1699574309
}
```

Atualizar a configuração da indexação de objeto

Para indexar dados de localização, atualize sua configuração de indexação para incluir os dados de localização. Como seus dados de localização estão armazenados em uma sombra nomeada neste tutorial, defina `thingIndexingMode` como `REGISTRY` (com um requisito mínimo), defina `namedShadowIndexingMode` como `ON` e adicione seus dados de localização à configuração. Neste exemplo, você deve adicionar o nome da sombra nomeada e o caminho dos dados de localização da sombra para `filter`.

1. Execute o comando para atualizar sua configuração de indexação para indexação de localização.

```
aws iot update-indexing-configuration --cli-input-json '{
"thingIndexingConfiguration": { "thingIndexingMode": "REGISTRY",
"thingConnectivityIndexingMode": "OFF",
"deviceDefenderIndexingMode": "OFF",
"namedShadowIndexingMode": "ON",
"filter": {
    "namedShadowNames": ["Bike1-shadow"],
    "geoLocations":[{
```

```
        "name":"shadow.name.Bike1-shadow.reported.coordinates"
    }]
},
"customFields": [
{ "name":"attributes.battery",
"type":"Number"}] } }'
```

O comando não produz saída. Talvez seja necessário aguardar um instante até a atualização ser concluída. Para verificar o status, execute o comando da CLI describe-index. Se você vê `indexStatus` mostrar: `ACTIVE`, sua atualização de indexação de objeto está concluída.

2. Execute o comando para verificar a configuração de indexação. Esta etapa é opcional.

```
aws iot get-indexing-configuration
```

A saída poderá ser parecida com o seguinte:

```
{
    "thingIndexingConfiguration": {
        "thingIndexingMode": "REGISTRY",
        "thingConnectivityIndexingMode": "OFF",
        "deviceDefenderIndexingMode": "OFF",
        "namedShadowIndexingMode": "ON",
        "managedFields": [
            {
                "name": "shadow.name.*.hasDelta",
                "type": "Boolean"
            },
            {
                "name": "registry.version",
                "type": "Number"
            },
            {
                "name": "registry.thingTypeName",
                "type": "String"
            },
            {
                "name": "registry.thingGroupNames",
                "type": "String"
            },
            {
                "name": "shadow.name.*.version",
```

```
                "type": "Number"
            },
            {
                "name": "thingName",
                "type": "String"
            },
            {
                "name": "thingId",
                "type": "String"
            }
        ],
        "customFields": [
            {
                "name": "attributes.battery",
                "type": "Number"
            }
        ],
        "filter": {
            "namedShadowNames": [
                "Bike1-shadow"
            ],
            "geoLocations": [
                {
                    "name": "shadow.name.Bike1-shadow.reported.coordinates",
                    "order": "LatLon"
                }
            ]
        }
    },
    "thingGroupIndexingConfiguration": {
        "thingGroupIndexingMode": "OFF"
    }
}
```

Executar consulta geográfica

Agora você atualizou sua configuração de indexação para incluir os dados de localização. Tente criar algumas consultas geográficas e executá-las para ver se você consegue obter os resultados de pesquisa desejados. Uma consulta geográfica deve seguir a Sintaxe da consulta. Você pode encontrar alguns exemplos úteis de geoconsultas em ???.

No comando de exemplo a seguir, você usa a consulta geográfica `shadow.name.Bike1-shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km` para pesquisar dispositivos que estão dentro da distância radial de 15,5 km do ponto central com coordenadas (47,6204, -122,3491).

```
aws iot search-index --query-string "shadow.name.Bike1-
shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

Como você tem um dispositivo localizado nas coordenadas “lat”: 47.6153, “lon”: -122.3333, que fica a uma distância de 15,5 km do ponto central, você deve conseguir ver esse dispositivo (Bike-1) na saída. A saída poderá ser parecida com o seguinte:

```
{
    "things": [
        {
            "thingName": "Bike-1",
            "thingId": "df9cf01d-b0c8-48fe-a2e2-e16cff6b23df",
            "attributes": {
                "acqDate": "06/09/23",
                "battery": "35",
                "model": "OEM-2302-12"
            },
            "shadow": "{\"reported\":{\"coordinates\":{\"lat\":47.6153,\"lon
\":-122.3333}},\"metadata\":{\"reported\":{\"coordinates\":{\"lat\":{\"timestamp
\":1699572906},\"lon\":{\"timestamp\":1699572906}}}},\"hasDelta\":false,\"version\":1}"
        }
    ]
}
```

Para obter mais informações, consulte ???.

Métricas de frota

As métricas de frota são um recurso da indexação de frotas, um serviço gerenciado que permite indexar, pesquisar e agregar os dados de seus dispositivos. AWS IoT Você pode usar métricas de frota para monitorar o estado agregado dos dispositivos da sua frota CloudWatchao longo do tempo, incluindo a revisão da taxa de desconexão dos dispositivos da sua frota ou das alterações médias do nível da bateria em um período especificado.

Usando métricas de frota, você pode criar consultas de agregação cujos resultados são continuamente emitidos CloudWatchcomo métricas para analisar tendências e criar alarmes. Para tarefas de monitoramento, é possível especificar as consultas de agregação de diferentes tipos de agregação (estatística, cardinalidade e percentual). É possível salvar todas as suas consultas de agregação para criar métricas de frota para reutilização posterior.

Tutorial de inicialização

Neste tutorial, você criará uma métrica de frota para monitorar as temperaturas de seus sensores e detectar possíveis anomalias. Ao criar a métrica da frota, você define uma consulta de agregação para detectar o número de sensores com temperaturas superiores a 80 graus Fahrenheit. Você especifica a consulta a ser executada a cada 60 segundos e os resultados da consulta são emitidos para CloudWatch, onde é possível visualizar o número de sensores que apresentam riscos potenciais de alta temperatura e definir alarmes. Para concluir este tutorial, você utilizará a AWS CLI.

Neste tutorial, você aprenderá:

- Configuração
- Criação de métricas de frota
- Exibir métricas em CloudWatch
- Limpeza de recursos

Este tutorial leva cerca de 15 minutos para ser concluído.

Pré-requisitos

- Instale a versão mais recente da AWS CLI
- Familiarize-se com a consulta de dados agregados
- Familiarize-se com o uso de métricas da Amazon CloudWatch

Configuração

Para usar as métricas de frota, ative a indexação de frotas. Para habilitar a indexação de frotas para suas objetos ou grupos de objetos com fontes de dados especificadas e configurações associadas, siga as instruções de Gerenciar a indexação de objetos e Gerenciar a indexação de grupos de objetos.

Realização da configuração

1. Execute o seguinte comando para ativar a indexação de frotas e especificar as fontes de dados a partir das quais pesquisar.

```
aws iot update-indexing-configuration \
--thing-indexing-configuration
 "thingIndexingMode=REGISTRY_AND_SHADOW,customFields=[{name=attributes.temperature,type=Num
{name=attributes.rackId,type=String},
{name=attributes.stateNormal,type=Boolean}],thingConnectivityIndexingMode=STATUS" \
```

   O exemplo de comando da CLI anterior permite que a indexação de frotas ofereça suporte à pesquisa de dados de registro, dados de sombra e status de conectividade de objetos usando o índice `AWS_Things`.

   Pode levar alguns minutos para que a alteração de configuração seja concluída. Antes de criar métricas de frota, verifique se a indexação da frota está ativada.

   Execute o seguinte comando da CLI para verificar se a indexação da frota foi ativada:

```
aws --region us-east-1 iot describe-index --index-name "AWS_Things"
```

   Para obter mais informações, consulte Habilitar a indexação de itens.

2. Execute o script bash a seguir para criar dez objetos e descrevê-las.

```
# Bash script. Type `bash` before running in other shells.

Temperatures=(70 71 72 73 74 75 47 97 98 99)
Racks=(Rack1 Rack1 Rack2 Rack2 Rack3 Rack4 Rack5 Rack6 Rack6 Rack6)
IsNormal=(true true true true true true false false false false)

for ((i=0; i < 10; i++))
do
  thing=$(aws iot create-thing --thing-name "TempSensor$i" --attribute-
payload attributes="{temperature=${Temperatures[@]:$i:1},rackId=${Racks[@]:
$i:1},stateNormal=${IsNormal[@]:$i:1}}")
  aws iot describe-thing --thing-name "TempSensor$i"
done
```

O script cria dez objetos para representar dez sensores. Cada objeto tem atributos de `temperature`, `rackId` e `stateNormal`, conforme descrito na seguinte tabela:

Atributo	Tipo de dados	Description
`temperature`	Número	Valor da temperatura em Fahrenheit
`rackId`	String	ID do rack do servidor contendo sensores
`stateNormal`	Booleano	Se o valor da temperatura do sensor é normal ou não

A saída desse script contém dez arquivos JSON. Um dos arquivos JSON tem a seguinte aparência:

```
{
    "version": 1,
    "thingName": "TempSensor0",
    "defaultClientId": "TempSensor0",
    "attributes": {
        "rackId": "Rack1",
        "stateNormal": "true",
        "temperature": "70"
    },
    "thingArn": "arn:aws:iot:region:account:thing/TempSensor0",
    "thingId": "example-thing-id"
}
```

Para obter mais informações, consulte Criar um objeto.

Criação de métricas de frota

Para criar uma métrica de frota

1. Execute o comando a seguir para criar uma métrica de frota chamada*high_temp_FM*. Você cria a métrica da frota para monitorar o número de sensores com temperaturas superiores a 80 graus Fahrenheit in. CloudWatch

```
aws iot create-fleet-metric --metric-name "high_temp_FM" --query-string
 "thingName:TempSensor* AND attributes.temperature >80" --period 60 --aggregation-
field "attributes.temperature" --aggregation-type name=Statistics,values=count
```

--metric-name

Tipo de dados: string. O parâmetro `--metric-name` especifica o nome da métrica da frota. Neste exemplo, você está criando uma métrica de frota chamada high_temp_FM.

--query-string

Tipo de dados: string. O parâmetro `--query-string` especifica a string de consulta. Neste exemplo, a string de consulta significa consultar todas as coisas com nomes que começam com TempSensore com temperaturas superiores a 80 graus Fahrenheit. Para obter mais informações, consulte Sintaxe de consulta.

--period

Tipo de dados: inteiro. O parâmetro `--period` especifica o tempo, em segundos, até a recuperação dos dados agregados. Neste exemplo, está especificado que a métrica de frota criada por você recupera os dados agregados a cada 60 segundos.

--aggregation-field

Tipo de dados: string. O parâmetro `--aggregation-field` especifica o atributo a ser avaliado. Neste exemplo, o atributo a ser avaliado é a temperatura.

--aggregation-type

O parâmetro `--aggregation-type` especifica o resumo estatístico a ser exibido na métrica da frota. Para tarefas de monitoramento, é possível personalizar as propriedades das consultas de agregação conforme os diferentes tipos de agregação (estatística, cardinalidade e percentual).

Neste exemplo, você especifica contagem para o tipo de agregação e Estatísticas para retornar a contagem de dispositivos que têm atributos que correspondem à consulta, em outras palavras, para retornar a contagem dos dispositivos com nomes começando com TempSensore com temperaturas superiores a 80 graus Fahrenheit. Para obter mais informações, consulte Consultar dados agregados.

A saída desse comando é semelhante à seguinte:

```
{
    "metricArn": "arn:aws:iot:region:111122223333:fleetmetric/high_temp_FM",
    "metricName": "high_temp_FM"
}
```

> **Note**
>
> Pode levar um momento para que os pontos de dados sejam exibidos CloudWatch.

Para saber mais sobre a criação de uma métrica de frota, leia Gerenciar métricas de frota.

Se você não conseguir criar uma métrica de frota, leia Solução de problemas de métricas de frota.

2. (Opcional) Execute o seguinte comando para descrever uma métrica de frota nomeada high_temp_FM:

```
aws iot describe-fleet-metric --metric-name "high_temp_FM"
```

A saída desse comando é semelhante à seguinte:

```
{
    "queryVersion": "2017-09-30",
    "lastModifiedDate": 1625249775.834,
    "queryString": "*",
    "period": 60,
    "metricArn": "arn:aws:iot:region:111122223333:fleetmetric/high_temp_FM",
    "aggregationField": "registry.version",
    "version": 1,
    "aggregationType": {
        "values": [
```

```
            "count"
        ],
        "name": "Statistics"
    },
    "indexName": "AWS_Things",
    "creationDate": 1625249775.834,
    "metricName": "high_temp_FM"
}
```

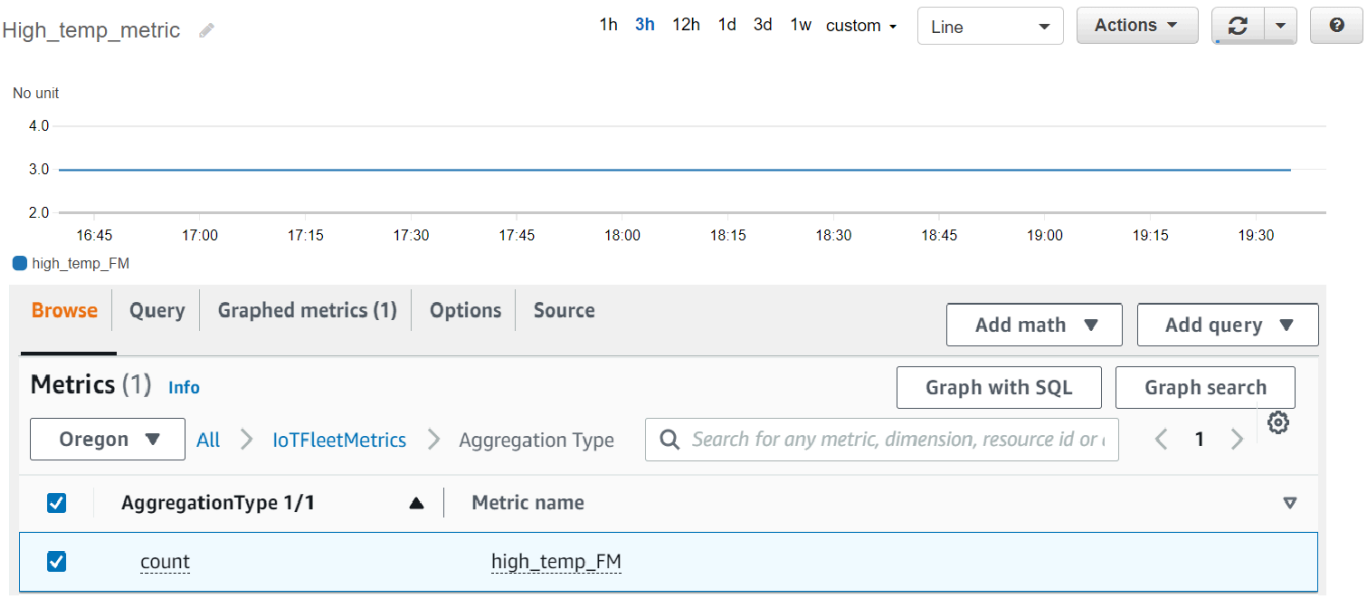
Veja as métricas da frota em CloudWatch

Depois de criar a métrica da frota, você pode visualizar os dados métricos em CloudWatch. Neste tutorial, você verá a métrica que mostra o número de sensores com nomes começando com TempSensore com temperaturas superiores a 80 graus Fahrenheit.

Para visualizar pontos de dados em CloudWatch

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No CloudWatch menu no painel esquerdo, escolha Métricas para expandir o submenu e, em seguida, escolha Todas as métricas. Isso abrirá a página com uma metade superior para exibir o gráfico e uma metade inferior que contém quatro seções com guias.
3. A primeira seção com guias Todas as métricas lista todas as métricas que você pode visualizar em grupos. Escolha Io TFleet Metrics. Aqui estão todas as métricas da sua frota.
4. Na seção Tipo de agregação da guia Todas as métricas, selecione Tipo de agregação para visualizar todas as métricas de frota criadas por você.
5. Selecione a métrica da frota para exibir o gráfico à esquerda da seção Tipo de agregação. Você verá o valor *count* à esquerda do nome da métrica, e esse é o valor do tipo de agregação que você especificou na seção Criar métricas de frota deste tutorial.
6. Escolha a segunda guia, chamada Métricas gráficas, à direita da guia Todas as métricas, para ver a métrica da frota que você escolheu na etapa anterior.

   Você deve conseguir ver um gráfico exibindo o número de sensores com temperaturas maiores que 80 graus Fahrenheit, como este:



> **Note**
>
> O atributo Período tem CloudWatch como padrão 5 minutos. É o intervalo de tempo entre os pontos de dados exibidos CloudWatch. É possível alterar a configuração de Período conforme as suas necessidades.

7. (Opcional) É possível definir um alarme métrico.

   1. No CloudWatch menu no painel esquerdo, escolha Alarmes para expandir o submenu e, em seguida, escolha Todos os alarmes.
   2. Na página Alarmes, selecione Criar alarme no canto superior direito. Siga as instruções de Criar alarme no console para criar um alarme adequado ao caso de uso. Para obter mais informações, consulte Usando CloudWatch alarmes da Amazon.

Para saber mais, leia Como usar CloudWatch métricas da Amazon.

Se você não conseguir ver os pontos de dados CloudWatch, leia Solução de problemas nas métricas da frota.

Limpeza

Exclusão de métricas de frotas

Deve-se usar o comando delete-fleet-metric da CLI para excluir as métricas de frota.

Execute o seguinte comando para excluir a métrica de frota nomeada high_temp_FM.

```
aws iot delete-fleet-metric --metric-name "high_temp_FM"
```

Limpeza de objetos

Deve-se usar o comando delete-thing da CLI para excluir objetos.

Execute o script a seguir para excluir as dez objetos criadas:

```
# Bash script. Type `bash` before running in other shells.

for ((i=0; i < 10; i++))
do
  thing=$(aws iot delete-thing --thing-name "TempSensor$i")
done
```

Para limpar as métricas em CloudWatch

CloudWatch não suporta a exclusão de métricas. As métricas expiram baseadas em seus cronogramas de retenção. Para saber mais, use as CloudWatch métricas da Amazon.

Gerenciar métricas de frota

Este tópico mostra como usar o AWS IoT console e gerenciar AWS CLI as métricas da sua frota.

Tópicos

- Gerenciar métricas de frota (console)
- Gerenciar métricas de frota (CLI)
- Autorize a marcação de recursos de IoT

Gerenciar métricas de frota (console)

As seções a seguir mostram como usar o AWS IoT console para gerenciar as métricas da sua frota. Habilite a indexação de frotas com as fontes de dados e configurações associadas antes de criar métricas de frota.

Habilitar a indexação de frotas

Caso já tenha ativado a indexação de frotas, pule esta seção.

Caso não tenha ativado a indexação de frotas, siga estas instruções.

1. Abra seu AWS IoT console em https://console.aws.amazon.com/iot/.
2. No AWS IoT menu, escolha Configurações.
3. Para ver as configurações detalhadas, na página Configurações, role para baixo até a seção Indexação de frotas.
4. Para atualizar suas configurações de indexação de frotas, à direita da seção Indexação de frotas, selecione Gerenciar indexação.
5. Na página Gerenciar indexação de frotas, atualize suas configurações de indexação de frotas conforme suas necessidades.
   - Configuração

     Para ativar a indexação de objetos, ative a Indexação de objetos e selecione as fontes de dados a partir das quais você deseja indexar.

     Para ativar a indexação de grupos de objetos, ative a Indexação de grupos de objetos.
   - Campos personalizados para agregação - opcional

     Os campos personalizados são uma lista de pares de nomes de campos e tipos de campos.

     Para adicionar um par de campos personalizado, selecione Adicionar novo campo. Insira um nome de campo personalizado, como `attributes.temperature`, e escolha um tipo de campo no menu Tipo de campo. Observe que o nome de um campo personalizado começa com `attributes.` e será salvo como um atributo para execução de consultas de agregações de objetos.

     Para atualizar e salvar a configuração, selecione Atualizar.

Criação de uma métrica de frota

1. Abra seu AWS IoT console em https://console.aws.amazon.com/iot/.
2. No AWS IoT menu, escolha Gerenciar e, em seguida, escolha Métricas da frota.
3. Na página Métricas de frota, selecione Criar métrica de frota e conclua as etapas de criação.
4. Na etapa 1, Configure as métricas da frota
   - Na seção Consulta, digite uma string de consulta para especificar as objetos ou grupos de objetos com as quais deseja realizar a pesquisa agregada. A string de consulta consiste

em um atributo e um valor. Em Propriedades, escolha o atributo desejado ou, caso ele não apareça na lista, digite o atributo no campo. Insira o valor depois de `:`. Um exemplo de string de consulta: `thingName:TempSensor*`. Para cada string de consulta inserida, pressione enter em seu teclado. Se você inserir diversas strings de consulta, especifique seu relacionamento escolhendo e, ou, e não ou ou não entre elas.

- Em Propriedades do relatório, selecione Nome do índice, Tipo de agregação e Campo de agregação em suas respectivas listas. Em sequência, selecione os dados que você deseja agregar em Selecionar dados, onde você pode selecionar vários valores de dados.
- Escolha Próximo.

5. Na etapa 2, Especifique as propriedades métricas da frota
   - No campo Nome da métrica da frota, digite um nome para a métrica da frota em criação.
   - No campo Descrição - opcional, digite uma descrição para a métrica da frota em criação. Esse campo é opcional.
   - Nos campos Horas e Minutos, insira a hora (com que frequência) você deseja que a métrica da frota emita dados. CloudWatch
   - Escolha Próximo.
6. Na etapa 3, Revise e crie
   - Revise as configurações da etapa 1 e etapa 2. Para editar as configurações, escolha Editar.
   - Selecione Criar métrica de frota.

Após a criação com êxito, a métrica da frota estará listada na página Métrica da frota.

Atualização de uma métrica de frota

1. Na página Métrica da frota, selecione a métrica da frota que você quer atualizar.
2. Na página Detalhes da métrica de frota, selecione Editar. Isso abre as etapas de criação nas quais você pode atualizar sua métrica de frota em qualquer uma das três etapas.
3. Depois de terminar de atualizar a métrica da frota, selecione Atualizar métrica da frota.

Exclusão de uma métrica de frota

1. Na página Métrica de frota, selecione a métrica de frota que você quer excluir.
2. Na próxima página, que exibe detalhes da métrica da sua frota, selecione Excluir.
3. Na caixa de diálogo, digite o nome da métrica de frota para confirmar a exclusão.

4. Selecione Excluir. Essa etapa exclui permanentemente a métrica de frota.

Gerenciar métricas de frota (CLI)

As seções a seguir mostram como usar o AWS CLI para gerenciar suas métricas de frota. Habilite a indexação de frotas com as fontes de dados e configurações associadas antes de criar métricas de frota. Para habilitar a indexação de frotas para suas objetos ou grupos de objetos siga as instruções de Gerenciar a indexação de objetos e Gerenciar a indexação de grupos de objetos.

Criação de uma métrica de frota

Você pode usar o comando create-fleet-metric CLI para criar uma métrica de frota.

```
aws iot create-fleet-metric --metric-name "YourFleetMetricName" --query-string
 "*" --period 60 --aggregation-field "registry.version" --aggregation-type
 name=Statistics,values=sum
```

A saída deste comando possui o nome e o nome do recurso da Amazon (ARN) da sua métrica de frota. A saída será exibida como a seguir:

```
{
    "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
    "metricName": "YourFleetMetricName"
}
```

Listagem de métricas de frota

Você pode usar o comando list-fleet-metric CLI para listar todas as métricas da frota em sua conta.

```
aws iot list-fleet-metrics
```

A saída deste comando possui todas as métricas de frota. A saída será exibida como a seguir:

```
{
    "fleetMetrics": [
        {
            "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/
YourFleetMetric1",
            "metricName": "YourFleetMetric1"
```

```
        },
        {
            "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/
YourFleetMetric2",
            "metricName": "YourFleetMetric2"
        }
    ]
}
```

Descrição de uma métrica de frota

Você pode usar o comando describe-fleet-metric CLI para exibir informações mais detalhadas sobre uma métrica de frota.

```
aws iot describe-fleet-metric --metric-name "YourFleetMetricName"
```

A saída do comando contém as informações detalhadas sobre a métrica de frota especificada. A saída será exibida como a seguir:

```
{
    "queryVersion": "2017-09-30",
    "lastModifiedDate": 1625790642.355,
    "queryString": "*",
    "period": 60,
    "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
    "aggregationField": "registry.version",
    "version": 1,
    "aggregationType": {
        "values": [
            "sum"
        ],
        "name": "Statistics"
    },
    "indexName": "AWS_Things",
    "creationDate": 1625790642.355,
    "metricName": "YourFleetMetricName"
}
```

Atualização de uma métrica de frota

Você pode usar o comando update-fleet-metric CLI para atualizar uma métrica de frota.

```
aws iot update-fleet-metric --metric-name "YourFleetMetricName" --query-string
 "*" --period 120 --aggregation-field "registry.version" --aggregation-type
 name=Statistics,values=sum,count --index-name AWS_Things
```

O comando update-fleet-metric não produz saída. Você pode usar o comando describe-fleet-metric CLI para ver o resultado.

```
{
    "queryVersion": "2017-09-30",
    "lastModifiedDate": 1625792300.881,
    "queryString": "*",
    "period": 120,
    "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
    "aggregationField": "registry.version",
    "version": 2,
    "aggregationType": {
        "values": [
            "sum",
            "count"
        ],
        "name": "Statistics"
    },
    "indexName": "AWS_Things",
    "creationDate": 1625792300.881,
    "metricName": "YourFleetMetricName"
}
```

Exclusão de uma métrica de frota

Use o comando delete-fleet-metric CLI para excluir uma métrica de frota.

```
aws iot delete-fleet-metric --metric-name "YourFleetMetricName"
```

Esse comando não produz saída alguma se a exclusão obtiver êxito ou se você especificar uma métrica de frota inexistente.

Para obter mais informações, consulte Solução de problemas de métricas de frota.

Autorize a marcação de recursos de IoT

Para um controle melhor sobre as métricas da frota que você pode criar, modificar ou usar, você pode anexar tags às métricas da frota.

Para marcar as métricas de frota que você cria usando Console de gerenciamento da AWS ou AWS CLI, você deve incluir a `iot:TagResource` ação na sua política do IAM para conceder permissões ao usuário. Se sua política do IAM não incluir `iot:TagResource`, qualquer ação para criar uma métrica de frota com uma tag retornará um erro `AccessDeniedException`.

Para informações gerais sobre a marcação de recursos, consulte Marcar seus recursos de AWS IoT.

Exemplo de política do IAM

Consulte o seguinte exemplo de políticas do IAM que concedem permissões de marcação ao criar uma métrica da frota:

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Action": [
    "iot:TagResource"
   ],
   "Effect": "Allow",
   "Resource": [
    "arn:aws:iot:*:*:fleetmetric/*"
   ]
  },
  {
   "Action": [
    "iot:CreateFleetMetric"
   ],
   "Effect": "Allow",
   "Resource": [
    "arn:aws:iot:*:*:index/*",
    "arn:aws:iot:*:*:fleetmetric/*"
   ]
  }
 ]
}
```

Para obter mais informações, consulte Ações, recursos e chaves de condição do AWS IoT.

Entrega de arquivos baseada em MQTT

Uma opção que você pode usar para gerenciar arquivos e transferi-los para AWS IoT dispositivos em sua frota é a entrega de arquivos baseada em MQTT. Com esse recurso na AWS nuvem, você pode criar um stream que contém vários arquivos, atualizar os dados do stream (a lista e as descrições dos arquivos), obter os dados do stream e muito mais. AWS IoT A entrega de arquivos baseada em MQTT pode transferir dados em pequenos blocos para seus dispositivos de IoT, usando o protocolo MQTT com suporte para mensagens de solicitação e resposta em JSON ou CBOR.

Para obter mais informações sobre como transferir dados de e para dispositivos de IoT usando AWS IoT, consulte. Conecte dispositivos a AWS IoT

Tópicos

- O que é um stream?
- Gerenciando um stream na AWS nuvem
- Usando a entrega de arquivos AWS IoT baseada em MQTT em dispositivos
- Um exemplo de caso de uso no FreeRTOS OTA

O que é um stream?

Em AWS IoT, um stream é um recurso endereçável publicamente que é uma abstração de uma lista de arquivos que podem ser transferidos para um dispositivo de IoT. Um stream típico contém as seguintes informações:

- Um nome do recurso da Amazon (ARN) que identifica exclusivamente um stream em um determinado momento. Esse ARN tem o padrão `arn:`*`partition`*`:iot:`*`region`*`:`*`account-ID`*`:stream/`*`stream ID`*.
- Um ID de stream que identifica seu stream e é usado (e geralmente obrigatório) em AWS Command Line Interface (AWS CLI) ou comandos do SDK.
- Uma descrição do stream que fornece uma descrição do recurso do stream.
- Uma versão do stream que identifica uma versão específica do stream. Como os dados do stream podem ser modificados imediatamente antes que os dispositivos iniciem a transferência de dados, a versão do stream pode ser usada pelos dispositivos para impor uma verificação de consistência.
- Uma lista de arquivos que podem ser transferidos para dispositivos. Para cada arquivo na lista, o stream registra um ID de arquivo, o tamanho do arquivo e as informações de endereço do arquivo,

que consistem, por exemplo, no nome do bucket do Amazon S3, na chave do objeto e na versão do objeto.

- Uma função AWS Identity and Access Management (IAM) que concede à entrega de arquivos AWS IoT baseada em MQTT a permissão para ler arquivos de stream armazenados no armazenamento de dados.

AWS IoT A entrega de arquivos baseada em MQTT fornece as seguintes funcionalidades para que os dispositivos possam transferir dados da AWS nuvem:

- Transferência de dados usando o protocolo MQTT.
- Suporte para formatos JSON ou CBOR.
- A capacidade de descrever um fluxo (API `DescribeStream`) para acessar uma lista de arquivos de fluxo, uma versão do fluxo e informações relacionadas.
- A capacidade de enviar dados em pequenos blocos (API `GetStream`) para que dispositivos com restrições de hardware possam receber os blocos.
- Suporte para um tamanho de bloco dinâmico por solicitação, para suportar dispositivos com diferentes capacidades de memória.
- Otimização para solicitações de streaming simultâneas quando vários dispositivos solicitam blocos de dados do mesmo arquivo de stream.
- Amazon S3 como armazenamento de dados para arquivos de stream.
- Support para publicação de registros de transferência de dados da entrega de arquivos AWS IoT baseada em MQTT para o. CloudWatch

Para quotas de entrega de arquivos baseadas em MQTT, consulte AWS IoT Core Service Quotas no Referência geral da AWS.

Gerenciando um stream na AWS nuvem

AWS IoT fornece AWS SDK e AWS CLI comandos que você pode usar para gerenciar um stream na AWS nuvem. Você pode usar esses comandos para fazer o seguinte:

- Criar um stream. CLI / SDK
- Descreva um stream para obter informações. CLI / SDK
- Liste streams em seu. Conta da AWSCLI / SDK

- Atualize a lista de arquivos ou a descrição do stream em um stream. CLI / SDK
- Exclui um stream. CLI / SDK

> **Note**
>
> No momento, os streams não são visíveis no Console de gerenciamento da AWS. Você deve usar o AWS SDK AWS CLI ou para gerenciar um stream em AWS IoT. Além disso, o Embedded C SDK é o único SDK que suporta transferências de arquivos baseadas em MQTT.

Antes de usar a entrega de arquivos AWS IoT baseada em MQTT de seus dispositivos, você deve garantir que as seguintes condições sejam atendidas para seus dispositivos, conforme mostrado nas próximas seções:

- Uma política que reflete as permissões corretas necessárias para transmitir dados via MQTT.
- Seu dispositivo pode se conectar ao AWS IoT Device Gateway.
- Uma declaração de política informando que você pode marcar recursos. Se `CreateStream` for chamado com tags, então `iot:TagResource` é obrigatório.

Antes de usar a entrega de arquivos AWS IoT baseada em MQTT de seus dispositivos, você deve seguir as etapas nas próximas seções para garantir que seus dispositivos estejam devidamente autorizados e possam se conectar ao AWS IoT Device Gateway.

Conceda permissões aos seus dispositivos

Você pode seguir as etapas em Criar uma AWS IoT política para criar uma política de dispositivo ou usar uma política de dispositivo existente. Anexe a política aos certificados associados aos dispositivos e adicione as permissões a seguir à política do dispositivo.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
```

```
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive",
                "iot:Publish"
            ],
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topic/$aws/things/
${iot:Connection.Thing.ThingName}/streams/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": [
            "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/streams/*"
            ]
        }
    ]
}
```

Conecte seus dispositivos a AWS IoT

É necessário se conectar aos dispositivos que usam a entrega de arquivos AWS IoT baseada em MQTT. AWS IoT AWS IoT A entrega de arquivos baseada em MQTT se AWS IoT integra à AWS nuvem, portanto, seus dispositivos devem se conectar diretamente ao endpoint do plano de dados. AWS IoT

> **Note**
>
> O ponto final do plano de AWS IoT dados é específico para a região Conta da AWS e. Você deve usar o endpoint da Conta da AWS e da região na qual seus dispositivos estão registrados. AWS IoT

Consulte Conecte-se a AWS IoT Core para obter mais informações.

TagResource Uso

O API `CreateStream` cria um fluxo para fornecer um ou mais arquivos grandes em blocos por meio do MQTT.

Uma chamada de API `CreateStream` bem-sucedida requer as seguintes permissões:

- `iot:CreateStream`
- `iot:TagResource` (se `CreateStream` estiver com tags)

A política que suporta essas duas permissões é mostrada abaixo:

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Action": [
            "iot:CreateStream",
            "iot:TagResource"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:iot:us-east-1:123456789012:stream/streamId"
    }
}
```

A ação da declaração de política `iot:TagResource` é necessária para garantir que um usuário não possa criar ou atualizar uma tag em um recurso sem as permissões adequadas. Sem a ação de declaração de política específica de `iot:TagResource`, a chamada de `CreateStream` API retornará e `AccessDeniedException` se a solicitação vier com tags.

Para obter mais informações, consulte os links a seguir:

- CreateStream
- TagResource
- Tag

Usando a entrega de arquivos AWS IoT baseada em MQTT em dispositivos

Para iniciar o processo de transferência de dados, um dispositivo deve receber um conjunto de dados inicial, que inclua, no mínimo, um ID de stream. Você pode usar um AWS IoT Empregos para agendar tarefas de transferência de dados para seus dispositivos incluindo o conjunto de dados inicial no documento da tarefa. Quando um dispositivo recebe o conjunto de dados inicial, ele deve iniciar a interação com a entrega de arquivos AWS IoT baseada em MQTT. Para trocar dados com a entrega de arquivos AWS IoT baseada em MQTT, um dispositivo deve:

- Use o protocolo MQTT para assinar o Tópicos de entrega de arquivos baseados em MQTT.
- Envie solicitações e aguarde para receber as respostas usando mensagens MQTT.

Opcionalmente, você pode incluir um ID de arquivo de stream e uma versão de stream no conjunto de dados inicial. Enviar um ID de arquivo de stream para um dispositivo pode simplificar a programação do firmware/software do dispositivo, pois elimina a necessidade de fazer uma solicitação do dispositivo `DescribeStream` para obter esse ID. O dispositivo pode especificar a versão do stream em uma `GetStream` solicitação para impor uma verificação de consistência caso o stream tenha sido atualizado inesperadamente.

Use DescribeStream para obter dados de stream

AWS IoT A entrega de arquivos baseada em MQTT fornece a `DescribeStream` API para enviar dados de stream para um dispositivo. Os dados de stream retornados por essa API incluem o ID do stream, a versão do stream, a descrição do stream e uma lista de arquivos do stream, cada um com um ID de arquivo e o tamanho do arquivo em bytes. Com essas informações, um dispositivo pode selecionar arquivos arbitrários para iniciar o processo de transferência de dados.

> **Note**
>
> Você não precisa usar a `DescribeStream` API se seu dispositivo receber todos os arquivos de stream necessários IDs no conjunto de dados inicial.

Siga estas etapas para fazer uma solicitação `DescribeStream`.

1. Faça a assinatura no filtro de tópicos “aceitos” `$aws/things/ThingName/streams/StreamId/description/json`.
2. Faça a assinatura no filtro de tópicos “rejeitados” `$aws/things/ThingName/streams/StreamId/rejected/json`.
3. Publique uma `DescribeStream` solicitação enviando uma mensagem para `$aws/things/ThingName/streams/StreamId/describe/json`.
4. Se a solicitação for aceita, seu dispositivo receberá uma `DescribeStream` resposta no filtro de tópicos “aceitos”.
5. Se a solicitação for rejeitada, seu dispositivo receberá a resposta de erro no filtro de tópicos “rejeitado”.

> **Note**
>
> Se você `json` substituir por `cbor` nos tópicos e filtros de tópicos mostrados, seu dispositivo receberá mensagens no formato CBOR, que é mais compacto que o JSON.

DescribeStream pedido

Uma `DescribeStream` solicitação típica em JSON é semelhante ao exemplo a seguir.

```
{
    "c": "ec944cfb-1e3c-49ac-97de-9dc4aaad0039"
}
```

- (Opcional) "c" é o campo do token do cliente.

  O token do cliente não pode ter até 64 bytes. Um token de cliente com mais de 64 bytes causa uma resposta de erro e uma mensagem de erro `InvalidRequest`.

DescribeStream resposta

Uma resposta `DescribeStream` em JSON se parece com o seguinte exemplo.

```
{
    "c": "ec944cfb-1e3c-49ac-97de-9dc4aaad0039",
    "s": 1,
```

```
    "d": "This is the description of stream ABC.",
    "r": [
        {
            "f": 0,
            "z": 131072
        },
        {
            "f": 1,
            "z": 51200
        }
    ]
}
```

- "c" é o campo do token do cliente. Isso é retornado se tiver sido fornecido na `DescribeStream` solicitação. Use o token do cliente para associar a resposta à solicitação.
- "s" é a versão do stream como um número inteiro. Você pode usar essa versão para realizar uma verificação de consistência com as `GetStream` solicitações.
- "r" contém uma lista dos arquivos no stream.
  - "f" é o ID do arquivo de stream como um número inteiro.
  - "z" é o tamanho do arquivo de stream em número de bytes.
- "d" contém a descrição do stream.

Obter blocos de dados de um arquivo de stream

Você pode usar a API `GetStream` para que um dispositivo possa receber arquivos de stream em pequenos blocos de dados, para que ela possa ser usada por dispositivos que têm restrições no processamento de blocos grandes. Para receber um arquivo de dados inteiro, um dispositivo pode precisar enviar ou receber várias solicitações e respostas até que todos os blocos de dados sejam recebidos e processados.

GetStream pedido

Siga estas etapas para fazer uma solicitação `GetStream`.

1. Faça a assinatura no filtro de tópicos “aceitos” `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/data/json`.
2. Faça a assinatura no filtro de tópicos “rejeitados” `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/rejected/json`.

3. Publique uma solicitação `GetStream` para o tópico `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/get/json`.
4. Se a solicitação for aceita, seu dispositivo receberá uma ou mais respostas `GetStream` no filtro de tópicos “aceitos”. Cada mensagem de resposta contém informações básicas e uma carga útil de dados para um único bloco.
5. Repita as etapas 3 e 4 para receber todos os blocos de dados. Você deve repetir essas etapas se a quantidade de dados solicitada for maior que 128 KB. Você deve programar seu dispositivo para usar várias `GetStream` solicitações para receber todos os dados solicitados.
6. Se a solicitação for rejeitada, seu dispositivo receberá a resposta de erro no filtro de tópicos “rejeitado”.

> **Note**
>
> - Se você substituir “json” por “cbor” nos tópicos e filtros de tópicos mostrados, seu dispositivo receberá mensagens no formato CBOR, que é mais compacto que o JSON.
> - AWS IoT A entrega de arquivos baseada em MQTT limita o tamanho de um bloco a 128 KB. Se você fizer uma solicitação para um bloco com mais de 128 KB, a solicitação falhará.
> - Você pode fazer uma solicitação para vários blocos cujo tamanho total seja maior que 128 KB (por exemplo, se você fizer uma solicitação de cinco blocos de 32 KB cada, totalizando 160 KB de dados). Nesse caso, a solicitação não falha, mas seu dispositivo precisa fazer várias solicitações para receber todos os dados solicitados. O serviço enviará blocos adicionais à medida que seu dispositivo fizer solicitações adicionais. Recomendamos que você continue com uma nova solicitação somente após a resposta anterior ter sido recebida e processada corretamente.
> - Independentemente do tamanho total dos dados solicitados, você deve programar seu dispositivo para iniciar novas tentativas quando os blocos não forem recebidos ou não forem recebidos corretamente.

Uma `GetStream` solicitação típica em JSON é semelhante ao exemplo a seguir.

```
{
    "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380",
    "s": 1,
```

```
    "f": 0,
    "l": 4096,
    "o": 2,
    "n": 100,
    "b": "..."
}
```

- [opcional] "c" é o campo do token do cliente.

  O token do cliente pode ter até 64 bytes. Um token de cliente com mais de 64 bytes causa uma resposta de erro e uma mensagem de erro `InvalidRequest`.

- [opcional] "s" é o campo da versão do stream (um número inteiro).

  A entrega de arquivos baseada em MQTT aplica uma verificação de consistência com base nessa versão solicitada e na versão mais recente do stream na nuvem. Se a versão do stream enviada de um dispositivo em uma solicitação `GetStream` não corresponder à versão mais recente do stream na nuvem, o serviço enviará uma resposta de erro e uma mensagem de erro `VersionMismatch`. Normalmente, um dispositivo recebe a versão esperada (mais recente) do stream no conjunto de dados inicial ou na resposta a `DescribeStream`.

- "f" é o ID do arquivo de fluxo (um número inteiro no intervalo de 0 a 255).

  O ID do arquivo de stream é necessário quando você cria ou atualiza um stream usando o SDK AWS CLI ou. Se um dispositivo solicitar um arquivo de stream com uma ID que não existe, o serviço envia uma resposta de erro e uma mensagem de erro `ResourceNotFound`.

- "l" é o tamanho do bloco de dados em bytes (um número inteiro no intervalo de 256 a 131.072).

  Consulte Crie um bitmap para uma solicitação GetStream para obter instruções sobre como usar os campos de bitmap para especificar qual parte do arquivo de stream será retornada na resposta `GetStream`. Se um dispositivo especificar um tamanho de bloco fora do alcance, o serviço enviará uma resposta de erro e uma mensagem de erro `BlockSizeOutOfBounds`.

- [opcional] "o" é o deslocamento do bloco no arquivo de stream (um número inteiro no intervalo de 0 a 98.304).

  Consulte Crie um bitmap para uma solicitação GetStream para obter instruções sobre como usar os campos de bitmap para especificar qual parte do arquivo de stream será retornada na resposta `GetStream`. O valor máximo de 98.304 é baseado em um limite de tamanho de arquivo de stream de 24 MB e 256 bytes para o tamanho mínimo do bloco. Se não especificado, o padrão será 0.

- [opcional] "n" é o número de blocos solicitados (um número inteiro no intervalo de 0 a 98.304).

O campo “n” especifica (1) o número de blocos solicitados ou (2) quando o campo bitmap (“b”) é usado, um limite no número de blocos que serão retornados pela solicitação de bitmap. Esse segundo uso é opcional. Se não for definido, o padrão é 131072/. *DataBlockSize*

- [opcional] "b" é um bitmap que representa os blocos que estão sendo solicitados.

  Usando um bitmap, seu dispositivo pode solicitar blocos não consecutivos, o que torna mais conveniente lidar com novas tentativas após um erro. Consulte Crie um bitmap para uma solicitação GetStream para obter instruções sobre como usar os campos de bitmap para especificar qual parte do arquivo de fluxo será retornada na resposta `GetStream`. Para esse campo, converta o bitmap em uma string representando o valor do bitmap em notação hexadecimal. O bitmap deve ser menor do que 12.288 bytes.

> **Important**
>
> Tanto "n" quanto "b" devem ser especificados. Se nenhum deles for especificado, a solicitação `GetStream` poderá não ser válida quando o tamanho do arquivo for menor que 131.072 bytes (128 KB).

GetStream resposta

Uma resposta `GetStream` em JSON se parece com este exemplo para cada bloco de dados solicitado.

```
{
    "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380",
    "f": 0,
    "l": 4096,
    "i": 2,
    "p": "..."
}
```

- "`c`" é o campo do token do cliente. Isso é retornado se tiver sido fornecido na `GetStream` solicitação. Use o token do cliente para associar a resposta à solicitação.
- "`f`" é o ID do arquivo de stream ao qual a carga útil do bloco de dados atual pertence.
- "`l`" é o tamanho da carga útil do bloco de dados em bytes.

- "i" é o ID do bloco de dados contido na carga útil. Os blocos de dados são numerados a partir de 0.
- "p" contém a carga útil do bloco de dados. Esse campo é uma string, que representa o valor do bloco de dados na codificação Base64.

Crie um bitmap para uma solicitação GetStream

Você pode usar o campo bitmap (b) em uma `GetStream` solicitação para obter blocos não consecutivos de um arquivo de stream. Isso ajuda dispositivos com capacidade limitada de RAM a lidar com problemas de entrega de rede. Um dispositivo pode solicitar somente os blocos que não foram recebidos ou não foram recebidos corretamente. O bitmap determina quais blocos do arquivo de fluxo serão retornados. Para cada bit, que é definido como 1 no bitmap, um bloco correspondente do arquivo de stream será retornado.

Veja a seguir um exemplo de como especificar um bitmap e seus campos de suporte em uma solicitação `GetStream`. Por exemplo, você deseja receber um arquivo de stream em partes de 256 bytes (o tamanho do bloco). Pense em cada bloco de 256 bytes como tendo um número que especifica sua posição no arquivo, começando em 0. Portanto, o bloco 0 é o primeiro bloco de 256 bytes no arquivo, o bloco 1 é o segundo e assim por diante. Você deseja solicitar os blocos 20, 21, 24 e 43 do arquivo.

Compensação de blocos

Como o primeiro bloco é o número 20, especifique o deslocamento (campo o) como 20 para economizar espaço no bitmap.

Número de blocos

Para garantir que seu dispositivo não receba mais blocos do que pode suportar com recursos de memória limitados, você pode especificar o número máximo de blocos que devem ser retornados em cada mensagem enviada pela entrega de arquivos baseada em MQTT. Observe que esse valor será ignorado se o próprio bitmap especificar menos do que esse número de blocos ou se isso fizer com que o tamanho total das mensagens de resposta enviadas pela entrega de arquivos com base em MQTT seja maior que o limite de serviço de 128 KB por solicitação `GetStream`.

Mapa de bits de blocos

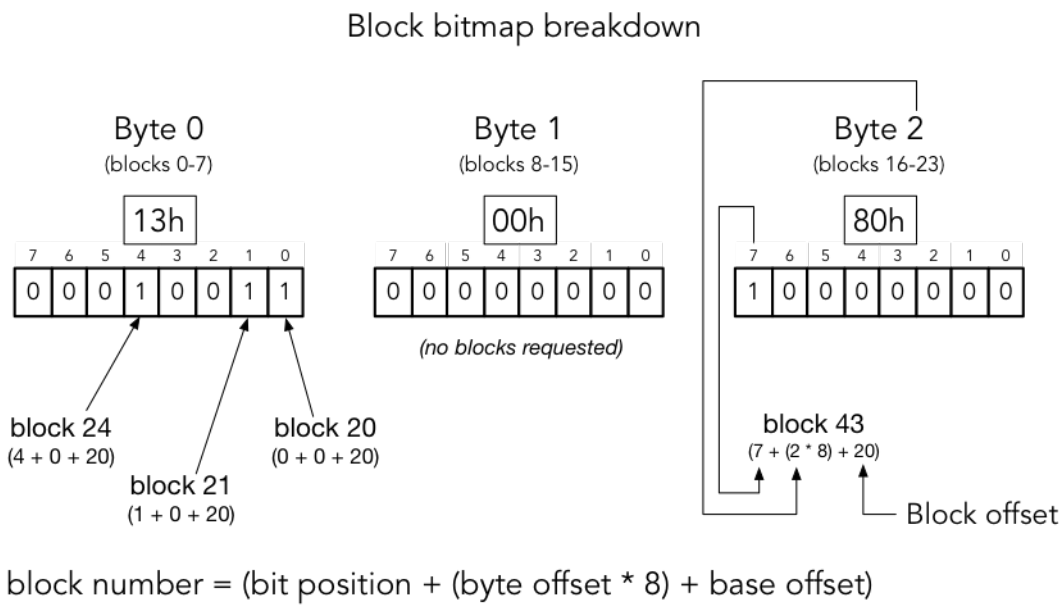
O bitmap em si é uma matriz de bytes não assinados expressos em notação hexadecimal e incluídos na solicitação `GetStream` como uma representação em sequência do número. Mas,

para estruturar essa string, vamos começar pensando no bitmap como uma longa sequência de bits (um número binário). Se um bit nessa sequência for definido como 1, o bloco correspondente do arquivo de stream será enviado de volta ao dispositivo. Para nosso exemplo, queremos receber os blocos 20, 21, 24 e 43, então devemos definir os bits 20, 21, 24 e 43 em nosso bitmap. Podemos usar o deslocamento do bloco para economizar espaço, então, depois de subtrairmos o deslocamento de cada número do bloco, queremos definir os bits 0, 1, 4 e 23, como no exemplo a seguir.

```
1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

Com a consideração de um byte (8 bits) por vez, isso é convencionalmente escrito como: "0b00010011", "0b00000000" e "0b10000000". O bit 0 aparece em nossa representação binária no final do primeiro byte e o bit 23 no início do último. Isso pode ser confuso, a menos que você conheça as convenções. O primeiro byte contém os bits 7-0 (nessa ordem), o segundo byte contém os bits 15-8, o terceiro byte contém os bits 23-16 e assim por diante. Em notação hexadecimal, isso é convertido em "0x130080".

> **Tip**
>
> Você pode converter o binário padrão em notação hexadecimal. Pegue quatro dígitos binários por vez e converta-os em seu equivalente hexadecimal. Por exemplo, “0001” se torna “1”, “0011” se torna “3” e assim por diante.



Juntando tudo isso, o JSON da nossa solicitação `GetStream` se parece com o seguinte.

```
{
    "c" : "1",
    "s" : 1,
    "l" : 256,
    "f" : 1,
    "o" : 20,
    "n" : 32,
    "b" : "130080"
}
```

- "c" é o campo do token do cliente.
- "s" é a versão esperada do stream.
- "l" é o tamanho da carga útil do bloco de dados em bytes.
- "f" é o ID do índice do arquivo de origem.
- "o" é o deslocamento do bloco.
- "n" é o número de blocos.
- "b" é o bitmap BlockID ausente a partir do deslocamento. Esse valor deve ser codificado em based64.

Tratamento de erros da entrega de arquivos AWS IoT baseada em MQTT

Uma resposta de erro que é enviada a um dispositivo para ambos `DescribeStream` e `GetStream` APIs contém um token de cliente, um código de erro e uma mensagem de erro. Uma resposta de erro típica é semelhante ao exemplo a seguir.

```
{
    "o": "BlockSizeOutOfBounds",
    "m": "The block size is out of bounds",
    "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380"
}
```

- "o" é o código de erro que indica o motivo de um erro ter ocorrido. Consulte os códigos de erro mais adiante nesta seção para obter mais detalhes.
- "m" é a mensagem de erro que contém detalhes do erro.

- "`c`" é o campo do token do cliente. Isso pode ser devolvido se tiver sido fornecido na solicitação `DescribeStream`. Você pode usar o token do cliente para associar a resposta à solicitação.

  O campo do token do cliente nem sempre é incluído em uma resposta de erro. Quando o token do cliente fornecido na solicitação não é válido ou está mal formado, ele não é retornado na resposta de erro.

> **Note**
>
> Para compatibilidade com versões anteriores, os campos na resposta de erro podem estar em formato não abreviado. Por exemplo, o código de erro pode ser designado pelos campos “código” ou “o” e o campo do token do cliente pode ser designado pelos campos “clientToken” ou “c”. Recomendamos que você use o formulário de abreviação mostrado acima.

InvalidTopic

O tópico MQTT da mensagem de stream é inválido.

InvalidJson

A solicitação Stream não é um documento JSON válido.

InvalidCbor

A solicitação Stream não é um documento CBOR válido.

InvalidRequest

A solicitação geralmente é identificada como mal formatada. Para ter mais informações, consulte a mensagem de erro.

Não autorizado

A solicitação não está autorizada a acessar os arquivos de dados de stream no meio de armazenamento, como o Amazon S3. Para ter mais informações, consulte a mensagem de erro.

BlockSizeOutOfBounds

O tamanho do bloco está fora dos limites. Consulte a seção “Entrega de arquivos baseada em MQTT” em AWS IoT Core Service Quotas.

OffsetOutOfBounds

O deslocamento está fora dos limites. Consulte a seção “Entrega de arquivos baseada em MQTT” em AWS IoT Core Service Quotas.

BlockCountLimitExceeded

O número de blocos de solicitação está fora dos limites. Consulte a seção “Entrega de arquivos baseada em MQTT” em AWS IoT Core Service Quotas.

BlockBitmapLimitExceeded

O tamanho do bitmap da solicitação está fora dos limites. Consulte a seção “Entrega de arquivos baseada em MQTT” em AWS IoT Core Service Quotas.

ResourceNotFound

O stream, os arquivos, as versões de arquivos ou os blocos solicitados não foram encontrados. Consulte a mensagem de erro para obter mais detalhes.

VersionMismatch

A versão do stream na solicitação não corresponde à versão do stream no atributo de entrega de arquivos baseado em MQTT. Isso indica que os dados do stream foram modificados desde que a versão do stream foi inicialmente recebida pelo dispositivo.

ETagIncompatibilidade

O S3 ETag no stream não corresponde à versão mais recente ETag do objeto S3.

InternalError

Ocorreu um erro interno na entrega de arquivos com base em MQTT.

Um exemplo de caso de uso no FreeRTOS OTA

O agente FreeRTOS OTA over-the-air () AWS IoT usa a entrega de arquivos baseada em MQTT para transferir imagens de firmware do FreeRTOS para dispositivos FreeRTOS. Para enviar o conjunto de dados inicial para um dispositivo, ele usa o serviço AWS IoT Job para agendar um trabalho de atualização OTA para dispositivos FreeRTOS.

Para uma implementação de referência de um cliente de entrega de arquivos baseado em MQTT, consulte os códigos do atendente FreeRTOS OTA na documentação do FreeRTOS.

Device Advisor

O Device Advisor é um recurso de teste totalmente gerenciado baseado em nuvem para validar dispositivos de IoT durante o desenvolvimento do software do dispositivo. O Device Advisor fornece testes pré-criados que você pode usar para validar dispositivos de IoT para conectividade confiável e segura antes de implantá-los na produção. AWS IoT Core Os testes pré-criados do Device Advisor ajudam você a validar o software do dispositivo em relação às práticas recomendadas de uso de TLS, MQTT, Sombra do dispositivo e Tarefas de IoT. Você também pode baixar relatórios de qualificação assinados para enviar à Rede de Parceiros da AWS para qualificar o dispositivo para o Catálogo de dispositivos de parceiros da AWS sem a necessidade de enviar o dispositivo e esperar que ele seja testado.

> **Note**
>
> O Device Advisor é compatível com as regiões us-east-1, us-west-2, ap-northeast-1 e eu-west-1.
> O Device Advisor oferece suporte a dispositivos e clientes que usam os protocolos MQTT e MQTT over WebSocket Secure (WSS) para publicar e assinar mensagens. Todos os protocolos suportam IPv4 IPv6 e.
> O Device Advisor é compatível com certificados de servidor RSA.

Qualquer dispositivo que tenha sido criado para se conectar AWS IoT Core pode aproveitar as vantagens do Device Advisor. Você pode acessar o Device Advisor pelo AWS IoT console ou usando o AWS CLI ou SDK. Quando estiver pronto para testar seu dispositivo, registre-o AWS IoT Core e configure o software do dispositivo com o endpoint Device Advisor. Em seguida, escolha os testes pré-criados, configure-os, execute-os no dispositivo e obtenha os resultados do teste junto com logs detalhados ou um relatório de qualificação.

O Device Advisor é um endpoint de teste na AWS nuvem. Você pode testar os dispositivos configurando-os para se conectarem ao endpoint de teste fornecido pelo Device Advisor. Depois que um dispositivo é configurado para se conectar ao endpoint de teste, você pode visitar o console do Device Advisor ou usar o AWS SDK para escolher os testes que deseja executar em seus dispositivos. Em seguida, o Device Advisor gerencia todo o ciclo de vida de um teste, incluindo o provisionamento de recursos, o agendamento do processo de teste, o gerenciamento da máquina de estado, o registro do comportamento do dispositivo, o registro em logs dos resultados e o fornecimento dos resultados finais na forma de um relatório de teste.

Protocolos TLS

O protocolo Transport Layer Security (TLS) é usado para criptografar dados confidenciais em redes não seguras, como a Internet. O protocolo TLS é o sucessor do protocolo Secure Sockets Layer (SSL).

O Device Advisor é compatível com os seguintes protocolos TLS:

- TLS 1.3 (recomendado)
- TLS 1.2

Protocolos, mapeamentos de porta e autenticação

O protocolo de comunicação do dispositivo é usado por um dispositivo ou cliente para se conectar ao agente de mensagens usando um endpoint do dispositivo. A tabela a seguir lista os protocolos com os quais os endpoints do Device Advisor são compatíveis e os métodos e portas de autenticação usados.

Protocolos, autenticação e mapeamentos de porta

Protocolo	Operações compatíveis	Autenticação	Porta	Nome do protocolo ALPN
Acabou o MQTT WebSocket	Publicar/assinar	Signature versão 4	443	N/D
MQTT	Publicar/assinar	Certificado do cliente X.509	8883	`x-amzn-mqtt-ca`
MQTT	Publicar/assinar	Certificado do cliente X.509	443	N/D

Este capítulo contém as seguintes seções:

- Configurar
- Conceitos básicos do Device Advisor no console
- Fluxo de trabalho do Device Advisor
- Fluxo de trabalho detalhado do console do Device Advisor
- Fluxo de trabalho do console de testes de longa duração

- Endpoints da VPC do Device Advisor (AWS PrivateLink)
- Casos de teste do Device Advisor

Configurar

Antes de usar o Device Advisor pela primeira vez, execute as seguintes tarefas:

Criar um objeto do IoT

Primeiro, crie um objeto do IoT e anexe um certificado a ela. Para ver um tutorial sobre como criar objetos, consulte Criar um objeto.

Criar um perfil do IAM a ser usado como perfil de dispositivo

> **Note**
>
> Você pode criar rapidamente o perfil do dispositivo com o console do Device Advisor. Para saber como configurar o perfil do dispositivo com o console do Device Advisor, consulte Conceitos básicos do Device Advisor no console.

1. Acesse o AWS Identity and Access Management console e faça login no que Conta da AWS você usa para testar o Device Advisor.
2. No painel de navegação à esquerda, escolha Políticas.
3. Escolha Criar política.
4. Em Criar política, faça o seguinte:
   a. Para Serviço, escolha IoT.
   b. Em Ações, faça uma das seguintes:
      - (Recomendado) Selecione ações com base na política anexada ao objeto ou certificado de IoT que você criou na seção anterior.
      - Pesquise as seguintes ações na caixa Filtrar ação e selecione-as:
        - `Connect`
        - `Publish`
        - `Subscribe`

- `Receive`
- `RetainPublish`

c. Em Recursos, restrinja o cliente, o tópico e os recursos do tópico. A restrição desses recursos é uma prática recomendada de segurança. Para restringir recursos, faça o seguinte:

i. Escolha Especificar ARN do recurso do cliente para a ação Conectar.

ii. Escolha Adicionar ARN e, em seguida, faça o seguinte:

> **Note**
>
> O clientId é o ID do cliente MQTT que o dispositivo usa para interagir com o Device Advisor.

- Especifique a Região, o accountId e o clientID no editor visual do ARN.
- Insira manualmente os nomes de recursos da Amazon (ARNs) dos tópicos de IoT com os quais você deseja executar seus casos de teste.

iii. Escolha Adicionar.

iv. Escolha Especificar ARN do recurso de tópico para a ação Receber e mais uma.

v. Escolha Adicionar ARN e, em seguida, faça o seguinte:

> **Note**
>
> O nome do tópico é o tópico do MQTT no qual o dispositivo publica mensagens.

- Especifique a Região, o accountID e o Nome do tópico no editor visual do ARN.
- Insira manualmente os tópicos ARNs de IoT com os quais você deseja executar seus casos de teste.

vi. Escolha Adicionar.

vii. Escolha Especificar ARN do recurso topicFilter para a ação Assinar.

viii. Escolha Adicionar ARN e, em seguida, faça o seguinte:

> **Note**
>
> O nome do tópico é o tópico do MQTT que o dispositivo assina.

- Especifique a Região, o accountID e o Nome do tópico no editor visual do ARN.
- Insira manualmente os tópicos ARNs de IoT com os quais você deseja executar seus casos de teste.

ix. Escolha Adicionar.

5. Escolha Próximo: tags.
6. Escolha Próximo: revisar.
7. Em Revisar política, insira um Nome para a política.
8. Selecione Criar política.
9. No painel de navegação à esquerda, escolha Perfis.
10. Selecione Criar perfil.
11. Em Tipo de entidade confiável, escolha Política de confiança personalizada.
12. Insira a política de confiança a seguir no campo Política de confiança personalizada. Para proteger do problema de confused deputy, adicione as chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount` à política.

> **Important**
>
> `aws:SourceArn` deve estar em conformidade com `format: arn:aws:iotdeviceadvisor:region:account-id:*`. Certifique-se de que a *`region`* corresponda à Região da AWS IoT e *`account-id`* corresponda ao ID da conta do cliente. Para obter mais informações, consulte Prevenção de confused deputy entre serviços.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAwsIoTCoreDeviceAdvisor",
```

```
            "Effect": "Allow",
            "Principal": {
                "Service": "iotdeviceadvisor.amazonaws.com"
        },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
            },
                "ArnLike": {
                    "aws:SourceArn":
 "arn:aws:iotdeviceadvisor:*:123456789012:suitedefinition/*"
            }
        }
        }
    ]
}
```

13. Escolha Próximo.
14. Selecione a política que você criou na Etapa 4.
15. Abra a seção Definir limite de permissões e escolha Usar um limite de permissões para controlar o número máximo de permissões de perfis.
16. Escolha Próximo.
17. Insira um Nome de perfil e uma Descrição de perfil.
18. Selecione Criar perfil.

Crie uma política gerenciada personalizada para que um usuário do IAM use o Device Advisor

1. Navegue até o console do IAM em https://console.aws.amazon.com/iam/. Se solicitado, insira suas credenciais da AWS para fazer login.
2. No painel de navegação à esquerda, escolha Políticas.
3. Escolha Criar política e escolha a guia JSON.
4. Adicione as permissões necessárias para usar o Device Advisor. O documento de política pode ser encontrado no tópico Práticas recomendadas de segurança.
5. Escolha Review Policy (Revisar política).
6. Preencha os campos Nome e Descrição.

7. Escolha Criar política.

Criar um usuário do IAM para usar o Device Advisor

> **Note**
>
> Recomendamos que você crie um usuário do IAM para usar ao executar testes do Device Advisor. Executar testes do Device Advisor por um usuário administrador pode representar riscos de segurança e não é recomendado.

1. Navegue até o console do IAM em https://console.aws.amazon.com/iam/Se solicitado, insira suas AWS credenciais para fazer login.
2. No painel de navegação à esquerda, escolha Usuários.
3. Escolha Add User (Adicionar usuário).
4. Digite um Nome de usuário.
5. Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do Console de gerenciamento da AWS. A forma de conceder acesso programático depende do tipo de usuário que está acessando AWS.

   Para conceder acesso programático aos usuários, selecione uma das seguintes opções:

Qual usuário precisa de acesso programático?	Para	Por
IAM	(Recomendado) Use as credenciais do console como credenciais temporárias para assinar solicitações programáticas para o AWS CLI, AWS SDKs ou. AWS APIs	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Login para desenvolvimento AWS local no Guia AWS Command Line Interface do usuário. • Para AWS SDKs isso, consulte Login para desenvolvimento AWS

Qual usuário precisa de acesso programático?	Para	Por
		local no Guia de referência de ferramentas AWS SDKs e ferramentas.
Identidade da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporárias para assinar solicitações programáticas para o AWS CLI AWS SDKs, ou. AWS APIs	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Configurando o AWS CLI para uso Centro de Identidade do AWS IAM no Guia do AWS Command Line Interface usuário. • Para AWS SDKs, ferramentas e AWS APIs, consulte a autenticação do IAM Identity Center no Guia de referência de ferramentas AWS SDKs e ferramentas.
IAM	Use credenciais temporárias para assinar solicitações programáticas para o AWS CLI AWS SDKs, ou. AWS APIs	Siga as instruções em Como usar credenciais temporárias com AWS recursos no Guia do usuário do IAM.

Qual usuário precisa de acesso programático?	Para	Por
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitações programáticas para o AWS CLI, AWS SDKs, ou. AWS APIs	Siga as instruções da interface que deseja utilizar. • Para isso AWS CLI, consulte Autenticação usando credenciais de usuário do IAM no Guia do AWS Command Line Interface usuário. • Para ferramentas AWS SDKs e ferramentas, consulte Autenticar usando credenciais de longo prazo no Guia de referência de ferramentas AWS SDKs e ferramentas. • Para isso AWS APIs, consulte Gerenciamento de chaves de acesso para usuários do IAM no Guia do usuário do IAM.

6. Escolha Próximo: Permissões.
7. Para conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

   - Usuários e grupos em Centro de Identidade do AWS IAM:

     Crie um conjunto de permissões. Siga as instruções em Criação de um conjunto de permissões no Guia do usuário do Centro de Identidade do AWS IAM .

   - Usuários gerenciados no IAM com provedor de identidades:

     Crie um perfil para a federação de identidades. Siga as instruções em Criando um perfil para um provedor de identidades de terceiros (federação) no Guia do Usuário do IAM.

   - Usuários do IAM:

- Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de um perfil para um usuário do IAM no Guia do usuário do IAM.
- (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adição de permissões a um usuário (console) no Guia do usuário do IAM.

8. Na caixa de pesquisa, digite o nome da política gerenciada pelo cliente que você criou. Em seguida, marque a caixa de seleção Nome da política.
9. Escolha Próximo: tags.
10. Escolha Próximo: revisar.
11. Escolha Criar usuário.
12. Escolha Fechar.

O Device Advisor exige acesso aos seus AWS recursos (itens, certificados e endpoints) em seu nome. O usuário do IAM deve ter as permissões necessárias para: O Device Advisor também publicará registros na Amazon CloudWatch se você anexar a política de permissões necessária ao seu usuário do IAM.

Configurar o dispositivo

O Device Advisor usa a extensão TLS Server Name Indication (SNI) para aplicar configurações de TLS. Os dispositivos devem usar essa extensão quando se conectam e transmitem um nome de servidor idêntico ao endpoint de teste do Device Advisor.

O Device Advisor permite a conexão TLS quando um teste está no estado `Running`. Ele nega a conexão TLS antes e depois de cada execução de teste. Por isso, recomenda-se usar o mecanismo de nova tentativa de conexão do dispositivo para uma experiência de teste totalmente automatizada com o Device Advisor. Você pode executar conjuntos de teste que incluem mais de um caso de teste, como TLS Connect, MQTT Connect e MQTT Publish. Se você executar vários casos de teste, recomendamos que o dispositivo tente se conectar ao nosso endpoint de teste a cada cinco segundos. Em seguida, você pode automatizar a execução de vários casos de teste em sequência.

> **Note**
>
> Para preparar o software do dispositivo para testes, recomendamos que você use um SDK que possa se conectar ao AWS IoT Core. Em seguida, você deve atualizar o SDK com o endpoint de teste do Device Advisor fornecido para sua Conta da AWS.

O Device Advisor é compatível com dois tipos de endpoints: endpoints em nível de conta e em nível de dispositivo. Escolha o endpoint que seja mais adequado ao caso de uso. Para executar simultaneamente vários conjuntos de testes para dispositivos diferentes, use um endpoint no nível do dispositivo.

Execute o seguinte comando para obter o endpoint no nível do dispositivo:

Para clientes do MQTT que usam certificados de cliente X.509:

```
aws iotdeviceadvisor get-endpoint --thing-arn your-thing-arn
```

or

```
aws iotdeviceadvisor get-endpoint --certificate-arn your-certificate-arn
```

Para WebSocket clientes com mais de MQTT que usam o Signature versão 4:

```
aws iotdeviceadvisor get-endpoint --device-role-arn your-device-role-arn --
authentication-method SignatureVersion4
```

Para executar um conjunto de testes por vez, escolha um endpoint no nível da conta. Execute o seguinte comando para obter o endpoint no nível da conta:

```
aws iotdeviceadvisor get-endpoint
```

Conceitos básicos do Device Advisor no console

Este tutorial ajuda você a começar AWS IoT Core Device Advisor a usar o console. O Device Advisor oferece recursos como testes obrigatórios e relatórios de qualificação assinados. Você pode usar esses testes e relatórios para qualificar e listar dispositivos no Catálogo de dispositivos do parceiro da AWS, conforme detalhado no programa de qualificação do AWS IoT Core.

Para obter mais informações sobre o uso do Device Advisor, consulte Fluxo de trabalho do Device Advisor e Fluxo de trabalho detalhado do console do Device Advisor.

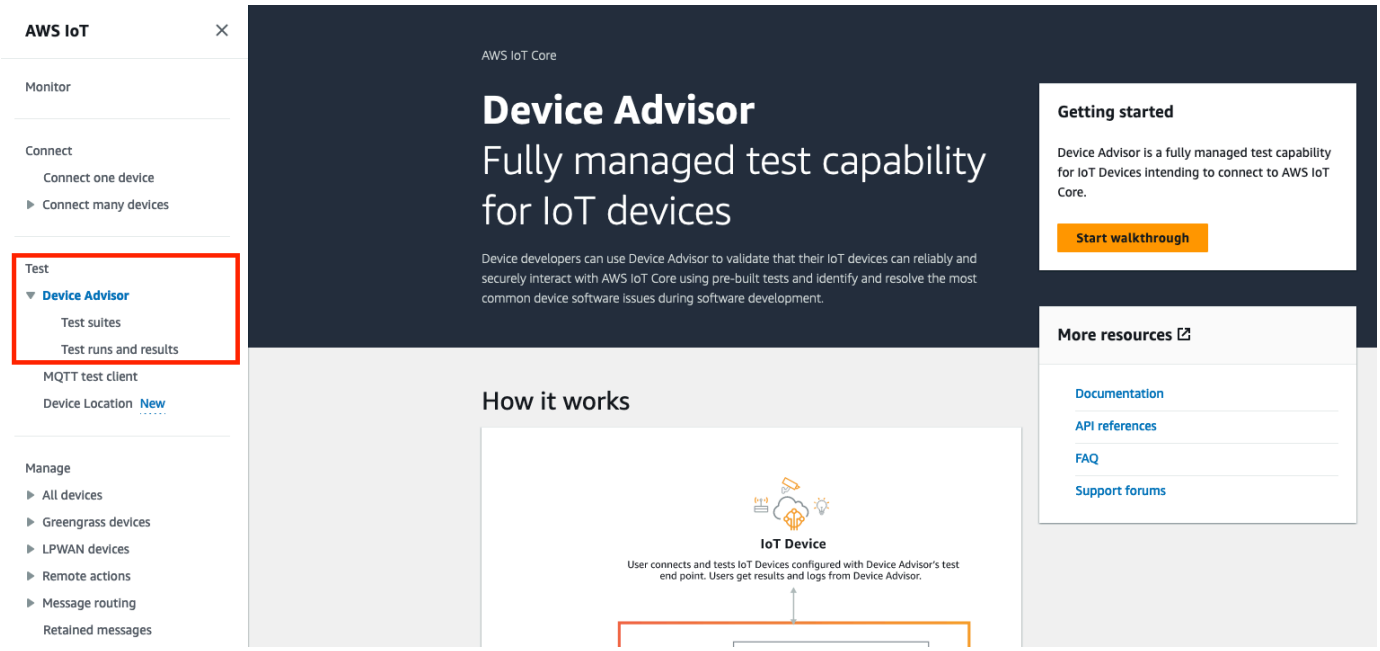
Para concluir este tutorial, siga as etapas descritas em Configurar.

> **Note**
>
> O Device Advisor é compatível com o seguinte Regiões da AWS:

- Leste dos EUA (N. da Virgínia)
- Oeste dos EUA (Oregon)
- Ásia-Pacífico (Tóquio)
- Europa (Irlanda)

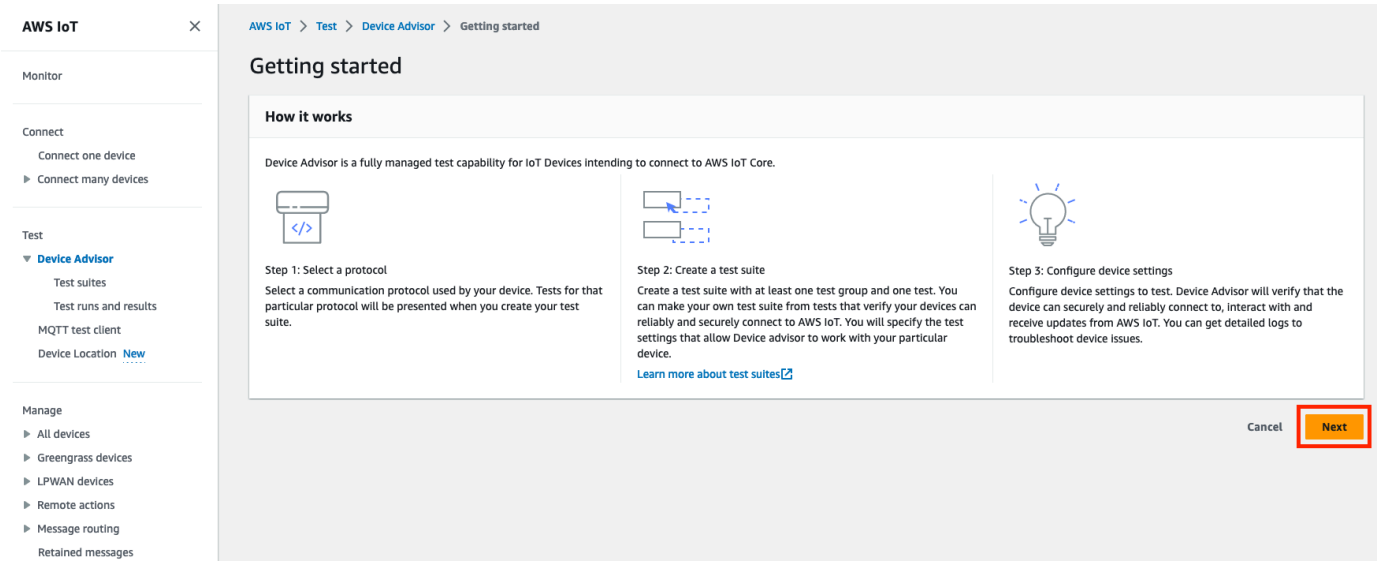
Introdução

1. No painel de navegação do console AWS IoT, em Teste, escolha Device Advisor. Em seguida, escolha o botão Iniciar passo a passo no console.

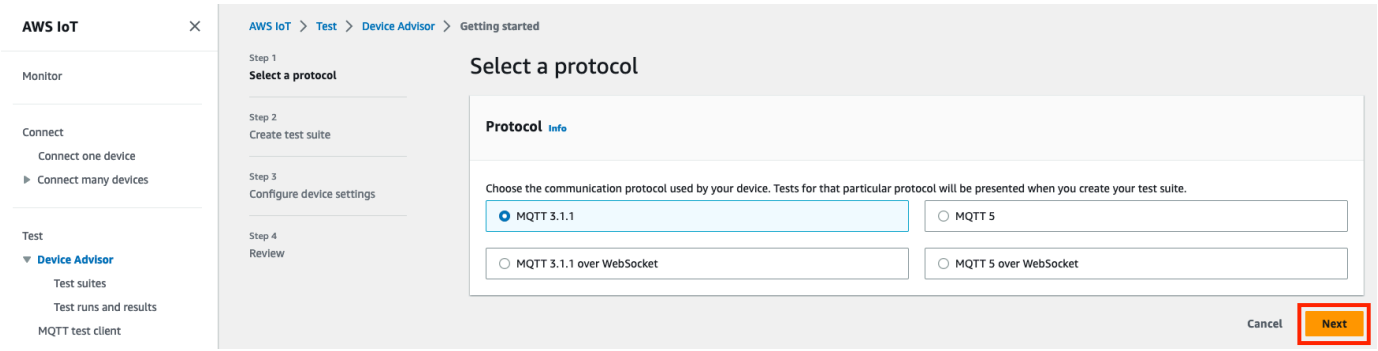


2. A página Conceitos básicos do Device Advisor fornece uma visão geral das etapas necessárias para criar um conjunto de testes e executar testes no dispositivo. Você também pode encontrar o endpoint de teste do Device Advisor para sua conta aqui. Você deve configurar o firmware ou o software no dispositivo usado para o teste para se conectar a esse endpoint de teste.

   Para concluir as etapas neste tutorial, primeiro é preciso criar um objeto e um certificado. Depois de revisar as informações em Como funciona, escolha Próximo.

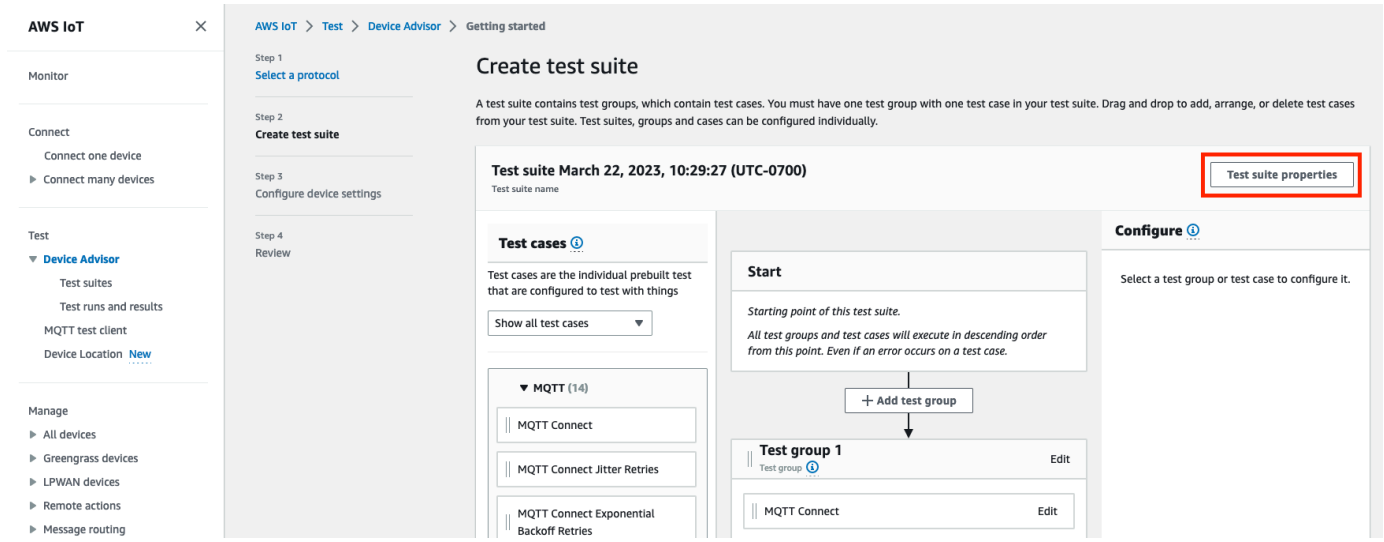


3. Na Etapa 1: selecionar um protocolo, selecione um protocolo nas opções listadas. Em seguida, escolha Próximo.



4. Na Etapa 2, você cria e configura um conjunto de testes personalizado. Um conjunto de testes personalizado deve ter pelo menos um grupo de teste, e cada grupo de teste deve ter pelo menos um caso de teste. Adicionamos o caso de teste do MQTT Connect para você começar.

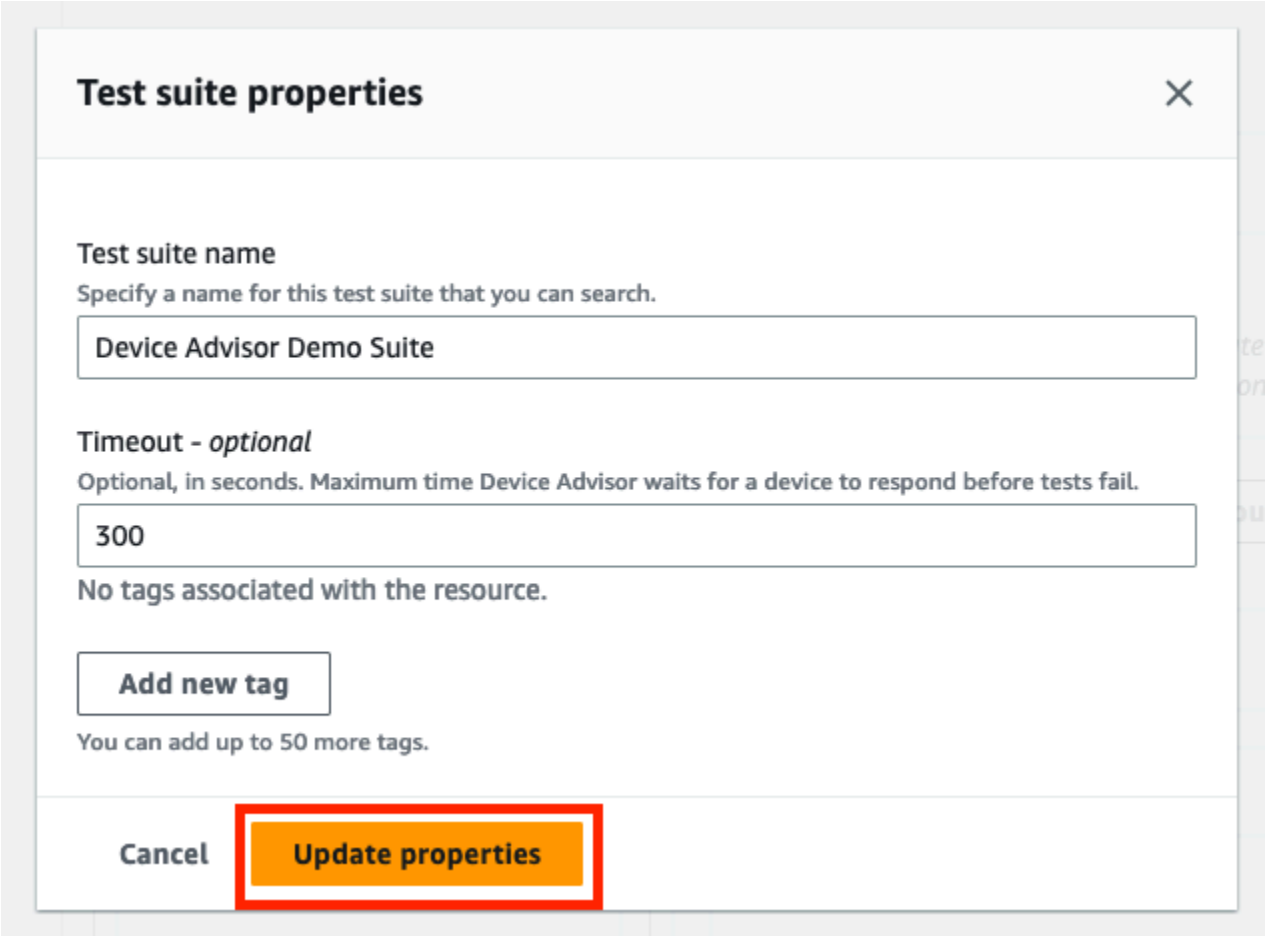
   Escolha Propriedades do conjunto de testes.



Forneça as propriedades do conjunto de testes ao criar o conjunto de testes. Você pode configurar as seguintes propriedades em nível de conjunto:

- Nome do conjunto de testes: insira um nome para o conjunto de testes.
- Tempo limite (opcional): o tempo limite (em segundos) para cada caso de teste no conjunto de testes atual. Se você não especificar um valor, o valor padrão será usado.
- Tags (opcional): adicione tags ao conjunto de testes.

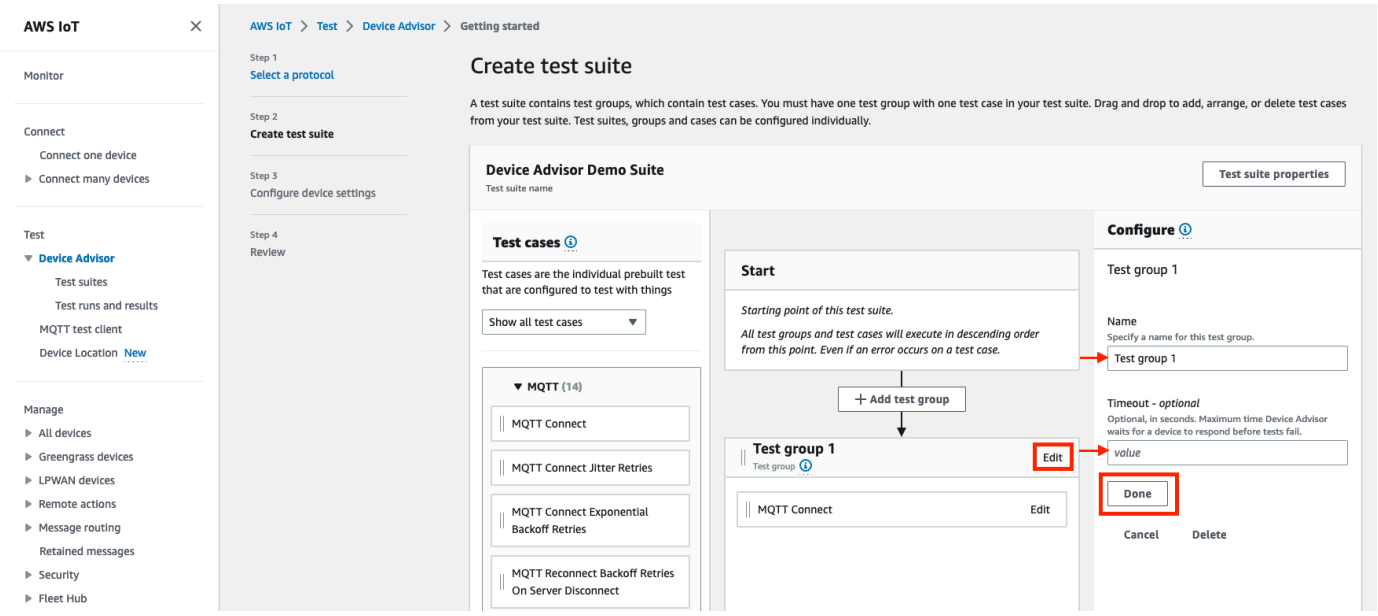
Quando terminar, escolha Atualizar propriedades.



5. (Opcional) Para atualizar a configuração do grupo do conjunto de testes, escolha o botão Editar ao lado do nome do grupo de teste.

   - Nome: insira um nome para o grupo de conjunto de teste.
   - Tempo limite (opcional): o tempo limite (em segundos) para cada caso de teste no conjunto de testes atual. Se você não especificar um valor, o valor padrão será usado.

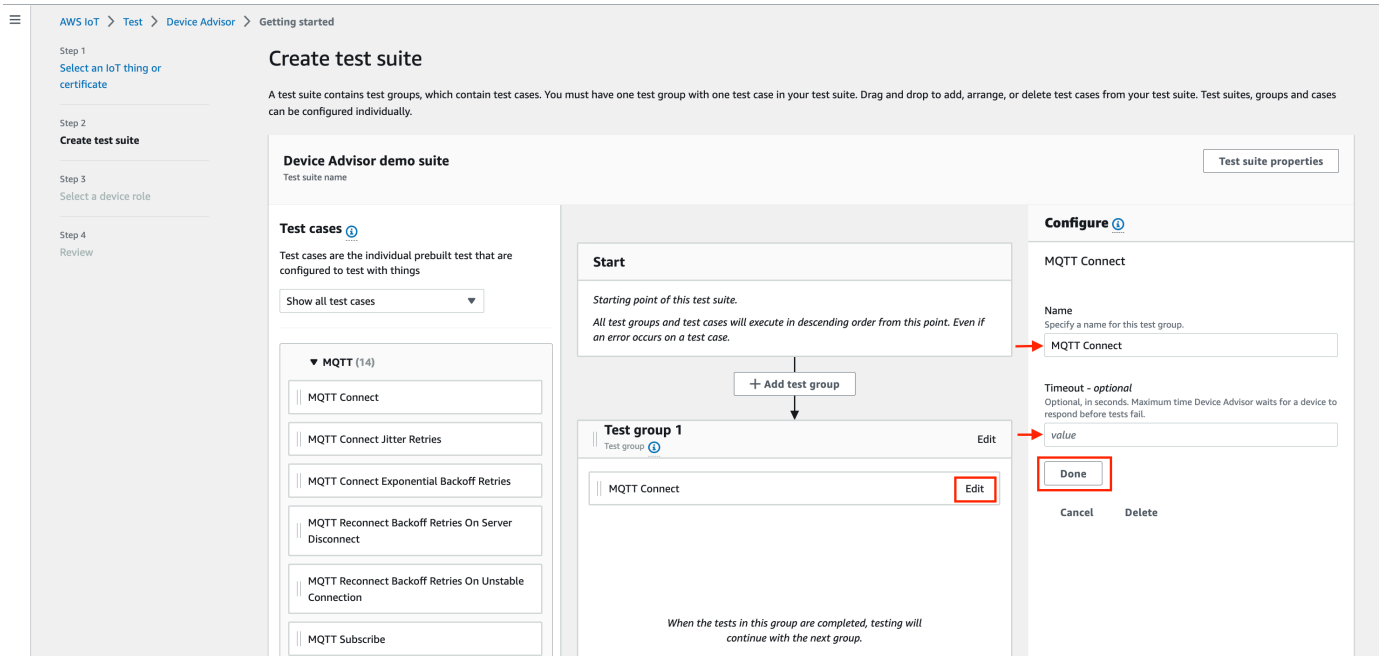
   Ao terminar, escolha Concluído para continuar.



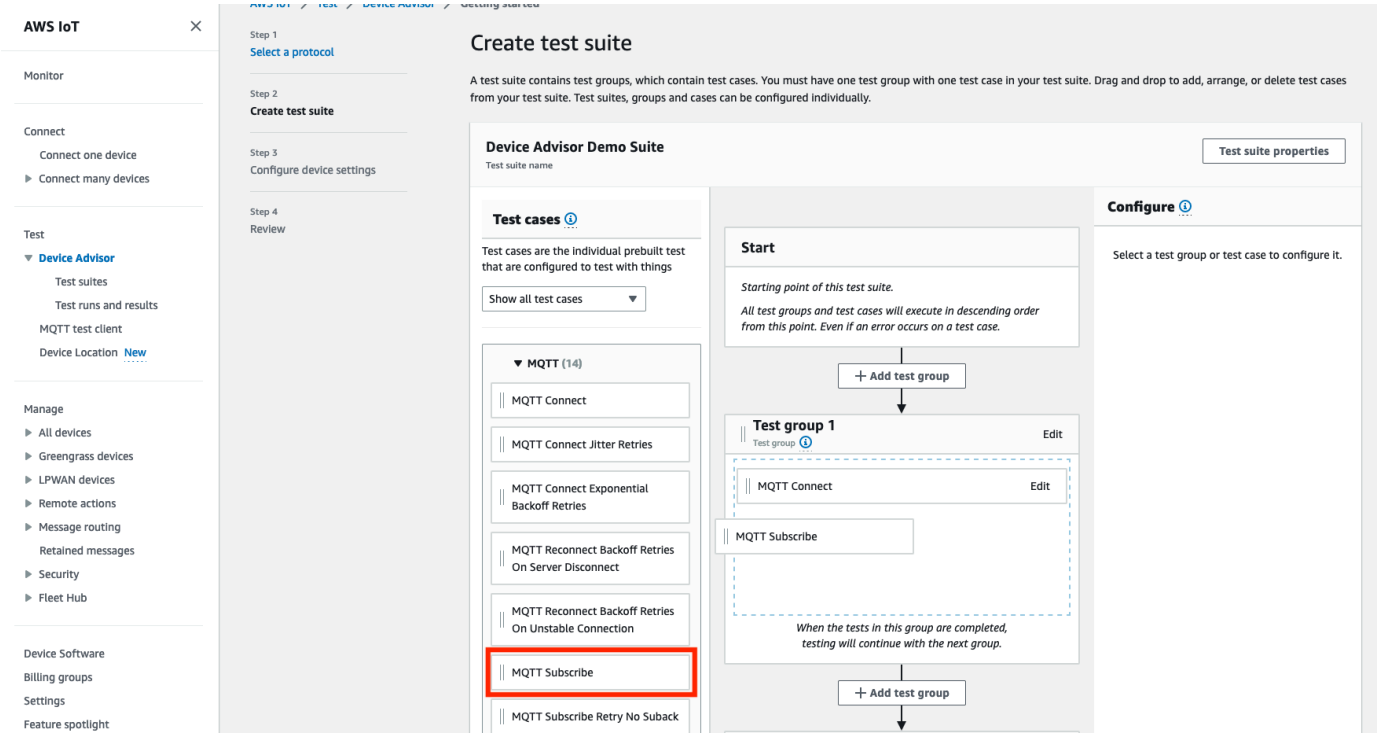
6. (Opcional) Para atualizar a configuração de um caso de teste, escolha o botão Editar ao lado do nome do caso de teste.

   - Nome: insira um nome para o grupo de conjunto de teste.
   - Tempo limite (opcional): o tempo limite (em segundos) para o caso de teste selecionado. Se você não especificar um valor, o valor padrão será usado.

   Ao terminar, escolha Concluído para continuar.

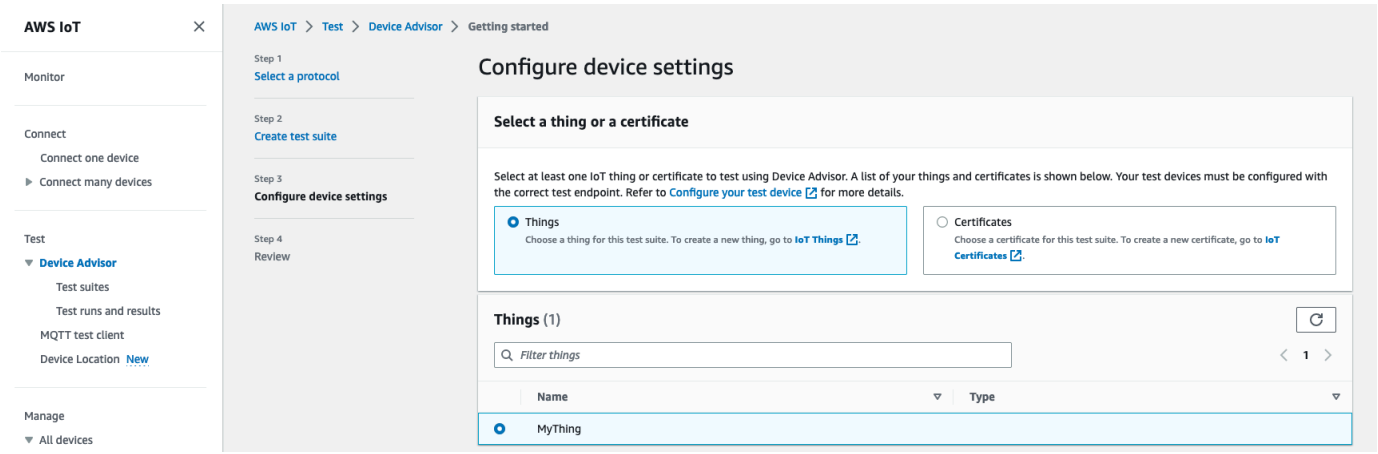


7. (Opcional) Para adicionar mais grupos de teste ao conjunto de testes, escolha Adicionar grupo de teste e siga as instruções na Etapa 5.
8. (Opcional) Para adicionar mais casos de teste, arraste os casos de teste na seção Casos de teste para qualquer um dos grupos de teste.



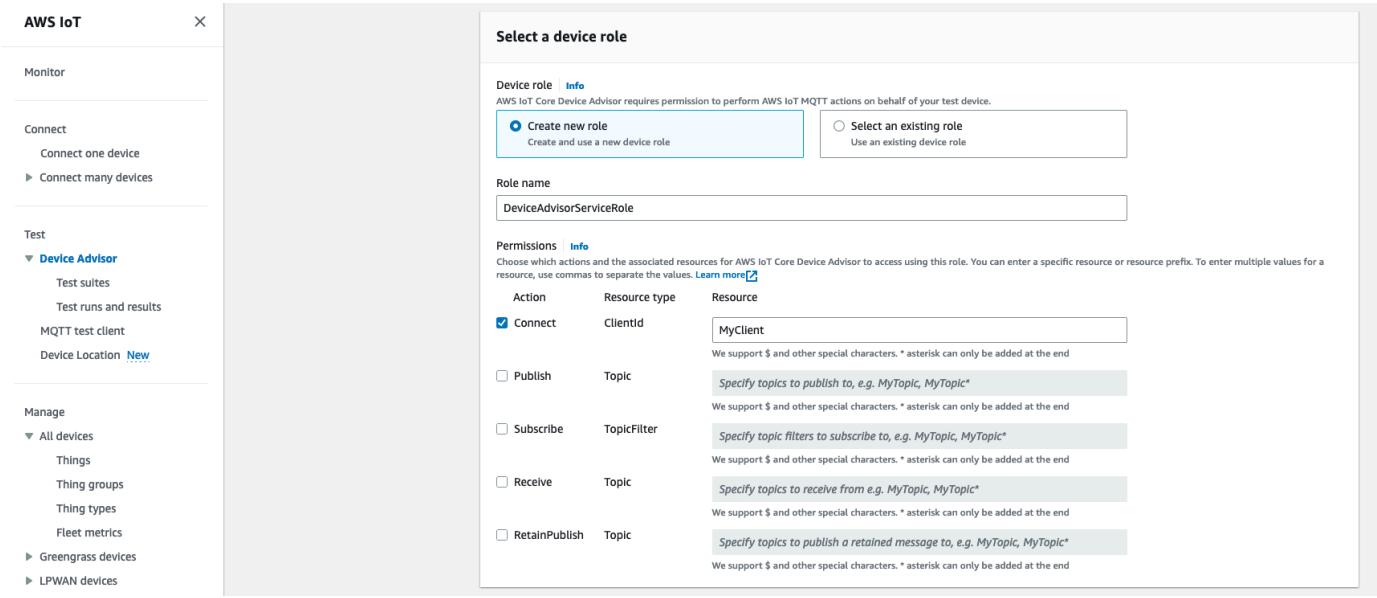
9. Você pode alterar a ordem dos grupos de teste e casos de teste. Para fazer alterações, arraste os casos de teste listados para cima ou para baixo na lista. O Device Advisor executa testes na ordem em que você os listou.

   Depois de configurar o conjunto de testes, escolha Próximo.
10. Na Etapa 3, selecione um AWS IoT item ou certificado para testar usando o Device Advisor. Se você não tiver nenhum objeto ou certificado existente, consulte Configuração.

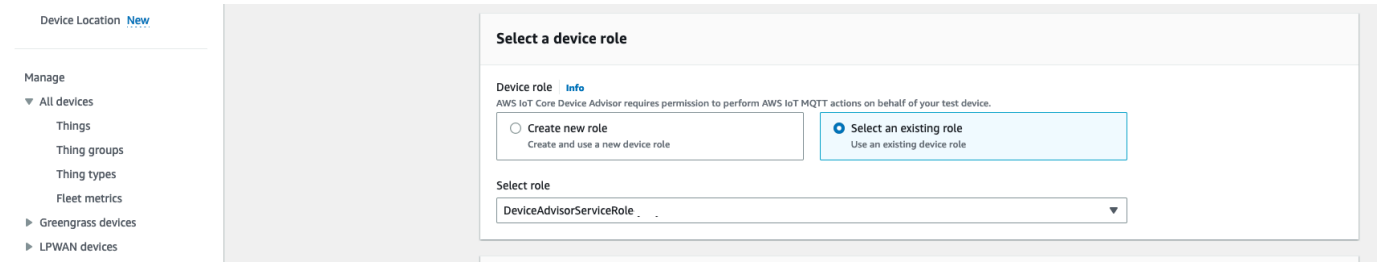


11. Você pode configurar uma função de dispositivo que o Device Advisor usa para realizar ações do AWS IoT MQTT em nome do seu dispositivo de teste. Somente para o caso de teste do MQTT Connect, a ação Conectar é selecionada automaticamente. Isso ocorre porque o perfil do dispositivo exige essa permissão para executar o conjunto de testes. Para outros casos de teste, as ações correspondentes são selecionadas.

    Forneça os valores dos recursos para cada uma das ações selecionadas. Por exemplo, para a ação Conectar, forneça o ID do cliente que o dispositivo usa para se conectar ao endpoint do Device Advisor. Você pode fornecer vários valores com valores separados por vírgula e valores de prefixo com um caractere curinga (*). Por exemplo, para fornecer permissão para publicar em qualquer tópico que comece com `MyTopic`, insira **`MyTopic*`** como valor do recurso.



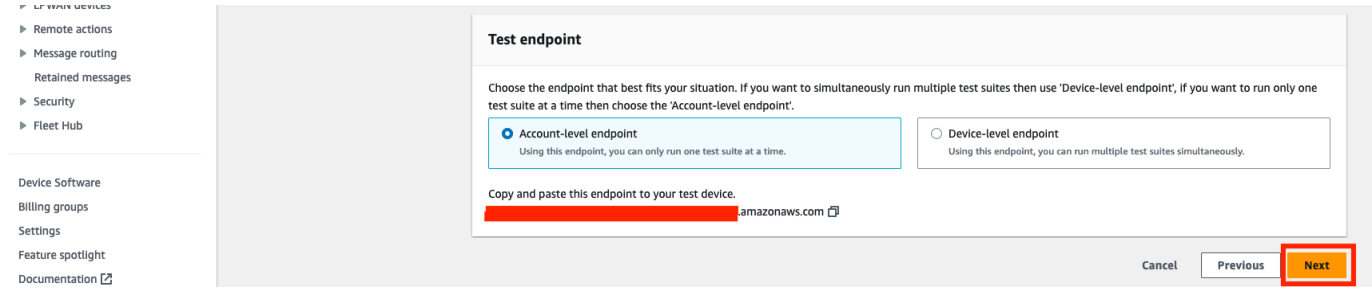
    Para usar um perfil de dispositivo criado anteriormente em Configuração, escolha Selecionar um perfil existente. Em seguida, escolha o perfil do dispositivo em Selecionar perfil.



    Configure o perfil do dispositivo com uma das duas opções fornecidas e escolha Próximo.

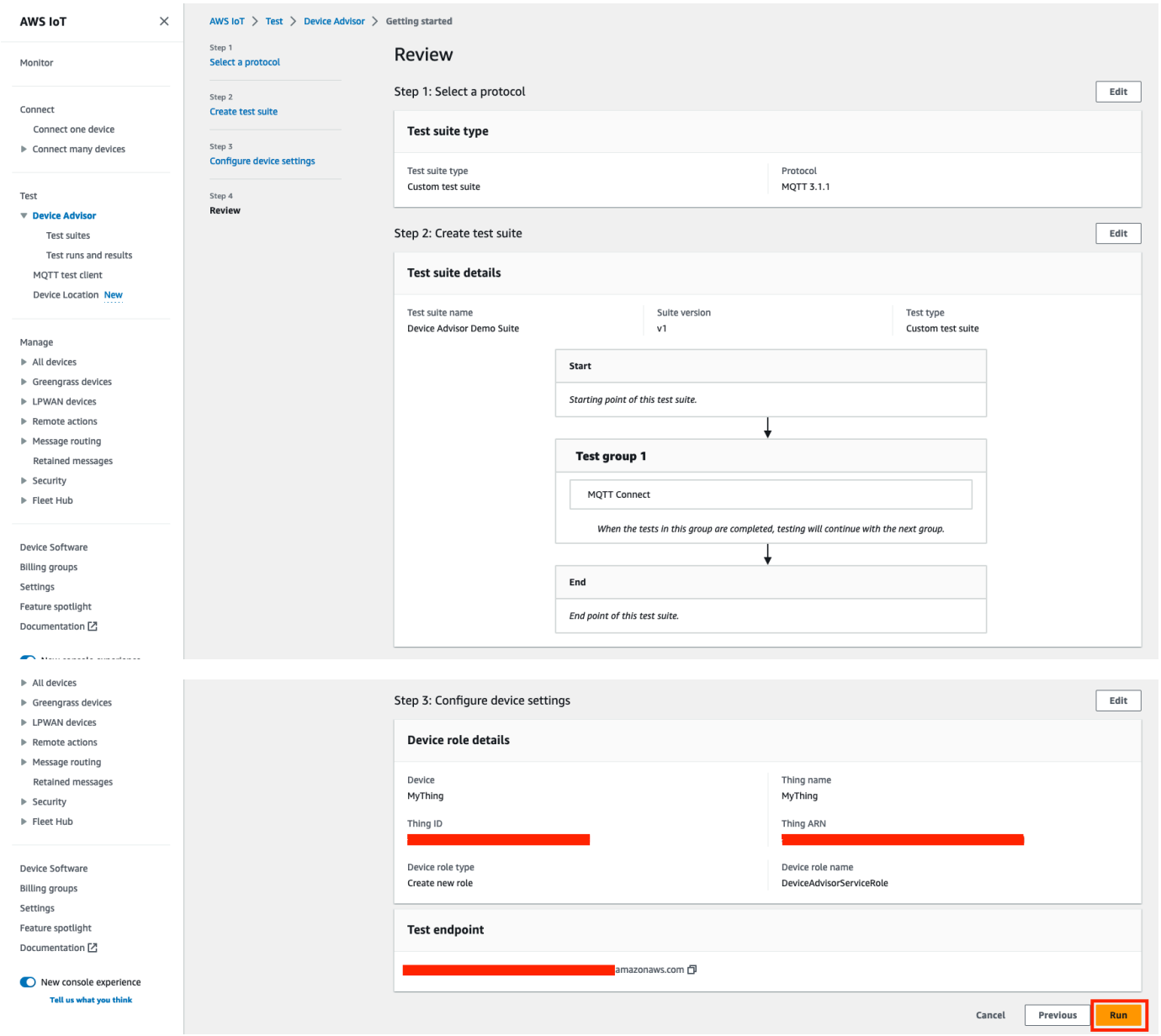
12. Na seção Endpoint de teste, selecione o endpoint que melhor se adapta ao caso de uso. Para executar várias suítes de testes simultaneamente com as mesmas Conta da AWS, selecione

Endpoint no nível do dispositivo. Para executar um conjunto de testes por vez, selecione Endpoint em nível de conta.

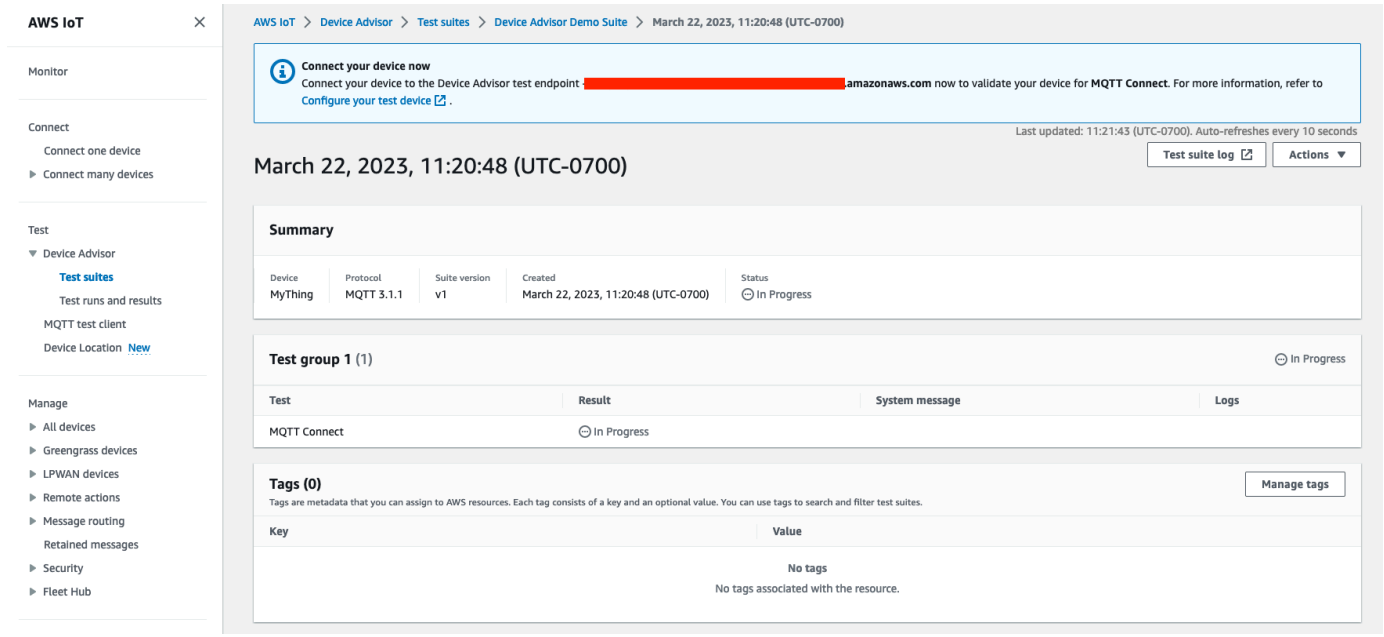


13. A Etapa 4 mostra uma visão geral do dispositivo de teste selecionado, do endpoint de teste, do conjunto de testes e do perfil do dispositivo de teste configurados. Selecione o botão Editar para qualquer seção que você deseja editar. Depois de confirmar a configuração de teste, escolha Executar para criar o conjunto de testes e executá-los.

> **Note**
>
> Para obter melhores resultados, você pode conectar o dispositivo de teste selecionado ao endpoint de teste do Device Advisor antes de iniciar a execução do conjunto de testes. Recomendamos que você tenha um mecanismo criado para que o dispositivo tente se conectar ao nosso endpoint de teste a cada cinco segundos por até um a dois minutos.



14. No painel de navegação, em Teste, escolha Device Advisor e, em seguida, escolha Execuções e resultados de testes. Selecione uma execução do conjunto de testes para ver os detalhes e os logs da execução.



15. Para acessar os CloudWatch registros da Amazon para a suíte, execute:

    - Escolha Registro da suíte de testes para ver os CloudWatch registros da execução da suíte de testes.
    - Escolha Registro do caso de teste para qualquer caso de teste para visualizar os registros específicos do caso CloudWatch de teste.

16. Com base nos resultados do teste, solucione os problemas do dispositivo até que todos os testes sejam aprovados.

Fluxo de trabalho do Device Advisor

Este tutorial explica como criar um conjunto de testes personalizado e executar testes no dispositivo que você deseja testar no console. Depois que os testes forem concluídos, você poderá visualizar os resultados do teste e os logs detalhados.

Pré-requisitos

Antes de iniciar este tutorial, conclua as etapas descritas em Configurar.

Criar uma definição de conjunto de teste

Primeiro, instale um AWS SDK.

Sintaxe de **`rootGroup`**

Um grupo raiz é uma string JSON que especifica quais casos de teste incluir no conjunto de testes. Ele também especifica todas as configurações necessárias para esses casos de teste. Use o grupo raiz para estruturar e ordenar o conjunto de testes com base em suas necessidades. A hierarquia de um conjunto de testes é:

```
test suite # test group(s) # test case(s)
```

Um conjunto de testes deve ter pelo menos um grupo de teste, e cada grupo de teste deve ter pelo menos um caso de teste. O Device Advisor executa testes na ordem em que você define os grupos de teste e os casos de teste.

Cada grupo raiz segue essa estrutura básica:

```
{
    "configuration": {  // for all tests in the test suite
        "": ""
    }
    "tests": [{
        "name": ""
        "configuration": {  // for all sub-groups in this test group
            "": ""
        },
        "tests": [{
            "name": ""
            "configuration": {  // for all test cases in this test group
                "": ""
            },
            "test": {
                "id": ""
                "version": ""
            }
        }]
    }]
}
```

No grupo raiz, você define o conjunto de testes com um `name`, um `configuration` e os `tests` que o grupo contém. O grupo `tests` contém as definições de testes individuais. Você define cada teste com um `name`, um `configuration` e um bloco `test` que define os casos de teste desse teste. Finalmente, cada caso de teste é definido com um `id` e um `version`.

Para obter informações sobre como usar os campos `"id"` e `"version"` para cada caso de teste (bloco `test`), consulte Casos de teste do Device Advisor. Essa seção também contém informações sobre as configurações `configuration` disponíveis.

O bloco a seguir é um exemplo de configuração de grupo raiz. Essas configurações especificam os casos de teste MQTT Connect Happy Case e MQTT Connect Exponential Backoff Retries, junto com as descrições dos campos de configuração.

```
{
    "configuration": {},  // Suite-level configuration
    "tests": [            // Group definitions should be provided here
      {
        "name": "My_MQTT_Connect_Group",  // Group definition name
        "configuration": {}               // Group definition-level configuration,
        "tests": [                        // Test case definitions should be provided
 here
        {
            "name": "My_MQTT_Connect_Happy_Case",  // Test case definition name
            "configuration": {
                "EXECUTION_TIMEOUT": 300        // Test case definition-level
 configuration, in seconds
            },
            "test": {
                "id": "MQTT_Connect",              // test case id
                "version": "0.0.0"                 // test case version
            }
        },
        {
            "name": "My_MQTT_Connect_Jitter_Backoff_Retries",  // Test case definition
 name
            "configuration": {
                "EXECUTION_TIMEOUT": 600                // Test case definition-level
 configuration,  in seconds
            },
            "test": {
                "id": "MQTT_Connect_Jitter_Backoff_Retries",  // test case id
                "version": "0.0.0"                                  // test case version
            }
        }]
    }]
}
```

Você deve fornecer a configuração do grupo raiz ao criar a definição de conjunto de testes. Salve o `suiteDefinitionId` que é retornado no objeto da resposta. Você pode usar esse ID para recuperar as informações de definição do conjunto de testes e executar o conjunto de testes.

Aqui está um exemplo do Java SDK:

```
response = iotDeviceAdvisorClient.createSuiteDefinition(
        CreateSuiteDefinitionRequest.builder()
            .suiteDefinitionConfiguration(SuiteDefinitionConfiguration.builder()
                .suiteDefinitionName("your-suite-definition-name")
                .devices(
                    DeviceUnderTest.builder()
                        .thingArn("your-test-device-thing-arn")
                        .certificateArn("your-test-device-certificate-arn")
                        .deviceRoleArn("your-device-role-arn") //if using SigV4 for
 MQTT over WebSocket
                        .build()
                )
                .rootGroup("your-root-group-configuration")
                .devicePermissionRoleArn("your-device-permission-role-arn")
                .protocol("MqttV3_1_1 || MqttV5 || MqttV3_1_1_OverWebSocket ||
 MqttV5_OverWebSocket")
                .build()
            )
            .build()
)
```

Obtenha uma definição de conjunto de teste

Depois de criar a definição de conjunto de testes, você recebe o `suiteDefinitionId` no objeto de resposta da operação da API `CreateSuiteDefinition`.

Quando a operação retorna o `suiteDefinitionId`, você pode ver novos campos `id` em cada grupo e a definição do caso de teste dentro do grupo raiz. Você pode usá-los IDs para executar um subconjunto da definição do seu conjunto de testes.

Exemplo de Java SDK:

```
response = iotDeviceAdvisorClient.GetSuiteDefinition(
    GetSuiteDefinitionRequest.builder()
        .suiteDefinitionId("your-suite-definition-id")
        .build()
```

```
)
```

Obtenha um endpoint de teste

Use a operação da API `GetEndpoint` para obter o endpoint de teste usado pelo dispositivo. Selecione o endpoint que melhor se adapta ao teste. Para executar vários conjuntos de testes simultaneamente, use o endpoint em nível de dispositivo fornecendo um `thing ARN`, um `certificate ARN` ou um `device role ARN`. Para executar uma única suíte de testes, não forneça argumentos à GetEndpoint operação para escolher o endpoint no nível da conta.

Exemplo do SDK:

```
response = iotDeviceAdvisorClient.getEndpoint(GetEndpointRequest.builder()
.certificateArn("your-test-device-certificate-arn")
.thingArn("your-test-device-thing-arn")
.deviceRoleArn("your-device-role-arn") //if using SigV4 for MQTT over WebSocket

.build())
```

Inicie a execução de um conjunto de testes

Depois de criar uma definição de conjunto de testes e configurar o dispositivo de teste para se conectar ao endpoint de teste do Device Advisor, execute o conjunto de testes com a API `StartSuiteRun`.

Para clientes do MQTT, use um `certificateArn` ou um `thingArn` para executar o conjunto de testes. Se ambos estiverem configurados, o certificado será usado se pertencer ao objeto.

Para MQTT em vez WebSocket do cliente, use `deviceRoleArn` para executar a suíte de testes. Se o perfil especificado for diferente do perfil especificado na definição do conjunto de testes, o perfil especificado substituirá o perfil definido.

Para `.parallelRun()`, use `true` se você usar um endpoint em nível de dispositivo para executar vários conjuntos de testes em paralelo usando uma Conta da AWS.

Exemplo do SDK:

```
response = iotDeviceAdvisorClient.startSuiteRun(StartSuiteRunRequest.builder()
.suiteDefinitionId("your-suite-definition-id")
.suiteRunConfiguration(SuiteRunConfiguration.builder()
```

```
    .primaryDevice(DeviceUnderTest.builder()
        .certificateArn("your-test-device-certificate-arn")
        .thingArn("your-test-device-thing-arn")
        .deviceRoleArn("your-device-role-arn") //if using SigV4 for MQTT over WebSocket

        .build())
    .parallelRun(true | false)
    .build())
.build())
```

Salve o `suiteRunId` da resposta. Você usará isso para recuperar os resultados da execução desse conjunto de testes.

Obtenha a execução de um conjunto de testes

Depois de iniciar a execução de um conjunto de testes, você pode verificar o progresso e os resultados com a API `GetSuiteRun`.

Exemplo do SDK:

```
// Using the SDK, call the GetSuiteRun API.

response = iotDeviceAdvisorClient.GetSuiteRun(
GetSuiteRunRequest.builder()
    .suiteDefinitionId("your-suite-definition-id")
    .suiteRunId("your-suite-run-id")
.build())
```

Interromper a execução de um conjunto de testes

Para interromper a execução de um conjunto de testes que ainda está em andamento, você pode chamar a operação da API `StopSuiteRun`. Depois de chamar a operação `StopSuiteRun`, o serviço inicia o processo de limpeza. Enquanto o serviço executa o processo de limpeza, o conjunto de testes executa atualizações de status para `Stopping`. O processo pode demorar vários minutos. Quando o processo estiver concluído, o conjunto de testes executará atualizações de status para `Stopped`. Depois que a execução de um teste for completamente interrompida, você poderá iniciar outra execução do conjunto de testes. Você pode verificar periodicamente o status de execução do conjunto usando a operação da API `GetSuiteRun`, conforme mostrado na seção anterior.

Exemplo do SDK:

```
// Using the SDK, call the StopSuiteRun API.

response = iotDeviceAdvisorClient.StopSuiteRun(
StopSuiteRun.builder()
    .suiteDefinitionId("your-suite-definition-id")
    .suiteRunId("your-suite-run-id")
.build())
```

Obtenha um relatório de qualificação para uma execução bem-sucedida do conjunto de testes de qualificação

Se você executar um conjunto de testes de qualificação concluído com êxito, poderá recuperar um relatório de qualificação com a operação da API `GetSuiteRunReport`. Você usa esse relatório de qualificação para qualificar o dispositivo com o programa de qualificação AWS IoT Core . Para determinar se o conjunto de testes é um conjunto de testes de qualificação, verifique se o parâmetro `intendedForQualification` está definido como `true`. Depois de chamar a operação da API `GetSuiteRunReport`, você pode baixar o relatório do URL retornado por até 90 segundos. Se passarem mais de 90 segundos desde a última vez em que você chamou a operação `GetSuiteRunReport`, chame a operação novamente para recuperar um novo URL válido.

Exemplo do SDK:

```
// Using the SDK, call the getSuiteRunReport API.

response = iotDeviceAdvisorClient.getSuiteRunReport(
    GetSuiteRunReportRequest.builder()
        .suiteDefinitionId("your-suite-definition-id")
        .suiteRunId("your-suite-run-id")
        .build()
)
```

Fluxo de trabalho detalhado do console do Device Advisor

Neste tutorial, você criará um conjunto de testes personalizado e executará testes no dispositivo que você deseja testar no console. Depois que os testes forem concluídos, você poderá visualizar os resultados do teste e os logs detalhados.

Tutoriais

- Pré-requisitos
- Criar uma definição de conjunto de teste
- Inicie a execução de um conjunto de testes
- Interromper a execução de um conjunto de testes (opcional)
- Exibir detalhes e logs da execução do conjunto de testes
- Baixe um relatório AWS IoT de qualificação

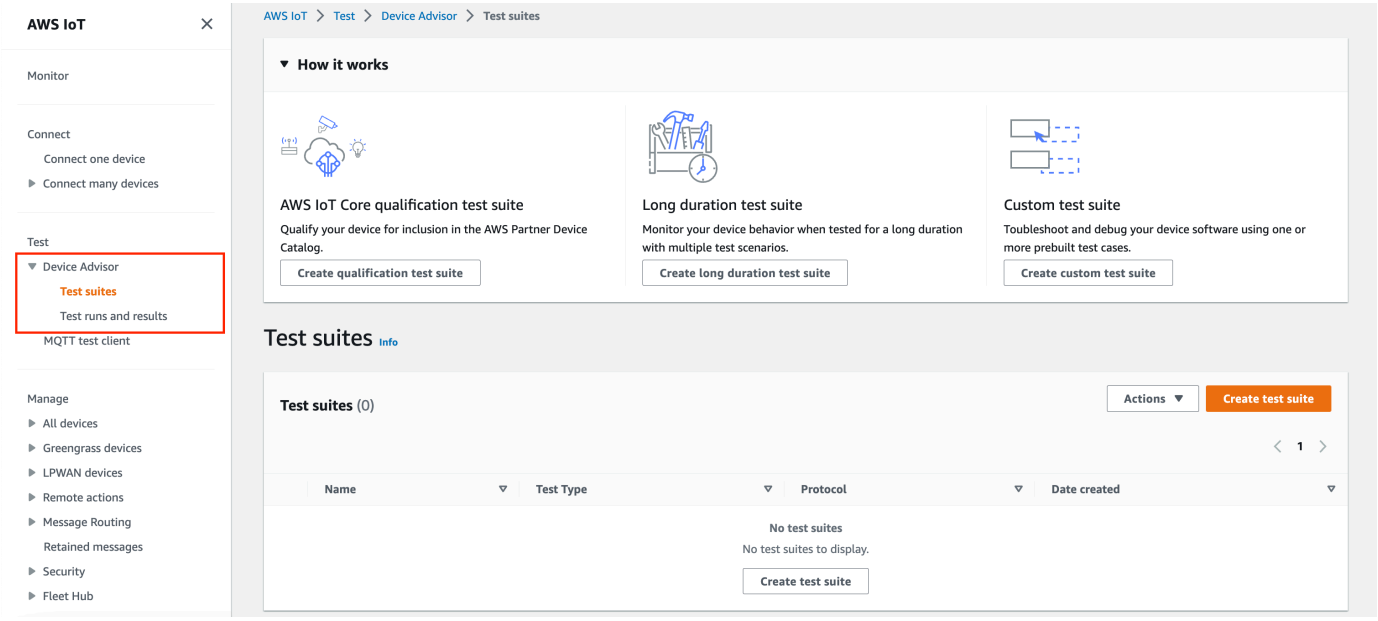
Pré-requisitos

Para concluir as etapas neste tutorial, você precisa criar um objeto e um certificado.

Criar uma definição de conjunto de teste

Crie um pacote de testes para que você possa executá-lo em seus dispositivos e realizar a verificação.

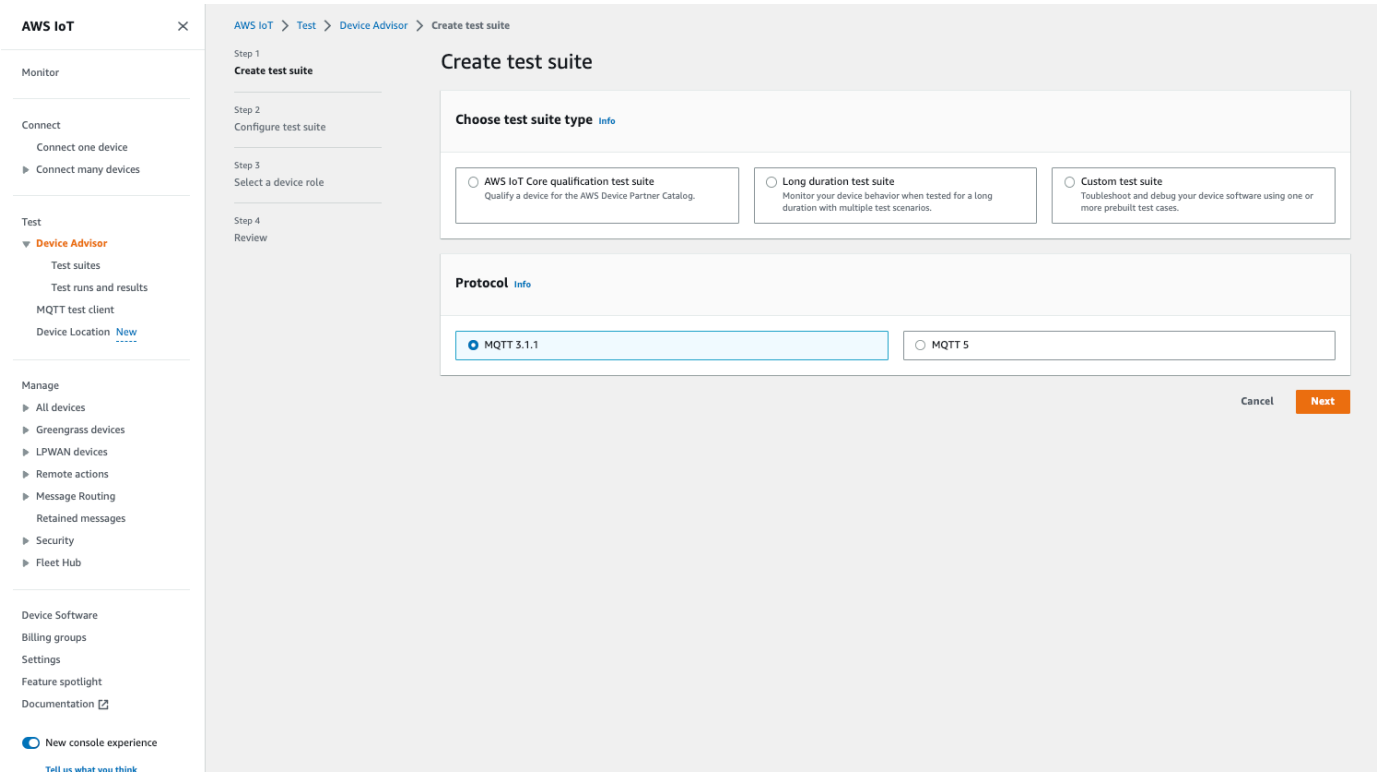
1. No console AWS IoT, no painel de navegação, expanda Teste, Device Advisor e escolha Conjuntos de testes.



   Escolha Criar conjunto de testes.

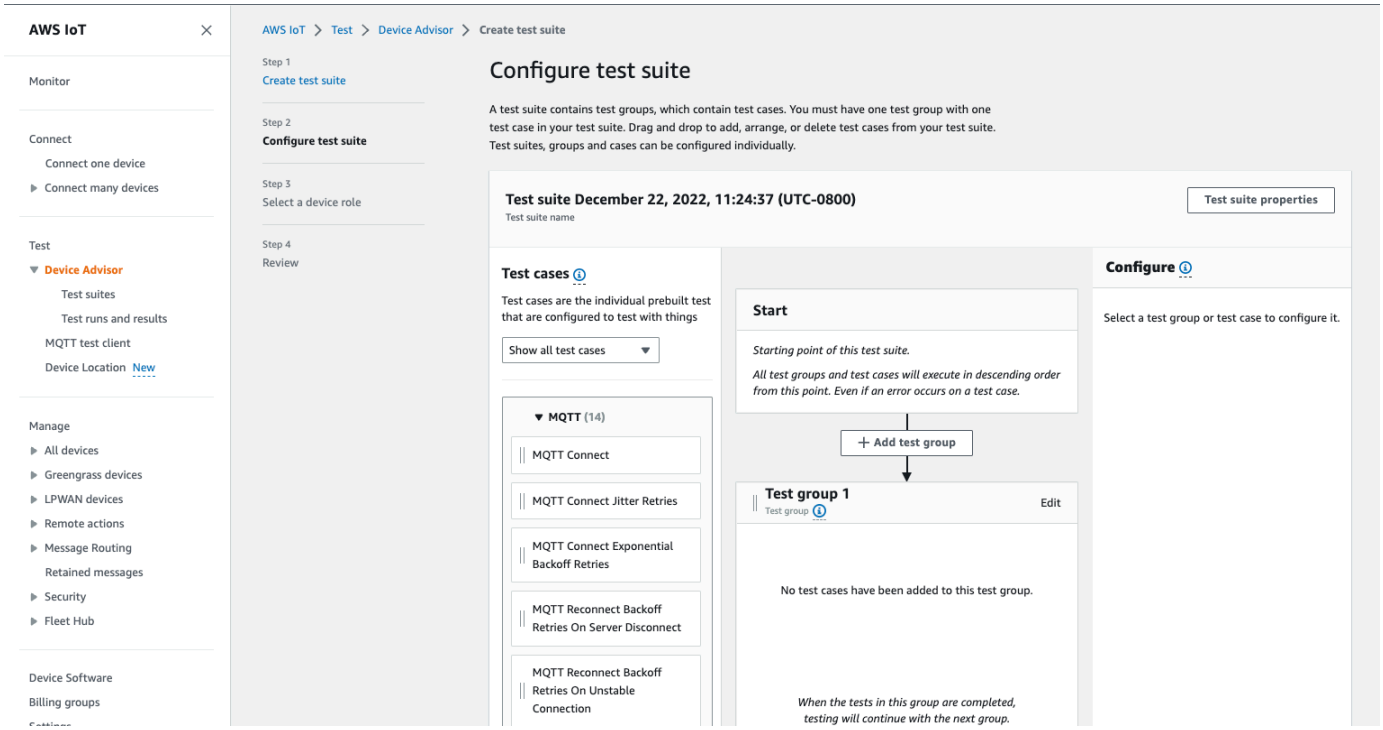
2. Selecione `Use the AWS Qualification test suite` ou `Create a new test suite`.

   Para protocolo, escolha MQTT 3.1.1 ou MQTT 5.

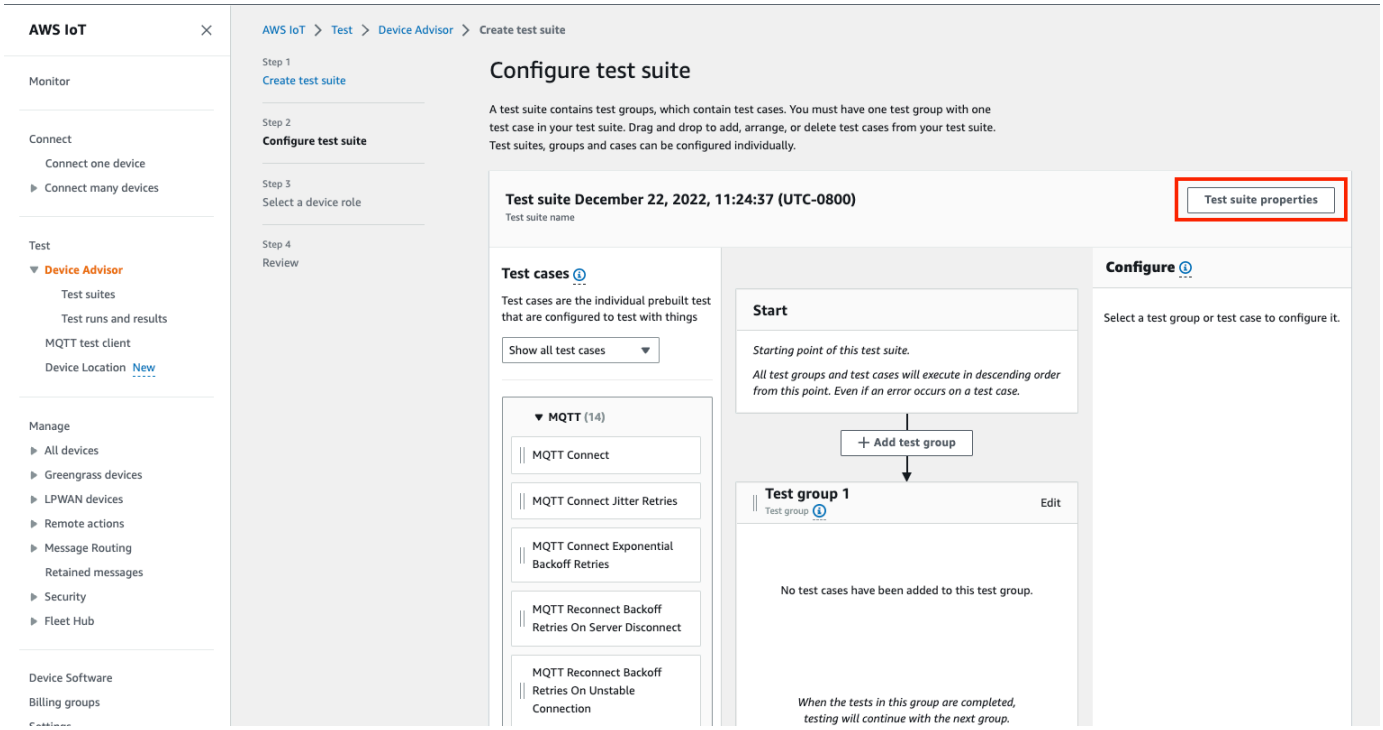


Selecione `Use the AWS Qualification test suite` para se qualificar e listar seu dispositivo no Catálogo de dispositivos AWS parceiros. Ao escolher essa opção, os casos de teste necessários para a qualificação do dispositivo para o programa de qualificação AWS IoT Core são pré-selecionados. Grupos de teste e casos de teste não podem ser adicionados nem removidos. Você ainda precisará configurar as propriedades do conjunto de testes.

Selecione `Create a new test suite` para criar e configurar um conjunto de testes personalizado. Recomendamos começar com essa opção para testes iniciais e solução de problemas. Um conjunto de testes personalizado deve ter pelo menos um grupo de teste, e cada grupo de teste deve ter pelo menos um caso de teste. Para fins deste tutorial, selecionaremos essa opção e escolheremos Próximo.



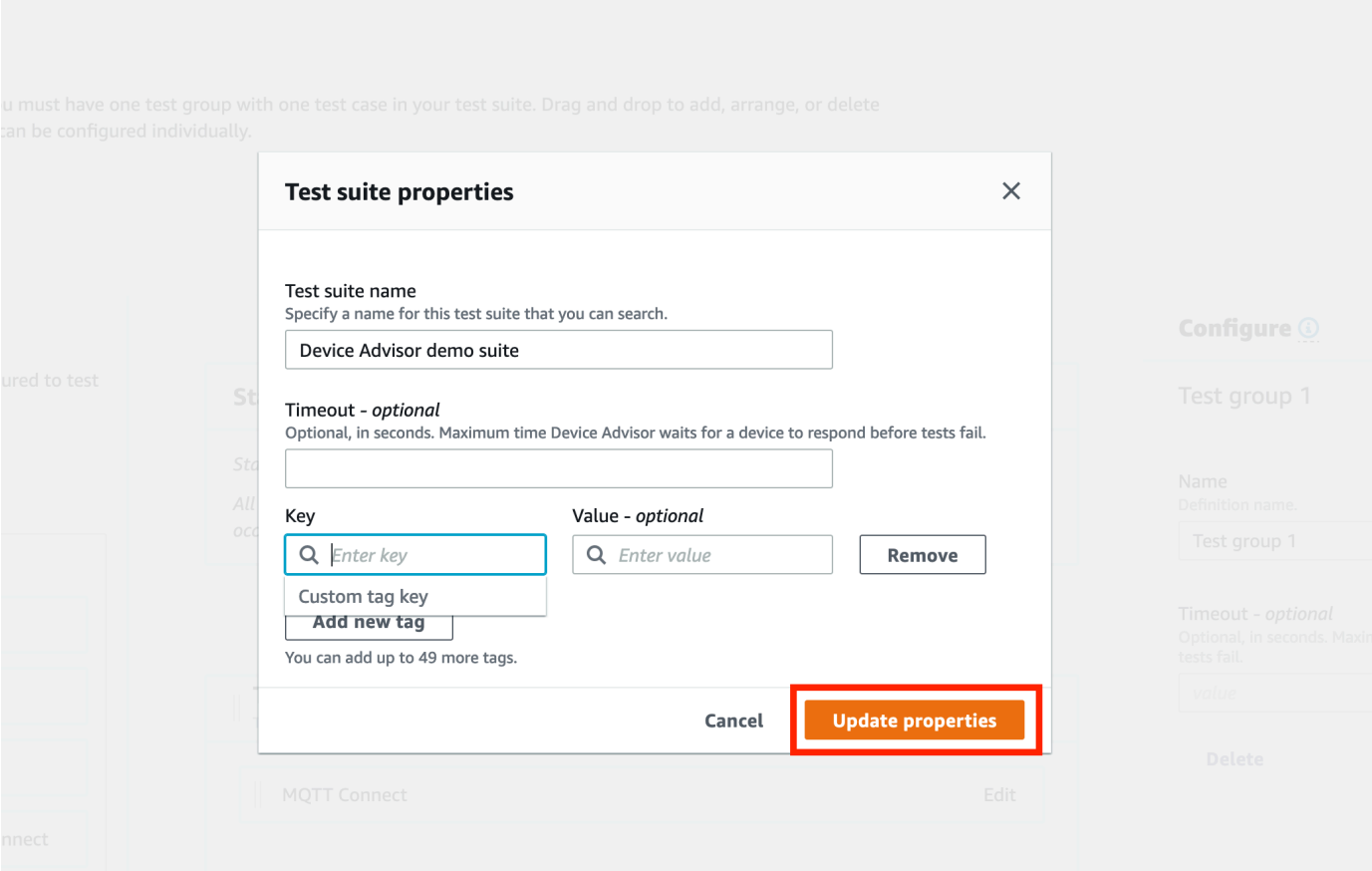
3. Escolha Propriedades do conjunto de testes. Você deve criar as propriedades do conjunto de testes ao criar o conjunto de testes.



Em Propriedades do conjunto de testes, preencha o seguinte:

- Nome do conjunto de teste: você pode criar o conjunto com um nome personalizado.

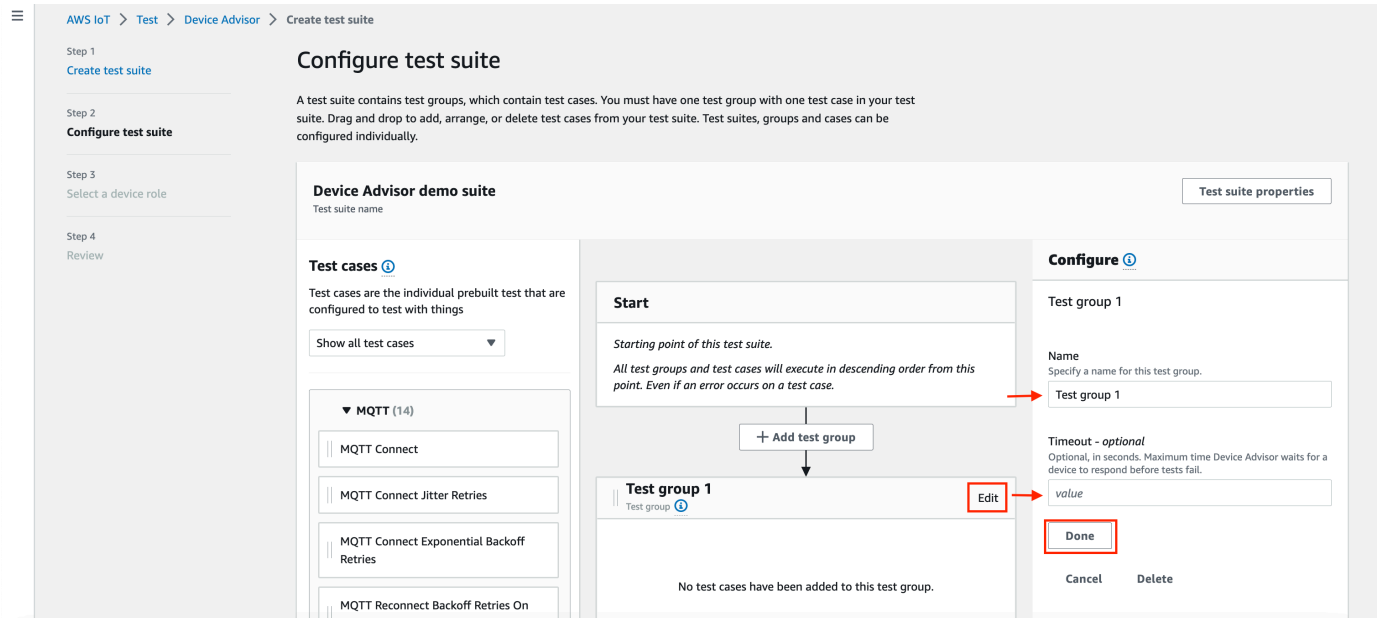
- Tempo limite (opcional): o tempo limite em segundos para cada caso de teste no conjunto de testes atual. Se você não especificar um valor, o valor padrão será usado.
- Tags (opcional): adicione tags ao conjunto de testes.



Quando terminar, escolha Atualizar propriedades.

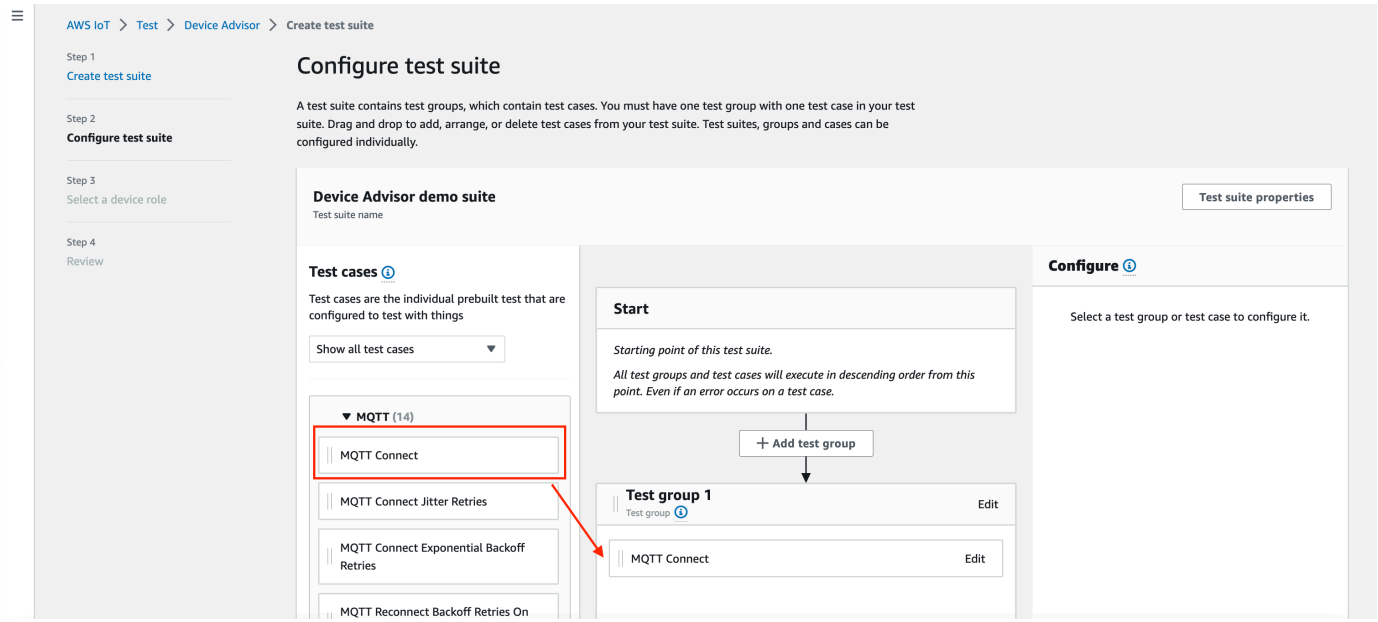
4. Para modificar a configuração em nível de grupo, em `Test group 1`, escolha Editar. Em seguida, insira um Nome para dar ao grupo um nome personalizado.

   Opcionalmente, você também pode inserir um valor de Tempo limite em segundos no grupo de teste selecionado. Se você não especificar um valor, o valor padrão será usado.



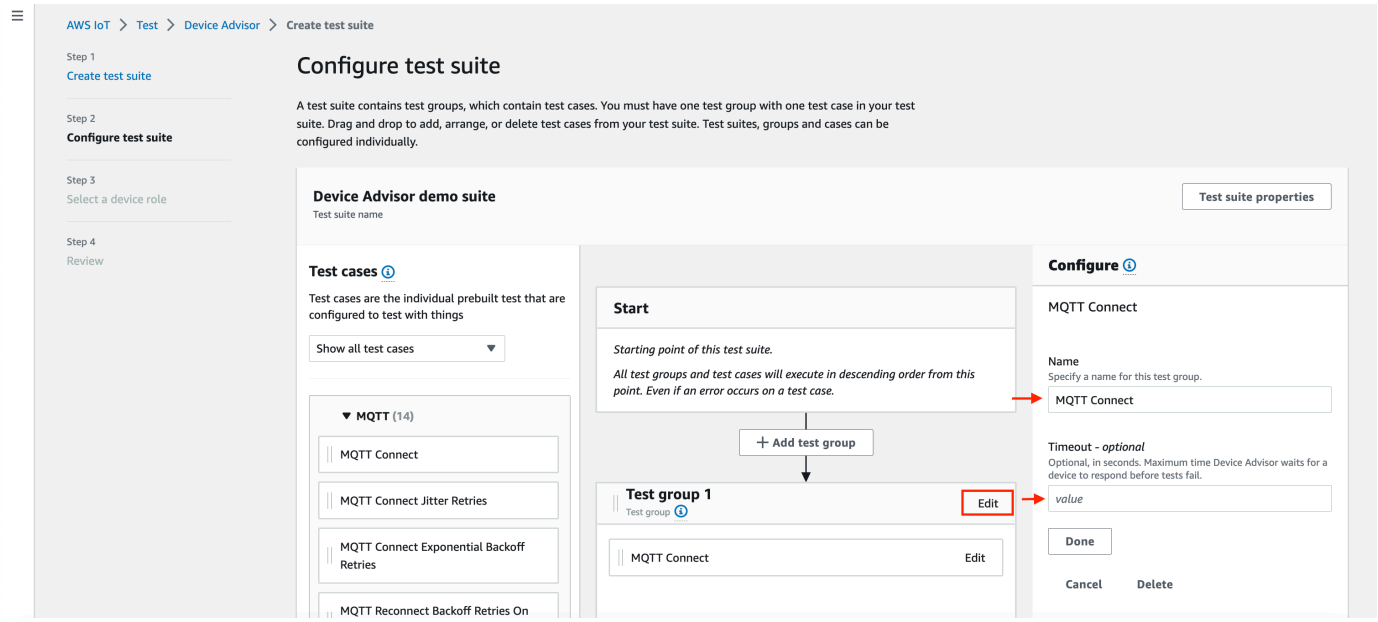
Escolha Concluído.

5. Arraste um dos casos de teste disponíveis de Casos de teste para o grupo de teste.



6. Para modificar a configuração do nível do caso de teste para o caso de teste que você adicionou ao grupo de teste, escolha Editar. Em seguida, insira um Nome para dar ao grupo um nome personalizado.

   Opcionalmente, você também pode inserir um valor de Tempo limite em segundos no grupo de teste selecionado. Se você não especificar um valor, o valor padrão será usado.



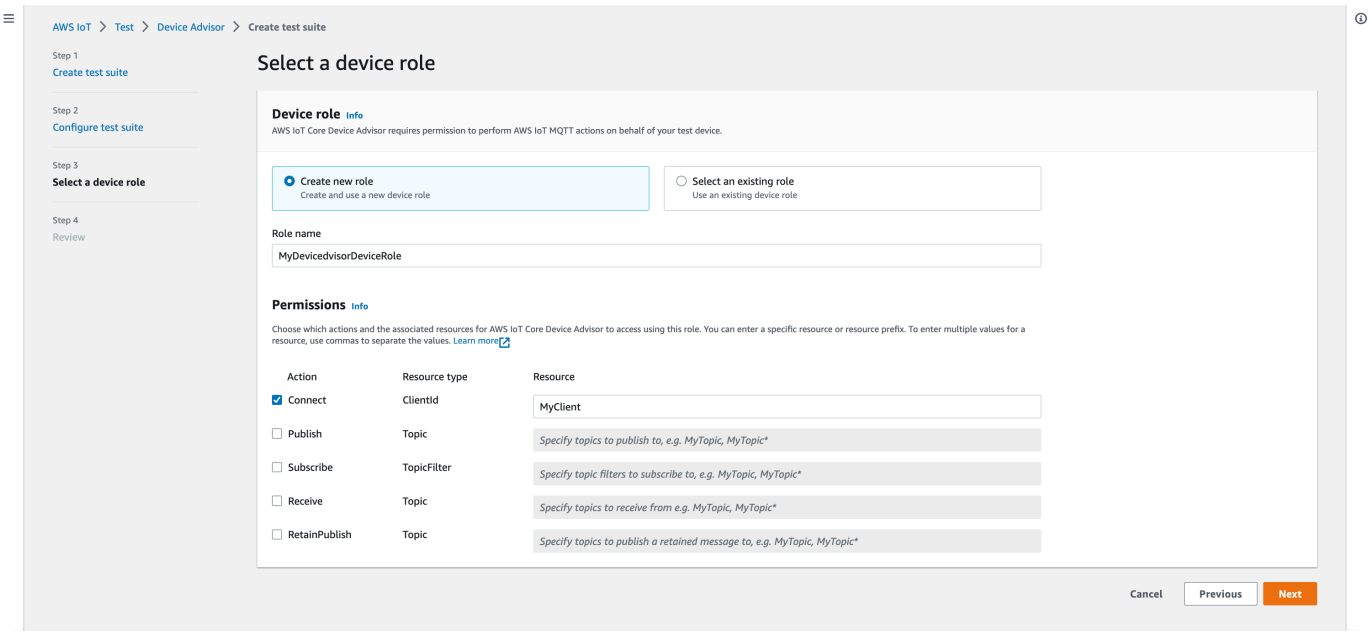
Selecione Concluído.

> **Note**
>
> Para adicionar mais grupos de teste ao conjunto de testes, escolha Adicionar grupo de teste. Siga as etapas anteriores para criar e configurar mais grupos de teste ou adicionar mais casos de teste a um ou mais grupos de teste. Grupos de teste e casos de teste podem ser reordenados escolhendo e arrastando um caso de teste até a posição desejada. O Device Advisor executa testes na ordem em que você define os grupos de teste e os casos de teste.

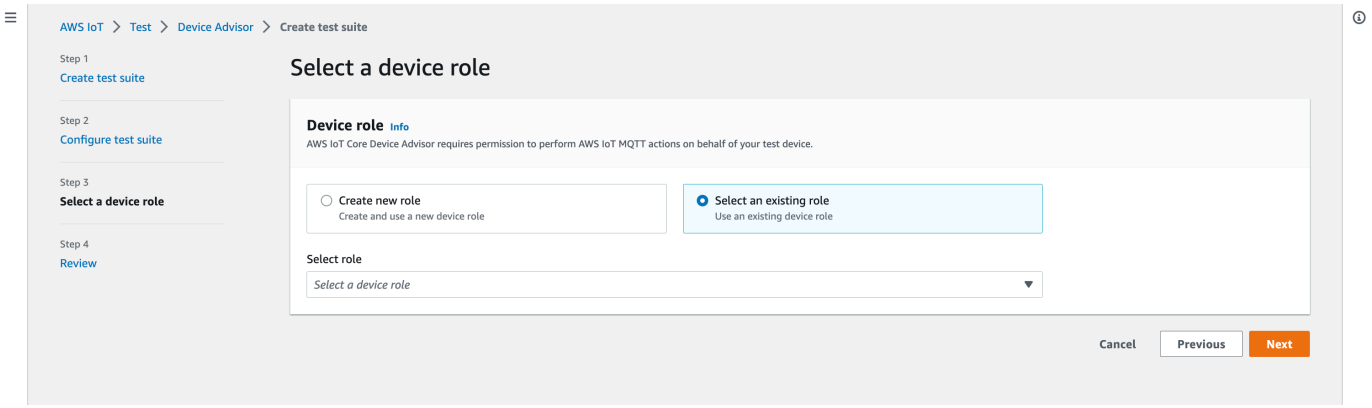
7. Escolha Próximo.
8. Na Etapa 3, configure uma função de dispositivo que o Device Advisor usará para realizar ações do AWS IoT MQTT em nome do seu dispositivo de teste.

   Se você selecionou o caso de teste do MQTT Connect somente na Etapa 2, a ação Conectar será verificada automaticamente, pois essa permissão é necessária no perfil do dispositivo para executar esse conjunto de testes. Se você selecionou outros casos de teste, as ações necessárias correspondentes serão verificadas. Certifique-se de que os valores dos recursos para cada uma das ações sejam fornecidos. Por exemplo, para a ação Conectar, forneça o ID do cliente com o qual o dispositivo se conectará ao endpoint do Device Advisor. Você pode fornecer vários valores usando vírgulas para separar os valores e também pode fornecer valores de prefixo usando um caractere curinga (*). Por exemplo, para fornecer permissão para publicar

em qualquer tópico que comece com `MyTopic`, você pode fornecer "`MyTopic*`" como o valor do recurso.



Se você já criou um perfil do dispositivo anteriormente e gostaria de usar esse perfil, selecione Selecionar um perfil existente e escolha o perfil do dispositivo em Selecionar perfil.



Configure o perfil do dispositivo usando uma das duas opções fornecidas e escolha Próximo.

9. Na Etapa 4, verifique se a configuração fornecida em cada uma das etapas está correta. Para editar a configuração fornecida para uma etapa específica, escolha Editar para a etapa correspondente.

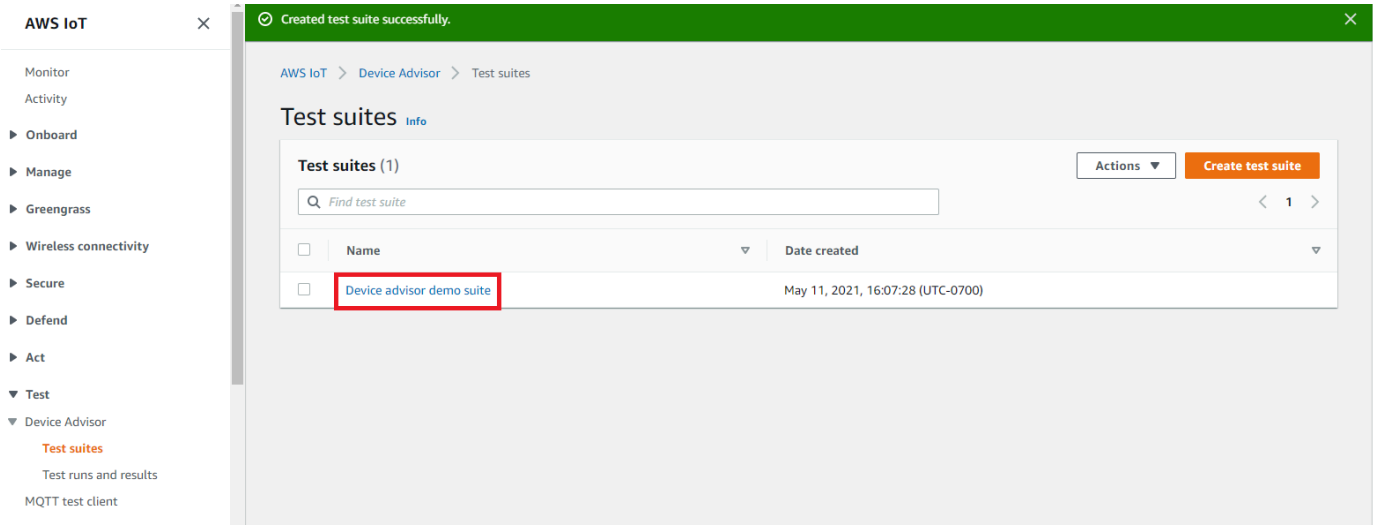
   Depois de verificar a configuração, escolha Criar conjunto de testes.

   O conjunto de testes deve ser criado e você será redirecionado para a página Conjuntos de testes, onde poderá ver todos os conjuntos que foram criados.

Se a criação do conjunto de testes falhar, verifique se o conjunto de testes, os grupos de teste, os casos de teste e o perfil do dispositivo foram configurados conforme as instruções anteriores.

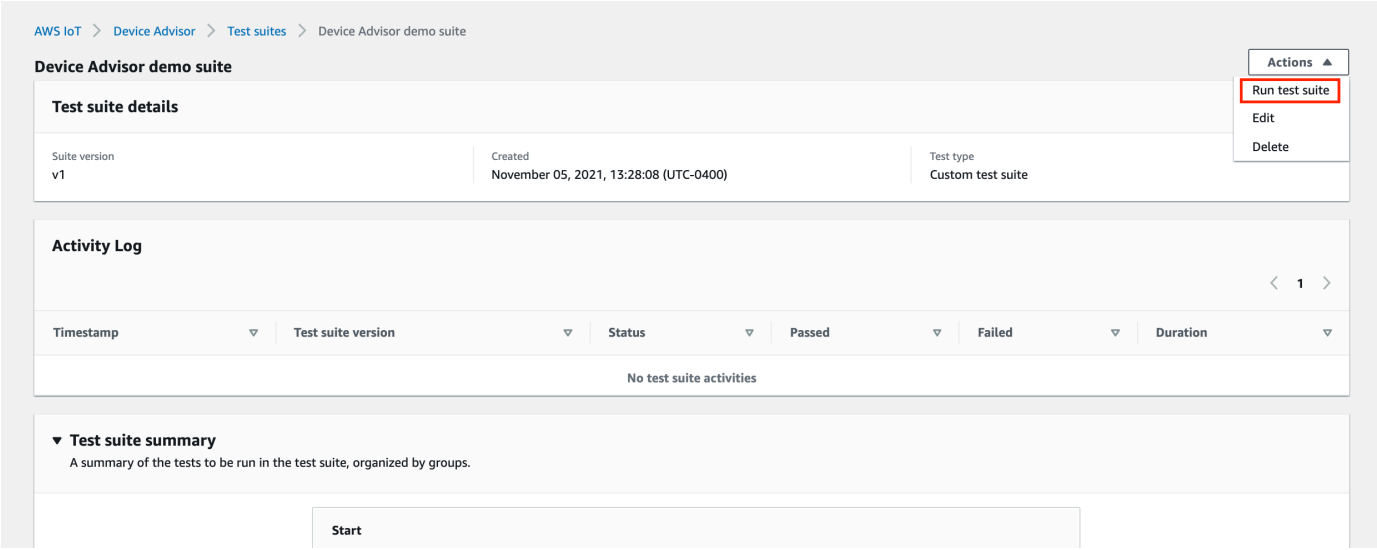
Inicie a execução de um conjunto de testes

1. No console AWS IoT, no painel de navegação, expanda Teste, Device Advisor e escolha Conjuntos de testes.
2. Escolha o conjunto de testes do qual você gostaria de ver os detalhes do conjunto de testes.



A página de detalhes do conjunto de testes exibe todas as informações relacionadas ao conjunto de testes.

3. Escolha Ações e, em seguida, Executar conjunto de testes.

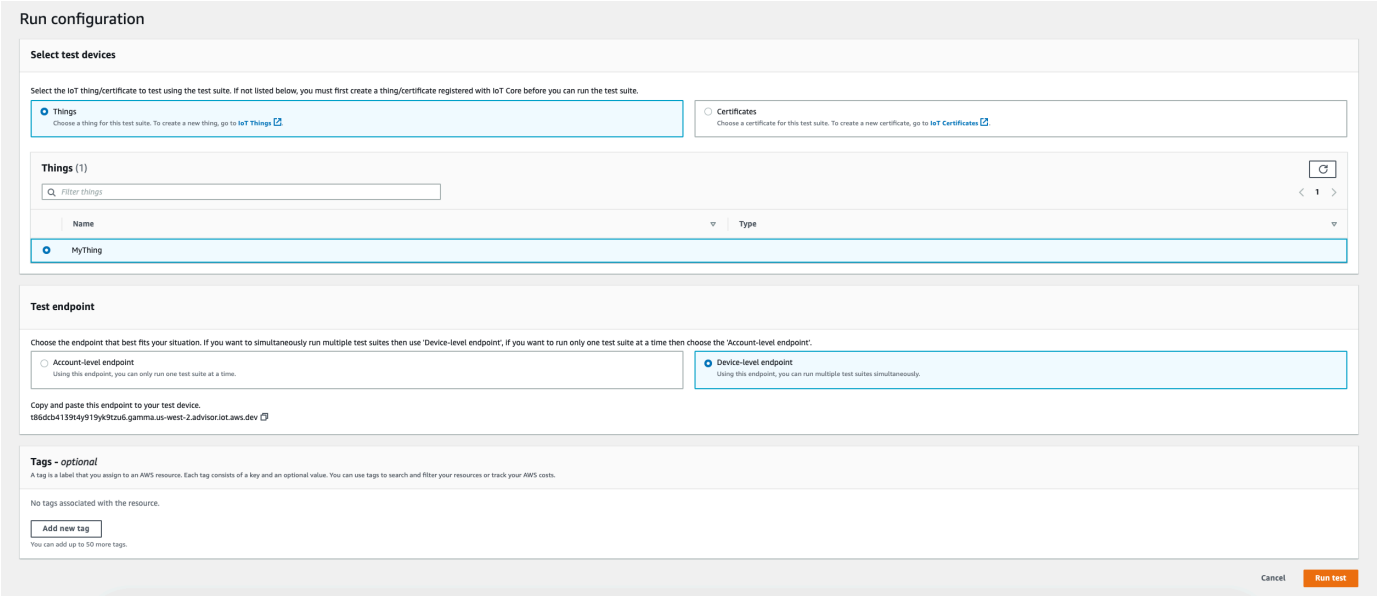


4. Em Executar configuração, você precisará selecionar AWS IoT algo ou certificado para testar usando o Device Advisor. Se você não tiver nenhum item ou certificado existente, primeiro crie AWS IoT Core recursos.

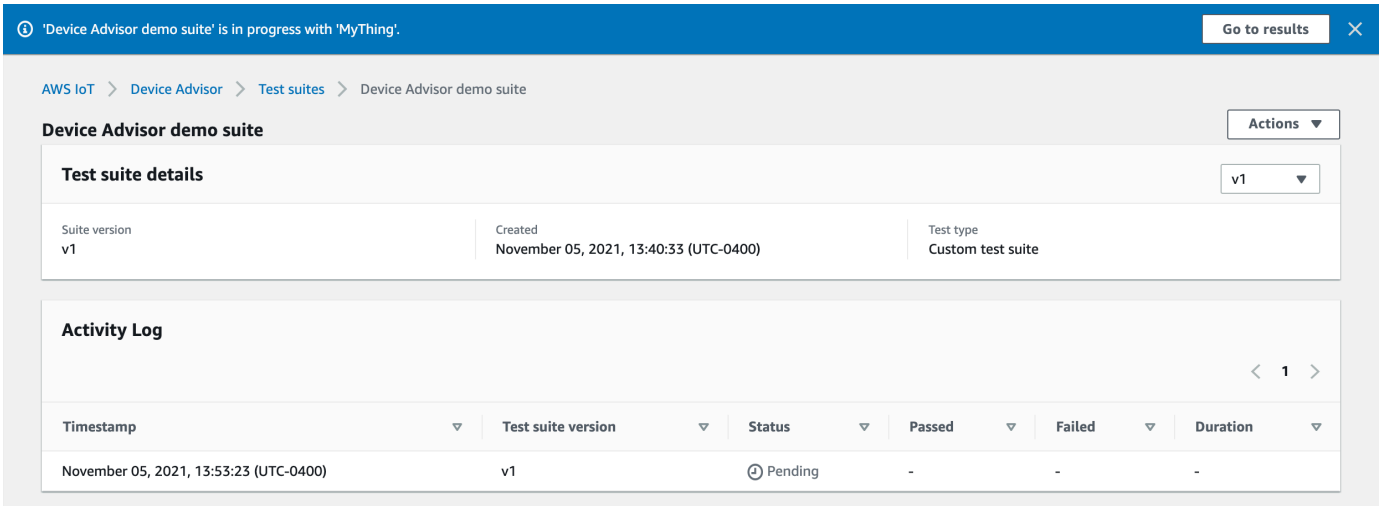
   Na seção Endpoint de teste, selecione o endpoint que melhor se adapta ao caso. Se você planeja executar várias suítes de teste simultaneamente usando a mesma AWS conta no futuro, selecione Endpoint em nível de dispositivo. Caso contrário, se planeja apenas executar um conjunto de testes por vez, selecione Endpoint em nível de conta.

   Configure o dispositivo de teste com o endpoint de teste do Device Advisor selecionado.

   Depois de selecionar um objeto ou um certificado e escolher um endpoint do Device Advisor, escolha Executar teste.

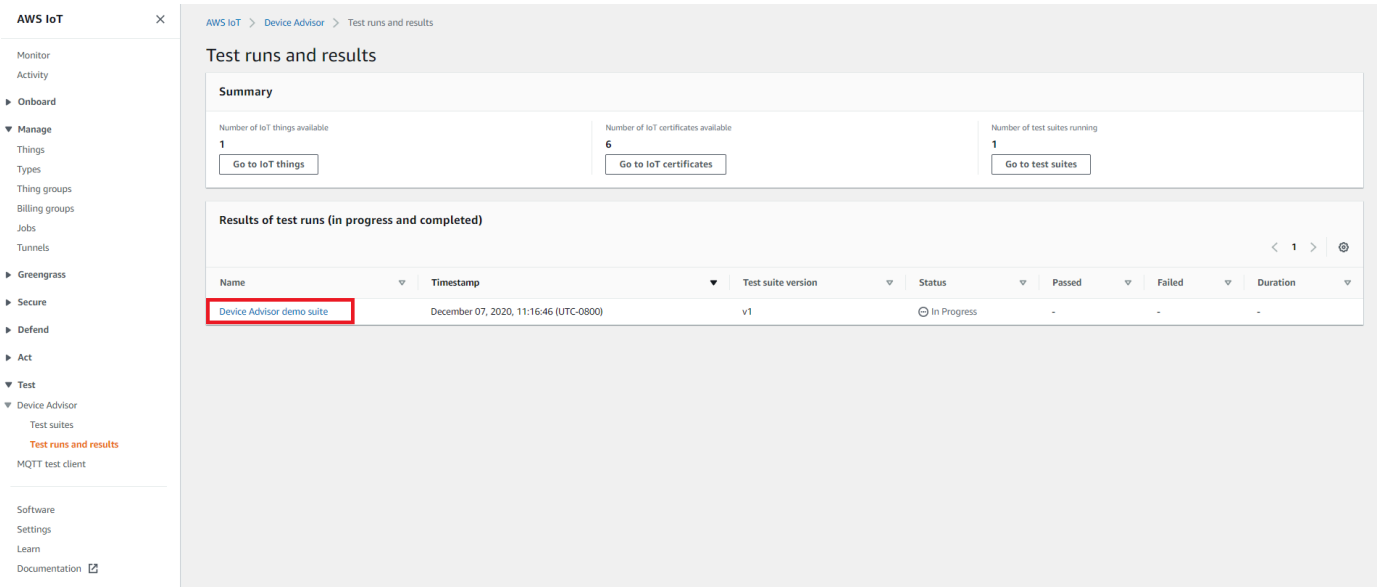


5. Escolha Ir para os resultados no banner superior para ver os detalhes da execução do teste.

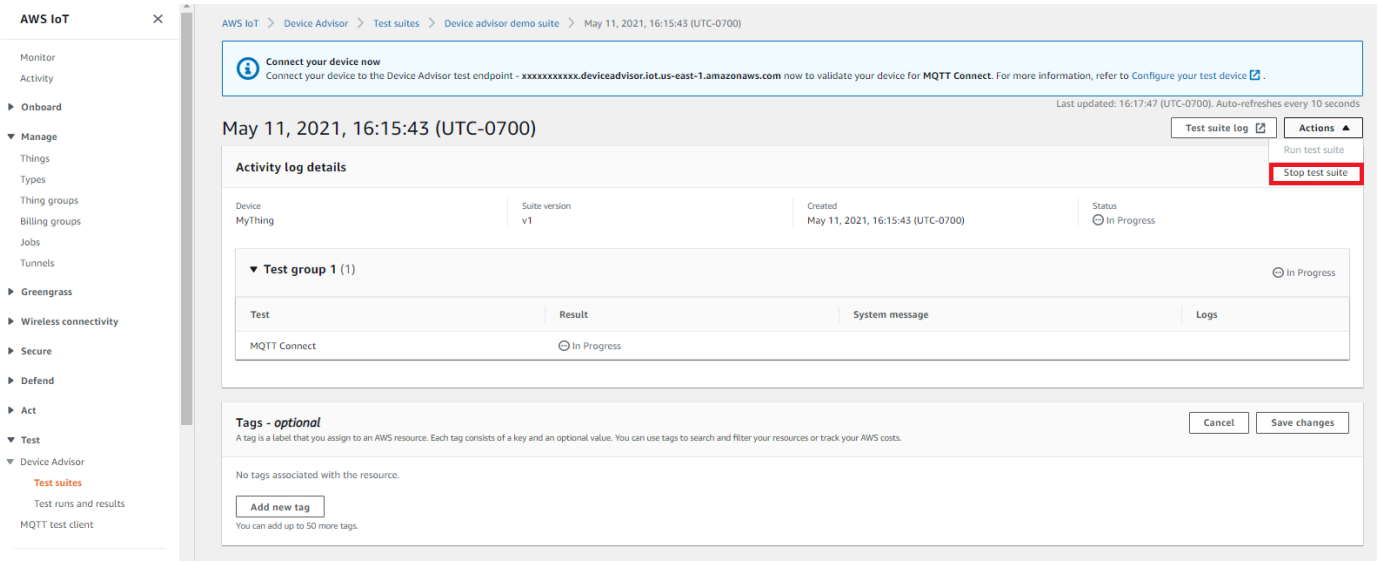


Interromper a execução de um conjunto de testes (opcional)

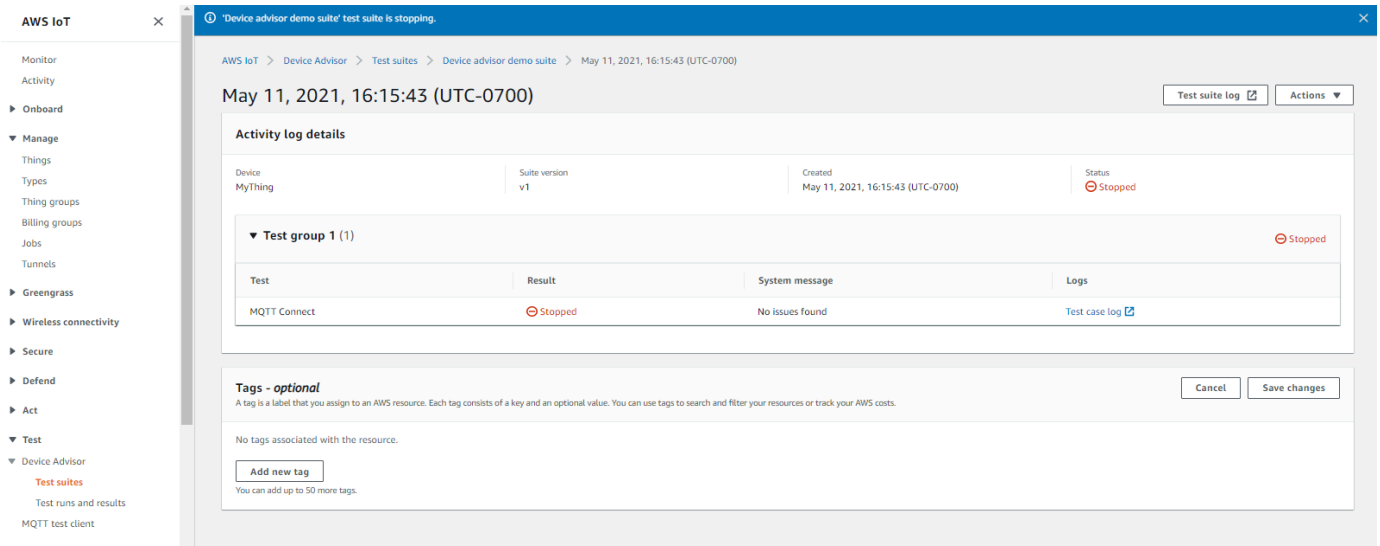
1. No console AWS IoT, no painel de navegação, expanda Teste, Device Advisor e escolha Execuções e resultados de testes.
2. Escolha o conjunto de testes em andamento que você deseja interromper.



3. Escolha Ações e, em seguida, Interromper conjunto de testes.



4. O processo de limpeza levará alguns minutos para ser concluído. Enquanto o processo de limpeza for executado, o status da execução do teste será `STOPPING`. Aguarde a conclusão do processo de limpeza e que o status do conjunto de testes mude para o status `STOPPED` antes de iniciar a execução de um novo conjunto.



Exibir detalhes e logs da execução do conjunto de testes

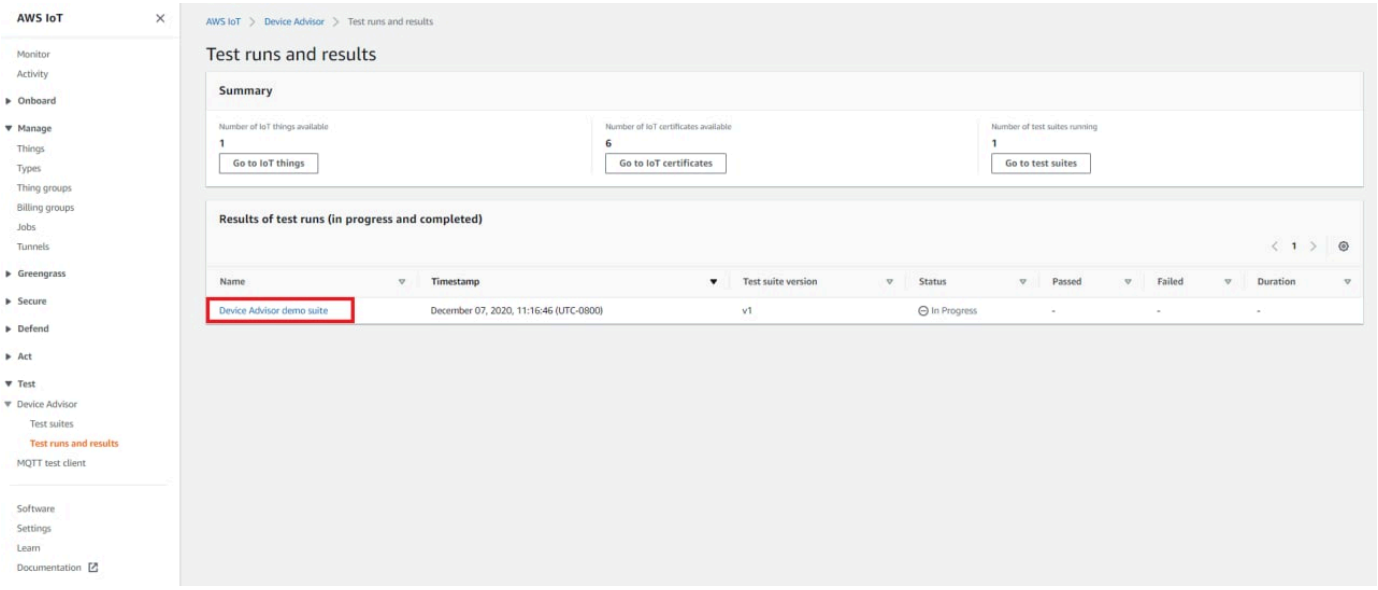
1. No console AWS IoT, no painel de navegação, expanda Teste, Device Advisor e escolha Execuções e resultados de testes.

   Esta página será exibida:

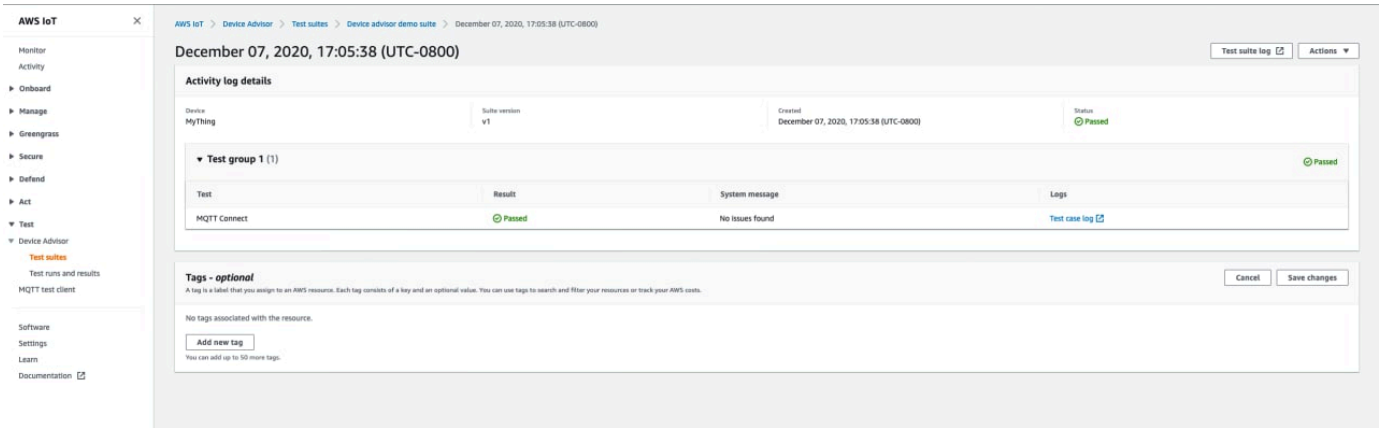
   - Número de objetos de IoT

- Número de certificados de IoT
- Número de conjuntos de teste atualmente em execução
- Todas as execuções do conjunto de testes que foram criadas

2. Escolha o conjunto de testes do qual você gostaria de ver os detalhes e logs da execução.



A página de resumo da execução exibe o status da execução atual do conjunto de testes. Essa página é atualizada automaticamente a cada 10 segundos. Recomendamos que você tenha um mecanismo criado para que o dispositivo tente se conectar ao nosso endpoint de teste a cada cinco segundos por um a dois minutos. Em seguida, você pode executar vários casos de teste em sequência de maneira automatizada.



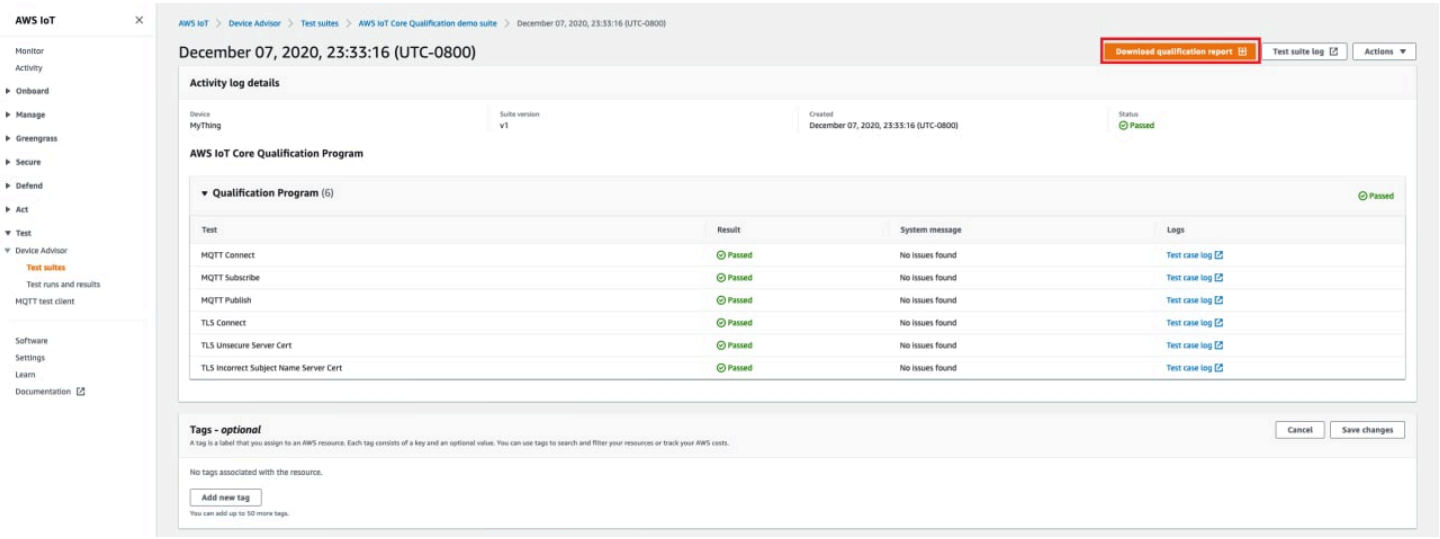
3. Para acessar os CloudWatch registros da execução da suíte de testes, escolha Registro da suíte de testes.

Para acessar CloudWatch os registros de qualquer caso de teste, escolha Registro de casos de teste.

4. Com base nos resultados do teste, solucione os problemas do dispositivo até que todos os testes sejam aprovados.

Baixe um relatório AWS IoT de qualificação

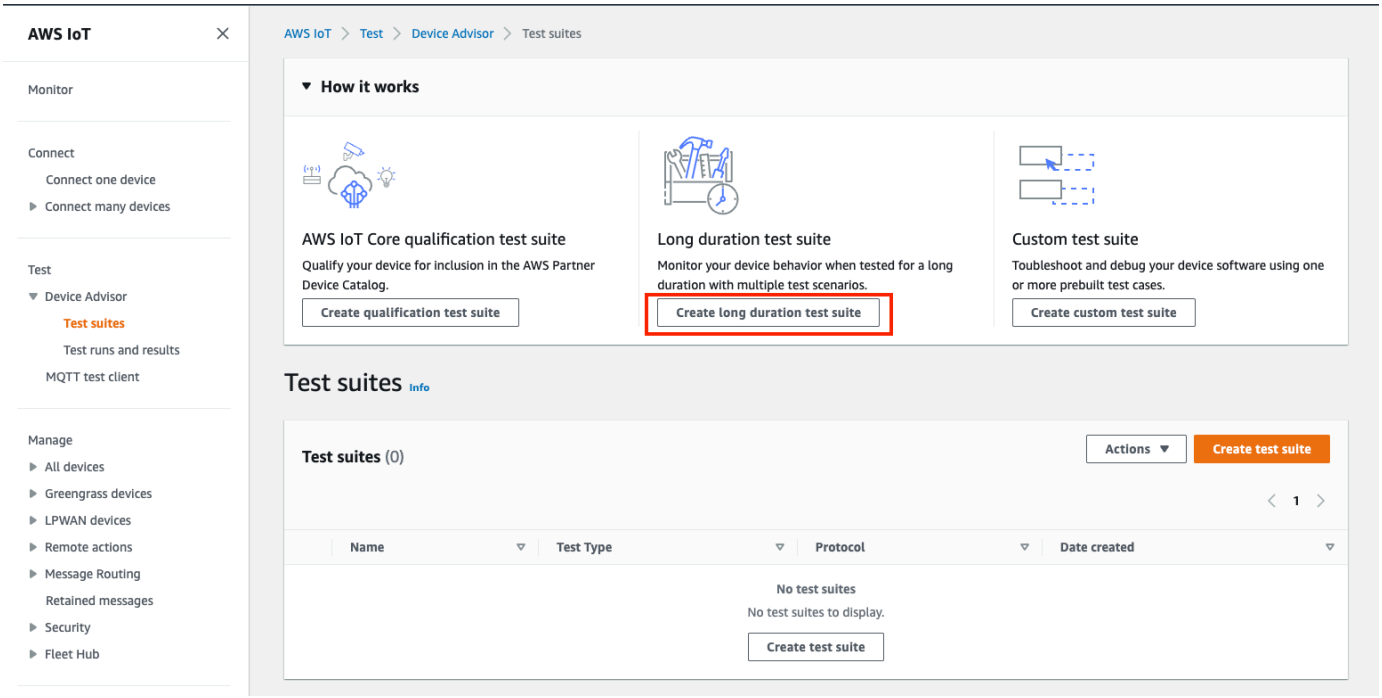
Se você escolheu a opção Usar a suíte de testes de AWS IoT qualificação ao criar uma suíte de testes e conseguiu executar uma suíte de testes de qualificação, poderá baixar um relatório de qualificação escolhendo Baixar relatório de qualificação na página de resumo da execução do teste.



Fluxo de trabalho do console de testes de longa duração

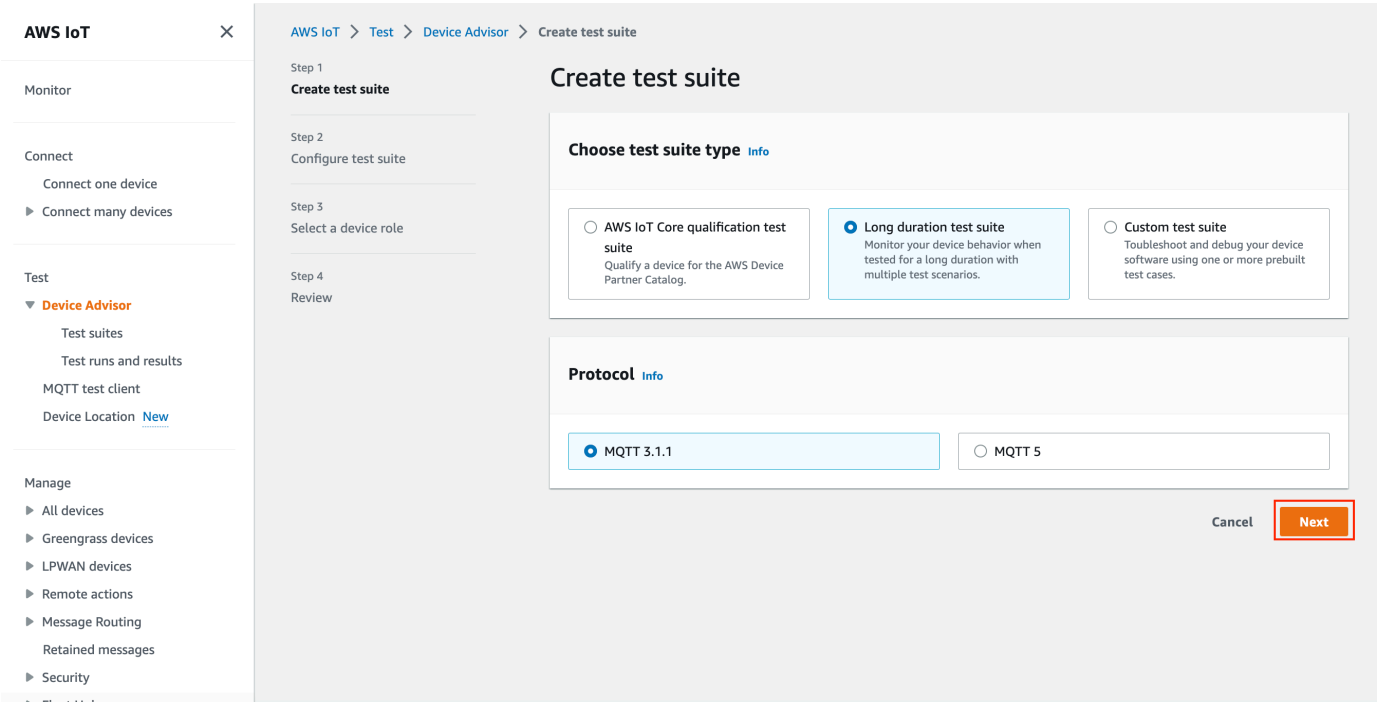
Este tutorial ajuda você a começar com os testes de Longa duração no Device Advisor usando o console. Para concluir o tutorial, siga as etapas em Configurar.

1. No console AWS IoT, no painel de navegação, expanda Teste, Device Advisor e escolha Conjuntos de testes. Na página, selecione Criar conjunto de testes de longa duração.



2. Na página Criar conjunto de testes, selecione Conjunto testes de longa duração e escolha Próximo.

   Para protocolo, escolha MQTT 3.1.1 ou MQTT 5.



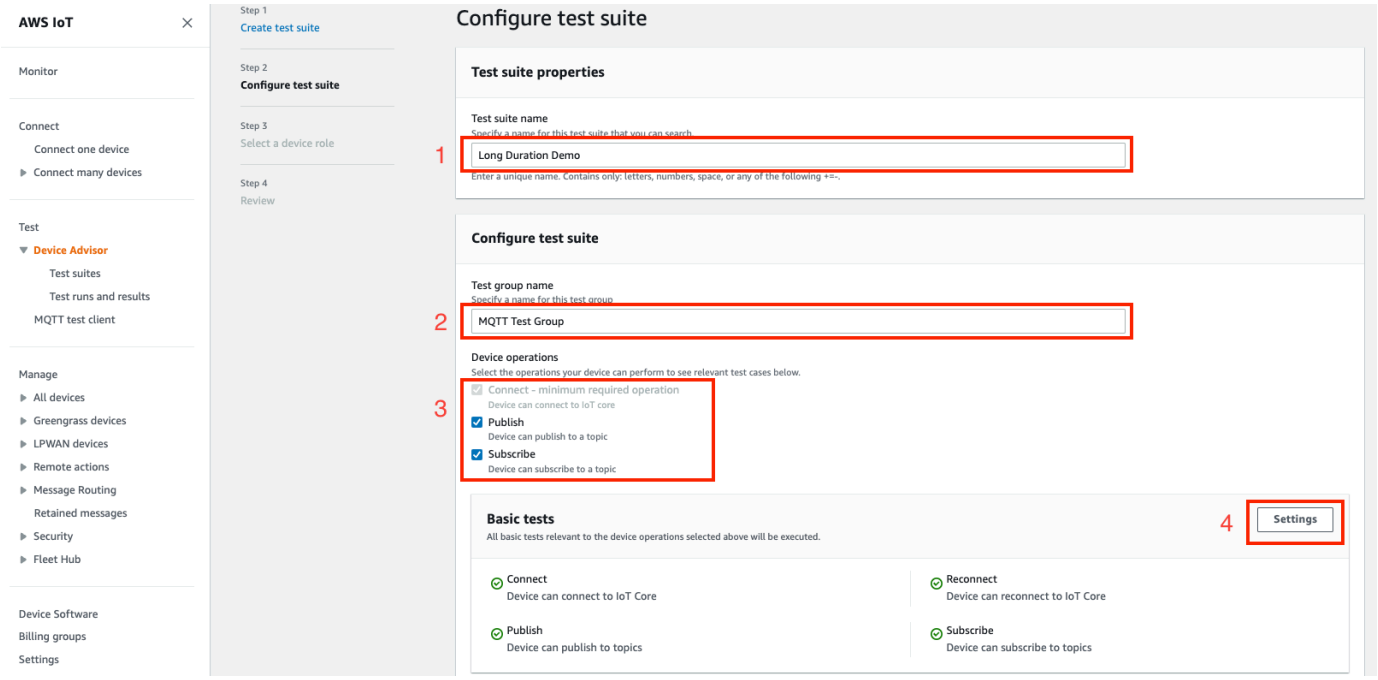
3. Faça o seguinte na página Configurar conjunto de teste:

   a. Atualize o campo Nome do conjunto de testes.

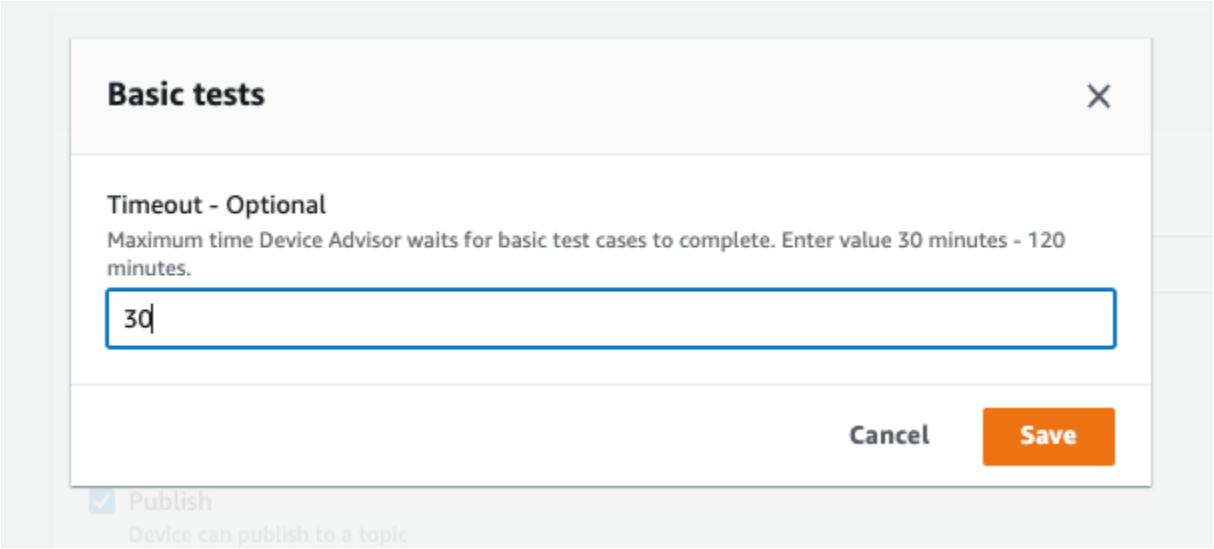
b. Atualize o campo Nome do grupo de testes.

c. Escolha as Operações do dispositivo que o dispositivo pode realizar. Isso selecionará os testes a serem executados.

d. Selecione a opção Configurações.



4. (Opcional) Insira o tempo máximo que o Device Advisor deve aguardar até que os testes básicos sejam concluídos. Selecione Salvar.



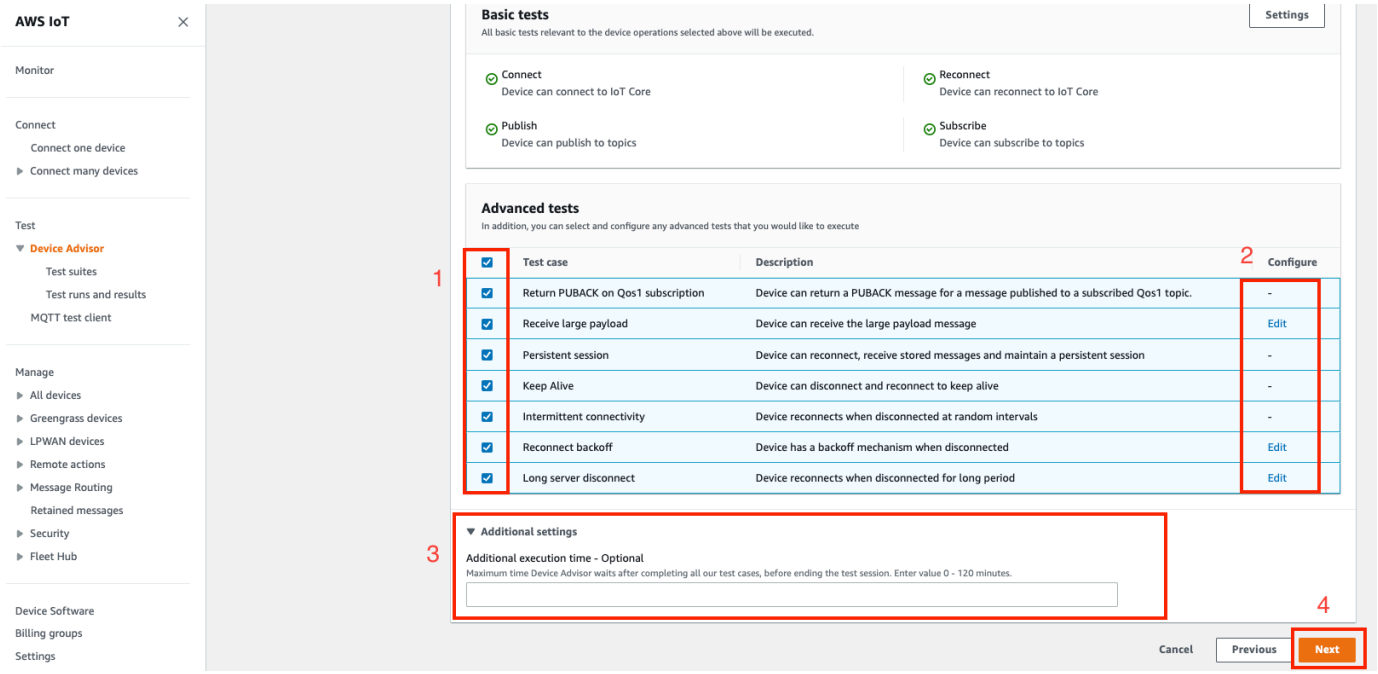
5. Faça o seguinte nas seções Testes avançados e Configurações adicionais.

a. Selecione ou desmarque os Testes avançados que você deseja executar como parte desse teste.

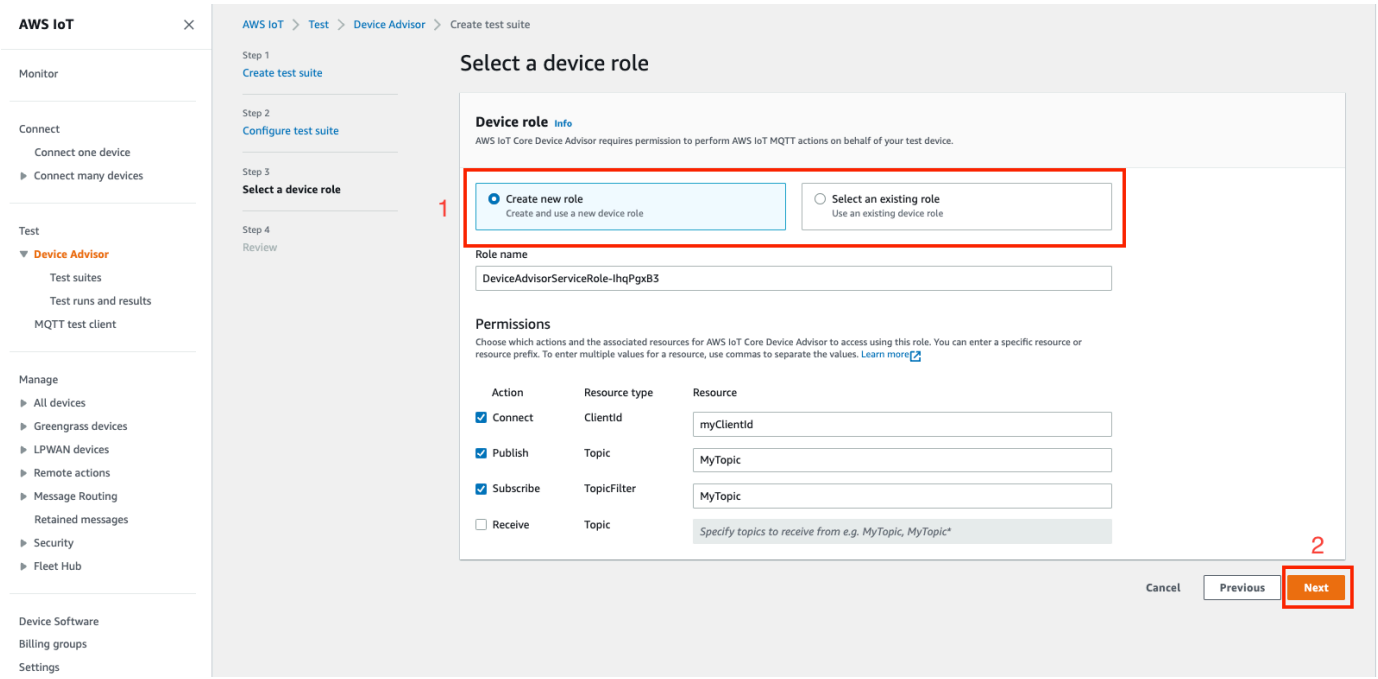
b. Edite as configurações dos testes quando aplicável.

c. Configure o Tempo de execução adicional na seção Configurações adicionais.

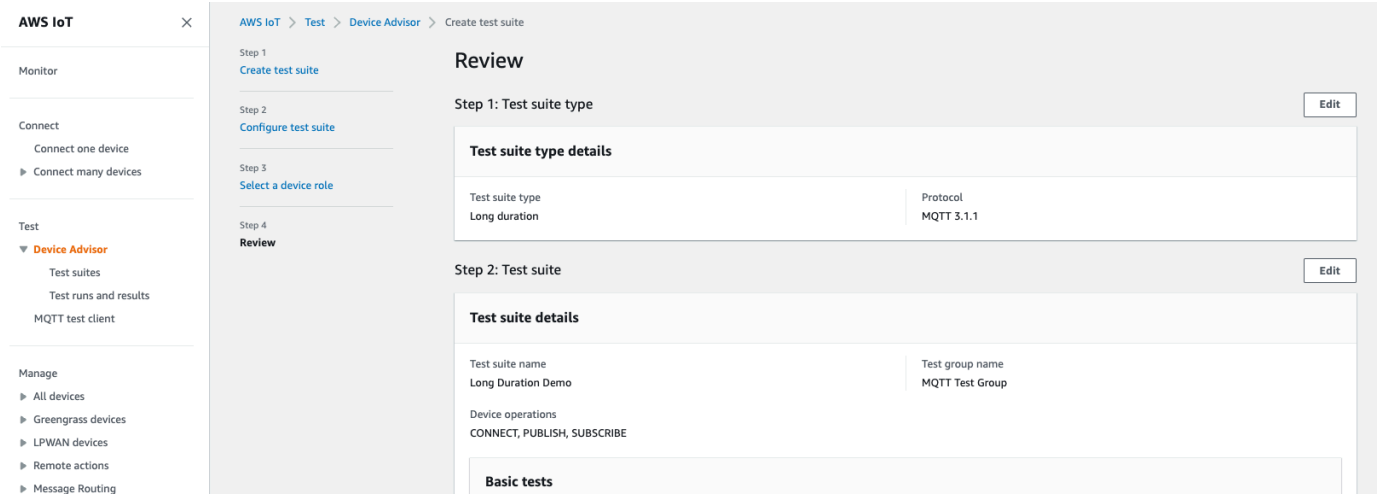
d. Escolha Próximo para fazer a próxima etapa.

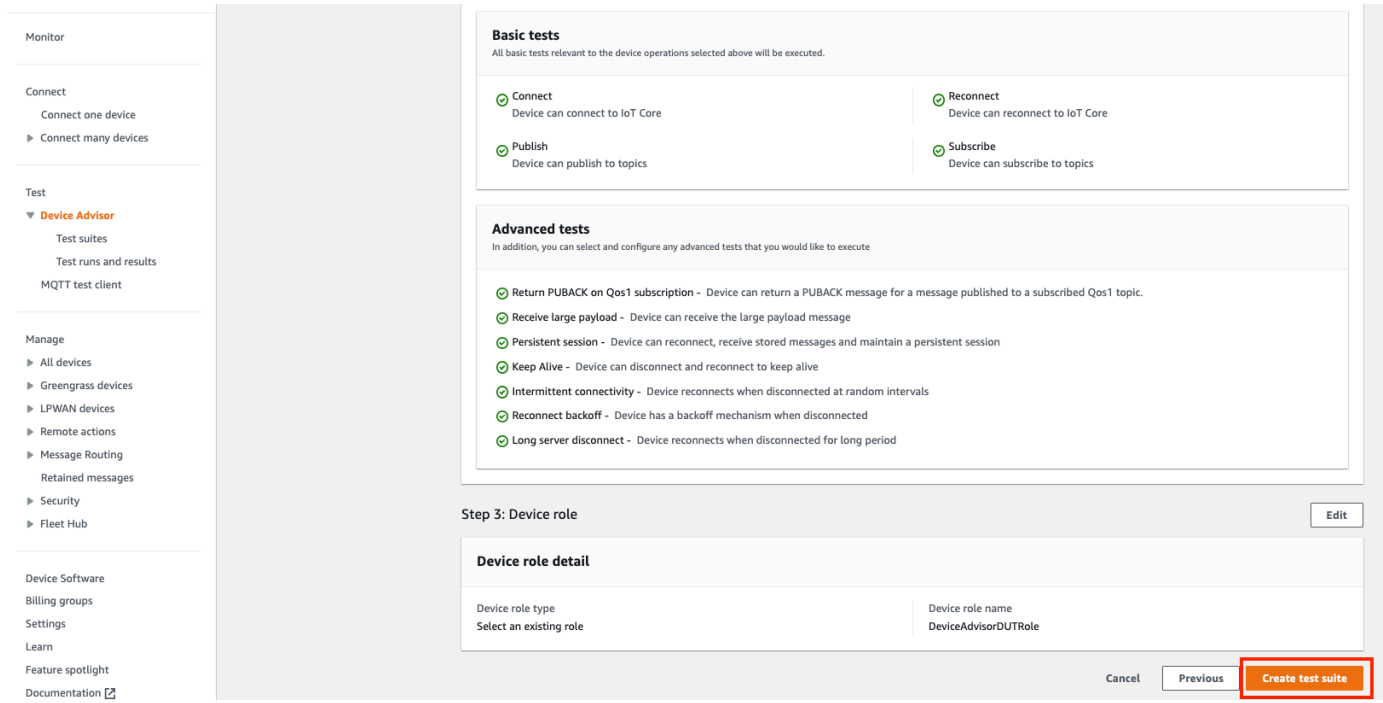


6. Nesta etapa, Crie um novo perfil ou Selecione um perfil existente. Para mais detalhes, consulte Criar um perfil do IAM a ser usado como perfil de dispositivo.

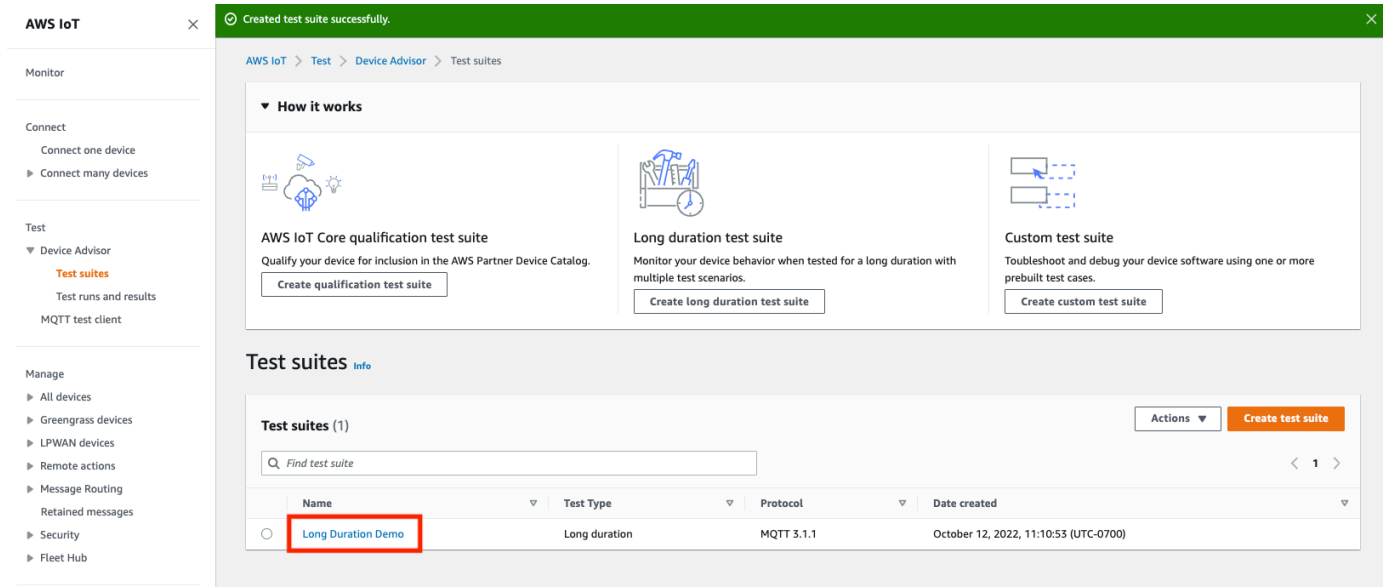


7. Revise todas as configurações criadas até essa etapa e selecione Criar conjunto de testes.

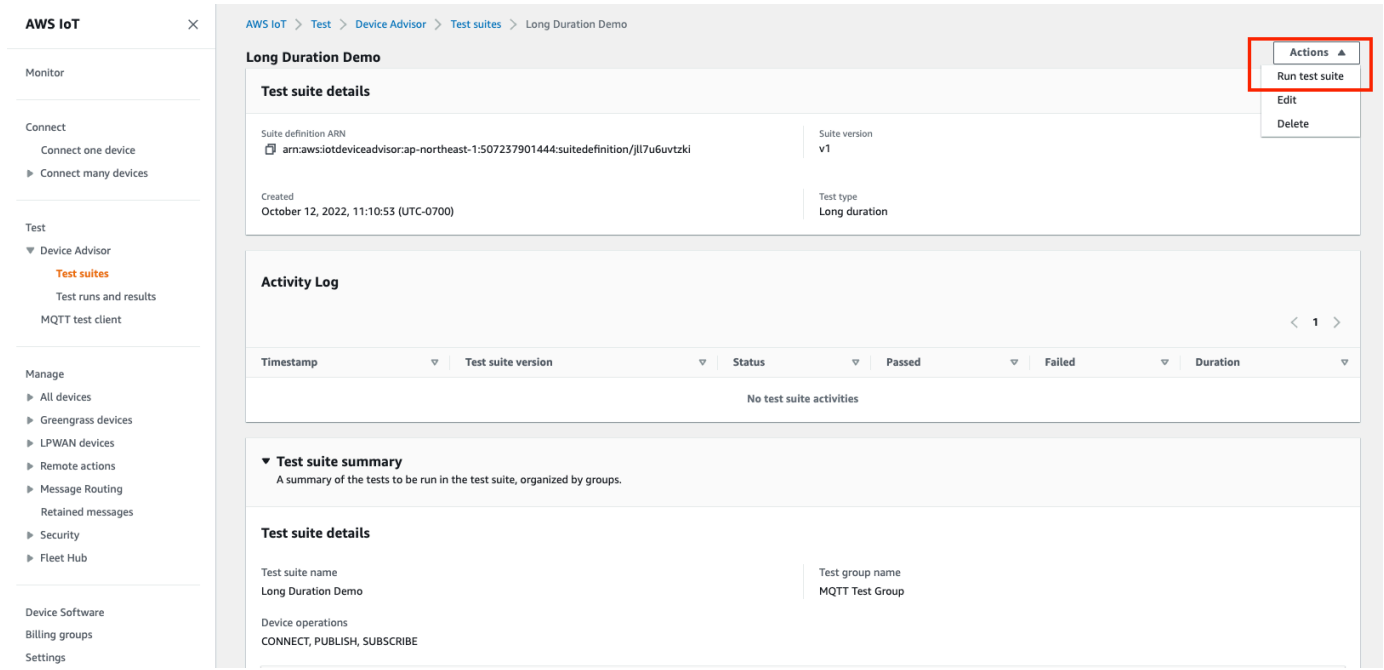




8. O conjunto de testes criado está na seção Conjuntos de testes. Selecione o conjunto para visualizar detalhes.



9. Para executar o conjunto de testes criado, selecione Ações e depois Executar conjunto de testes.

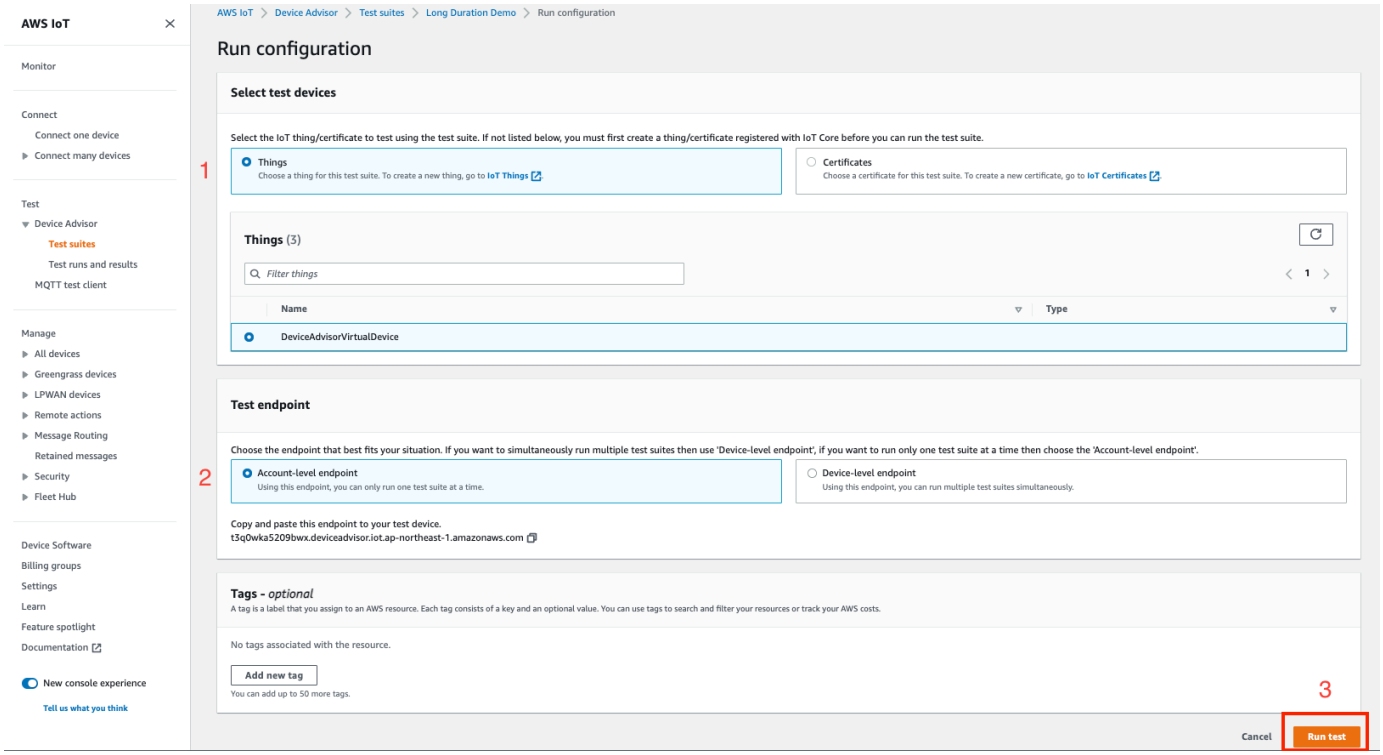


10. Escolha as opções de configuração na página Executar configuração.

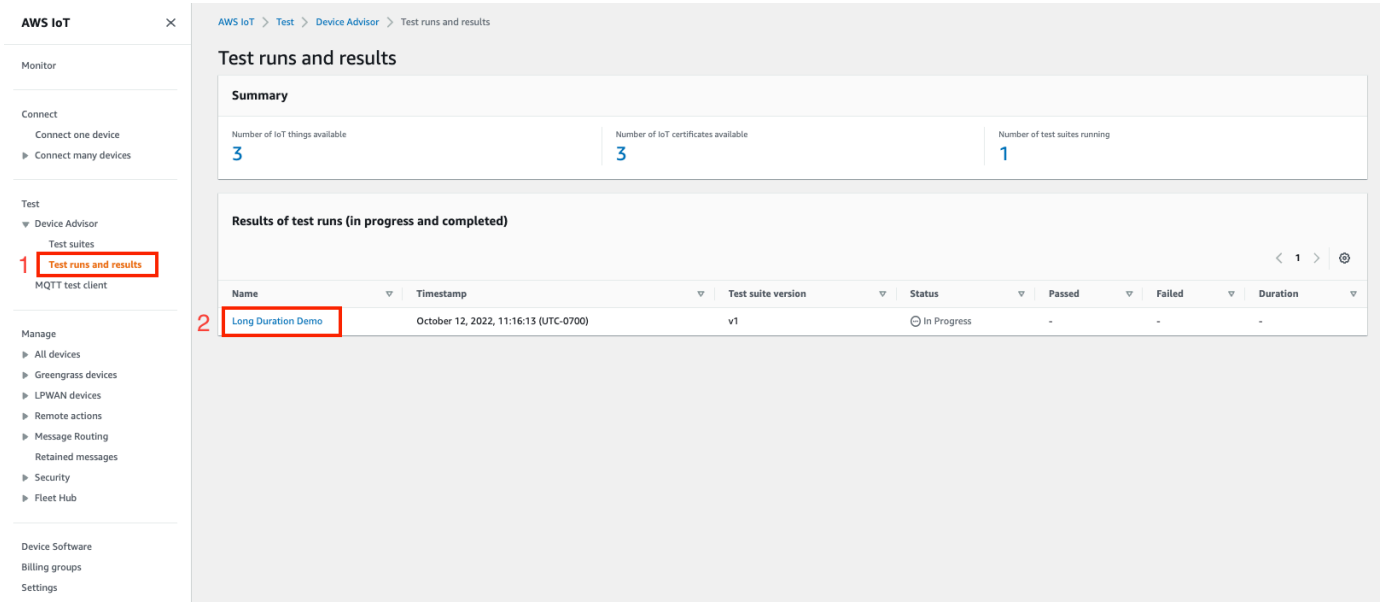
    a. Selecione as Objetos ou o Certificado para executar o teste.

    b. Selecione o Endpoint em nível de conta ou o Endpoint em nível de dispositivo.

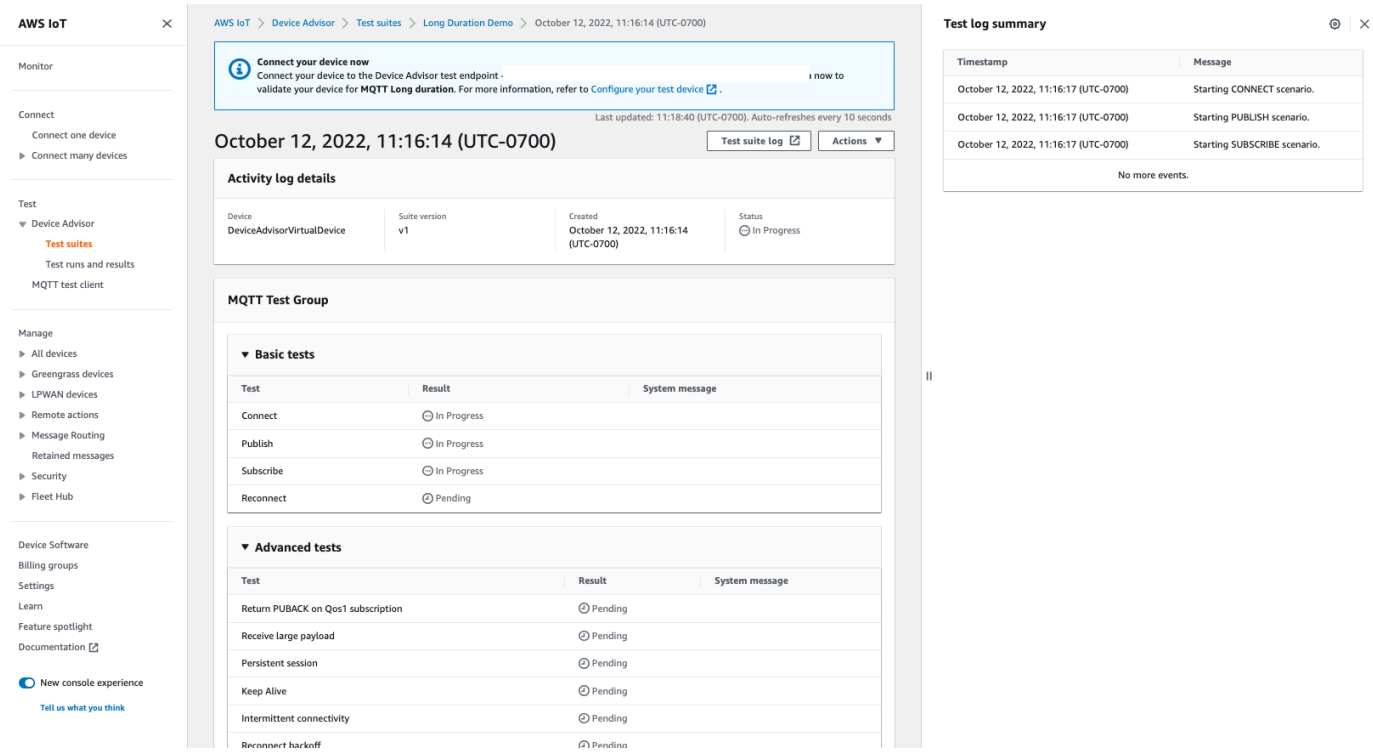
    c. Escolha Executar teste para executar o teste.



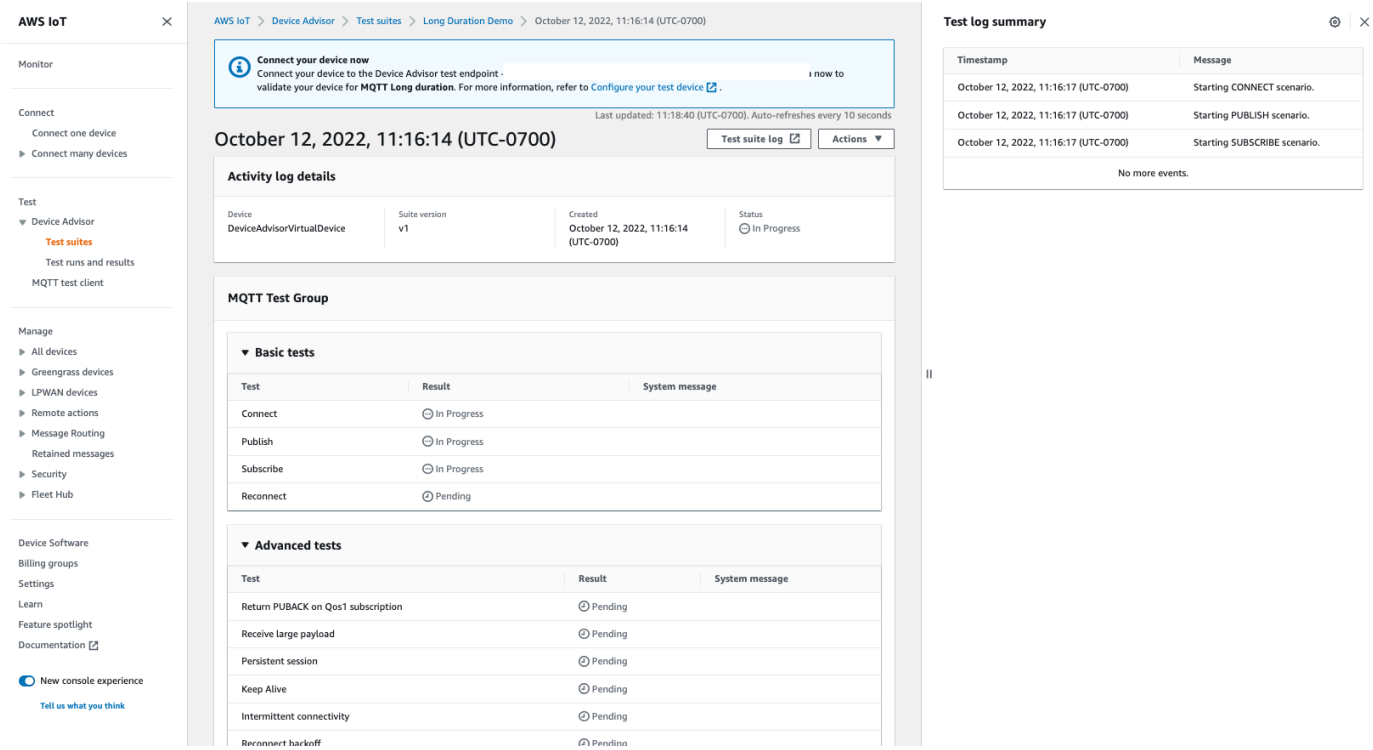
11. Para visualizar os resultados da execução do conjunto de testes, selecione Execuções e resultados de teste no painel de navegação esquerdo. Escolha o conjunto de testes que foi executado para ver os detalhes dos resultados.



12. A etapa anterior abre a página de resumo do teste. Todos os detalhes da execução do teste são exibidos nessa página. Quando o console solicitar o início da conexão do dispositivo, conecte-o ao endpoint fornecido. O progresso dos testes é visto nessa página.



13. O teste de Longa duração fornece um Resumo do log de teste adicional no painel lateral, que exibe todos os eventos importantes que ocorrem entre o dispositivo e o agente quase em tempo real. Para ver logs mais detalhados, clique em Log do caso de teste.



Endpoints da VPC do Device Advisor (AWS PrivateLink)

Você pode estabelecer uma conexão privada entre sua VPC e o endpoint de AWS IoT Core Device Advisor teste (plano de dados) criando uma interface VPC endpoint. Você pode usar esse endpoint para validar AWS IoT dispositivos para conectividade confiável e segura AWS IoT Core antes de implantá-los na produção. Os testes pré-criados do Device Advisor ajudam você a validar o software do dispositivo em relação às práticas recomendadas de uso de TLS, MQTT, Sombra do dispositivo e Tarefas do AWS IoT.

AWS PrivateLinkalimenta os endpoints da interface usados com seus dispositivos de IoT. Esse serviço ajuda você a acessar o endpoint de teste do AWS IoT Core Device Advisor sem um gateway da Internet, um dispositivo NAT, uma conexão VPN ou uma conexão do Direct Connect . As instâncias em sua VPC que enviam pacotes TCP e MQTT não precisam de endereços IP públicos para se comunicar com os endpoints de teste. AWS IoT Core Device Advisor Tráfego entre sua VPC e o AWS IoT Core Device Advisor que não sai. Nuvem AWS Qualquer comunicação TLS e MQTT entre dispositivos de IoT e casos de teste do Device Advisor permanece dentro dos recursos da sua Conta da AWS.

Cada endpoint de interface é representado por uma ou mais interfaces de rede elástica nas sub-redes.

Para saber mais sobre como usar os endpoints da VPC da interface, consulte Endpoints da VPC da interface (AWS PrivateLink) no Guia do usuário da Amazon VPC.

Considerações sobre AWS IoT Core Device Advisor VPC endpoints

Antes de configurar propriedades e limitações de endpoint da interface no Guia do usuário da Amazon VPC antes de configurar os endpoints da VPC da interface. Considere o seguinte antes de continuar:

- AWS IoT Core Device Advisor atualmente oferece suporte para fazer chamadas para o endpoint de teste do Device Advisor (plano de dados) a partir de sua VPC. Um agente de mensagens usa comunicações de plano de dados para enviar e receber dados. Ele faz isso com a ajuda dos pacotes TLS e MQTT. VPC endpoints para AWS IoT Core Device Advisor conectar seu AWS IoT dispositivo aos endpoints de teste do Device Advisor. As ações da API do ambiente de gerenciamento não são usadas por esse endpoint da VPC. Para criar ou executar uma suíte de testes ou outro plano de controle APIs, use o console, um AWS SDK ou uma interface de linha de AWS comando na Internet pública.

- Os seguintes Regiões da AWS oferecem suporte a endpoints VPC para: AWS IoT Core Device Advisor
  - Leste dos EUA (N. da Virgínia)
  - Oeste dos EUA (Oregon)
  - Ásia-Pacífico (Tóquio)
  - Europa (Irlanda)
- O Device Advisor oferece suporte ao MQTT com certificados de cliente X.509 e certificados de servidor RSA.
- As políticas de endpoint da VPC não são compatíveis no momento.
- Consulte os pré-requisitos de endpoint da VPC para obter instruções sobre como criar recursos que conectem os endpoints da VPC. Você deve criar uma VPC e sub-redes privadas para usar VPC endpoints. AWS IoT Core Device Advisor
- Há cotas em seus AWS PrivateLink recursos. Para obter mais informações, consulte as AWS PrivateLink cotas.
- Os VPC endpoints oferecem suporte somente ao tráfego. IPv4

Crie uma interface VPC endpoint para AWS IoT Core Device Advisor

Para começar a usar endpoints da VPC, crie um endpoint da VPC da interface. Em seguida, selecione AWS IoT Core Device Advisor como AWS service (Serviço da AWS) o. Se você estiver usando o AWS CLI, ligue describe-vpc-endpoint-servicespara confirmar que AWS IoT Core Device Advisor está presente em uma zona de disponibilidade em seu Região da AWS. Confirme se o grupo de segurança conectado ao endpoint permite a comunicação do protocolo TCP para tráfego MQTT e TLS. Por exemplo, na região Leste dos EUA (Norte da Virgínia), use o seguinte comando:

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-
east-1.deviceadvisor.iot
```

Você pode criar um VPC endpoint AWS IoT Core usando o seguinte nome de serviço:

- com.amazonaws.region.deviceadvisor.iot

Por padrão, o DNS privado está ativado para o endpoint. Isso garante que o uso do endpoint de teste padrão permaneça nas sub-redes privadas. Para obter seu endpoint em nível de conta ou

dispositivo, use o console AWS CLI ou um AWS SDK. Por exemplo, se você executa get-endpoint em uma sub-rede pública ou na Internet pública, você pode obter o endpoint e usá-lo para se conectar ao Device Advisor. Para mais informações, consulte Acessar um serviço por um endpoint de interface no Guia do usuário da Amazon VPC.

Para conectar clientes MQTT às interfaces de endpoint da VPC, AWS PrivateLink o serviço cria registros DNS em uma zona hospedada privada anexada à sua VPC. Esses registros DNS direcionam as solicitações do dispositivo AWS IoT para o endpoint da VPC.

Controle do acesso a AWS IoT Core Device Advisor mais de VPC endpoints

Você pode restringir o acesso ao dispositivo AWS IoT Core Device Advisor e permitir o acesso somente por meio de VPC endpoints usando as chaves de contexto de condição da VPC. AWS IoT Core é compatível com as seguintes chaves de contexto relacionadas à VPC:

- SourceVpc
- SourceVpce
- VPCSourceIp

> **Note**
>
> AWS IoT Core Device Advisor não oferece suporte às políticas de VPC endpoint no momento.

A política a seguir concede permissão para se conectar AWS IoT Core Device Advisor usando um ID de cliente que corresponda ao nome da coisa. Ele também publica em qualquer tópico prefixado pelo nome do objeto. A política depende da conexão do dispositivo a um endpoint da VPC com um ID específico do endpoint da VPC. Essa política nega tentativas de conexão com o endpoint de teste do AWS IoT Core Device Advisor público.

```
{
"Version":"2012-10-17",
    "Statement": [
        {
```

```
"Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ],
            "Condition": {
"StringEquals": {
"aws:SourceVpce": "vpce-1a2b3c4d"
            }
        }

        },
        {
"Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.ThingName}/*"
            ]
        }
    ]
}
```

Casos de teste do Device Advisor

O Device Advisor fornece testes pré-criados em seis categorias.

- TLS
- MQTT
- Shadow
- Execução de trabalho
- Políticas e permissões
- Testes de longa duração

Casos de teste do Device Advisor para se qualificar para o Programa de Qualificação de AWS Dispositivos.

O dispositivo deve passar nos testes a seguir para se qualificar conforme o Programa de Qualificação de Dispositivos da AWS.

> **Note**
>
> Esta é uma lista revisada dos testes de qualificação.

- TLS Connect ("TLS Connect")
- Certificado de servidor de nome de assunto incorreto do TLS(“Nome comum do assunto incorreto [CN]/Nome alternativo do assunto [SAN]”)
- TLS Unsecure Server Cert ("Not Signed By Recognized CA")
- Suporte de dispositivo TLS para pacotes de AWS IoT criptografia (“Suporte de dispositivo TLS AWS IoT para conjuntos de criptografia recomendados”)
- TLS Receive Maximum Size Fragments("TLS Receive Maximum Size Fragments")
- TLS Expired Server Cert("Expired server certificate")
- TLS Large Size Server Cert("TLS large Size Server Certificate")
- MQTT Connect (“Dispositivo envie CONNECT para AWS IoT Core (Happy case)”)
- MQTT Subscribe ("Can Subscribe (Happy Case)")
- MQTT Publish ("QoS0 (Happy Case)")
- MQTT Connect Jitter Retries (“Device connect retries with jitter backoff - No CONNACK response”)

TLS

Use esses testes para determinar se o protocolo de segurança da camada de transporte (TLS) entre seus dispositivos AWS IoT é seguro.

> **Note**
>
> O Device Advisor agora é compatível com o TLS 1.3.

Happy Path

TLS Connect

Valida se o dispositivo em teste pode concluir o handshake TLS para. AWS IoT Esse teste não valida a implementação do MQTT do dispositivo cliente.

Example Definição do caso de teste da API:

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Para obter melhores resultados, recomendamos um valor de tempo limite de dois minutos.

```
"tests":[
   {
      "name":"my_tls_connect_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
      },
      "test":{
         "id":"TLS_Connect",
         "version":"0.0.0"
      }
   }
]
```

Example Saídas do caso de teste:

- Aprovação — O dispositivo em teste concluiu o handshake TLS com. AWS IoT
- Passe com avisos — O dispositivo em teste concluiu o handshake TLS com AWS IoT, mas houve mensagens de aviso de TLS do dispositivo ou. AWS IoT
- Falha — O dispositivo em teste falhou ao concluir o handshake TLS AWS IoT devido a um erro de handshake.

O TLS recebe fragmentos de tamanho máximo

Esse caso de teste valida que o dispositivo pode receber e processar fragmentos de tamanho máximo do TLS. O dispositivo de teste deve assinar um tópico pré-configurado com QoS 1 para receber uma grande carga. É possível personalizar a carga com a configuração `${payload}`.

Example Definição do caso de teste da API:

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Para obter melhores resultados, recomendamos um valor de tempo limite de dois minutos.

```
"tests":[
   {
      "name":"TLS Receive Maximum Size Fragments",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
         "PAYLOAD_FORMAT":"{"message":"${payload}"}", // A string with a placeholder
 ${payload}, or leave it empty to receive a plain string.
         "TRIGGER_TOPIC": "test_1" // A topic to which a device will subscribe, and
 to which a test case will publish a large payload.
      },
      "test":{
         "id":"TLS_Receive_Maximum_Size_Fragments",
         "version":"0.0.0"
      }
   }
]
```

Pacotes de criptografia

Suporte a dispositivos TLS para pacotes de AWS IoT criptografia recomendados

Valida se os conjuntos de cifras na mensagem Hello do Cliente TLS do dispositivo em teste contêm os conjuntos de cifras do AWS IoT recomendados. Fornece mais informações sobre os conjuntos de cifras compatíveis com o dispositivo.

Example Definição do caso de teste da API:

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 2 minutos.

```
"tests":[
   {
      "name":"my_tls_support_aws_iot_cipher_suites_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
      },
      "test":{
         "id":"TLS_Support_AWS_IoT_Cipher_Suites",
         "version":"0.0.0"
      }
   }
]
```

Example Saídas do caso de teste:

- Aprovado — O dispositivo em conjuntos de cifras de teste contém pelo menos um dos conjuntos de AWS IoT cifras recomendados e não contém nenhum conjunto de cifras não suportado.
- Passe com avisos — Os conjuntos de cifras do dispositivo contêm pelo menos um conjunto de cifras da AWS IoT , mas:

  1. Ele não contém nenhum dos conjuntos de cifras recomendados
  2. Ele contém pacotes de criptografia que não são suportados pelo. AWS IoT

  Sugerimos que você verifique se todos os conjuntos de cifras não compatíveis são seguros.
- Falha — O dispositivo em conjuntos de criptografia de teste não contém nenhum dos conjuntos de criptografia AWS IoT compatíveis.

Certificado de servidor de tamanho maior

Certificado de servidor de tamanho maior do TLS

Valida se o dispositivo pode concluir o handshake TLS com AWS IoT quando recebe e processa um certificado de servidor de tamanho maior. O tamanho do certificado do servidor (em bytes) usado por esse teste é maior do que o usado atualmente no caso de teste do TLS Connect e do IoT Core em 20. Durante esse caso de teste AWS IoT , testa o espaço do buffer do seu dispositivo para TLS. Se o espaço do buffer for grande o suficiente, o handshake TLS será concluído sem erros. Esse teste não valida a implementação do MQTT do dispositivo. O caso de teste ocorre após a conclusão do processo de handshake do TLS.

Example Definição do caso de teste da API:

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Para obter melhores resultados, recomendamos um valor de tempo limite de dois minutos. Se esse caso de teste falhar, mas o caso de teste do TLS Connect for aprovado, recomendamos que você aumente o limite de espaço de buffer do dispositivo para TLS. Aumentar os limites de espaço de buffer garante que o dispositivo possa processar um certificado de servidor de tamanho maior caso o tamanho aumente.

```
"tests":[
   {
      "name":"my_tls_large_size_server_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
      },
      "test":{
         "id":"TLS_Large_Size_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example Saídas do caso de teste:

- Aprovação — O dispositivo em teste concluiu o handshake TLS com AWS IoT.

- Passe com avisos — O dispositivo em teste concluiu o handshake TLS com AWS IoT, mas há mensagens de aviso de TLS do dispositivo ou. AWS IoT
- Falha — O dispositivo em teste falhou ao concluir o handshake TLS AWS IoT devido a um erro durante o processo de handshake.

Certificado de servidor TLS desprotegido

Não assinado por uma CA reconhecida

Valida que o dispositivo em teste fecha a conexão se for apresentado um certificado de servidor sem uma assinatura válida da CA ATS. Um dispositivo só deve se conectar a um endpoint que apresente um certificado válido.

Example Definição do caso de teste da API:

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 2 minutos.

```
"tests":[
   {
      "name":"my_tls_unsecure_server_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
      },
      "test":{
         "id":"TLS_Unsecure_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example Saídas do caso de teste:

- Passe — O dispositivo em teste fechou a conexão.
- Falha — O dispositivo em teste concluiu o handshake TLS com. AWS IoT

Certificado de servidor de nome de assunto incorreto do TLS/Nome comum do assunto incorreto (CN)/Nome alternativo do assunto (SAN)

Valida se o dispositivo em teste fecha a conexão se for apresentado um certificado de servidor para um nome de domínio diferente do solicitado.

Example Definição do caso de teste da API:

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 2 minutos.

```
"tests":[
   {
      "name":"my_tls_incorrect_subject_name_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",   // in seconds
      },
      "test":{
         "id":"TLS_Incorrect_Subject_Name_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example Saídas do caso de teste:

- Passe — O dispositivo em teste fechou a conexão.
- Falha — O dispositivo em teste concluiu o handshake TLS com. AWS IoT

Certificado de servidor TLS expirado

Certificado de servidor expirado

Valida que o dispositivo em teste fecha a conexão se for apresentado um certificado de servidor expirado.

Example Definição do caso de teste da API:

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 2 minutos.

```
"tests":[
   {
      "name":"my_tls_expired_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
      },
      "test":{
         "id":"TLS_Expired_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example Saídas do caso de teste:

- Aprovação — O dispositivo em teste se recusa a concluir o handshake TLS com. AWS IoT O dispositivo envia uma mensagem de alerta do TLS antes de fechar a conexão.
- Passe com avisos — O dispositivo em teste se recusa a concluir o handshake TLS com AWS IoT. No entanto, ele não envia uma mensagem de alerta do TLS antes de fechar a conexão.
- Falha — O dispositivo em teste conclui o handshake TLS com. AWS IoT

MQTT

CONECTAR, DESCONECTAR e RECONECTAR

“Dispositivo envie CONNECT para AWS IoT Core (Happy case)”

Valida se o dispositivo em teste envia uma solicitação CONNECT.

Definição do caso de teste da API:

> **Note**
>
> EXECUTION_TIMEOUT tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 2 minutos.

```
"tests":[
   {
      "name":"my_mqtt_connect_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",   // in seconds
      },
      "test":{
         "id":"MQTT_Connect",
         "version":"0.0.0"
      }
   }
]
```

“O dispositivo pode retornar o PUBACK a um tópico arbitrário para QoS1”

Esse caso de teste verificará se o dispositivo (cliente) pode retornar uma mensagem PUBACK se tiver recebido uma mensagem de publicação do agente após assinar um tópico com QoS1.

O conteúdo e o tamanho da carga são configuráveis para esse caso de teste. Se o tamanho da carga estiver configurado, o Device Advisor substituirá o valor do conteúdo da carga e enviará uma carga predefinida para o dispositivo com o tamanho desejado. O tamanho da carga é um valor entre 0 e 128 e não pode exceder 128 KB. O AWS IoT Core rejeita solicitações de publicação e conexão maiores que 128 KB, conforme visto na página de limites e cotas do protocolo e do agente de mensagens do AWS IoT Core.

Definição do caso de teste da API:

> **Note**
>
> EXECUTION_TIMEOUT tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 2 minutos. PAYLOAD_SIZE pode ser configurado para um valor entre 0 e 128 kilobytes. A definição de um tamanho de carga substitui o conteúdo da carga, pois

o Device Advisor enviará uma carga predefinida com o tamanho determinado de volta ao dispositivo.

```
"tests":[
{
        "name":"my_mqtt_client_puback_qos1",
        "configuration": {
            // optional:"TRIGGER_TOPIC": "myTopic",
            "EXECUTION_TIMEOUT":"300", // in seconds
            "PAYLOAD_FOR_PUBLISH_VALIDATION":"custom payload",
            "PAYLOAD_SIZE":"100" // in kilobytes
        },
        "test": {
            "id": "MQTT_Client_Puback_QoS1",
            "version": "0.0.0"
        }
    }
]
```

“Tentativas de conexão do dispositivo com recuo de variação de sinal - Sem resposta de CONNACK”

Valida se o dispositivo em teste usa o recuo de variação de sinal adequado ao se reconectar com o agente por pelo menos cinco vezes. O agente registra a data e hora do dispositivo sob a solicitação CONNECT do teste, realiza a validação do pacote, faz uma pausa sem enviar um CONNACK para o dispositivo em teste e espera que o dispositivo em teste reenvie a solicitação. A sexta tentativa de conexão tem permissão para passar, e o CONNACK tem permissão para fluir de volta para o dispositivo em teste.

O processo anterior é executado novamente. No total, esse caso de teste exige que o dispositivo se conecte pelo menos 12 vezes no total. Os registros de data e hora coletados são usados para validar se o recuo de variação de sinal é usado pelo dispositivo em teste. Se o dispositivo em teste tiver um atraso de recuo estritamente exponencial, esse caso de teste será aprovado com avisos.

Recomendamos a implementação do mecanismo Recuo exponencial e variação de sinal no dispositivo em teste para passar neste caso de teste.

Definição do caso de teste da API:

> **Note**
>
> EXECUTION_TIMEOUT tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 4 minutos.

```
"tests":[
   {
      "name":"my_mqtt_jitter_backoff_retries_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",    // in seconds
      },
      "test":{
         "id":"MQTT_Connect_Jitter_Backoff_Retries",
         "version":"0.0.0"
      }
   }
]
```

“Tentativas de conexão do dispositivo com recuo exponencial - Sem resposta de CONNACK”

Valida se o dispositivo em teste usa o recuo exponencial adequado ao se reconectar com o agente por pelo menos cinco vezes. O agente registra a data e a hora do dispositivo sob a solicitação CONNECT do teste, realiza a validação do pacote, faz uma pausa sem enviar um CONNACK para o dispositivo cliente e espera que o dispositivo em teste reenvie a solicitação. Os registros de data e hora coletados são usados para validar se um recuo exponencial é usado pelo dispositivo em teste.

Recomendamos a implementação do mecanismo Recuo exponencial e variação de sinal no dispositivo em teste para passar neste caso de teste.

Definição do caso de teste da API:

> **Note**
>
> EXECUTION_TIMEOUT tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 4 minutos.

```
"tests":[
   {
      "name":"my_mqtt_exponential_backoff_retries_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"600",  // in seconds
      },
      "test":{
         "id":"MQTT_Connect_Exponential_Backoff_Retries",
         "version":"0.0.0"
      }
   }
]
```

“Reconexão do dispositivo com recuo de variação de sinal - Após a desconexão do servidor”

Valida se um dispositivo em teste usa a variação de sinal e o recuo necessários ao se reconectar após ser desconectado do servidor. O Device Advisor desconecta o dispositivo do servidor por pelo menos cinco vezes e observa o comportamento do dispositivo na reconexão do MQTT. O Device Advisor registra a data e a hora da solicitação CONNECT para o dispositivo em teste, realiza a validação do pacote, faz uma pausa sem enviar um CONNACK para o dispositivo cliente e espera que o dispositivo em teste reenvie a solicitação. Os registros de data e hora coletados são usados para validar se o dispositivo em teste usa variação de sinal e recuo ao se reconectar. Se o dispositivo em teste tiver um recuo estritamente exponencial ou não implementar um mecanismo de recuo exponencial de variação de sinal adequado, esse caso de teste será aprovado com avisos. Se o dispositivo em teste tiver implementado um mecanismo de recuo linear ou um mecanismo de recuo constante, o teste falhará.

Para passar nesse caso de teste, recomendamos a implementação do mecanismo Recuo exponencial e variação de sinal no dispositivo em teste.

Definição do caso de teste da API:

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 4 minutos.

O número de tentativas de reconexão a serem validadas para recuo pode ser alterado especificando `RECONNECTION_ATTEMPTS`. O número deve estar entre 5 e 10. O valor padrão é 5.

```
"tests":[
   {
      "name":"my_mqtt_reconnect_backoff_retries_on_server_disconnect",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "RECONNECTION_ATTEMPTS": 5
      },
      "test":{
         "id":"MQTT_Reconnect_Backoff_Retries_On_Server_Disconnect",
         "version":"0.0.0"
      }
   }
]
```

“Reconexão do dispositivo com recuo de variação de sinal - Em conexão instável”

Valida se um dispositivo em teste usa a variação de sinal e o recuo necessários ao se reconectar em uma conexão instável. O Device Advisor desconecta o dispositivo do servidor após cinco conexões bem-sucedidas e observa o comportamento do dispositivo na reconexão do MQTT. O Device Advisor registra a data e a hora da solicitação CONNECT para o dispositivo em teste, realiza a validação do pacote, envia de volta um CONNACK, desconecta-se, registra a data e a hora da desconexão e espera que o dispositivo em teste reenvie a solicitação. Os registros de data e hora coletados são usados para validar se o dispositivo em teste usa variação de sinal e recuo ao se reconectar após conexões bem-sucedidas, mas instáveis. Se o dispositivo em teste tiver um recuo estritamente exponencial ou não implementar um mecanismo de recuo exponencial de variação de sinal adequado, esse caso de teste será aprovado com avisos. Se o dispositivo em teste tiver implementado um mecanismo de recuo linear ou um mecanismo de recuo constante, o teste falhará.

Para passar nesse caso de teste, recomendamos a implementação do mecanismo Recuo exponencial e variação de sinal no dispositivo em teste.

Definição do caso de teste da API:

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 4 minutos.
> O número de tentativas de reconexão a serem validadas para recuo pode ser alterado especificando `RECONNECTION_ATTEMPTS`. O número deve estar entre 5 e 10. O valor padrão é 5.

```
"tests":[
   {
      "name":"my_mqtt_reconnect_backoff_retries_on_unstable_connection",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "RECONNECTION_ATTEMPTS": 5
      },
      "test":{
         "id":"MQTT_Reconnect_Backoff_Retries_On_Unstable_Connection",
         "version":"0.0.0"
      }
   }
]
```

Publicar

“QoS0 (Happy Case)”

Valida se o dispositivo em teste publica uma mensagem com QoS0 ou QoS1. Você também pode validar o tópico da mensagem e da carga especificando o valor do tópico e a carga nas configurações de teste.

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 2 minutos.

```
"tests":[
```

```
   {
      "name":"my_mqtt_publish_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "TOPIC_FOR_PUBLISH_VALIDATION": "my_TOPIC_FOR_PUBLISH_VALIDATION",
         "PAYLOAD_FOR_PUBLISH_VALIDATION": "my_PAYLOAD_FOR_PUBLISH_VALIDATION",
      },
      "test":{
         "id":"MQTT_Publish",
         "version":"0.0.0"
      }
   }
]
```

“Tentativa de publicação de QoS1 - Sem PUBACK”

Valida que o dispositivo em teste republica uma mensagem enviada com QoS1, se o agente não enviar PUBACK. Você também pode validar o tópico da mensagem especificando esse tópico nas configurações de teste. O dispositivo cliente não deve se desconectar antes de republicar a mensagem. Esse teste também valida se a mensagem republicada tem o mesmo identificador de pacote que a original. Durante a execução do teste, se o dispositivo perder a conexão e se reconectar, o caso de teste será reiniciado sem falhas, e o dispositivo deverá executar as etapas do caso de teste novamente.

Definição do caso de teste da API:

> **Note**
>
> EXECUTION_TIMEOUT tem um valor padrão de cinco minutos. É recomendado por pelo menos 4 minutos.

```
"tests":[
   {
      "name":"my_mqtt_publish_retry_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "TOPIC_FOR_PUBLISH_VALIDATION": "my_TOPIC_FOR_PUBLISH_VALIDATION",
         "PAYLOAD_FOR_PUBLISH_VALIDATION": "my_PAYLOAD_FOR_PUBLISH_VALIDATION",
```

```
      },
      "test":{
         "id":"MQTT_Publish_Retry_No_Puback",
         "version":"0.0.0"
      }
   }
]
```

“Publicar mensagens retidas”

Valida se o dispositivo em teste publica uma mensagem com `retainFlag` definida como true. Você também pode validar o tópico e a carga da mensagem definindo o valor do tópico e a carga nas configurações de teste. Se o `retainFlag` enviado dentro do pacote PUBLICAR não estiver definido como true, o caso de teste falhará.

Definição do caso de teste da API:

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 2 minutos. Para executar esse caso de teste, adicione a ação `iot:RetainPublish` no perfil do dispositivo.

```
"tests":[
   {
      "name":"my_mqtt_publish_retained_messages_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds

 "TOPIC_FOR_PUBLISH_RETAINED_VALIDATION": "my_TOPIC_FOR_PUBLISH_RETAINED_VALIDATION",

 "PAYLOAD_FOR_PUBLISH_RETAINED_VALIDATION": "my_PAYLOAD_FOR_PUBLISH_RETAINED_VALIDATION",
      },
      "test":{
         "id":"MQTT_Publish_Retained_Messages",
         "version":"0.0.0"
      }
   }
]
```

“Publicar com propriedade do usuário”

Valida se o dispositivo em teste publica uma mensagem com a propriedade de usuário correta. Você pode validar a propriedade do usuário definindo o par nome-valor nas configurações de teste. Se a propriedade do usuário não for fornecida ou não corresponder, o caso de teste falhará.

Definição do caso de teste da API:

> **Note**
>
> Esse é o MQTT5 único caso de teste.
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 2 minutos.

```
"tests":[
   {
      "name":"my_mqtt_user_property_test",
      "test":{
        "USER_PROPERTIES": [
            {"name": "name1", "value":"value1"},
            {"name": "name2", "value":"value2"}
        ],
        "EXECUTION_TIMEOUT":"300",  // in seconds
      },
      "test":{
         "id":"MQTT_Publish_User_Property",
         "version":"0.0.0"
      }
   }
]
```

Assinar

“Pode assinar (Happy Case)”

Valida se o dispositivo em teste assina os tópicos do MQTT. Você também pode validar o tópico que o dispositivo em teste assina especificando esse tópico nas configurações de teste.

Definição do caso de teste da API:

> **Note**
>
> EXECUTION_TIMEOUT tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 2 minutos.

```
"tests":[
   {
      "name":"my_mqtt_subscribe_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "TOPIC_LIST_FOR_SUBSCRIPTION_VALIDATION":
["my_TOPIC_FOR_PUBLISH_VALIDATION_a","my_TOPIC_FOR_PUBLISH_VALIDATION_b"]
      },
      "test":{
         "id":"MQTT_Subscribe",
         "version":"0.0.0"
      }
   }
]
```

“Nova tentativa de assinar - Sem SUBACK”

Valida se o dispositivo em teste tenta novamente uma assinatura com falha dos tópicos do MQTT. O servidor, então, espera e não envia um SUBACK. Se o dispositivo cliente não repetir a assinatura, o teste falhará. O dispositivo cliente deve tentar novamente a assinatura com falha com o mesmo ID de pacote. Você também pode validar o tópico que o dispositivo em teste assina especificando esse tópico nas configurações de teste. Durante a execução do teste, se o dispositivo perder a conexão e se reconectar, o caso de teste será reiniciado sem falhas, e o dispositivo deverá executar as etapas do caso de teste novamente.

Definição do caso de teste da API:

> **Note**
>
> EXECUTION_TIMEOUT tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 4 minutos.

```
"tests":[
   {
      "name":"my_mqtt_subscribe_retry_test",
      "configuration":{
         "EXECUTION_TIMEOUT":"300",  // in seconds
         // optional:
         "TOPIC_LIST_FOR_SUBSCRIPTION_VALIDATION":
["myTOPIC_FOR_PUBLISH_VALIDATION_a","my_TOPIC_FOR_PUBLISH_VALIDATION_b"]
      },
      "test":{
         "id":"MQTT_Subscribe_Retry_No_Suback",
         "version":"0.0.0"
      }
   }
]
```

Keep-alive

“Matt No Ack PingResp”

Este caso de teste valida se o dispositivo em teste se desconecta quando não recebe uma resposta de ping. Como parte desse caso de teste, o Device Advisor bloqueia respostas enviadas AWS IoT Core para solicitações de publicação, assinatura e ping. Também valida se o dispositivo em teste desconecta a conexão do MQTT.

Definição do caso de teste da API:

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Recomendamos um tempo limite maior que 1,5 vezes o valor `keepAliveTime`.
> O máximo `keepAliveTime` não deve ser maior que 230 segundos para este teste.

```
"tests":[
    {
       "name":"Mqtt No Ack PingResp",
       "configuration":
          //optional:
          "EXECUTION_TIMEOUT":"306",   // in seconds
```

```
        },
        "test":{
           "id":"MQTT_No_Ack_PingResp",
           "version":"0.0.0"
        }
    }
]
```

Sessão persistente

“Sessão persistente (Happy Case)”

Este caso de teste valida o comportamento do dispositivo quando desconectado de uma sessão persistente. O caso de teste verifica se o dispositivo pode se reconectar, retomar as assinaturas dos tópicos acionadores sem assinar de novo explicitamente, receber as mensagens armazenadas nos tópicos e funcionar conforme o esperado durante uma sessão persistente. Quando esse caso de teste é aprovado, ele indica que o dispositivo cliente é capaz de manter uma sessão persistente com o AWS IoT Core broker da maneira esperada. Para obter mais informações sobre sessões AWS IoT persistentes, consulte Usando sessões persistentes do MQTT.

Nesse caso de teste, espera-se que o dispositivo cliente faça CONNECT com o AWS IoT Core com o sinalizador de sessão limpa definido como false e, em seguida, assine um tópico acionador. Após uma assinatura bem-sucedida, o dispositivo será desconectado pelo AWS IoT Core Device Advisor. Enquanto o dispositivo estiver em um estado desconectado, uma carga de mensagem de QoS 1 será armazenada nesse tópico. O Device Advisor permitirá então que o dispositivo cliente se reconecte ao endpoint de teste. Nesse ponto, como há uma sessão persistente, espera-se que o dispositivo cliente retome assinaturas de tópicos sem enviar nenhum pacote SUBSCRIBE adicional e receba a mensagem de QoS 1 do agente. Após a reconexão, se o dispositivo cliente se inscrever novamente em seu tópico acionador enviando um pacote SUBSCRIBE adicional, and/or se o cliente não receber a mensagem armazenada do tópico acionador, o caso de teste falhará.

Definição do caso de teste da API:

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de pelo menos 4 minutos. Na primeira conexão, o dispositivo

cliente precisa assinar explicitamente um `TRIGGER_TOPIC` que não estava assinado antes. Para passar no caso de teste, o dispositivo cliente deve assinar com sucesso o `TRIGGER_TOPIC` com um QoS 1. Depois de se reconectar, espera-se que o dispositivo cliente entenda que há uma sessão persistente ativa; portanto, ele deve aceitar a mensagem armazenada enviada pelo tópico acionador e retornar PUBACK para essa mensagem específica.

```
"tests":[
   {
      "name":"my_mqtt_persistent_session_happy_case",
      "configuration":{
         //required:
         "TRIGGER_TOPIC": "myTrigger/topic",
         // optional:
         // if Payload not provided, a string will be stored in the trigger topic to
 be sent back to the client device
         "PAYLOAD": "The message which should be received from AWS IoT Broker after
 re-connecting to a persistent session from the specified trigger topic.",

         "EXECUTION_TIMEOUT":"300" // in seconds
      },
      "test":{
         "id":"MQTT_Persistent_Session_Happy_Case",
         "version":"0.0.0"
      }
   }
]
```

“Sessão persistente - Expiração da sessão”

Este caso de teste ajuda a validar o comportamento do dispositivo quando um dispositivo desconectado se reconecta a uma sessão persistente expirada. Depois que a sessão expirar, esperamos que o dispositivo assine novamente os tópicos assinados anteriormente, enviando explicitamente um novo pacote SUBSCRIBE.

Durante a primeira conexão, esperamos que o dispositivo de teste SE CONECTE ao agente de AWS IoT, pois seu `CleanSession` sinalizador está definido como falso para iniciar uma sessão persistente. O dispositivo deve então assinar um tópico acionador. Em seguida, o dispositivo é desconectado pelo AWS IoT Core Device Advisor, após uma assinatura bem-sucedida e o início

de uma sessão persistente. Após a desconexão, o AWS IoT Core Device Advisor permite que o dispositivo de teste se reconecte ao endpoint de teste. Nesse ponto, quando o dispositivo de teste envia outro pacote CONNECT, o AWS IoT Core Device Advisor envia de volta um pacote CONNACK que indica que a sessão persistente expirou. O dispositivo de teste precisa interpretar esse pacote corretamente e espera-se que ele assine novamente o mesmo tópico acionador quando a sessão persistente for encerrada. Se o dispositivo de teste não assinar novamente o tópico acionador, o caso de teste falhará. Para que o teste seja aprovado, o dispositivo precisa entender que a sessão persistente acabou e enviar de volta um novo pacote SUBSCRIBE para o mesmo tópico acionador na segunda conexão.

Se esse caso de teste for aprovado em um dispositivo de teste, isso indicará que o dispositivo é capaz de lidar com a reconexão após a expiração da sessão persistente de uma forma esperada.

Definição do caso de teste da API:

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de pelo menos 4 minutos. O dispositivo de teste precisa assinar explicitamente um `TRIGGER_TOPIC`, o qual não assinava antes. Para passar no caso de teste, o dispositivo de teste deve enviar um pacote CONNECT com o sinalizador `CleanSession` definido como false e assinar com êxito um tópico acionador com um QoS 1. Depois de uma conexão bem-sucedida, AWS IoT Core o Device Advisor desconecta o dispositivo. Após a desconexão, o AWS IoT Core Device Advisor permite que o dispositivo se reconecte, e espera-se que o dispositivo se inscreva novamente, `TRIGGER_TOPIC` pois o AWS IoT Core Device Advisor teria encerrado a sessão persistente.

```
"tests":[
   {
      "name":"my_expired_persistent_session_test",
      "configuration":{
         //required:
         "TRIGGER_TOPIC": "myTrigger/topic",
         // optional:
         "EXECUTION_TIMEOUT":"300" // in seconds
      },
      "test":{
```

```
         "id":"MQTT_Expired_Persistent_Session",
         "version":"0.0.0"
      }
   }
]
```

Shadow

Use esses testes para verificar se seus dispositivos em teste usam o serviço AWS IoT Device Shadow corretamente. Consulte AWS IoT Serviço Device Shadow para obter mais informações. Se esses casos de teste estiverem configurados no conjunto de testes, será necessário fornecer um objeto ao iniciar a execução do conjunto.

O MQTT over não WebSocket é suportado no momento.

Publicar

“O dispositivo publica o estado após a conexão (Happy Case)”

Valida se um dispositivo pode publicar seu estado depois de se conectar ao AWS IoT Core

Definição do caso de teste da API:

> **Note**
>
> EXECUTION_TIMEOUT tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 2 minutos.

```
"tests":[
   {
      "name":"my_shadow_publish_reported_state",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300", // in seconds
         "SHADOW_NAME": "SHADOW_NAME",
         "REPORTED_STATE": {
            "STATE_ATTRIBUTE": "STATE_VALUE"
         }
      },
      "test":{
```

```
         "id":"Shadow_Publish_Reported_State",
         "version":"0.0.0"
      }
   }
]
```

Os `REPORTED_STATE` podem ser fornecidos para validação adicional do estado exato da sombra do dispositivo, após a conexão. Por padrão, esse caso de teste valida o estado de publicação do dispositivo.

Se *`SHADOW_NAME`* não for fornecido, o caso de teste procurará mensagens publicadas em prefixos de tópicos do tipo de sombra Sem nome (clássico) por padrão. Forneça um nome de sombra se o dispositivo usar o tipo de sombra nomeado. Consulte Como usar sombras em dispositivos para obter mais informações.

Atualizar

“O dispositivo atualiza o estado reportado para o estado desejado (Happy Case)”

Valida se o dispositivo lê todas as mensagens de atualização recebidas e sincroniza o estado do dispositivo para corresponder às propriedades de estado desejadas. O dispositivo deve publicar o último estado relatado após a sincronização. Se o dispositivo já tiver uma sombra existente antes de executar o teste, certifique-se de que o estado desejado configurado para o caso de teste e o estado relatado existente ainda não correspondam. Você pode identificar as mensagens de atualização do Shadow enviadas pelo Device Advisor examinando o `ClientToken`campo no documento Shadow como ele será`DeviceAdvisorShadowTestCaseSetup`.

Definição do caso de teste da API:

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 2 minutos.

```
"tests":[
   {
      "name":"my_shadow_update_reported_state",
      "configuration": {
```

```
         "DESIRED_STATE": {
            "STATE_ATTRIBUTE": "STATE_VALUE"
         },
         // optional:
         "EXECUTION_TIMEOUT":"300", // in seconds
         "SHADOW_NAME": "SHADOW_NAME"
      },
      "test":{
         "id":"Shadow_Update_Reported_State",
         "version":"0.0.0"
      }
   }
]
```

O DESIRED_STATE deve ter pelo menos um atributo e um valor associado.

Se SHADOW_NAME não for fornecido, o caso de teste procurará mensagens publicadas em prefixos de tópicos do tipo de sombra Sem nome (clássico) por padrão. Forneça um nome de sombra se o dispositivo usar o tipo de sombra nomeado. Consulte Como usar sombras em dispositivos para obter mais informações.

Execução de trabalho

“O dispositivo pode concluir a execução de um trabalho”

Esse caso de teste ajuda você a validar se seu dispositivo é capaz de receber atualizações usando o AWS IoT Jobs e publicar o status de atualizações bem-sucedidas. Para obter mais informações sobre AWS IoT trabalhos, consulte Trabalhos.

Para executar esse caso de teste com êxito, há dois tópicos da AWS reservados que você precisa conceder ao Perfil de dispositivo. Para assinar mensagens relacionadas a atividades de trabalhos, use os tópicos notify e notify-next. O perfil de dispositivo deve conceder a ação PUBLICAR para os seguintes tópicos:

- $aws/things/thingName/jobs/jobId/get
- $aws/things/thingName/jobs/jobId/update

É recomendável conceder ações de SUBSCRIBE e RECEIVE para os seguintes tópicos:

- $aws/things/ ThingName/jobs/get/accepted
- $aws/things/thingName/jobs/jobId/get/rejected

- $aws/things/thingName/jobs/jobId/update/accepted
- $aws/things/thingName/jobs/jobId/update/rejected

É recomendável conceder ações de SUBSCRIBE para o seguinte tópico:

- $aws/things/thingName/jobs/notify-next

Para obter mais informações sobre esses tópicos reservados, consulte tópicos reservados para Trabalhos de AWS IoT.

O MQTT over não WebSocket é suportado no momento.

Definição do caso de teste da API:

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de cinco minutos. Recomendamos um valor de tempo limite de 3 minutos. Dependendo do documento do AWS IoT Job ou da fonte fornecida, ajuste o valor do tempo limite (por exemplo, se um trabalho levar muito tempo para ser executado, defina um valor de tempo limite maior para o caso de teste). Para executar o teste, é necessário um documento de AWS IoT trabalho válido ou uma ID de trabalho já existente. Um documento AWS IoT Job pode ser fornecido como um documento JSON ou um link S3. Se um documento de trabalho for fornecido, fornecer um ID de trabalho é opcional. Se um ID de trabalho for fornecido, o Device Advisor usará esse ID ao criar o AWS IoT Job em seu nome. Se o documento do trabalho não for fornecido, você poderá fornecer um ID existente que esteja na mesma região em que você está executando o caso de teste. Nesse caso, o Device Advisor usará esse AWS IoT Job ao executar o caso de teste.

```
"tests": [
   {
      "name":"my_job_execution",
      "configuration": {
         // optional:
         // Test case will create a job task by using either JOB_DOCUMENT or
 JOB_DOCUMENT_SOURCE.
         // If you manage the job task on your own, leave it empty and provide the
 JOB_JOBID (self-managed job task).
         // JOB_DOCUMENT is a JSON formatted string
         "JOB_DOCUMENT": "{
```

```
            \"operation\":\"reboot\",
            \"files\" : {
               \"fileName\" : \"install.py\",
               \"url\" : \"${aws:iot:s3-presigned-url:https://s3.amazonaws.com/
bucket-name/key}\"
            }
         }",
         // JOB_DOCUMENT_SOURCE is an S3 link to the job document. It will be used
 only if JOB_DOCUMENT is not provided.
         "JOB_DOCUMENT_SOURCE": "https://s3.amazonaws.com/bucket-name/key",
         // JOB_JOBID is mandatory, only if neither document nor document source is
 provided. (Test case needs to know the self-managed job task id).
         "JOB_JOBID": "String",
         // JOB_PRESIGN_ROLE_ARN is used for the presign Url, which will replace the
 placeholder in the JOB_DOCUMENT field
         "JOB_PRESIGN_ROLE_ARN": "String",
         // Presigned Url expiration time. It must be between 60 and 3600 seconds,
 with the default value being 3600.
         "JOB_PRESIGN_EXPIRES_IN_SEC": "Long"
         "EXECUTION_TIMEOUT": "300", // in seconds
      },
      "test": {
         "id": "Job_Execution",
         "version": "0.0.0"
      }
   }
]
```

Para obter mais informações sobre como criar e usar documentos de trabalho, consulte o documento de trabalho.

Políticas e permissões

Você pode usar os testes a seguir para determinar se as políticas anexadas aos certificados dos dispositivos seguem as melhores práticas padrão.

O MQTT over não WebSocket é suportado no momento.

“As políticas anexadas ao certificado do dispositivo não contêm curingas”

Valida se as políticas de permissão associadas a um dispositivo seguem as melhores práticas e não concedem ao dispositivo mais permissões do que o necessário.

Definição do caso de teste da API:

> **Note**
>
> `EXECUTION_TIMEOUT` tem um valor padrão de 1 minuto. Recomendamos definir um tempo limite de pelo menos 30 segundos.

```
"tests":[
   {
        "name":"my_security_device_policies",
        "configuration": {
            // optional:
            "EXECUTION_TIMEOUT":"60"    // in seconds
        },
        "test": {
            "id": "Security_Device_Policies",
            "version": "0.0.0"
        }
    }
]
```

Testes de longa duração

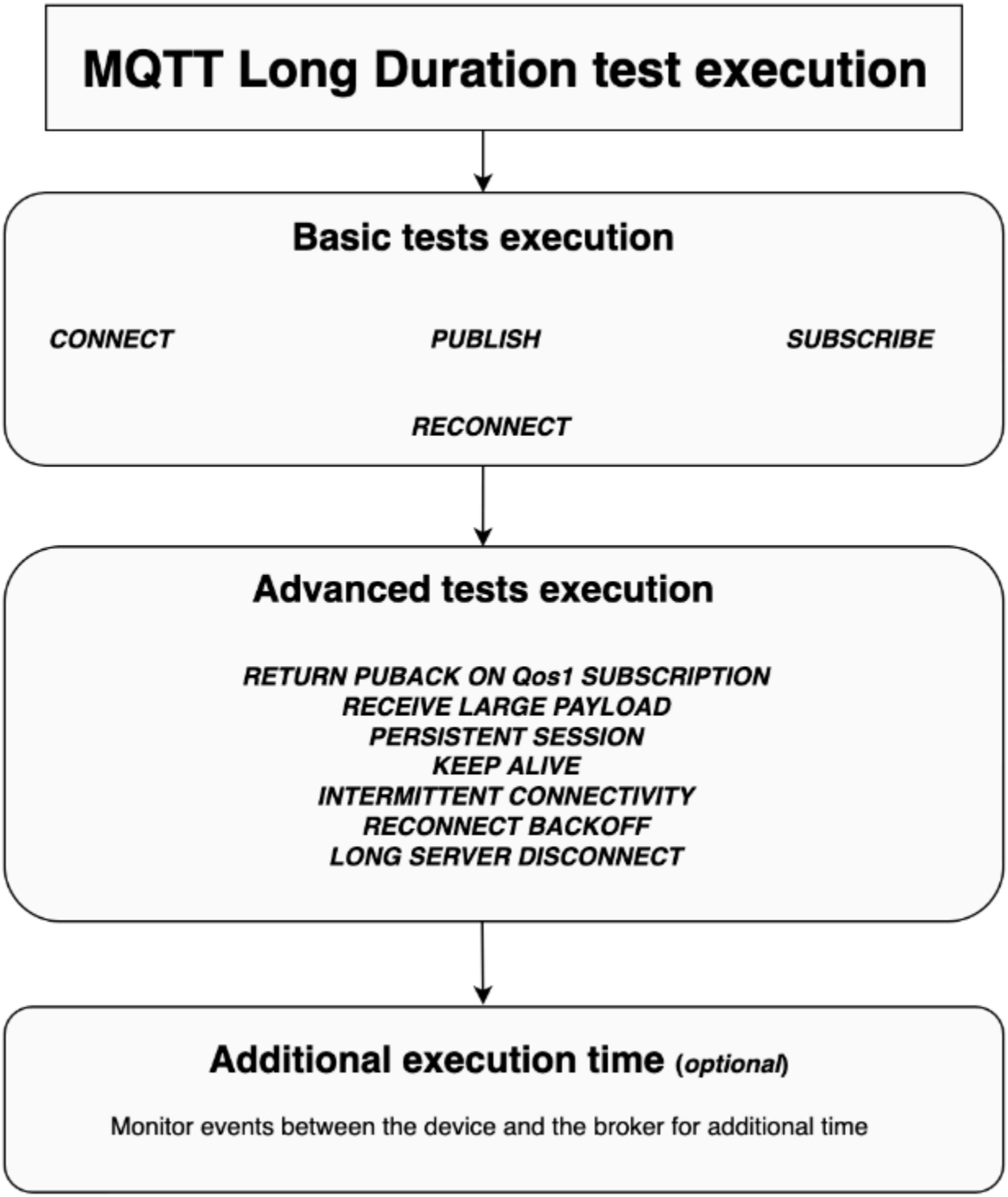
Os testes de longa duração são um novo conjunto de testes que monitora o comportamento de um dispositivo quando ele opera por longos períodos de tempo. Em comparação com a execução de testes individuais que se concentram em comportamentos específicos de um dispositivo, o teste de longa duração examina o comportamento do dispositivo em uma variedade de cenários do mundo real ao longo da vida útil do dispositivo. O Device Advisor organiza os testes na ordem mais eficiente possível. O teste gera resultados e logs, incluindo um log de resumo com métricas úteis sobre o desempenho do dispositivo durante o teste.

Caso de teste de longa duração do MQTT

No caso de teste de longa duração do MQTT, o comportamento do dispositivo é observado inicialmente em cenários felizes, como MQTT Connect, Subscribe, Publish e Reconnect. Em seguida, o dispositivo é observado em vários cenários de falha complexos, como MQTT Reconnect Backoff, Long Server Disconnect e Intermittent Connectivity.

Fluxo de execução de casos de teste de longa duração do MQTT

Há três fases na execução de um caso de teste de longa duração do MQTT:



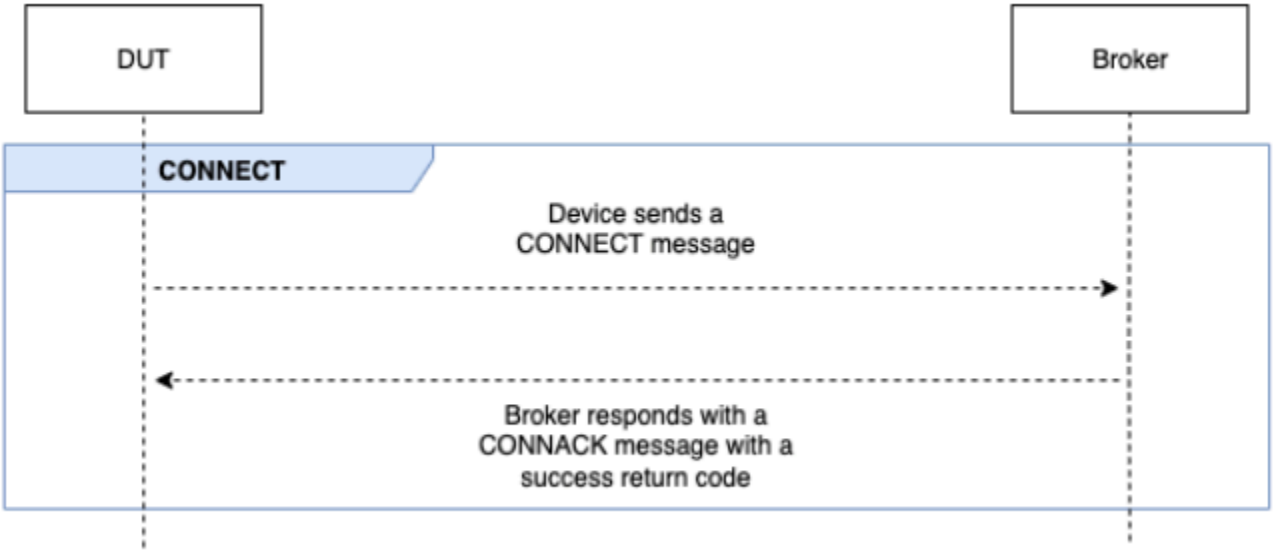
Execução de testes básicos

Nesta fase, o caso de teste executa testes simples em paralelo. O teste valida se o dispositivo tem as operações selecionadas na configuração.

O conjunto de testes básicos pode incluir o seguinte, com base nas operações selecionadas:

CONECTAR

Esse cenário valida se o dispositivo é capaz de fazer uma conexão bem-sucedida com o agente.

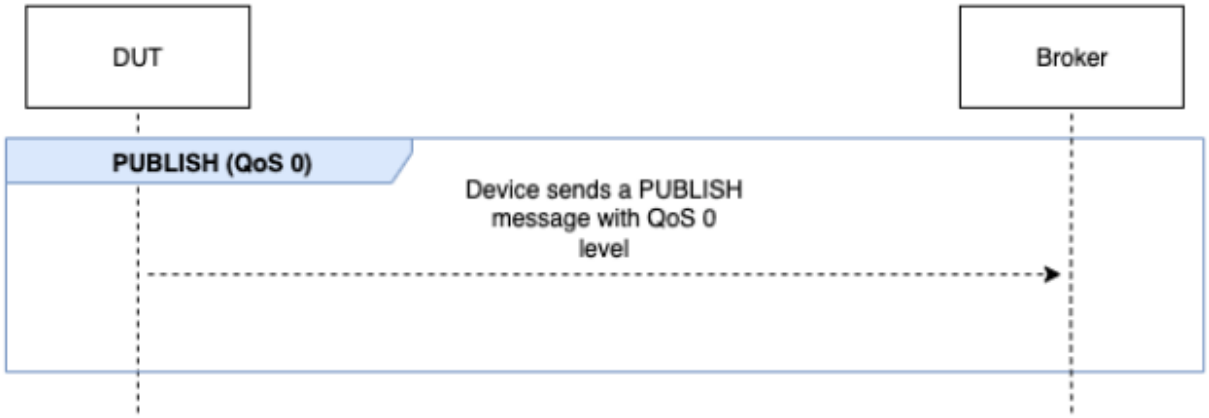


PUBLICAR

Este cenário valida se o dispositivo publica com êxito no agente.

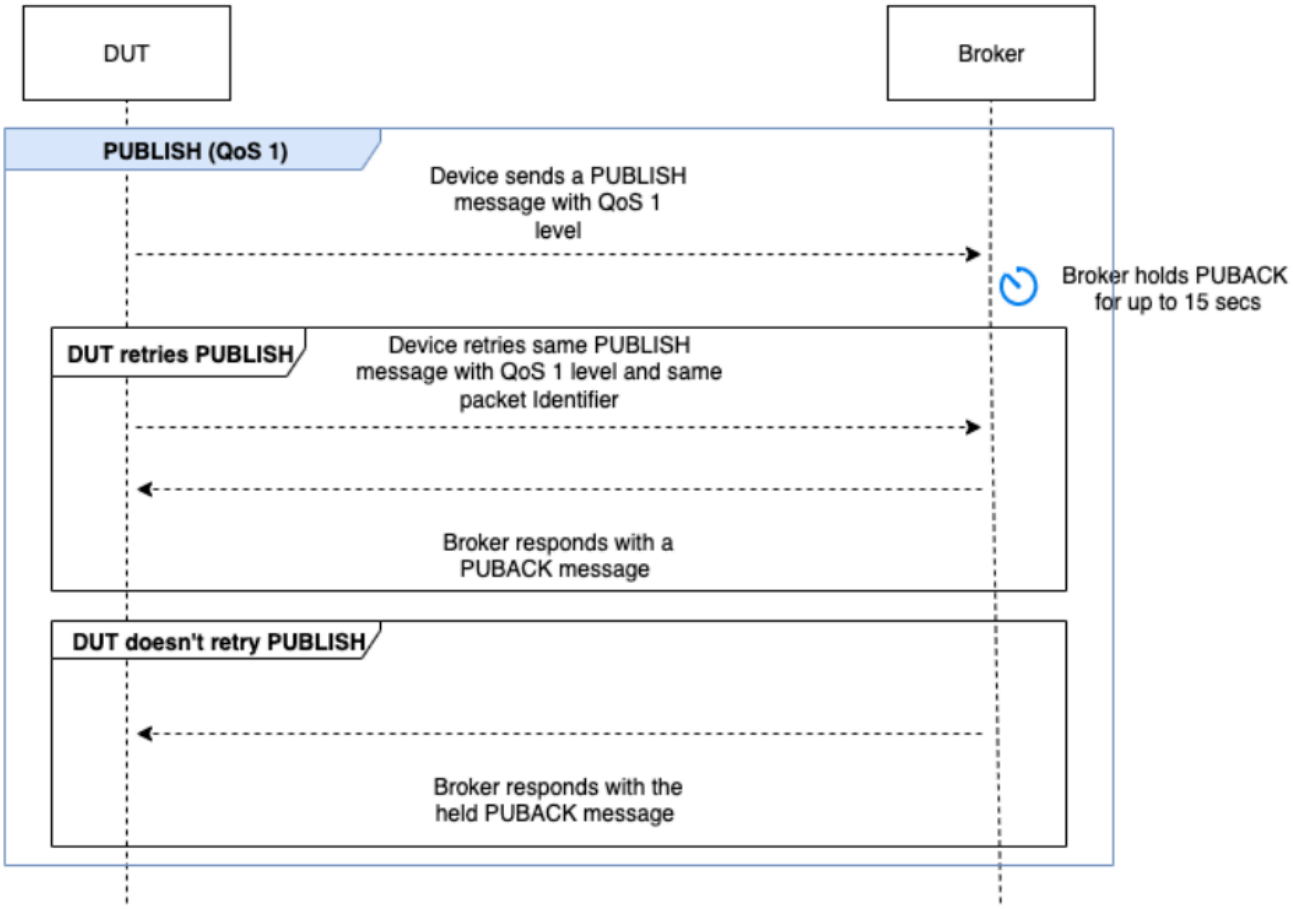
QoS 0

Este caso de teste valida se o dispositivo envia com êxito uma mensagem `PUBLISH` ao agente durante uma publicação com QoS 0. O teste não espera que a mensagem `PUBACK` seja recebida pelo dispositivo.



QoS 1

Neste caso de teste, espera-se que o dispositivo envie duas mensagens `PUBLISH` ao agente com QoS 1. Após a primeira mensagem `PUBLISH`, o agente espera por até 15 segundos antes de responder. O dispositivo deve repetir a mensagem `PUBLISH` original com o mesmo identificador de pacote dentro da janela de 15 segundos. Se isso acontecer, o agente responde com uma mensagem `PUBACK`, e o teste é validado. Se o dispositivo não tentar novamente a `PUBLISH`, a `PUBACK` original será enviada ao dispositivo, e o teste será marcado como Aprovado com avisos, junto com uma mensagem do sistema. Durante a execução do teste, se o dispositivo perder a conexão e se reconectar, o caso de teste será reiniciado sem falhas, e o dispositivo deverá executar as etapas do cenário de teste novamente.

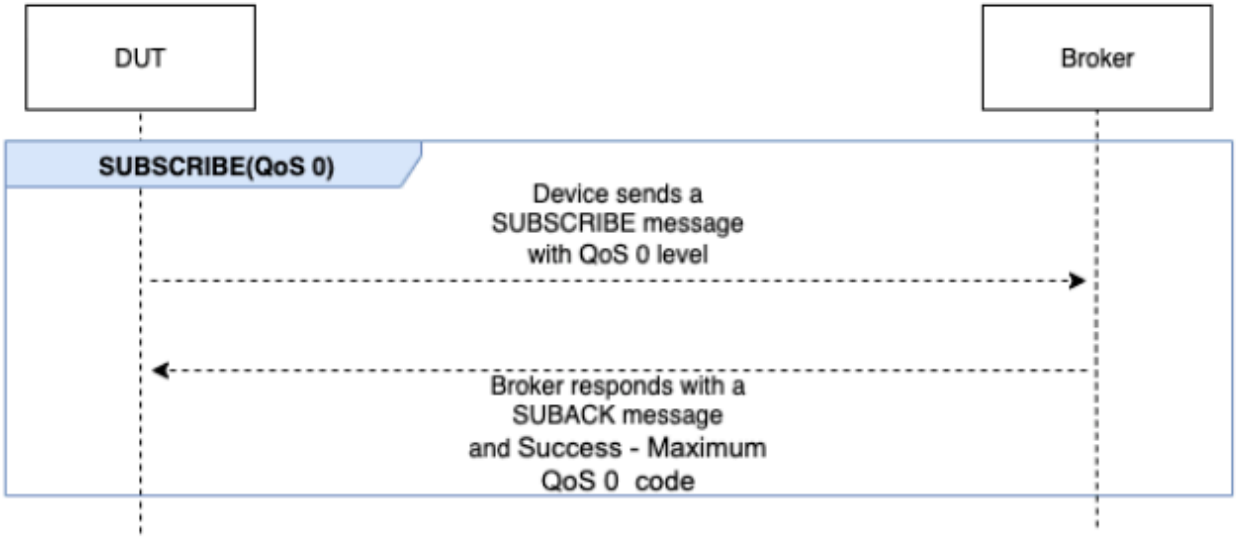


ASSINAR

Este cenário valida se o dispositivo assina com êxito o agente.

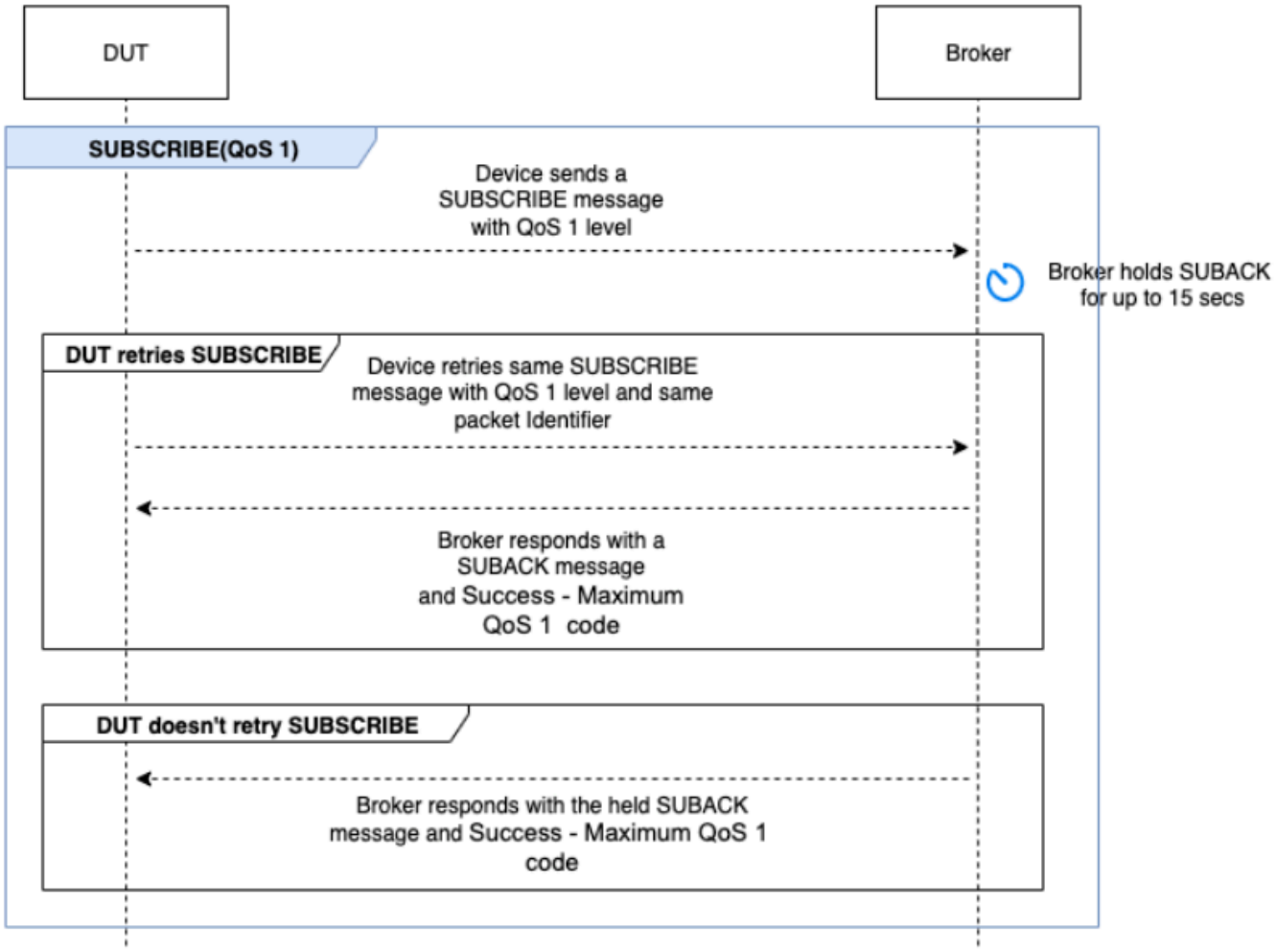
QoS 0

Este caso de teste valida se o dispositivo envia com êxito uma mensagem `SUBSCRIBE` ao agente durante uma assinatura com QoS 0. O teste não espera que o dispositivo receba uma mensagem SUBACK.



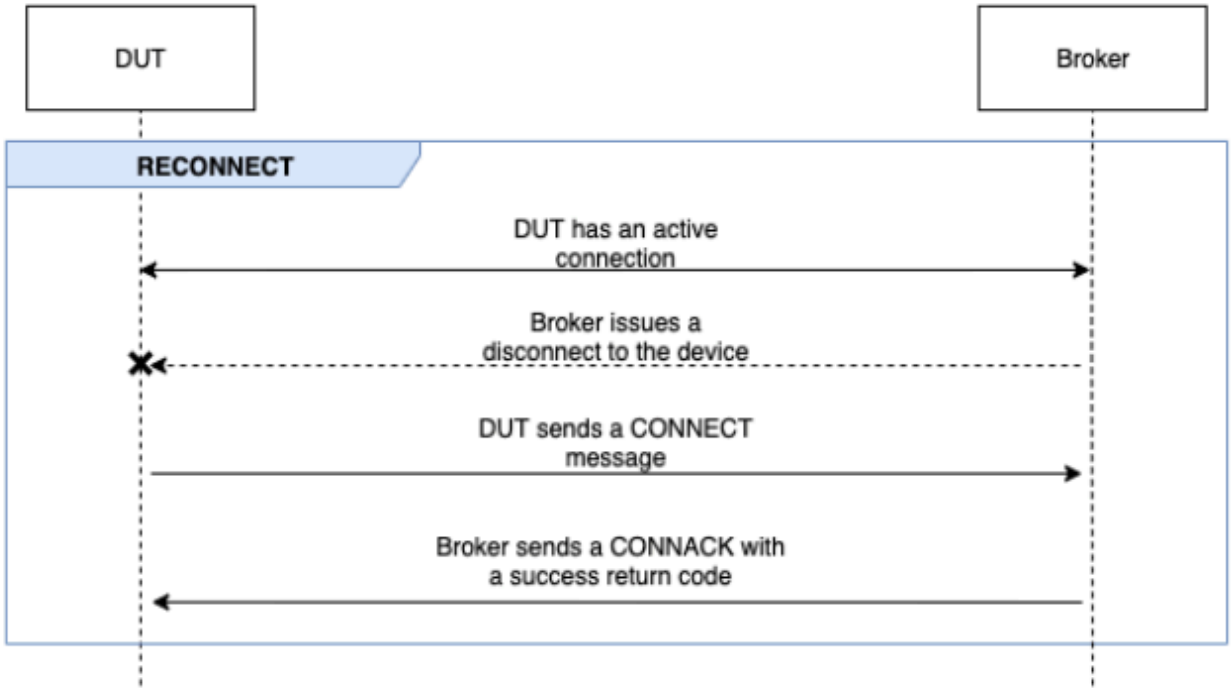
QoS 1

Neste caso de teste, espera-se que o dispositivo envie duas mensagens `SUBSCRIBE` ao agente com QoS 1. Após a primeira mensagem `SUBSCRIBE`, o agente espera por até 15 segundos antes de responder. O dispositivo deve repetir a mensagem `SUBSCRIBE` original com o mesmo identificador de pacote dentro da janela de 15 segundos. Se isso acontecer, o agente responde com uma mensagem `SUBACK`, e o teste é validado. Se o dispositivo não tentar novamente a `SUBSCRIBE`, a `SUBACK` original será enviada ao dispositivo, e o teste será marcado como Aprovado com avisos, junto com uma mensagem do sistema. Durante a execução do teste, se o dispositivo perder a conexão e se reconectar, o caso de teste será reiniciado sem falhas, e o dispositivo deverá executar as etapas do cenário de teste novamente.



RECONECTAR

Este cenário valida se o dispositivo se reconecta com êxito ao agente após o dispositivo ser desconectado de uma conexão bem-sucedida. O Device Advisor não desconectará o dispositivo se ele estiver conectado mais de uma vez anteriormente durante o conjunto de testes. Em vez disso, ele marcará o teste como Aprovado.



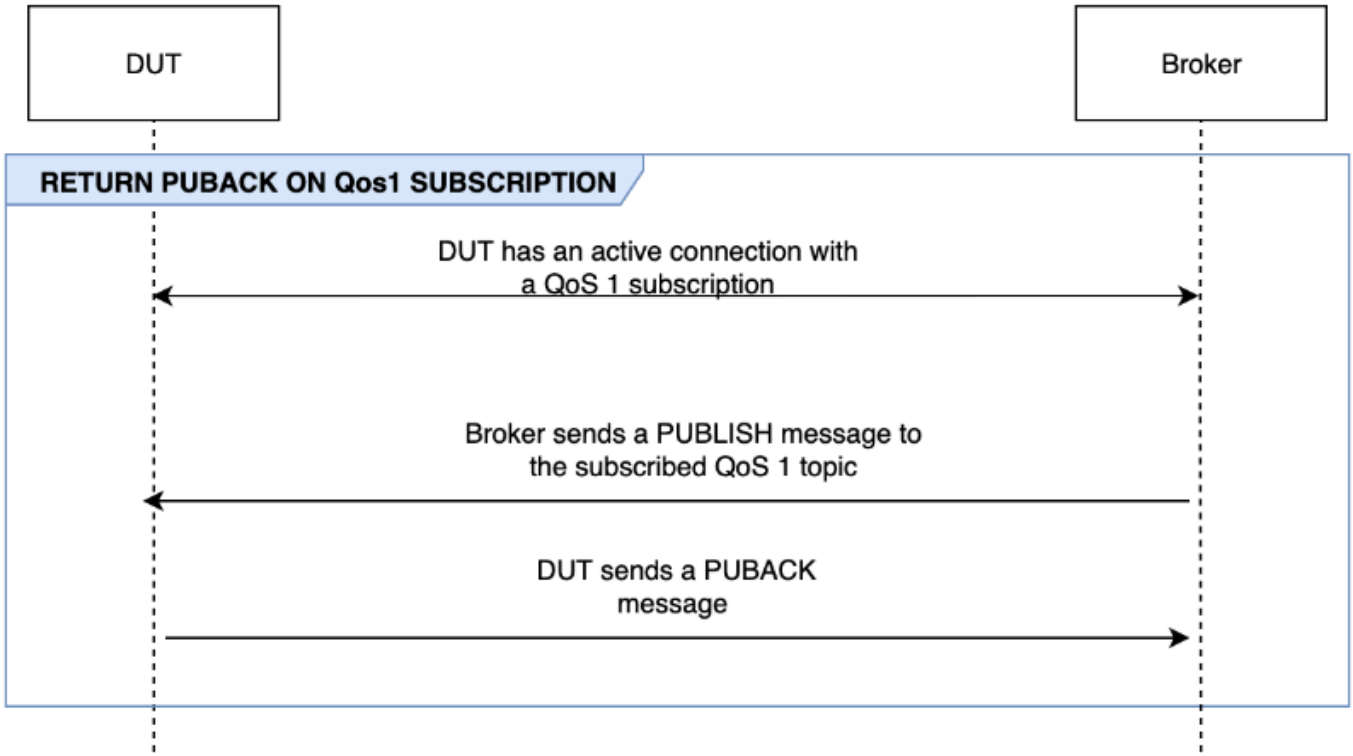
Execução de testes avançados

Nesta fase, o caso de teste executa testes mais complexos em série para validar se o dispositivo segue as melhores práticas. Esses testes avançados estão disponíveis para seleção e podem ser excluídos se não forem necessários. Cada teste avançado tem o próprio valor de tempo limite com base no que o cenário exige.

RETURN PUBACK ON QoS 1 SUBSCRIPTION

> **Note**
>
> Selecione este cenário somente se o dispositivo for capaz de realizar assinaturas de QoS 1.

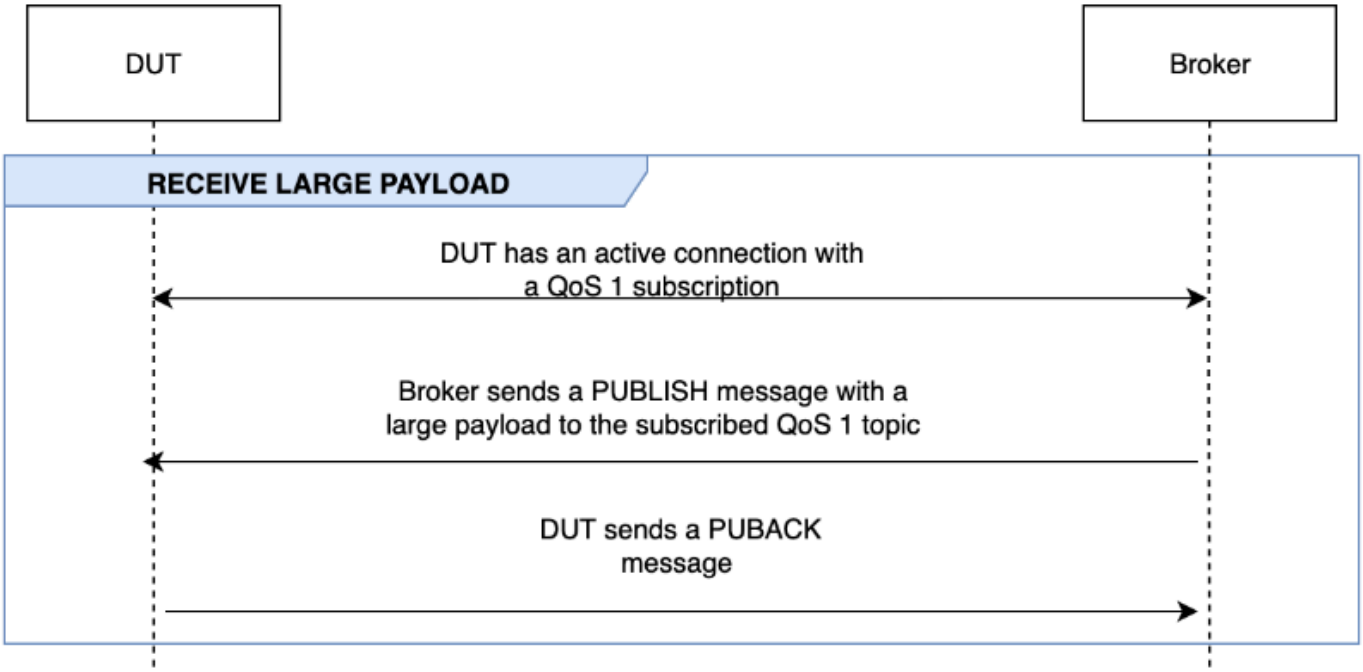
Esse cenário valida se, depois que o dispositivo assina um tópico e recebe uma mensagem `PUBLISH` do agente, ele retorna uma mensagem `PUBACK`.



RECEIVE LARGE PAYLOAD

> **Note**
>
> Selecione este cenário somente se o dispositivo for capaz de realizar assinaturas de QoS 1.

Esse cenário valida se o dispositivo responde com uma mensagem `PUBACK` após receber uma mensagem `PUBLISH` do agente para um tópico de QoS 1 com uma grande carga. O formato da carga esperada pode ser configurado usando a opção `LONG_PAYLOAD_FORMAT`.



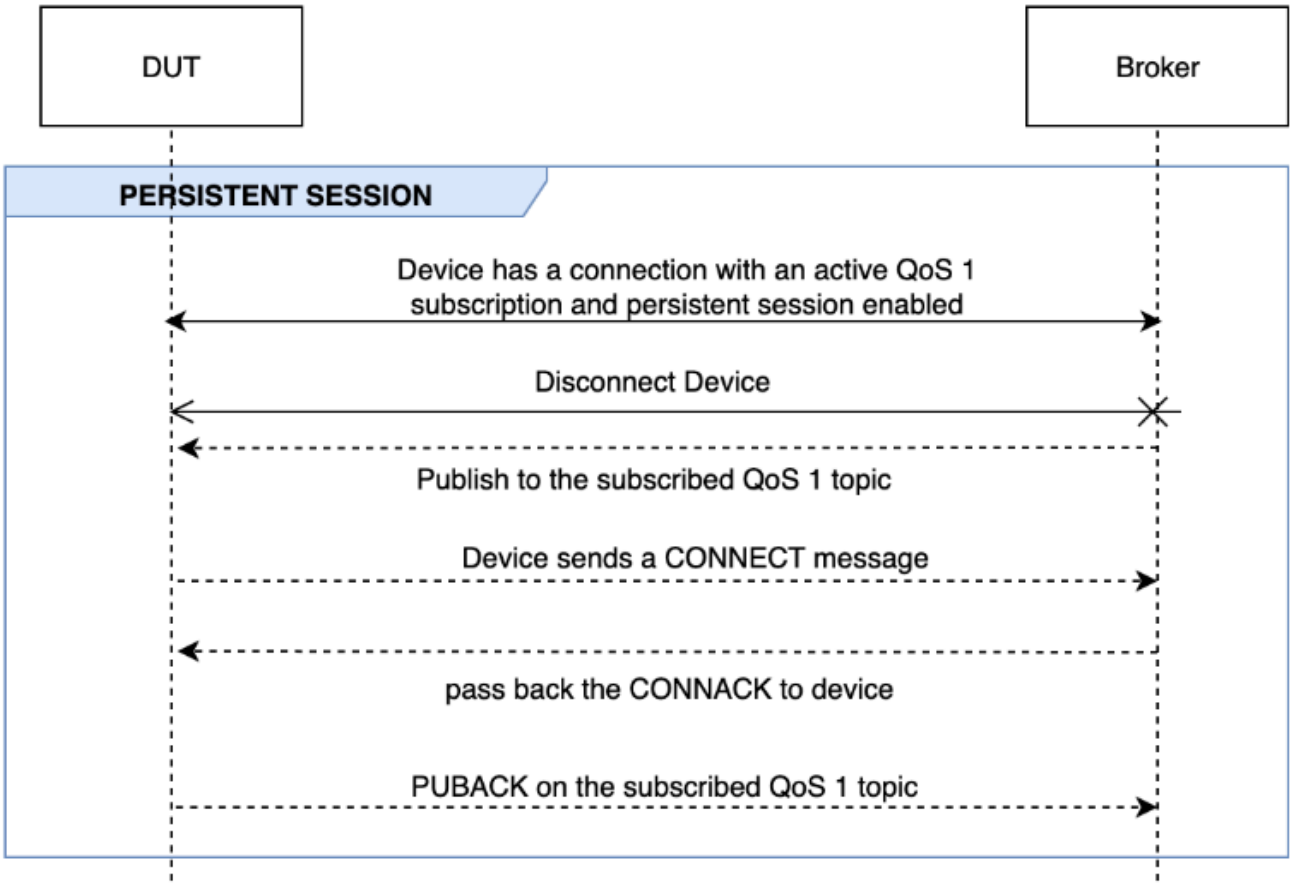
SESSÃO PERSISTENTE

> **Note**
>
> Selecione este cenário somente se o dispositivo for capaz de realizar assinaturas de QoS 1 e puder manter uma sessão persistente.

Esse cenário valida o comportamento do dispositivo na manutenção de sessões persistentes. O teste é validado quando as seguintes condições são atendidas:

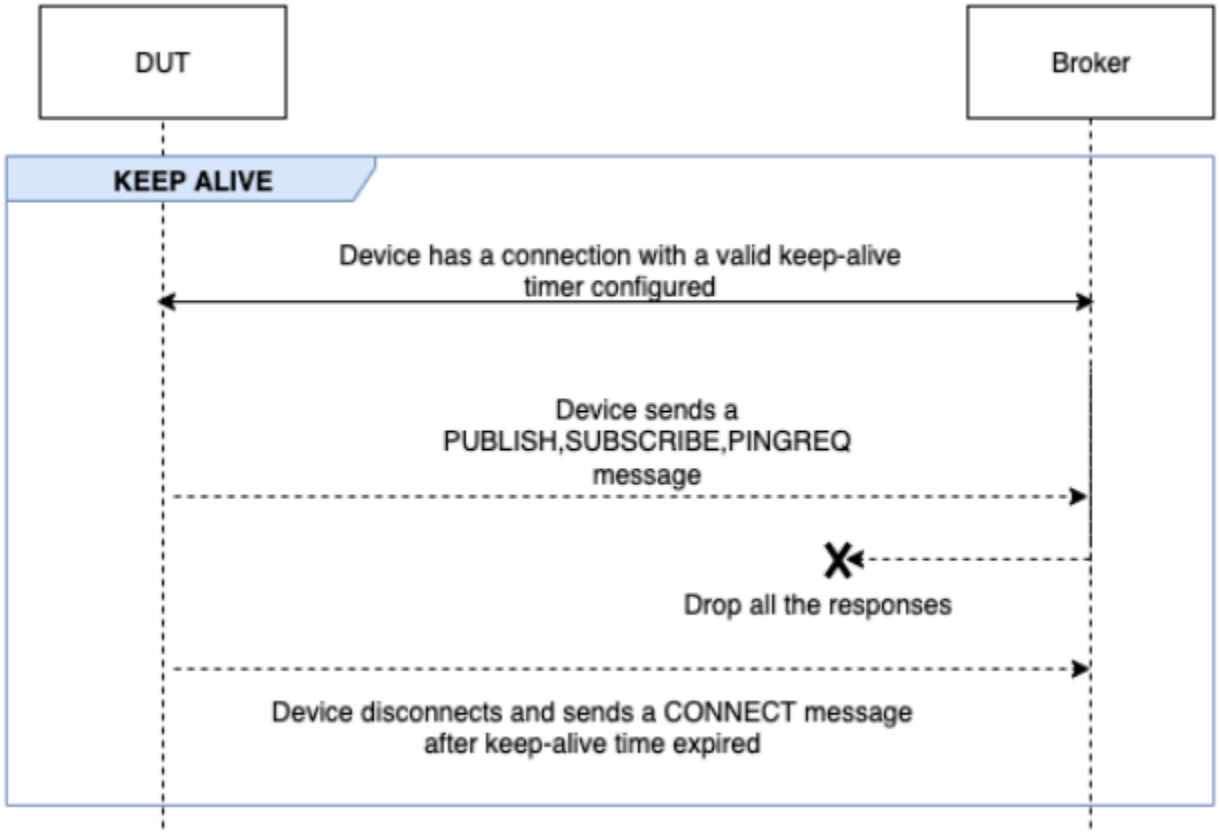
- O dispositivo se conecta ao agente com uma assinatura ativa de QoS 1 e sessões persistentes ativadas.
- O dispositivo se desconecta com sucesso do agente durante a sessão.
- O dispositivo se reconecta ao agente e retoma as assinaturas dos tópicos acionadores sem assinar de novo esses tópicos explicitamente.
- O dispositivo recebe com sucesso as mensagens armazenadas pelo agente para os tópicos assinados e funciona conforme o esperado.

Para obter mais informações sobre sessões AWS IoT persistentes, consulte Usando sessões persistentes do MQTT.



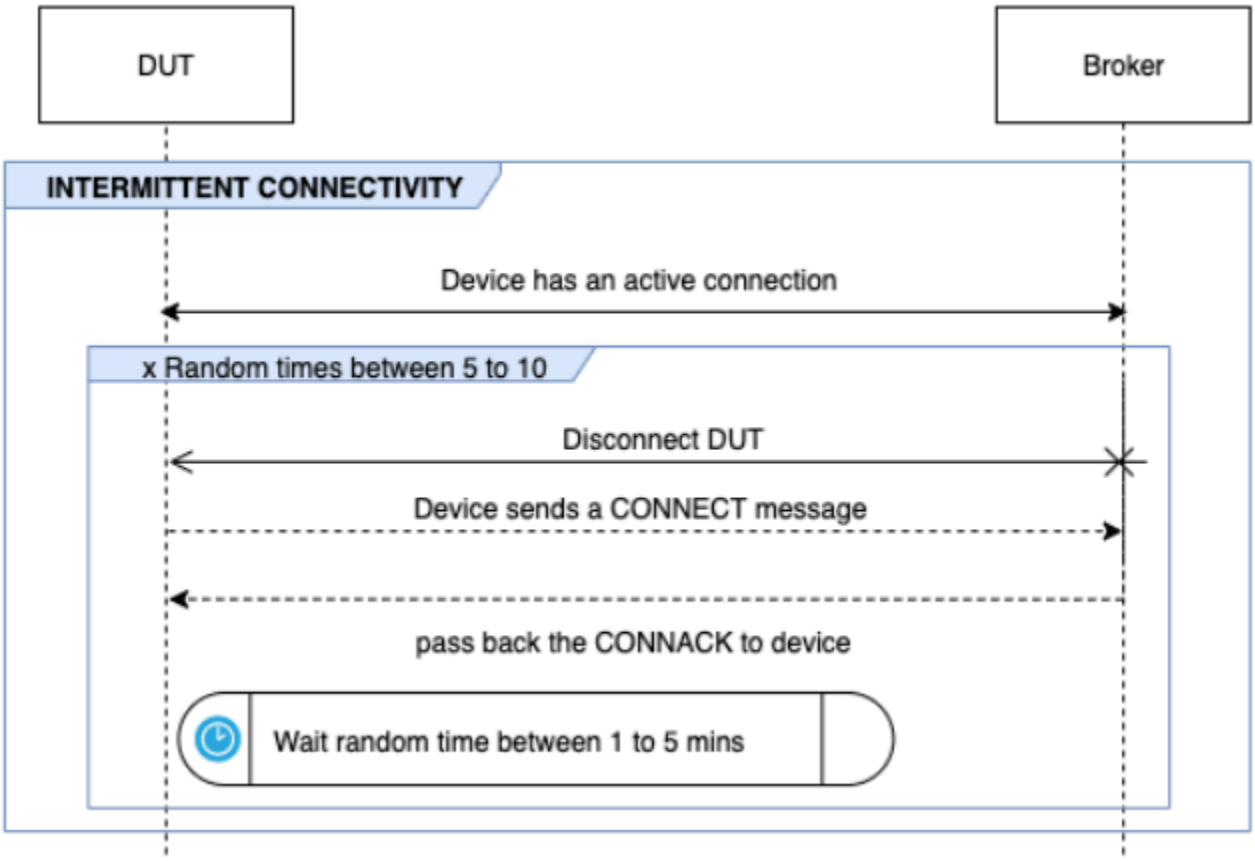
KEEP ALIVE

Este cenário valida se o dispositivo se desconecta com sucesso depois de não receber uma resposta de ping do agente. A conexão deve ter um cronômetro de keep-alive válido configurado. Como parte desse teste, o agente bloqueia todas as respostas enviadas para mensagens `PUBLISH`, `SUBSCRIBE` e `PINGREQ`. Também valida se o dispositivo em teste desconecta a conexão do MQTT.



CONECTIVIDADE INTERMITENTE

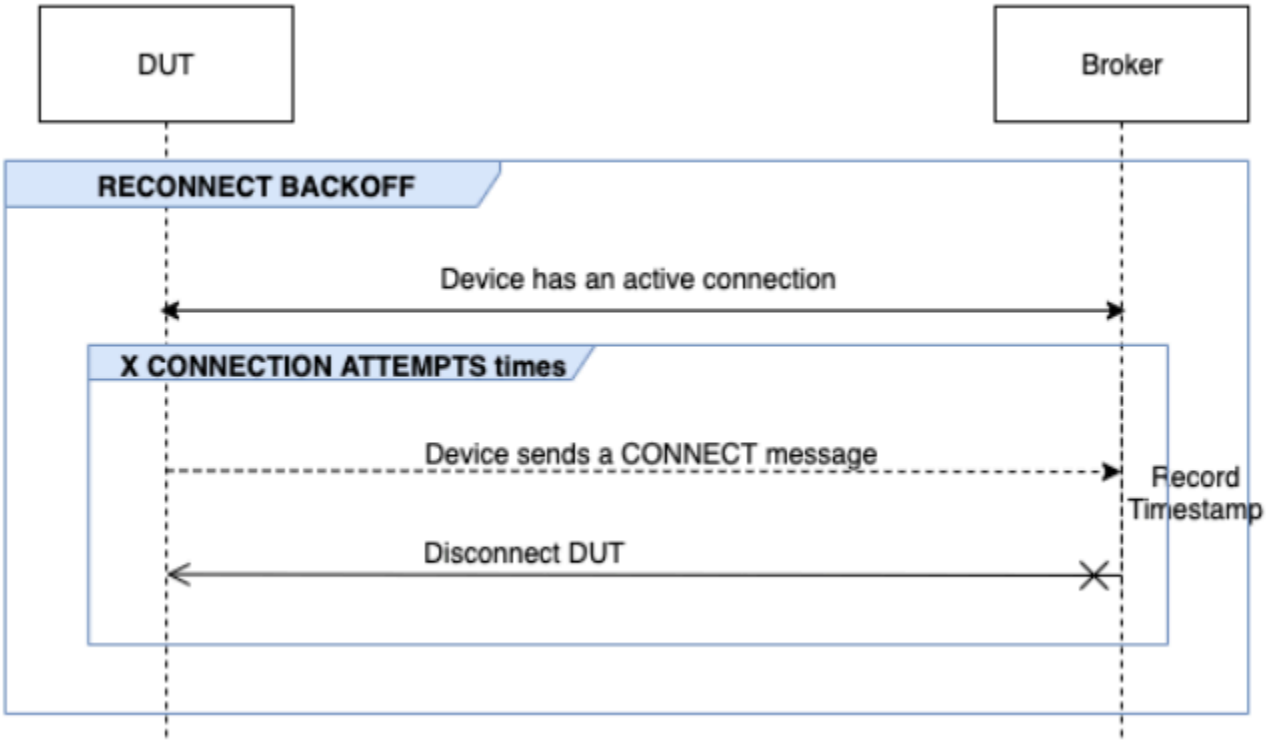
Este cenário valida se o dispositivo pode se conectar novamente ao agente depois que ele desconecta o dispositivo em intervalos aleatórios por um período de tempo aleatório.



RECONNECT BACKOFF

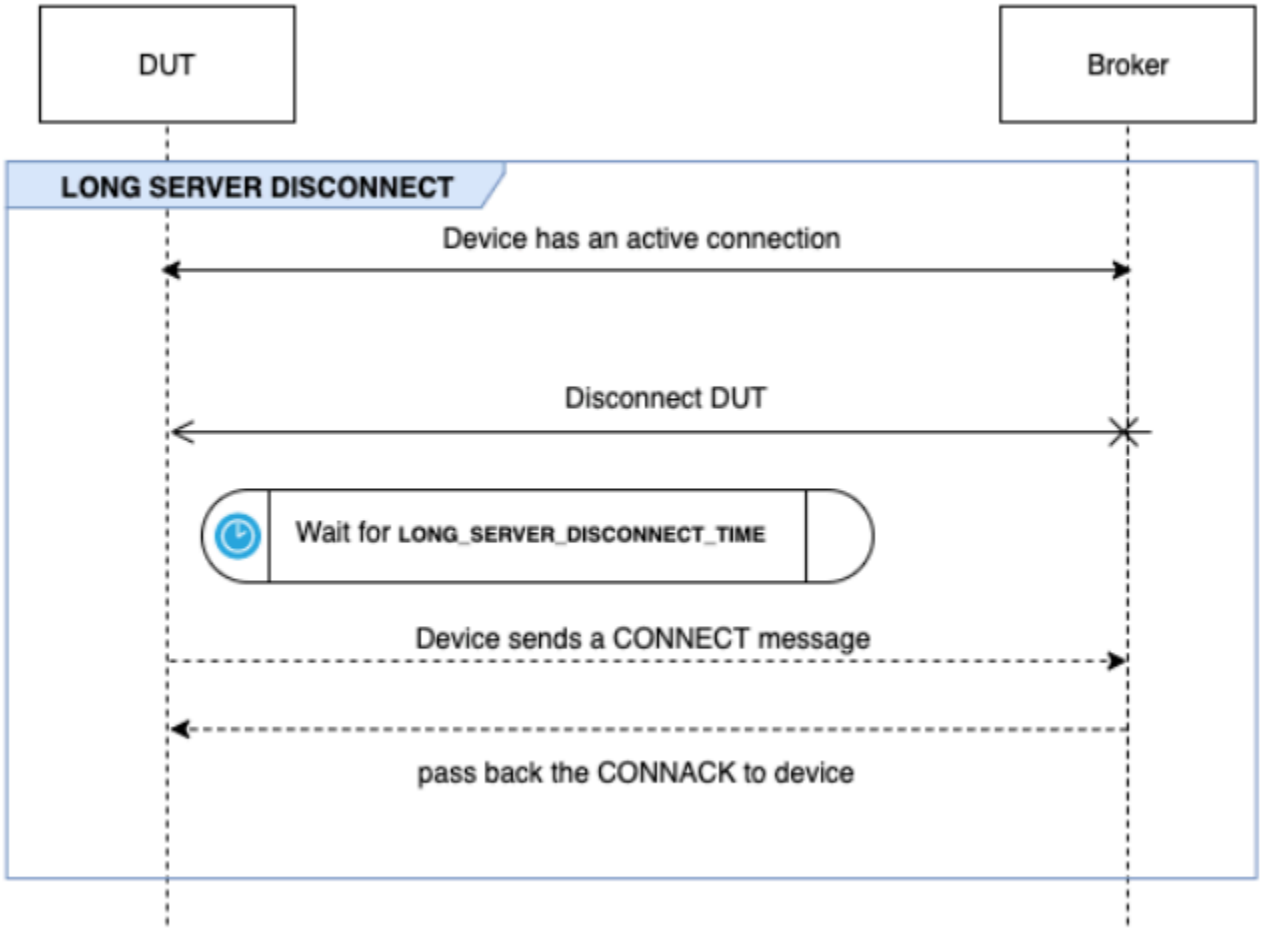
Este cenário valida se o dispositivo tem um mecanismo de recuo implementado quando o agente se desconecta dele várias vezes. O Device Advisor relata o tipo de recuo como exponencial, variação de sinal, linear ou constante. O número de tentativas de recuo é configurável usando a opção `BACKOFF_CONNECTION_ATTEMPTS`. O valor padrão é 5. O valor é configurável entre 5 e 10.

Para passar nesse teste, recomendamos a implementação do mecanismo Recuo exponencial e variação de sinal no dispositivo em teste.



DESCONEXÃO LONGA DO SERVIDOR

Este cenário valida se o dispositivo pode se reconectar com êxito após o agente desconectar o dispositivo por um longo período de tempo (até 120 minutos). A hora da desconexão do servidor pode ser configurada usando a opção `LONG_SERVER_DISCONNECT_TIME`. O valor padrão são 120 minutos. Esse valor é configurável de 30 a 120 minutos.



Tempo de execução adicional

O tempo de execução adicional é o tempo que o teste espera após concluir todos os testes acima e antes de encerrar o caso de teste. Os clientes usam esse período adicional para monitorar e registrar todas as comunicações entre o dispositivo e o agente. O tempo de execução adicional pode ser configurado usando a opção `ADDITIONAL_EXECUTION_TIME`. Por padrão, essa opção é definida como 0 minutos e pode ser de 0 a 120 minutos.

Opções de configuração de teste de longa duração do MQTT

Todas as opções de configuração fornecidas para o teste de longa duração do MQTT são opcionais. As seguintes opções estão disponíveis:

OPERATIONS

A lista de operações que o dispositivo executa, como `CONNECT`, `PUBLISH` e `SUBSCRIBE`. O caso de teste executa cenários com base nas operações especificadas. As operações que não são especificadas são consideradas válidas.

```
{
"OPERATIONS": ["PUBLISH", "SUBSCRIBE"]
//by default the test assumes device can CONNECT
}
```

SCENARIOS

Com base nas operações selecionadas, o caso de teste executa cenários para validar o comportamento do dispositivo. Há dois tipos de cenários:

- Os Cenários básicos são testes simples que validam se o dispositivo pode realizar as operações selecionadas acima como parte da configuração. Eles são pré-selecionados com base nas operações especificadas na configuração. Mais nenhuma entrada é necessária na configuração.
- Cenários avançados são cenários mais complexos que são executados em relação ao dispositivo para validar se ele segue as práticas recomendadas quando se depara com as condições do mundo real. Eles são opcionais e podem ser passados como uma matriz de cenários para a entrada de configuração do conjunto de testes.

```
{
    "SCENARIOS": [      // list of advanced scenarios
                "PUBACK_QOS_1",
                "RECEIVE_LARGE_PAYLOAD",
                "PERSISTENT_SESSION",
                "KEEP_ALIVE",
                "INTERMITTENT_CONNECTIVITY",
                "RECONNECT_BACK_OFF",
                "LONG_SERVER_DISCONNECT"
    ]
}
```

BASIC_TESTS_EXECUTION_TIME_OUT:

O tempo máximo que o caso de teste aguardará até que todos os testes básicos sejam concluídos. O valor padrão são 60 minutos. Esse valor é configurável de 30 a 120 minutos.

LONG_SERVER_DISCONNECT_TIME:

O tempo necessário para que o caso de teste se desconecte e reconecte o dispositivo durante o teste de Desconexão longa do servidor. O valor padrão são 60 minutos. Esse valor é configurável de 30 a 120 minutos.

ADDITIONAL_EXECUTION_TIME:

A configuração dessa opção fornece uma janela de tempo após a conclusão de todos os testes, para monitorar eventos entre o dispositivo e o agente. O valor padrão é 0 minutos. Esse valor é configurável de 0 a 120 minutos.

BACKOFF_CONNECTION_ATTEMPTS:

Essa opção configura o número de vezes que o dispositivo é desconectado pelo caso de teste. Isso é usado pelo teste Recuo de reconexão. O valor padrão é 5 tentativas. Esse valor é configurável de 5 a 10.

LONG_PAYLOAD_FORMAT:

O formato da carga da mensagem que o dispositivo espera quando o caso de teste é publicado em um tópico de QoS 1 assinado pelo dispositivo.

Definição do caso de teste da API:

```
{
"tests":[
   {
      "name":"my_mqtt_long_duration_test",
      "configuration": {
         // optional
         "OPERATIONS": ["PUBLISH", "SUBSCRIBE"],
         "SCENARIOS": [
            "LONG_SERVER_DISCONNECT",
            "RECONNECT_BACK_OFF",
            "KEEP_ALIVE",
            "RECEIVE_LARGE_PAYLOAD",
            "INTERMITTENT_CONNECTIVITY",
            "PERSISTENT_SESSION",
         ],
         "BASIC_TESTS_EXECUTION_TIMEOUT": 60, // in minutes (60 minutes by default)
         "LONG_SERVER_DISCONNECT_TIME": 60,   // in minutes (120 minutes by default)
         "ADDITIONAL_EXECUTION_TIME": 60,     // in minutes (0 minutes by default)
```

```
         "BACKOFF_CONNECTION_ATTEMPTS": "5",
         "LONG_PAYLOAD_FORMAT":"{"message":"${payload}"}"
      },
      "test":{
         "id":"MQTT_Long_Duration",
         "version":"0.0.0"
      }
   }
 ]
}
```

Log de resumo do caso de teste de longa duração do MQTT

O caso de teste de longa duração do MQTT é executado por mais tempo do que os casos de teste normais. É fornecido um log de resumo separado, que lista eventos importantes, como conexões de dispositivos, publicação e assinatura durante a execução. Os detalhes incluem o que foi testado, o que não foi testado e o que falhou. No final do log, o teste inclui um resumo de todos os eventos que aconteceram durante a execução do caso de teste. Isso inclui:

- O temporizador Keep Alive está configurado no dispositivo.
- Sinalizador de sessão persistente configurado no dispositivo.
- O número de conexões do dispositivo durante a execução do teste.
- O tipo de recuo de reconexão do dispositivo, se validado para o teste de recuo de reconexão.
- Os tópicos nos quais o dispositivo publicou durante a execução do caso de teste.
- Os tópicos que o dispositivo assinou durante a execução do caso de teste.

AWS IoT Core Localização do dispositivo

Antes de usar o recurso de localização do AWS IoT Core dispositivo, revise os Termos e Condições desse recurso. Observe que AWS pode transmitir seus parâmetros de solicitação de pesquisa de geolocalização, como os dados de localização usados para realizar pesquisas e outras informações para o provedor de dados terceirizado escolhido, que podem estar fora do Região da AWS que você está usando atualmente. O provedor terceirizado e o solucionador a serem usados são escolhidos com base na carga útil de entrada recebida. Para obter mais informações, consulte os Termos de Serviço da AWS.

Use a localização AWS IoT Core do dispositivo para testar a localização dos seus dispositivos de IoT usando solucionadores de terceiros. Os solucionadores são algoritmos fornecidos por fornecedores terceiros que resolvem dados de medição e estimam a localização do dispositivo. Ao identificar a localização dos dispositivos, você pode rastreá-los e depurá-los em campo para solucionar quaisquer problemas.

Os dados de medição coletados de várias fontes são resolvidos, e as informações de localização geográfica são relatadas como uma carga útil do GeoJSON. O formato GeoJSON é um formato usado para codificar estruturas de dados geográficos. A carga contém as coordenadas de latitude e longitude da localização do seu dispositivo, que são baseadas no sistema de coordenadas do Sistema Geodésico Mundial (). WGS84

Tópicos

- Tipos de medição e solucionadores
- Como funciona a localização do AWS IoT Core dispositivo
- Como usar a localização AWS IoT Core do dispositivo
- Como resolver a localização dos dispositivos de IoT
- Resolvendo a localização do dispositivo usando a localização do AWS IoT Core dispositivo (tópicos do MQTT)
- Solucionadores de localização e carga do dispositivo

Tipos de medição e solucionadores

AWS IoT Core A Device Location faz parceria com fornecedores terceirizados para resolver os dados de medição e fornecer uma localização estimada do dispositivo. A tabela a seguir mostra os tipos de medição e os solucionadores de localização de terceiros, além de informações sobre os dispositivos compatíveis. Para obter informações sobre dispositivos LoRa WAN e como configurar a localização do dispositivo para eles, consulte Configurando a posição dos recursos de LoRa WAN.

> **Note**
>
> Dispositivos gerais de IoT e dispositivos Sidewalk podem usar os tópicos do MQTT de localização do dispositivo para obter as informações de localização. Para tipos de medição de endereço Wi-Fi, celular e IP, se os dispositivos publicarem os dados de medição nos tópicos reservados no formato GeoJSON definido AWS IoT Core , a localização do dispositivo poderá resolver a localização do dispositivo. Para o tipo de medição GNSS, o dispositivo deve ter o LR11xx chip para escanear os dados de medição para obter as informações de localização resolvidas usando o solucionador GNSS. Para obter informações sobre como obter informações de localização para dispositivos LoRa WAN, consulte Configurando a posição dos recursos de LoRa WAN na documentação sobre AWS IoT conexões sem fio.

Tipos de medição e solucionadores

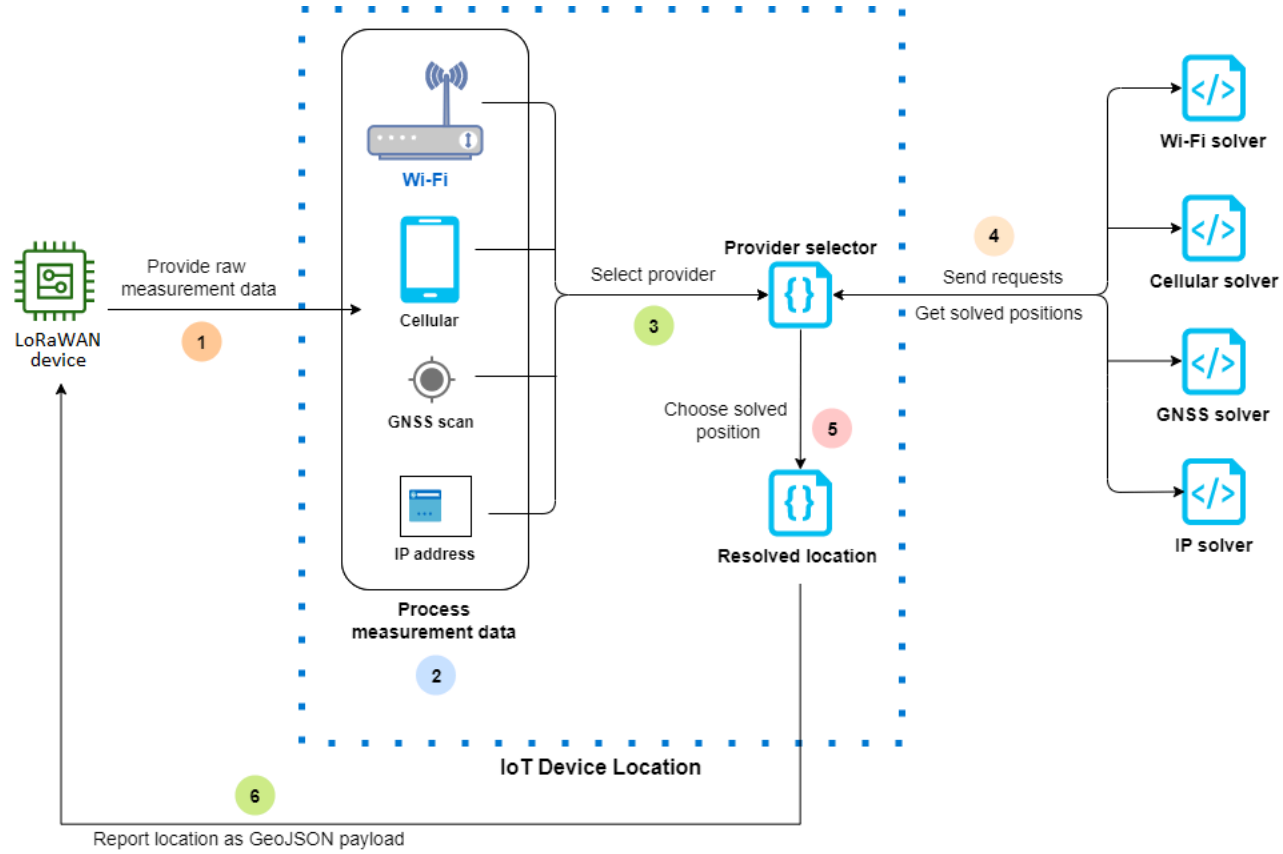
Tipo de medição	Números de terceiros	Dispositivos compatíveis
Pontos de acesso Wi-Fi	Solucionador baseado em Wi-Fi	Dispositivos gerais de IoT, LoRa WAN e dispositivos Amazon Sidewalk
Torres de rádio celular: dados GSM, LTE, CDMA, SCDMA, WCMDA e TD-SCDMA	Solucionador baseado em celular	Dispositivos gerais de IoT, LoRa WAN e dispositivos Amazon Sidewalk
IP address (endereço de IP)	Solucionador de pesquisa reversa de IP	Qualquer dispositivo de IoT que se conecte por TCP/IP

Tipo de medição	Números de terceiros	Dispositivos compatíveis
Dados de varredura de GNSS (mensagens NAV)	Solucionador GNSS	Dispositivos gerais de IoT, LoRa WAN e dispositivos Amazon Sidewalk
Bluetooth Low Energy (BLE)	Solucionador baseado em BLE	Dispositivos Amazon Sidewalk

Para obter mais informações sobre os solucionadores de localização e exemplos que mostram a carga do dispositivo para os vários tipos de medição, consulte Solucionadores de localização e carga do dispositivo.

Como funciona a localização do AWS IoT Core dispositivo

O diagrama a seguir mostra como a localização do AWS IoT Core dispositivo coleta dados de medição e resolve as informações de localização dos seus dispositivos.



As etapas a seguir mostram como a localização AWS IoT Core do dispositivo funciona.

1. Receba dados de medição

   Os dados brutos de medição relacionados ao local do dispositivo são enviados primeiro do dispositivo. Os dados de medição são especificados como uma carga JSON.

2. Processar dados de medição

   Os dados de medição são processados e o AWS IoT Core Device Location escolhe os dados de medição a serem usados, que podem ser Wi-Fi, celular, digitalização GNSS ou informações de endereço IP.

3. Escolha o solucionador

   O solucionador terceiro é escolhido com base nos dados de medição. Por exemplo, se os dados de medição contiverem informações de Wi-Fi e endereço IP, ele escolherá o solucionador Wi-Fi e o solucionador de pesquisa reversa de IP.

4. Obtenha a localização resolvida

   Uma solicitação de API é enviada aos provedores de solução solicitando a resolução do local. AWS IoT Core A localização do dispositivo então obtém as informações estimadas de geolocalização dos solucionadores.

5. Escolha o local resolvido

   As informações de localização resolvidas e sua precisão são comparadas, e a localização do AWS IoT Core dispositivo escolhe os resultados da geolocalização com a maior precisão.

6. Informações de local de saída

   As informações de localização geográfica são enviadas para você como uma carga útil GeoJSON. A carga contém as coordenadas WGS84 geográficas, as informações de precisão, os níveis de confiança e a data e hora em que a localização resolvida foi obtida.

Como usar a localização AWS IoT Core do dispositivo

As etapas a seguir mostram como usar a localização AWS IoT Core do dispositivo.

1. Forneça dados de medição

   Especifique os dados brutos de medição relacionados à localização do dispositivo como uma carga JSON. Para recuperar os dados de medição da carga útil, acesse os registros do dispositivo ou use os registros e copie CloudWatch as informações dos dados da carga útil. A carga JSON deve conter um ou mais tipos de medição de dados. Para exemplos que mostram o formato da carga de vários solucionadores, consulte Solucionadores de localização e carga do dispositivo.

2. Resolver informações de localização

   Usando a página Localização do dispositivo no AWS IoT console ou a operação da GetPositionEstimateAPI, transmita os dados de medição da carga útil e resolva a localização do dispositivo. AWS IoT Core Em seguida, a localização do dispositivo escolhe o solucionador com a maior precisão e relata a localização do dispositivo. Para obter mais informações, consulte Como resolver a localização dos dispositivos de IoT.

3. Copiar informações de localização

   Verifique as informações de geolocalização que foram resolvidas pelo AWS IoT Core Device Location e relatadas como uma carga GeoJSON. Você pode copiar a carga útil para uso com seus aplicativos e outros AWS service (Serviço da AWS). Por exemplo, você pode enviar seus dados de localização geográfica para o Amazon Location Service usando a ação de Local AWS IoT regra.

Os tópicos a seguir mostram como usar a localização AWS IoT Core do dispositivo e exemplos de carga útil de localização do dispositivo.

- Como resolver a localização dos dispositivos de IoT
- Solucionadores de localização e carga do dispositivo

Como resolver a localização dos dispositivos de IoT

Use a localização AWS IoT Core do dispositivo para decodificar os dados de medição de seus dispositivos e resolver a localização do dispositivo usando solucionadores de terceiros. A localização resolvida é gerada como uma carga GeoJSON com as coordenadas geográficas e informações de precisão. Você pode resolver a localização do seu dispositivo no AWS IoT console, na API AWS IoT sem fio ou AWS CLI.

Tópicos

- Como resolver o local do dispositivo (console)
- Como resolver o local do dispositivo (API)
- Como solucionar problemas com erros ao resolver a localização

Como resolver o local do dispositivo (console)

Para resolver o local do dispositivo (console)

1. Acesse a página Localização do dispositivo no AWS IoT console.
2. Obtenha os dados de medição da carga útil dos registros do seu dispositivo ou dos CloudWatch registros e insira-os na seção Resolver posição via carga útil.

   O código a seguir mostra um exemplo de carga JSON. A carga contém dados de medição de celular e Wi-Fi. Se a carga contiver tipos adicionais de dados de medição, o solucionador com a melhor precisão será usado. Para ter mais informações e exemplos de carga, consulte the section called “Solucionadores de localização e carga do dispositivo”.

   > **Note**
   >
   > A carga JSON deve conter pelo menos um tipo de dados de medição.

```
{
    "Timestamp": 1664313161,
    "Ip":{
        "IpAddress": "54.240.198.35"
    },
    "WiFiAccessPoints": [{
        "MacAddress": "A0:EC:F9:1E:32:C1",
        "Rss": -77
    }],
    "CellTowers": {
        "Gsm": [{
            "Mcc": 262,
            "Mnc": 1,
            "Lac": 5126,
            "GeranCid": 16504,
            "GsmLocalId": {
```

```
                "Bsic": 6,
                "Bcch": 82
            },
            "GsmTimingAdvance": 1,
            "RxLevel": -110,
            "GsmNmr": [{
                "Bsic": 7,
                "Bcch": 85,
                "RxLevel": -100,
                "GlobalIdentity": {
                    "Lac": 1,
                    "GeranCid": 1
                }
            }]
        }],
        "Wcdma": [{
            "Mcc": 262,
            "Mnc": 7,
            "Lac": 65535,
            "UtranCid": 14674663,
            "WcdmaNmr": [{
                    "Uarfcndl": 10786,
                    "UtranCid": 14674663,
                    "Psc": 149
                },
                {
                    "Uarfcndl": 10762,
                    "UtranCid": 14674663,
                    "Psc": 211
                }
            ]
        }],
        "Lte": [{
            "Mcc": 262,
            "Mnc": 2,
            "EutranCid": 2898945,
            "Rsrp": -50,
            "Rsrq": -5,
            "LteNmr": [{
                    "Earfcn": 6300,
                    "Pci": 237,
                    "Rsrp": -60,
                    "Rsrq": -6,
                    "EutranCid": 2898945
```

```
                },
                {
                    "Earfcn": 6300,
                    "Pci": 442,
                    "Rsrp": -70,
                    "Rsrq": -7,
                    "EutranCid": 2898945
                }
            ]
        }]
    }
}
```

3. Para resolver as informações de local, escolha Resolver.

   As informações de localização são do tipo blob e retornadas como uma carga que usa o formato GeoJSON, que é um formato usado para codificar estruturas de dados geográficos. A carga contém:

   - As coordenadas WGS84 geográficas, que incluem as informações de latitude e longitude. Também pode incluir informações de altitude.
   - O tipo de informação de localização relatada, como Ponto. Um tipo de localização de ponto representa a localização como WGS84 latitude e longitude, codificada como um ponto GeoJSON.
   - As informações de precisão horizontal e vertical, que indicam a diferença, em metros, entre as informações de local estimadas pelos solucionadores e o local real do dispositivo.
   - O nível de confiança, que indica a incerteza na resposta da estimativa de localização. O valor padrão é 0,68, o que indica uma probabilidade de 68% de que o local real do dispositivo esteja dentro do raio de incerteza do local estimado.
   - A cidade, o estado, o país e o código postal em que o dispositivo está localizado. Essas informações serão relatadas somente quando o solucionador de pesquisa reversa de IP for usado.
   - As informações de registro de data e hora, que correspondem à data e à hora em que o local foi resolvido. Ele usa o formato de registro de data/hora Unix.

   O código a seguir mostra um exemplo de carga GeoJSON retornada pela resolução da localização.

> **Note**
>
> Se AWS IoT Core o Device Location reportar erros ao tentar resolver o local, você poderá solucionar os erros e resolver o local. Para obter mais informações, consulte Como solucionar problemas com erros ao resolver a localização.


```
{
    "coordinates": [
        13.376076698303223,
        52.51823043823242
    ],
    "type": "Point",
    "properties": {
        "verticalAccuracy": 45,
        "verticalConfidenceLevel": 0.68,
        "horizontalAccuracy": 303,
        "horizontalConfidenceLevel": 0.68,
        "country": "USA",
        "state": "CA",
        "city": "Sunnyvalue",
        "postalCode": "91234",
        "timestamp": "2022-11-18T12:23:58.189Z"
    }
}
```


4. Vá para a seção Localização do recurso e verifique as informações de geolocalização relatadas pela Localização AWS IoT Core do dispositivo. Você pode copiar a carga útil para uso com outros aplicativos e AWS service (Serviço da AWS) s. Por exemplo, você pode usar a Local para enviar os dados de localização geográfica para o Amazon Location Service.

Como resolver o local do dispositivo (API)

Para resolver a localização do dispositivo usando a API AWS IoT sem fio, use a operação da GetPositionEstimateAPI ou o comando da get-position-estimateCLI. Especifique os dados de medição da carga como entrada e execute a operação da API para resolver o local do dispositivo.

> **Note**
>
> A operação da API `GetPositionEstimate` não armazena nenhuma informação de dispositivo ou estado e não pode ser usada para recuperar dados históricos de localização. Ele executa uma operação única que resolve os dados de medição e produz a localização estimada. Para recuperar as informações de localização, você deve especificar as informações da carga sempre que realizar essa operação de API.

O comando a seguir mostra um exemplo de como resolver a localização usando essa operação de API.

> **Note**
>
> Ao executar o comando da CLI `get-position-estimate`, você deve especificar o arquivo JSON de saída como a primeira entrada. Esse arquivo JSON armazenará as informações de localização estimadas obtidas como resposta da CLI no formato GeoJSON. Por exemplo, o comando a seguir armazena as informações de localização no *`locationout.json`* arquivo.

```
aws iotwireless get-position-estimate locationout.json \
    --ip IpAddress=""54.240.198.35"" \
    --wi-fi-access-points \
        MacAddress="A0:EC:F9:1E:32:C1",Rss=-75 \
        MacAddress="A0:EC:F9:15:72:5E",Rss=-67
```

Este exemplo inclui pontos de acesso Wi-Fi e endereço IP como tipos de medição. AWS IoT Core A localização do dispositivo escolhe entre o solucionador Wi-Fi e o solucionador de pesquisa reversa de IP e seleciona o solucionador com maior precisão.

A localização resolvida é retornada como uma carga que usa o formato GeoJSON, que é um formato usado para codificar estruturas de dados geográficos. Em seguida, ele é armazenado no *`locationout.json`* arquivo. A carga contém as coordenadas de WGS84 latitude e longitude, informações de precisão e nível de confiança, o tipo de dados de localização e a data e hora em que a localização foi resolvida.

```
{
    "coordinates": [
        13.37704086303711,
```

```
        52.51865005493164
    ],
    "type": "Point",
    "properties": {
        "verticalAccuracy": 707,
        "verticalConfidenceLevel": 0.68,
        "horizontalAccuracy": 389,
        "horizontalConfidenceLevel": 0.68,
        "country": "USA",
        "state": "CA",
        "city": "Sunnyvalue",
        "postalCode": "91234",
        "timestamp": "2022-11-18T14:03:57.391Z"
    }
}
```

Como solucionar problemas com erros ao resolver a localização

Ao tentar resolver o local, você pode ver qualquer um dos códigos de erro a seguir. AWS IoT Core A localização do dispositivo pode gerar um erro ao usar a operação da `GetPositionEstimate` API ou então se referir ao número da linha correspondente ao erro no AWS IoT console.

- Erro 400

  Esse erro indica que o formato da carga útil do dispositivo JSON não pode ser validado pela AWS IoT Core localização do dispositivo. O erro pode ocorrer porque:

  - Os dados de medição do JSON estão formatados incorretamente.
  - A carga contém somente as informações do registro de data/hora.
  - Os parâmetros dos dados de medição, como o endereço IP, não são válidos.

  Para resolver esse erro, verifique se o JSON está formatado corretamente e contém dados de um ou mais tipos de medição como entrada. Se o endereço IP for inválido, para obter informações sobre como você pode fornecer um endereço IP válido para resolver o erro, consulte Solucionador de pesquisa reversa de IP.

- Erro 403

  Esse erro indica que você não tem as permissões para realizar a operação da API ou usar o AWS IoT console para recuperar a localização do dispositivo. Para resolver esse erro, verifique se você tem as permissões necessárias para realizar essa ação. Esse erro pode ocorrer se sua Console de gerenciamento da AWS sessão ou seu token de AWS CLI sessão tiverem expirado. Para resolver

esse erro, atualize o token da sessão para usar o. ou saia do Console de gerenciamento da AWS e AWS CLI, em seguida, faça login usando suas credenciais.

- Erro 404

  Esse erro indica que nenhuma informação de localização foi encontrada ou resolvida pela Localização AWS IoT Core do dispositivo. O erro pode ocorrer devido a casos como dados insuficientes na entrada de dados de medição. Por exemplo:

  - As informações do endereço MAC ou da torre celular não são suficientes.
  - O endereço IP não está disponível para pesquisar e recuperar a localização.
  - A carga do GNSS não é suficiente.

  Para resolver o erro nesses casos, verifique se os dados de medição contêm informações suficientes necessárias para resolver o local do dispositivo.

- Erro 500

  Esse erro indica que ocorreu uma exceção interna do servidor quando Local do dispositivo AWS IoT Core tentou resolver o local. Para tentar corrigir esse erro, atualize a sessão e tente enviar novamente os dados de medição a serem resolvidos.

Resolvendo a localização do dispositivo usando a localização do AWS IoT Core dispositivo (tópicos do MQTT)

Você pode usar tópicos reservados do MQTT para obter as informações de localização mais recentes para seus dispositivos com o recurso Localização do AWS IoT Core dispositivo.

Formato dos tópicos MQTT de local do dispositivo

Os tópicos reservados para a localização do AWS IoT Core dispositivo usam o seguinte prefixo:

`$aws/device_location/{customer_device_id}/`

Para criar um tópico completo, primeiro substitua *`customer_device_id`* pelo ID exclusivo que você usa para identificar o dispositivo. Recomendamos que você especifique o`WirelessDeviceId`, como para dispositivos LoRa WAN e Sidewalk, e*`thingName`*, se o seu dispositivo está registrado como AWS IoT algo. Em seguida, você acrescenta o tópico ao esboço do tópico, como `get_position_estimate` ou `get_position_estimate/accepted`, conforme mostrado na seção a seguir.

> **Note**
>
> O *{customer_device_id}* contém somente letras minúsculas, números e traços. Ao assinar tópicos de local do dispositivo, você só pode usar o sinal de adição (+) como um caractere curinga. Por exemplo, você pode usar o curinga + para o *{customer_device_id}* para obter as informações de localização dos dispositivos. Quando você assina o tópico `$aws/device_location/+/get_position_estimate/accepted`, uma mensagem é publicada com as informações de localização dos dispositivos que correspondem a qualquer ID do dispositivo, caso tenha sido resolvida com sucesso.

A seguir estão os tópicos reservados usados para interagir com a localização AWS IoT Core do dispositivo.

Tópicos MQTT de local do dispositivo

Tópico	Operações permitidas	Description
$ aws/device_location/ / get_position_estimate *customer_device_id*	Publicar	Um dispositivo publica este tópico para que os dados brutos de medição digitalizados sejam resolvidos pela localização AWS IoT Core do dispositivo.
$ aws/device_location/ / get_position_estimate/ accepted *customer_ device_id*	Assinar	AWS IoT Core A localização do dispositivo publica as informações de localização neste tópico quando resolve com êxito a localizaç ão do dispositivo.
$ aws/device_location/ / get_position_estimate/ rejeitado *customer_ device_id*	Assinar	AWS IoT Core A localização do dispositivo publica as informações de erro neste tópico quando não consegue resolver a localização do dispositivo.

Política para tópicos MQTT de local do dispositivo

Para receber mensagens de tópicos de localização do dispositivo, seu dispositivo deve usar uma política que permita que ele se conecte ao gateway do AWS IoT dispositivo e assine os tópicos do MQTT.

Veja a seguir um exemplo da política necessária para receber mensagens para os vários tópicos.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/
device-123/get_position_estimate"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/
device-123/get_position_estimate/accepted",
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/
device-123/get_position_estimate/rejected"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
device_location/device-123/get_position_estimate/accepted",
```

```
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
device_location/device-123/get_position_estimate/rejected"
            ]
        }
    ]
}
```

Tópicos de local do dispositivo e carga

Veja a seguir os tópicos de localização do AWS IoT Core dispositivo, o formato da carga útil da mensagem e um exemplo de política para cada tópico.

Tópicos

- /get_position_estimate
- /get_position_estimate/accepted
- /get_position_estimate/rejected

/get_position_estimate

Publique uma mensagem neste tópico para que os dados brutos de medição do dispositivo sejam resolvidos pela localização AWS IoT Core do dispositivo.

```
$aws/device_location/customer_device_id/get_position_estimate
```

AWS IoT Core A localização do dispositivo responde publicando em um /get_position_estimate/accepted ou/get_position_estimate/rejected.

> **Note**
>
> A mensagem publicada neste tópico deve ser uma carga JSON válida. Se a mensagem de entrada não estiver no formato JSON válido, você não receberá nenhuma resposta. Para obter mais informações, consulte Carga da mensagem.

Carga útil da mensagem

O formato da carga útil da mensagem segue uma estrutura semelhante ao corpo da solicitação de operação da API AWS IoT sem fio, `GetPositionEstimate`. Ela contém:

- Uma string `Timestamp` opcional, que corresponde à data e hora em que a localização foi resolvida. A string `Timestamp` pode ter um comprimento mínimo de 1 e um máximo de 10.
- Uma string `MessageId` opcional, que pode ser usada para mapear a solicitação para a resposta. Se você especificar essa string, a mensagem publicada nos tópicos `get_position_estimate/accepted` ou `get_position_estimate/rejected` conterá esse `MessageId`. A string `MessageID` pode ter um comprimento mínimo de 1 e um máximo de 256.
- Os dados de medição do dispositivo que contém um ou mais dos seguintes tipos de medição:
  - `WiFiAccessPoint`
  - `CellTowers`
  - `IpAddress`
  - `Gnss`

Veja a seguir um exemplo de carga de mensagem.

```
{
    "Timestamp": "1664313161",
    "MessageId": "ABCD1",
    "WiFiAccessPoints":  [
        {
            "MacAddress": "A0:EC:F9:1E:32:C1",
            "Rss": -66
        }
    ],
    "Ip":{
        "IpAddress": "54.192.168.0"
    },
    "Gnss":{
      "Payload":"8295A614A2029517F4F77C0A7823B161A6FC57E25183D96535E3689783F6CA48",
      "CaptureTime":1354393948
    }
}
```

Exemplo de política

Veja a seguir um exemplo da política necessária:

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/
device-123/get_position_estimate"
            ]
        }
    ]
}
```

/get_position_estimate/accepted

AWS IoT Core A localização do dispositivo publica uma resposta a esse tópico ao retornar as informações de localização resolvidas do seu dispositivo. As informações de localização são retornadas no formato GeoJSON.

```
$aws/device_location/customer_device_id/get_position_estimate/accepted
```

Veja a seguir a carga da mensagem e um exemplo de política.

Carga útil da mensagem

O exemplo a seguir é da carga da mensagem no formato GeoJSON. Se você especificou a `MessageId` em seus dados brutos de medição e a localização do AWS IoT Core dispositivo resolveu as informações de localização com êxito, a carga da mensagem retornará as mesmas `MessageId` informações.

```
{
    "coordinates": [
        13.37704086303711,
        52.51865005493164
    ],
    "type": "Point",
    "properties": {
        "verticalAccuracy": 707,
        "verticalConfidenceLevel": 0.68,
```

```
        "horizontalAccuracy": 389,
        "horizontalConfidenceLevel": 0.68,
        "country": "USA",
        "state": "CA",
        "city": "Sunnyvalue",
        "postalCode": "91234",
        "timestamp": "2022-11-18T14:03:57.391Z",
        "messageId": "ABCD1"
    }
}
```

Exemplo de política

Veja a seguir um exemplo da política necessária:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/
device_location/device-123/get_position_estimate/accepted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/
device-123/get_position_estimate/accepted"
            ]
        }
    ]
}
```

/get_position_estimate/rejected

AWS IoT Core A localização do dispositivo publica uma resposta de erro para esse tópico quando não consegue resolver a localização do dispositivo.

```
$aws/device_location/customer_device_id/get_position_estimate/rejected
```

Veja a seguir a carga da mensagem e um exemplo de política. Para obter mais informações sobre os erros, consulte Como solucionar problemas com erros ao resolver a localização.

Carga útil da mensagem

Veja a seguir um exemplo da carga útil da mensagem que fornece o código de erro e a mensagem, indicando por que a localização do AWS IoT Core dispositivo falhou ao resolver as informações de localização. Se você especificou um `MessageId` ao fornecer seus dados de medição brutos e a localização do AWS IoT Core dispositivo falhar em resolver as informações de localização, as mesmas `MessageId` informações serão retornadas na carga útil da mensagem.

```
{
    "errorCode": 500,
    "errorMessage":"Internal server error",
    "messageId": "ABCD1"
}
```

Exemplo de política

Veja a seguir um exemplo da política necessária:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/device_location/
device-123/get_position_estimate/rejected"
      ]
```

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/device_location/device-123/
get_position_estimate/rejected"
      ]
    }
  ]
}
```

Solucionadores de localização e carga do dispositivo

Os solucionadores de localização são algoritmos que podem ser usados para resolver a localização de seus dispositivos de IoT. AWS IoT Core A localização do dispositivo é compatível com os seguintes solucionadores de localização. Você verá exemplos do formato de carga JSON para esses tipos de medição, os dispositivos suportados pelo solucionador e como a localização é resolvida.

Para resolver o local do dispositivo, especifique um ou mais desses tipos de dados de medição. Uma única localização resolvida será retornada para todos os dados de medição combinados.

Tópicos

- Solucionador baseado em Wi-Fi
- Solucionador baseado em celular
- Solucionador de pesquisa reversa de IP
- Solucionador GNSS
- Solucionador BLE para dispositivos habilitados para Amazon Sidewalk

Solucionador baseado em Wi-Fi

Use o solucionador baseado em Wi-Fi para resolver a localização usando as informações de varredura dos pontos de acesso Wi-Fi. O solucionador suporta a tecnologia WLAN e pode ser usado para calcular a localização do dispositivo para dispositivos IoT gerais conectados por TCP/IP, dispositivos LoRa sem fio WAN e dispositivos habilitados para Amazon Sidewalk.

Os dispositivos LoRa WAN devem ter o chipset LoRa Edge, que pode decodificar as informações de digitalização Wi-Fi recebidas. LoRa O Edge é uma plataforma de potência ultrabaixa que integra um LoRa transceptor de longo alcance, um scanner GNSS de várias constelações e um scanner MAC Wi-Fi passivo voltado para aplicativos de geolocalização. Quando uma mensagem de uplink é recebida do dispositivo, os dados da varredura de Wi-Fi são enviados para a Localização do AWS IoT Core dispositivo, e a localização é estimada com base nos resultados da verificação de Wi-Fi. As informações decodificadas são passadas para o solucionador baseado em Wi-Fi para recuperar as informações de localização.

Para saber mais sobre como resolver dados de localização para dispositivos habilitados para Amazon Sidewalk, visite o guia do AWS IoT Core para Amazon Sidewalk desenvolvedor.

Exemplo de carga de solucionador baseado em Wi-Fi

O código a seguir mostra um exemplo da carga JSON do dispositivo que contém os dados de medição. Quando AWS IoT Core o Device Location recebe esses dados como entrada, ele envia uma solicitação HTTP ao provedor do solucionador para resolver as informações de localização. Para recuperar as informações, especifique valores para o endereço MAC e intensidade do sinal recebido (RSS). Para fazer isso, forneça a carga JSON usando esse formato ou use o parâmetro de WiFiAccessPointsobjeto da operação da GetPositionEstimateAPI.

```
{
    "Timestamp": 1664313161,    // optional
    "WiFiAccessPoints": [
        {
            "MacAddress": "A0:EC:F9:1E:32:C1",  // required
            "Rss": -75                          // required
        }
    ]
}
```

Solucionador baseado em celular

Você pode usar o solucionador baseado em celular para resolver a localização usando dados de medição obtidos das torres de rádio celular. O solucionador oferece suporte às tecnologias a seguir. Uma única informação de localização resolvida é obtida, mesmo se você incluir dados de medição de qualquer uma ou de todas essas tecnologias.

- GSM
- CDMA

- WCDMA
- TD-SCDMA
- LTE

Exemplos de carga de solucionador baseado em celular

O código a seguir mostra um exemplo da carga JSON do dispositivo que contém os dados de medição celular. Quando AWS IoT Core o Device Location recebe esses dados como entrada, ele envia uma solicitação HTTP ao provedor do solucionador para resolver as informações de localização. Para recuperar as informações, você fornece a carga JSON usando esse formato no console ou especifica valores para o CellTowersparâmetro da operação da GetPositionEstimateAPI. Você pode fornecer os dados de medição especificando valores para parâmetros usando qualquer uma ou todas essas tecnologias celulares.

Evolução de longo prazo (LTE)

Ao usar esses dados de medição, você deve especificar informações como a rede e o código do país da rede móvel e parâmetros adicionais opcionais, incluindo informações sobre o ID da localização. Veja a seguir um exemplo do formato da carga. Para mais informações sobre esses parâmetros, consulte Objeto LTE.

```
{
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
        "Lte": [
        {
          "Mcc": int,                   // required
          "Mnc": int,                   // required
          "EutranCid": int,             // required. Make sure that you use int for
 EutranCid.
          "Tac": int,                   // optional
          "LteLocalId": {               // optional
              "Pci": int,               // required
              "Earfcn": int,            // required
          },
          "LteTimingAdvance": int,      // optional
          "Rsrp": int,                  // optional
          "Rsrq": float,                // optional
          "NrCapable": boolean,         // optional
          "LteNmr": [                   // optional
```

```
                {
                    "Pci": int,         // required
                    "Earfcn": int,      // required
                    "EutranCid": int,   // required
                    "Rsrp": int,        // optional
                    "Rsrq": float       // optional
                }
            ]
        }
    ]
  }
}
```

Sistema Global de Comunicações Móveis (GSM)

Ao usar esses dados de medição, você deve especificar informações como a rede e o código do país da rede móvel, informações de estação base e parâmetros adicionais opcionais. Veja a seguir um exemplo do formato da carga. Para obter mais informações sobre esses parâmetros, consulte Objeto GSM.

```
{
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
        "Gsm": [
        {
          "Mcc": int,                   // required
          "Mnc": int,                   // required
          "Lac": int,                   // required
          "GeranCid": int,              // required
          "GsmLocalId": {               // optional
              "Bsic": int,              // required
              "Bcch": int,              // required
          },
          "GsmTimingAdvance": int,      // optional
          "RxLevel": int,               // optional
          "GsmNmr": [                   // optional
            {
                "Bsic": int,            // required
                "Bcch": int,            // required
                "RxLevel": int,         // optional
                "GlobalIdentity": {
                    "Lac": int,         // required
                    "GeranCid": int     // required
```

```
                }
            }
        ]
    }
]
}
```

Acesso múltiplo por divisão de código (CDMA)

Ao usar esses dados de medição, você deve especificar informações, como a potência do sinal e informações de identificação, da estação base e parâmetros adicionais opcionais. Veja a seguir um exemplo do formato da carga. Para obter mais informações sobre esses parâmetros, consulte Objeto CDMA.

```
{
    "Timestamp": 1664313161,                // optional
    "CellTowers": {
        "Cdma": [
        {
            "SystemId": int,                // required
            "NetworkId": int,               // required
            "BaseStationId": int,           // required
            "RegistrationZone": int,        // optional
            "CdmaLocalId": {                // optional
              "PnOffset": int,              // required
              "CdmaChannel": int,           // required
            },
            "PilotPower": int,              // optional
            "BaseLat": float,               // optional
            "BaseLng": float,               // optional
            "CdmaNmr": [                    // optional
                {
                    "PnOffset": int,        // required
                    "CdmaChannel": int,     // required
                    "PilotPower": int,      // optional
                    "BaseStationId": int    // optional
                }
            ]
        }
    ]
}
}
```

Acesso múltiplo por divisão de código de ondas longas (CDMA)

Ao usar esses dados de medição, você deve especificar informações, como o código de rede e do país, potência do sinal e informações de identificação, da estação base e parâmetros adicionais opcionais. Veja a seguir um exemplo do formato da carga. Para obter mais informações sobre esses parâmetros, consulte Objeto CDMA.

```
{
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
        "Wcdma": [
        {
          "Mcc": int,                   // required
          "Mnc": int,                   // required
          "UtranCid": int,              // required
          "Lac": int,                   // optional
          "WcdmaLocalId": {             // optional
              "Uarfcndl": int,          // required
              "Psc": int,               // required
          },
          "Rscp": int,                  // optional
          "Pathloss": int,              // optional
          "WcdmaNmr": [                 // optional
                {
                  "Uarfcndl": int,      // required
                  "Psc": int,           // required
                  "UtranCid": int,      // required
                  "Rscp": int,          // optional
                  "Pathloss": int,      // optional
                }
             ]
          }
       ]
    }
}
```

Acesso múltiplo por divisão de código síncrona por divisão de tempo (TD-SCDMA)

Ao usar esses dados de medição, você deve especificar informações, como o código de rede e do país, potência do sinal e informações de identificação, da estação base e parâmetros adicionais opcionais. Veja a seguir um exemplo do formato da carga. Para obter mais informações sobre esses parâmetros, consulte Objeto CDMA.

```
{
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
        "Tdscdma": [
        {
          "Mcc": int,                   // required
          "Mnc": int,                   // required
          "UtranCid": int,              // required
          "Lac": int,                   // optional
          "TdscdmaLocalId": {           // optional
              "Uarfcn": int,            // required
              "CellParams": int,        // required
          },
          "TdscdmaTimingAdvance": int,  // optional
          "Rscp": int,                  // optional
          "Pathloss": int,              // optional
          "TdscdmaNmr": [               // optional
                {
                  "Uarfcn": int,        // required
                  "CellParams": int,    // required
                  "UtranCid": int,      // optional
                  "Rscp": int,          // optional
                  "Pathloss": int,      // optional
                }
             ]
         }
      ]
   }
}
```

Solucionador de pesquisa reversa de IP

Você pode usar o solucionador de pesquisa reversa de IP para resolver a localização usando o endereço IP como entrada. O solucionador pode obter as informações de localização dos dispositivos que foram provisionados com. AWS IoT Especifique as informações do endereço IP usando um formato que seja o IPv6 padrão padrão IPv4 ou o padrão comprimido IPv6 hexadecimal. Em seguida, você obtém a estimativa de localização resolvida, incluindo informações adicionais, como cidade e país onde o dispositivo está localizado.

> **Note**
>
> Ao usar a pesquisa reversa de IP, você concorda em não usá-la com a finalidade de identificar ou localizar uma residência específica ou um endereço.

Exemplo de carga do solucionador de pesquisa reversa de IP

O código a seguir mostra um exemplo da carga JSON do dispositivo que contém os dados de medição. Quando o AWS IoT Core Device Location recebe as informações do endereço IP nos dados de medição, ele consulta essas informações no banco de dados do provedor do solucionador, que é então usado para resolver as informações de localização. Para recuperar as informações, forneça a carga JSON usando esse formato ou especifique valores para o parâmetro Ip da operação da GetPositionEstimateAPI.

> **Note**
>
> Quando esse solucionador é usado, a cidade, o estado, o país e o código postal em que o dispositivo está localizado também são informados, além das coordenadas. Para ver um exemplo, consulte Como resolver o local do dispositivo (console).

```
{
    "Timestamp": 1664313161,
    "Ip":{
        "IpAddress":"54.240.198.35"
      }
}
```

Solucionador GNSS

Use o solucionador Sistema de navegação por satélite (GNSS) para recuperar o local do dispositivo usando as informações contidas nas mensagens de resultado de varredura de GNSS ou nas mensagens NAV. Opcionalmente, você pode fornecer informações adicionais de assistência GNSS, o que reduz o número de variáveis que o solucionador deve usar para pesquisar sinais. Ao fornecer essas informações de assistência, que incluem a posição, a altitude, o tempo de captura e as informações de precisão, o solucionador pode identificar facilmente os satélites à vista e calcular o local do dispositivo.

Esse solucionador pode ser usado com dispositivos LoRa WAN e dispositivos habilitados para Amazon Sidewalk, bem como outros dispositivos que foram provisionados com dispositivos de AWS IoT que foram provisionados com. AWS IoT Para dispositivos IoT gerais, se os dispositivos são compatíveis com a estimativa de localização usando GNSS, quando as informações de varredura de GNSS são recebidas do dispositivo, os transceptores resolvem as informações de local. Para dispositivos LoRa WAN, os dispositivos devem ter o chipset LoRa Edge. Quando uma mensagem de uplink é recebida do dispositivo, os dados de escaneamento GNSS são enviados para a LoRa WAN ou AWS IoT para o AWS IoT Amazon Sidewalk, e a localização é estimada com base nos resultados da verificação dos transceptores.

Exemplo de carga do solucionador GNSS

O código a seguir mostra um exemplo da carga JSON do dispositivo que contém os dados de medição. Quando o AWS IoT Core Device Location recebe as informações de varredura GNSS contendo a carga nos dados de medição, ele usa os transceptores e qualquer informação de assistência adicional incluída para pesquisar sinais e resolver as informações de localização. Para recuperar as informações, forneça a carga JSON usando esse formato ou especifique valores para o parâmetro Gnss da operação da API. GetPositionEstimate

> **Note**
>
> Antes AWS IoT Core que a localização do dispositivo possa resolver a localização do dispositivo, você deve remover o byte de destino da carga.

```
{
    "Timestamp": 1664313161,                // optional
    "Gnss": {
        "AssistAltitude": number,           // optional
        "AssistPosition": [ number ],       // optional
        "CaptureTime": number,              // optional
        "CaptureTimeAccuracy": number,      // optional
        "Payload": "string",                // required
        "Use2DSolver": boolean              // optional
   }
}
```

Solucionador BLE para dispositivos habilitados para Amazon Sidewalk

> **Note**
>
> Para localização baseada em Bluetooth Low Energy, AWS IoT retorna as coordenadas de localização com base na localização aproximada dos Sidewalk Gateways próximos que estão conectados ao Amazon Sidewalk e têm o recurso Community Finding ativado. Os dados de localização do gateway são AWS conteúdo e são fornecidos a você exclusivamente com a finalidade de ajudá-lo a localizar seus dispositivos conectados ao Amazon Sidewalk, e você só deve usar os dados para essa finalidade. Você só deve usar e acessar os dados de localização por meio da interface e da funcionalidade que geralmente disponibilizamos para você, e não deve tentar reidentificar, fazer engenharia reversa ou remapear quaisquer dados de localização do Gateway fornecidos por nós a você.

Os dispositivos finais do Amazon Sidewalk conectados AWS IoT Core para Amazon Sidewalk podem resolver seus dados de localização usando mensagens de uplink BLE, Wi-Fi ou GNSS publicadas pelo dispositivo final Amazon Sidewalk. Para obter mais informações, consulte o guia do AWS IoT Core para Amazon Sidewalk desenvolvedor.

Mensagens de eventos

Esta seção contém informações sobre mensagens publicadas AWS IoT quando itens ou trabalhos são atualizados ou alterados. Para obter informações sobre o AWS IoT Events serviço que permite criar detectores para monitorar seus dispositivos em busca de falhas ou alterações na operação e para acionar ações quando elas ocorrerem, consulte AWS IoT Events.

Como as mensagens de eventos são geradas

AWS IoT publica mensagens de eventos quando determinados eventos ocorrem. Por exemplo, os eventos são gerados pelo registro quando as objetos são adicionadas, atualizadas ou excluídas. Cada evento faz com que uma única mensagem de evento seja enviada. As mensagens de eventos são publicadas por meio do MQTT com uma carga JSON. O conteúdo da carga depende do tipo do evento.

> **Note**
>
> Há garantia de que as mensagens de eventos sejam publicadas uma vez. É possível que elas sejam publicadas mais de uma vez. A ordenação das mensagens de eventos não é garantida.

Política para receber mensagens de eventos

Para receber mensagens de eventos, seu dispositivo deve usar uma política apropriada que permita que ele se conecte ao gateway do AWS IoT dispositivo e assine os tópicos de eventos do MQTT. Você também deve assinar os filtros apropriados dos tópicos.

Veja a seguir um exemplo da política necessária para o recebimento de eventos de ciclo de vida:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe",
                "iot:Receive"
```

```
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:/$aws/events/*"
            ]
        }
    ]
}
```

Habilitar eventos para AWS IoT

Antes que os assinantes dos tópicos reservados possam receber mensagens, você deve habilitar as mensagens de eventos do Console de gerenciamento da AWS ou usando a API ou a CLI. Para obter informações sobre as mensagens de eventos que as diferentes opções gerenciam, consulte a Tabela de configurações de AWS IoT eventos.

- Para habilitar mensagens de eventos, acesse a guia Configurações do AWS IoT console e, na seção Mensagens baseadas em eventos, escolha Gerenciar eventos. Você pode especificar os eventos que deseja gerenciar.
- Para controlar quais tipos de eventos são publicados usando a API ou a CLI, chame a UpdateEventConfigurationsAPI ou use o comando da CLIupdate-event-configurations. Por exemplo:

```
aws iot update-event-configurations --event-configurations "{\"THING\":{\"Enabled\":
 true}}"
```

> **Note**
>
> Todas as aspas (") são recuadas com uma barra invertida (\).

Você pode obter a configuração atual do evento chamando a DescribeEventConfigurationsAPI ou usando o comando describe-event-configurations CLI. Por exemplo: .

```
aws iot describe-event-configurations
```

Tabela de configurações de eventos de AWS IoT

Categoria de evento (AWS IoT Console: Configurações: mensagens baseadas em eventos)	Valor da chave **eventConfigurations** (AWS CLI/API)	Tópico de mensagens de evento
(Só pode ser configurado usando a AWS CLI/API)	`CA_CERTIFICATE`	`$aws/events/certificates/registered/` *caCertificateId*
(Só pode ser configurado usando a AWS CLI/API)	`CERTIFICATE`	`$aws/events/presence/connected/` *clientId*
(Só pode ser configurado usando a AWS CLI/API)	`CERTIFICATE`	`$aws/events/presence/disconnected/` *clientId*
(Só pode ser configurado usando a AWS CLI/API)	`CERTIFICATE`	`$aws/events/subscriptions/subscribed/` *clientId*
(Só pode ser configurado usando a AWS CLI/API)	`CERTIFICATE`	`$aws/events/subscriptions/unsubscribed/` *clientId*
Trabalho concluído, cancelado	`JOB`	`$aws/events/job/`*jobID*`/canceled`
Trabalho concluído, cancelado	`JOB`	`$aws/events/job/`*jobID*`/cancellation_in_progress`
Trabalho concluído, cancelado	`JOB`	`$aws/events/job/`*jobID*`/completed`
Trabalho concluído, cancelado	`JOB`	`$aws/events/job/`*jobID*`/deleted`

Categoria de evento (AWS IoT Console: Configurações: mensagens baseadas em eventos)	Valor da chave **`eventConfigurations`** (AWS CLI/API)	Tópico de mensagens de evento
Trabalho concluído, cancelado	`JOB`	`$aws/events/job/`*`jobID`*`/deletion_in_progress`
Execução do trabalho: sucesso, falha, rejeitado, cancelado, removido	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/canceled`
Execução do trabalho: sucesso, falha, rejeitado, cancelado, removido	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/deleted`
Execução do trabalho: sucesso, falha, rejeitado, cancelado, removido	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/failed`
Execução do trabalho: sucesso, falha, rejeitado, cancelado, removido	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/rejected`
Execução do trabalho: sucesso, falha, rejeitado, cancelado, removido	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/removed`
Execução do trabalho: sucesso, falha, rejeitado, cancelado, removido	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/succeeded`
Execução do trabalho: sucesso, falha, rejeitado, cancelado, removido	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/timed_out`

Categoria de evento (AWS IoT Console: Configurações: mensagens baseadas em eventos)	Valor da chave **eventConfigurations** (AWS CLI/API)	Tópico de mensagens de evento
Objeto: criado, atualizado, excluído	`THING`	`$aws/events/thing/` *thingName* `/created`
Objeto: criado, atualizado, excluído	`THING`	`$aws/events/thing/` *thingName* `/updated`
Objeto: criado, atualizado, excluído	`THING`	`$aws/events/thing/` *thingName* `/deleted`
Grupo de objetos: adicionado, removido	`THING_GROUP`	`$aws/events/thingGroup/` *thingGroupName* `/created`
Grupo de objetos: adicionado, removido	`THING_GROUP`	`$aws/events/thingGroup/` *thingGroupName* `/updated`
Grupo de objetos: adicionado, removido	`THING_GROUP`	`$aws/events/thingGroup/` *thingGroupName* `/deleted`
Hierarquia de grupos de objetos: adicionada, removida	`THING_GROUP_HIERARCHY`	`$aws/events/thingGroupHierarchy/thingGroup/` *parentThingGroupName* `/childThingGroup/` *childThingGroupName* `/added`

Categoria de evento (AWS IoT Console: Configurações: mensagens baseadas em eventos)	Valor da chave **eventConfigurations** (AWS CLI/API)	Tópico de mensagens de evento
Hierarquia de grupos de objetos: adicionada, removida	`THING_GROUP_HIERARCHY`	`$aws/events/thingGroupHierarchy/thingGroup/` *parentThingGroupName* `/childThingGroup/` *childThingGroupName* `/removed`
Associação ao grupo de objetos: adicionada, removida	`THING_GROUP_MEMBERSHIP`	`$aws/events/thingGroupMembership/thingGroup/` *thingGroupName* `/thing/` *thingName* `/added`
Associação ao grupo de objetos: adicionada, removida	`THING_GROUP_MEMBERSHIP`	`$aws/events/thingGroupMembership/thingGroup/` *thingGroupName* `/thing/` *thingName* `/removed`
Tipo de objeto: criada, atualizada, excluída	`THING_TYPE`	`$aws/events/thingType/` *thingTypeName* `/created`
Tipo de objeto: criada, atualizada, excluída	`THING_TYPE`	`$aws/events/thingType/` *thingTypeName* `/updated`
Tipo de objeto: criada, atualizada, excluída	`THING_TYPE`	`$aws/events/thingType/` *thingTypeName* `/deleted`

Categoria de evento (AWS IoT Console: Configurações: mensagens baseadas em eventos)	Valor da chave **`eventConfigurations`** (AWS CLI/API)	Tópico de mensagens de evento

Associação do tipo de objeto: adicionada, removida	`THING_TYPE_ASSOCIATION`	`$aws/events/thingTypeAssociation/thing/` *`thingName`* `/thingType/` *`thingTypeName`* `/added`
		
		`$aws/events/thingTypeAssociation/thing/` *`thingName`* `/thingType/` *`thingTypeName`* `/removed`

Eventos de registro

O registro pode publicar mensagens de eventos quando objetos, tipos de objetos e grupos de objetos são criados, atualizados ou excluídos. No entanto, esses eventos não estão disponíveis por padrão. Para obter mais informações sobre como ativar esses eventos, consulte Habilitar eventos para AWS IoT.

O tipo de evento ocorreu com os seguintes tipos de evento:

- Eventos de objetos
- Eventos de tipos de objeto
- Eventos de grupos de objetos

Eventos de objetos

Coisa Created/Updated/Deleted

O registro publica as seguintes mensagens de eventos quando as objetos são criadas, atualizadas ou excluídas:

- `$aws/events/thing/thingName/created`
- `$aws/events/thing/thingName/updated`
- `$aws/events/thing/thingName/deleted`

As mensagens contêm os seguintes exemplos de carga útil:

```
{
    "eventType" : "THING_EVENT",
    "eventId" : "f5ae9b94-8b8e-4d8e-8c8f-b3266dd89853",
    "timestamp" : 1234567890123,
    "operation" : "CREATED|UPDATED|DELETED",
    "accountId" : "123456789012",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "thingName" : "MyThing",
    "versionNumber" : 1,
    "thingTypeName" : null,
    "attributes": {
                "attribute3": "value3",
                "attribute1": "value1",
                "attribute2": "value2"
    }
}
```

As cargas úteis contêm os seguintes atributos:

eventType

Defina como "THING_EVENT".

eventId

Um ID de evento exclusivo (sequência).

timestamp

A data e hora do UNIX de quando o evento ocorreu.

operação

A operação que acionou o evento. Os valores válidos são:

- CREATED
- UPDATED

- EXCLUÍDO

accountId

Sua Conta da AWS identidade.

thingId

O ID do objeto que está sendo criada, atualizada ou excluída.

thingName

O nome do objeto que está sendo criada, atualizada ou excluída.

versionNumber

A versão do objeto que está sendo criada, atualizada ou excluída. Esse valor é definido como 1 quando um objeto é criada. Ele é incrementado em 1 toda vez que o objeto é atualizada.

thingTypeName

O tipo do objeto associado ao objeto, se existir. Caso contrário, `null`.

attributes

Uma coleção de pares nome-valor associados ao objeto.

Eventos de tipos de objeto

Eventos relacionados ao tipo de objeto:

- Tipo de coisa Created/Updated/Deprecated/Undeprecated/Deleted
- Tipo de objeto associado ou desassociado de um objeto

Tipo de coisa Created/Updated/Deprecated/Undeprecated/Deleted

O registro publica as seguintes mensagens de evento quando os tipos de coisa são criados, atualizados, suspensos, recuperados ou excluídos:

- `$aws/events/thingType/`*`thingTypeName`*`/created`
- `$aws/events/thingType/`*`thingTypeName`*`/updated`
- `$aws/events/thingType/`*`thingTypeName`*`/deleted`

A mensagem contém as seguintes cargas úteis de exemplo:

```
{
    "eventType" : "THING_TYPE_EVENT",
    "eventId" : "8827376c-4b05-49a3-9b3b-733729df7ed5",
    "timestamp" : 1234567890123,
    "operation" : "CREATED|UPDATED|DELETED",
    "accountId" : "123456789012",
    "thingTypeId" : "c530ae83-32aa-4592-94d3-da29879d1aac",
    "thingTypeName" : "MyThingType",
    "isDeprecated" : false|true,
    "deprecationDate" : null,
    "searchableAttributes" : [ "attribute1", "attribute2", "attribute3" ],
    "propagatingAttributes": [
        {
     "userPropertyKey": "key",
     "thingAttribute": "model"
        },
        {
     "userPropertyKey": "key",
     "connectionAttribute": "iot:ClientId"
        }
    ],
    "description" : "My thing type"
}
```

As cargas úteis contêm os seguintes atributos:

eventType

Defina como "THING_TYPE_EVENT".

eventId

Um ID de evento exclusivo (sequência).

timestamp

A data e hora do UNIX de quando o evento ocorreu.

operação

A operação que acionou o evento. Os valores válidos são:

- CREATED
- UPDATED
- EXCLUÍDO

accountId

Sua Conta da AWS identidade.

thingTypeId

O ID do tipo de coisa que está sendo criada, atualizada, suspensa ou excluída.

thingTypeName

O nome do tipo de coisa que está sendo criada, atualizada, suspensa ou excluída.

isDeprecated

`true` se o tipo do objeto estiver preterido. Caso contrário, `false`.

deprecationDate

A data e hora do UNIX em que o tipo de objeto foi preterido.

searchableAttributes

Uma coleção de pares nome-valor associados ao tipo de objeto que pode ser usada para pesquisa.

propagatingAttributes

Lista de atributos de propagação. Um atributo de propagação pode conter um atributo de coisa, um atributo de conexão e uma chave de propriedade do usuário. Para acessar mais informações, consulte Adicionar atributos de propagação para enriquecimento de mensagens.

descrição

Uma descrição do tipo de objeto.

Tipo de objeto associado ou desassociado de um objeto

O registro publica as seguintes mensagens de eventos quando um tipo de objeto é associado ou desassociado de um objeto.

- `$aws/events/thingTypeAssociation/thing/`*`thingName`*`/thingType/`*`typeName`*`/added`
- `$aws/events/thingTypeAssociation/thing/`*`thingName`*`/thingType/`*`typeName`*`/removed`

Veja a seguir um exemplo de carga `added`. As cargas das mensagens `removed` são semelhantes.

```
{
    "eventId" : "87f8e095-531c-47b3-aab5-5171364d138d",
    "eventType" : "THING_TYPE_ASSOCIATION_EVENT",
    "operation" : "ADDED",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "thingName": "myThing",
    "thingTypeName" : "MyThingType",
    "timestamp" : 1234567890123,
}
```

As cargas úteis contêm os seguintes atributos:

eventId

Um ID de evento exclusivo (sequência).

eventType

Defina como "THING_TYPE_ASSOCIATION_EVENT".

operação

A operação que acionou o evento. Os valores válidos são:

- ADICIONADO
- REMOVIDO

thingId

O ID do objeto cuja associação do tipo foi alterada.

thingName

O nome do objeto cuja associação do tipo foi alterada.

thingTypeName

O tipo de objeto associado ou não mais associado ao objeto.

timestamp

A data e hora do UNIX de quando o evento ocorreu.

Eventos de grupos de objetos

Eventos relacionados ao grupo de objetos:

- Thing Group Created/Updated/Deleted
- Objeto adicionado ou removido de um grupo de objetos
- Grupo de objetos adicionado ou removido de um grupo de objetos

Thing Group Created/Updated/Deleted

O registro publica as seguintes mensagens de eventos quando um grupo de objetos é criado, atualizado ou excluído.

- `$aws/events/thingGroup/`*`groupName`*`/created`
- `$aws/events/thingGroup/`*`groupName`*`/updated`
- `$aws/events/thingGroup/`*`groupName`*`/deleted`

Veja a seguir um exemplo de carga `updated`. As cargas para mensagens `created` e `deleted` são semelhantes.

```
{
  "eventType": "THING_GROUP_EVENT",
  "eventId": "8b9ea8626aeaa1e42100f3f32b975899",
  "timestamp": 1603995417409,
  "operation": "UPDATED",
  "accountId": "571EXAMPLE833",
  "thingGroupId": "8757eec8-bb37-4cca-a6fa-403b003d139f",
  "thingGroupName": "Tg_level5",
  "versionNumber": 3,
  "parentGroupName": "Tg_level4",
  "parentGroupId": "5fce366a-7875-4c0e-870b-79d8d1dce119",
  "description": "New description for Tg_level5",
  "rootToParentThingGroups": [
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/TgTopLevel",
      "groupId": "36aa0482-f80d-4e13-9bff-1c0a75c055f6"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level1",
      "groupId": "bc1643e1-5a85-4eac-b45a-92509cbe2a77"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level2",
      "groupId": "0476f3d2-9beb-48bb-ae2c-ea8bd6458158"
```

```
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level3",
      "groupId": "1d9d4ffe-a6b0-48d6-9de6-2e54d1eae78f"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level4",
      "groupId": "5fce366a-7875-4c0e-870b-79d8d1dce119"
    }
  ],
  "attributes": {
    "attribute1": "value1",
    "attribute3": "value3",
    "attribute2": "value2"
  },
  "dynamicGroupMappingId": null
}
```

As cargas úteis contêm os seguintes atributos:

eventType

Defina como "THING_GROUP_EVENT".

eventId

Um ID de evento exclusivo (sequência).

timestamp

A data e hora do UNIX de quando o evento ocorreu.

operação

A operação que acionou o evento. Os valores válidos são:

- CREATED
- UPDATED
- EXCLUÍDO

accountId

Sua Conta da AWS identidade.

thingGroupId

O ID do grupo de objetos que está sendo criado, atualizado ou excluído.

thingGroupName

O nome do grupo de objetos que está sendo criado, atualizado ou excluído.

versionNumber

A versão do grupo de objetos. Esse valor é definido como 1 quando um grupo de objetos é criado. Ele é incrementado em 1 toda vez que o grupo de objetos é atualizado.

parentGroupName

O nome do grupo de objetos pai, se houver.

parentGroupId

O ID do grupo de objetos pai, se houver.

descrição

Uma descrição do grupo de objetos.

rootToParentThingGroups

Um conjunto de informações sobre o grupo de objetos pai. Há um elemento para cada grupo de objetos principal, começando pelo grupo de objetos raiz e continuando até o grupo de objetos principal. Cada entrada contém o `groupArn` e `groupId`. do grupo de objetos.

attributes

Uma coleção de pares nome-valor associados ao grupo de objetos.

Objeto adicionado ou removido de um grupo de objetos

O registro publica as seguintes mensagens de eventos quando um objeto é adicionada ou removida de um grupo de objetos.

- `$aws/events/thingGroupMembership/thingGroup/`*`thingGroupName`*`/thing/`*`thingName`*`/added`
- `$aws/events/thingGroupMembership/thingGroup/`*`thingGroupName`*`/thing/`*`thingName`*`/removed`

As mensagens contêm os seguintes exemplos de carga útil:

```
{
```

```
    "eventType" : "THING_GROUP_MEMBERSHIP_EVENT",
    "eventId" : "d684bd5f-6f6e-48e1-950c-766ac7f02fd1",
    "timestamp" : 1234567890123,
    "operation" : "ADDED|REMOVED",
    "accountId" : "123456789012",
    "groupArn" : "arn:aws:iot:ap-northeast-2:123456789012:thinggroup/
MyChildThingGroup",
    "groupId" : "06838589-373f-4312-b1f2-53f2192291c4",
    "thingArn" : "arn:aws:iot:ap-northeast-2:123456789012:thing/MyThing",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "membershipId" : "8505ebf8-4d32-4286-80e9-c23a4a16bbd8"
}
```

As cargas úteis contêm os seguintes atributos:

eventType

Defina como "THING_GROUP_MEMBERSHIP_EVENT".

eventId

O ID do evento.

timestamp

A data e hora do UNIX de quando o evento ocorreu.

operação

`ADDED` quando um objeto é adicionada a um grupo de objetos. `REMOVED` quando um objeto é removida de um grupo de objetos.

accountId

Sua Conta da AWS identidade.

groupArn

O ARN do grupo de objetos.

groupId

O ID do group.

thingArn

O ARN do objeto que foi adicionada ou removida do grupo de objetos.

thingId

O ID do objeto que foi adicionada ou removida do grupo de objetos.

membershipId

Um ID que representa a relação entre o objeto e o grupo de objetos. Esse valor é gerado quando você adiciona um objeto a um grupo de objetos.

Grupo de objetos adicionado ou removido de um grupo de objetos

O registro publica as seguintes mensagens de eventos quando um grupo de objetos é adicionado ou removido de um outro grupo de objetos.

- `$aws/events/thingGroupHierarchy/thingGroup/`*`parentThingGroupName`*`/childThingGroup/`*`childThingGroupName`*`/added`
- `$aws/events/thingGroupHierarchy/thingGroup/`*`parentThingGroupName`*`/childThingGroup/`*`childThingGroupName`*`/removed`

A mensagem contém as seguintes cargas úteis de exemplo:

```
{
    "eventType" : "THING_GROUP_HIERARCHY_EVENT",
    "eventId" : "264192c7-b573-46ef-ab7b-489fcd47da41",
    "timestamp" : 1234567890123,
    "operation" : "ADDED|REMOVED",
    "accountId" : "123456789012",
    "thingGroupId" : "8f82a106-6b1d-4331-8984-a84db5f6f8cb",
    "thingGroupName" : "MyRootThingGroup",
    "childGroupId" : "06838589-373f-4312-b1f2-53f2192291c4",
    "childGroupName" : "MyChildThingGroup"
}
```

As cargas úteis contêm os seguintes atributos:

eventType

Defina como "THING_GROUP_HIERARCHY_EVENT".

eventId

O ID do evento.

timestamp

A data e hora do UNIX de quando o evento ocorreu.

operação

`ADDED` quando um objeto é adicionada a um grupo de objetos. `REMOVED` quando um objeto é removida de um grupo de objetos.

accountId

Sua Conta da AWS identidade.

thingGroupId

O ID do grupo de objetos pai.

thingGroupName

O nome do grupo de objetos pai.

childGroupId

O ID do grupo de objetos filho.

childGroupName

O nome do grupo de objetos filho.

Eventos de trabalho

O serviço AWS IoT Jobs publica em tópicos reservados no protocolo MQTT quando os trabalhos estão pendentes, concluídos ou cancelados e quando um dispositivo relata sucesso ou falha ao executar um trabalho. Os dispositivos ou os aplicativos de gerenciamento e monitoramento podem acompanhar o status de trabalhos assinando esses tópicos.

Como habilitar eventos de trabalhos

As mensagens de resposta do serviço AWS IoT Jobs não passam pelo agente de mensagens e não podem ser assinadas por outros clientes ou regras. Para assinar mensagens relacionadas a atividades de trabalhos, use os tópicos `notify` e `notify-next`. Para obter mais informações sobre os tópicos de trabalhos, consulte Tópicos de trabalhos.

Para ser notificado sobre atualizações de trabalhos, ative esses eventos de trabalhos usando o Console de gerenciamento da AWS, ou usando a API ou a CLI. Para obter mais informações, consulte Habilitar eventos para AWS IoT.

Como funcionam os eventos de trabalho

Como cancelamento ou exclusão do trabalho podem levar algum tempo, duas mensagens são enviadas para indicar o início e o fim de uma solicitação. Por exemplo, quando uma solicitação de cancelamento é iniciada, uma mensagem é enviada para o tópico `$aws/events/job/jobID/cancellation_in_progress`. Quando uma solicitação de cancelamento é concluída, uma mensagem é enviada para o tópico `$aws/events/job/jobID/canceled`.

Um processo similar ocorre para uma solicitação de exclusão do trabalho. Os aplicativos de gerenciamento e de monitoramento podem assinar esses tópicos para acompanhar os status de trabalhos. Para obter mais informações sobre publicação e assinatura em tópicos MQTT, consulte the section called “Protocolos de comunicação do dispositivo”.

Tipos de eventos do trabalho

Os seguintes exemplos mostram os diferentes tipos de eventos de trabalho:

Job Completed/Canceled/Deleted

O serviço AWS IoT Jobs publica uma mensagem em um tópico do MQTT quando um trabalho é concluído, cancelado, excluído ou quando o cancelamento ou a exclusão estão em andamento:

- `$aws/events/job/jobID/completed`
- `$aws/events/job/jobID/canceled`
- `$aws/events/job/jobID/deleted`
- `$aws/events/job/jobID/cancellation_in_progress`
- `$aws/events/job/jobID/deletion_in_progress`

A mensagem `completed` contém as seguintes cargas úteis de exemplo:

```
{
  "eventType": "JOB",
  "eventId": "7364ffd1-8b65-4824-85d5-6c14686c97c6",
  "timestamp": 1234567890,
  "operation": "completed",
  "jobId": "27450507-bf6f-4012-92af-bb8a1c8c4484",
  "status": "COMPLETED",
  "targetSelection": "SNAPSHOT|CONTINUOUS",
  "targets": [
    "arn:aws:iot:us-east-1:123456789012:thing/a39f6f91-70cf-4bd2-a381-9c66df1a80d0",
```

```
    "arn:aws:iot:us-east-1:123456789012:thinggroup/2fc4c0a4-6e45-4525-
a238-0fe8d3dd21bb"
  ],
  "description": "My Job Description",
  "completedAt": 1234567890123,
  "createdAt": 1234567890123,
  "lastUpdatedAt": 1234567890123,
  "jobProcessDetails": {
    "numberOfCanceledThings": 0,
    "numberOfRejectedThings": 0,
    "numberOfFailedThings": 0,
    "numberOfRemovedThings": 0,
    "numberOfSucceededThings": 3
  }
}
```

A mensagem `canceled` contém a seguinte carga de exemplo.

```
{
  "eventType": "JOB",
  "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
  "timestamp": 1234567890,
  "operation": "canceled",
  "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
  "status": "CANCELED",
  "targetSelection": "SNAPSHOT|CONTINUOUS",
  "targets": [
    "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
    "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
  ],
  "description": "My job description",
  "createdAt": 1234567890123,
  "lastUpdatedAt": 1234567890123
}
```

A mensagem `deleted` contém a seguinte carga de exemplo.

```
{
      "eventType": "JOB",
      "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
      "timestamp": 1234567890,
```

```
      "operation": "deleted",
      "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
      "status": "DELETED",
      "targetSelection": "SNAPSHOT|CONTINUOUS",
      "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
      ],
      "description": "My job description",
      "createdAt": 1234567890123,
      "lastUpdatedAt": 1234567890123,
      "comment": "Comment for this operation"
    }
```

A mensagem `cancellation_in_progress` contém as seguintes cargas úteis de exemplo:

```
{
      "eventType": "JOB",
      "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
      "timestamp": 1234567890,
      "operation": "cancellation_in_progress",
      "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
      "status": "CANCELLATION_IN_PROGRESS",
      "targetSelection": "SNAPSHOT|CONTINUOUS",
      "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
      ],
      "description": "My job description",
      "createdAt": 1234567890123,
      "lastUpdatedAt": 1234567890123,
      "comment": "Comment for this operation"
    }
```

A mensagem `deletion_in_progress` contém as seguintes cargas úteis de exemplo:

```
{
      "eventType": "JOB",
      "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
```

```
      "timestamp": 1234567890,
      "operation": "deletion_in_progress",
      "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
      "status": "DELETION_IN_PROGRESS",
      "targetSelection": "SNAPSHOT|CONTINUOUS",
      "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
      ],
      "description": "My job description",
      "createdAt": 1234567890123,
      "lastUpdatedAt": 1234567890123,
      "comment": "Comment for this operation"
    }
```

Status terminal da execução do trabalho

O serviço AWS IoT Jobs publica uma mensagem quando um dispositivo atualiza a execução de uma tarefa para o status do terminal:

- `$aws/events/jobExecution/`*`jobID`*`/succeeded`
- `$aws/events/jobExecution/`*`jobID`*`/failed`
- `$aws/events/jobExecution/`*`jobID`*`/rejected`
- `$aws/events/jobExecution/`*`jobID`*`/canceled`
- `$aws/events/jobExecution/`*`jobID`*`/timed_out`
- `$aws/events/jobExecution/`*`jobID`*`/removed`
- `$aws/events/jobExecution/`*`jobID`*`/deleted`

A mensagem contém as seguintes cargas úteis de exemplo:

```
{
  "eventType": "JOB_EXECUTION",
  "eventId": "cca89fa5-8a7f-4ced-8c20-5e653afb3572",
  "timestamp": 1234567890,
  "operation": "succeeded|failed|rejected|canceled|removed|timed_out",
  "jobId": "154b39e5-60b0-48a4-9b73-f6f8dd032d27",
  "thingArn": "arn:aws:iot:us-east-1:123456789012:myThing/6d639fbc-8f85-4a90-924d-
a2867f8366a7",
  "status": "SUCCEEDED|FAILED|REJECTED|CANCELED|REMOVED|TIMED_OUT",
```

```
  "statusDetails": {
    "key": "value"
  }
}
```

Eventos de execução de comandos

AWS IoT publica mensagens de eventos em tópicos do MQTT quando as execuções de comandos mudam de status. Você pode usar esses eventos para monitorar o progresso da execução do comando e criar aplicativos que respondam às mudanças de status.

Tópicos de eventos de execução de comandos

Os eventos de execução do comando são publicados no seguinte tópico do MQTT:

```
$aws/events/commandExecution/commandId/status
```

Em que:

- *commandId*é o identificador do comando.
- *status*é o status da execução do comando. Os valores válidos são: `CREATED`, `IN_PROGRESS`, `SUCCEEDED`, `FAILED`, `REJECTED`, `TIMED_OUT`.

Para receber notificações para todos os comandos e todos os status, assine o tópico a seguir usando caracteres curinga:

```
$aws/events/commandExecution/+/#
```

Mensagem do evento de execução do comando

Quando o status de execução de um comando muda, AWS IoT publica uma mensagem de evento no tópico correspondente do MQTT. A mensagem contém as seguintes cargas úteis de exemplo:

```
{
    "executionId": "2bd65c51-4cfd-49e4-9310-d5cbfdbc8554",
    "status": "FAILED",
    "statusReason": {
        "reasonCode": "DEVICE_TOO_BUSY",
        "reasonDescription": ""
```

```
    },
    "eventType": "COMMAND_EXECUTION",
    "commandArn": "arn:aws:iot:us-east-1:123456789012:command/0b9d9ddf-
e873-43a9-8e2c-9fe004a90086",
    "targetArn": "arn:aws:iot:us-east-1:123456789012:thing/5006c3fc-
de96-4def-8427-7eee36c6f2bd",
    "timestamp": 1717708862107
}
```

As cargas contêm os seguintes atributos:

ID de execução

Um identificador exclusivo para a execução do comando (string).

status

O status da execução do comando. Os valores válidos são: `CREATED`, `IN_PROGRESS`, `SUCCEEDED`, `FAILED`, `REJECTED`, `TIMED_OUT`.

statusReason

Um objeto contendo informações adicionais sobre o status, se disponível. Contém `reasonCode` e `reasonDescription` campos.

eventType

Defina como “COMMAND_EXECUTION”.

Comando ARN

O Amazon Resource Name (ARN) do comando.

TargetARN

O ARN do dispositivo de destino (coisa ou cliente) para a execução do comando.

timestamp

A data e hora do UNIX de quando o evento ocorreu.

Eventos de ciclo de vida

AWS IoT pode publicar eventos do ciclo de vida sobre os tópicos do MQTT. Esses eventos estão disponíveis por padrão e não podem ser desativados.

> **Note**
>
> As mensagens de ciclo de vida podem ser enviadas fora de ordem. Você pode receber mensagens duplicadas.
> `thingName` só será incluído se o cliente estiver se conectando usando o recurso de coisa exclusiva.

Neste tópico:

- Eventos de conexão/desconexão
- Evento de falha na tentativa de conexão
- Eventos de assinatura/cancelamento de assinatura

Eventos de conexão/desconexão

> **Note**
>
> Com a indexação de frotas do AWS IoT Device Management, você pode pesquisar coisas, executar consultas agregadas e criar grupos dinâmicos com base em eventos de coisas. Connect/Disconnect Para obter mais informações, consulte Indexação de frota.

AWS IoT publica uma mensagem nos seguintes tópicos do MQTT quando um cliente se conecta ou desconecta:

- `$aws/events/presence/connected/`*`clientId`* – Um cliente se conectou apenas ao agente de mensagens.
- `$aws/events/presence/disconnected/`*`clientId`* – Um cliente se desconectou do agente de mensagens.

Veja a seguir uma lista dos elementos JSON contidos nas connection/disconnection mensagens publicadas no `$aws/events/presence/connected/`*`clientId`* tópico.

clientId

O ID do cliente que se conecta ou que se desconecta.

> **Note**
>
> Clientes IDs que contêm # ou + não recebem eventos de ciclo de vida.

thingName

O nome da sua coisa da IoT. `thingName` só será incluído se o cliente estiver se conectando usando o recurso de coisa exclusiva.

clientInitiatedDisconnect

Verdadeiro se o cliente iniciou a desconexão. Caso contrário, falso. Encontrado apenas em mensagens de desconexão.

disconnectReason

A razão pela qual o cliente está se desconectando. Encontrado apenas em mensagens de desconexão. A tabela a seguir contém valores válidos e se o agente enviará mensagens Last Will and Testament (LWT) quando a desconexão ocorrer.

Motivo da desconexão	Description	O agente enviará as mensagens LWT
`AUTH_ERROR`	Falha ao autenticar o cliente, ou a autorização falhou.	Sim[1]
`CLIENT_INITIATED_DISCONNECT`	O cliente indica que ele se desconectará. O cliente pode fazer isso enviando um pacote de `DISCONNECT` controle MQTT ou `Close frame` se o cliente estiver usando uma WebSocket conexão.	Não
`CLIENT_ERROR`	O cliente fez algo errado que faz com que ele se desconecte. Por exemplo, um cliente será desconectado para enviar mais de 1 pacote `CONNECT` MQTT na mesma conexão ou se o cliente tentar publicar com uma carga útil que exceda o limite de carga útil.	Sim

Motivo da desconexão	Description	O agente enviará as mensagens LWT
`CONNECTION_LOST`	A conexão cliente-servidor é cortada. Isso pode acontecer durante um período de alta latência de rede ou quando a conexão com a Internet é perdida.	Sim
`DUPLICATE_CLIENTID`	O cliente está usando um ID de cliente que já está em uso. Nesse caso, o cliente que já está conectado será desconectado com esse motivo de desconexão.	Sim
`FORBIDDEN_ACCESS`	O cliente não tem permissão para ser conectado . Por exemplo, um cliente com um endereço IP negado falhará ao se conectar.	Sim[1]
`MQTT_KEEP_ALIVE_TIMEOUT`	Se não houver nenhuma comunicação cliente-servidor para 1,5x do tempo keep-alive do cliente, o cliente será desconectado.	Sim
`SERVER_ERROR`	Desconectado devido a problemas inesperados no servidor.	Sim
`SERVER_INITIATED_DISCONNECT`	O servidor desconecta intencionalmente um cliente por razões operacionais.	Sim
`API_INITIATED_DISCONNECT`	O cliente foi desconectado usando a API `DeleteConnection` .	Sim[2]
`THROTTLED`	O cliente é desconectado por exceder um limite de controle.	Sim

Motivo da desconexão	Description	O agente enviará as mensagens LWT
`WEBSOCKET_TTL_EXPIRATION`	O cliente está desconectado porque a está conectado WebSocket há mais tempo do que seu time-to-live valor.	Sim
`CUSTOMAUTH_TTL_EXPIRATION`	O cliente está desconectado porque está conectado há mais tempo do que o time-to-live valor de seu autorizador personalizado.	Sim

[1]Se o dispositivo tiver uma conexão ativa antes de receber esse erro.

[2] Para evitar mensagens Last Will and Testament (LWT), configure `preventWillMessage=true` para substituir o comportamento de envio de LWT padrão da API `DeleteConnection`.

eventType

O tipo de evento. Os valores válidos são `connected` ou `disconnected`.

ipAddress

O endereço IP do cliente que está se conectando. Isso pode estar em IPv4 nosso IPv6 formato. Encontrado somente em mensagens de conexão.

principalIdentifier

A credencial usada para autenticar. Para certificados de autenticação mútua TLS, a credencial é o ID do certificado. Para outras conexões, são as credenciais do IAM.

sessionIdentifier

Um identificador globalmente exclusivo AWS IoT que existe durante toda a sessão.

timestamp

Uma estimativa de quando o evento ocorreu.

versionNumber

O número de versão para o evento de ciclo de vida. Esse é um valor inteiro longo que cresce monotonicamente para cada conexão do ID do cliente. O número da versão pode ser usado por um assinante para inferir a ordem dos eventos de ciclo de vida.

> **Note**
>
> As mensagens de conexão e desconexão para uma conexão do cliente têm o mesmo número de versão.
> O número da versão pode ignorar valores e não há garantia de que ele aumente de forma consistente em 1 para cada evento.
> Se um cliente não permanecer conectado por aproximadamente uma hora, o número da versão será redefinido como 0. Para sessões persistentes, o número da versão é redefinido para 0 após um cliente ter sido desconectado por mais tempo do que o configurado time-to-live (TTL) para a sessão persistente.

Uma mensagem de conexão tem a seguinte estrutura.

```
{
    "clientId": "186b5",
    "thingName": "exampleThing",
    "timestamp": 1573002230757,
    "eventType": "connected",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "principalIdentifier": "12345678901234567890123456789012",
    "ipAddress": "192.0.2.0",
    "versionNumber": 0
}
```

Uma mensagem de desconexão tem a seguinte estrutura.

```
{
    "clientId": "186b5",
    "thingName": "exampleThing",
    "timestamp": 1573002340451,
    "eventType": "disconnected",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "principalIdentifier": "12345678901234567890123456789012",
```

```
    "clientInitiatedDisconnect": true,
    "disconnectReason": "CLIENT_INITIATED_DISCONNECT",
    "versionNumber": 0
}
```

Gerenciamento de desconexões de clientes

A prática recomendada é sempre ter um estado de espera implementado para eventos de ciclo de vida, incluindo mensagens Last Will and Testament (LWT). Quando uma mensagem de desconexão é recebida, seu código deve aguardar um período e verificar se um dispositivo ainda está offline antes de realizar uma ação. Uma forma de fazer isso é usando Filas de atraso do SQS. Quando um cliente recebe um evento LWT ou de ciclo de vida, você pode colocar uma mensagem na fila (por exemplo, por 5 segundos). Quando essa mensagem se torna disponível e é processada (pelo Lambda ou outro serviço), você pode primeiro verificar se o dispositivo ainda está off-line antes de realizar outras ações.

Evento de falha na tentativa de conexão

AWS IoT publica uma mensagem no seguinte tópico do MQTT quando um cliente não está autorizado a se conectar ou quando um último testamento e testamento são configurados e o cliente não está autorizado a publicar nesse tópico do último testamento.

```
$aws/events/presence/connect_failed/clientId
```

Veja a seguir uma lista de elementos JSON contidos nas mensagens de autorização de conexão publicadas no tópico `$aws/events/presence/connect_failed/clientId`.

clientId

O ID do cliente que tentou, mas não conseguiu se conectar.

> **Note**
>
> Clientes IDs que contêm # ou + não recebem eventos de ciclo de vida.

thingName

O nome da sua coisa da IoT. `thingName` só será incluído se o cliente estiver se conectando usando o recurso de coisa exclusiva.

timestamp

Uma estimativa de quando o evento ocorreu.

eventType

O tipo de evento. O valor válido é `connect_failed`.

connectFailureReason

A razão pela qual a conexão falha. O valor válido é `AUTHORIZATION_FAILED`.

principalIdentifier

A credencial usada para autenticar. Para certificados de autenticação mútua TLS, a credencial é o ID do certificado. Para outras conexões, são as credenciais do IAM.

sessionIdentifier

Um identificador globalmente exclusivo AWS IoT que existe durante toda a sessão.

ipAddress

O endereço IP do cliente que está se conectando. Isso pode estar em IPv4 nosso IPv6 formato. Encontrado somente em mensagens de conexão.

Uma mensagem de falha de conexão tem a estrutura a seguir.

```
{
    "clientId": "186b5",
    "thingName": "exampleThing",
    "timestamp": 1460065214626,
    "eventType": "connect_failed",
    "connectFailureReason": "AUTHORIZATION_FAILED",
    "principalIdentifier": "12345678901234567890123456789012",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "ipAddress" : "192.0.2.0"
}
```

Eventos de assinatura/cancelamento de assinatura

AWS IoT publica uma mensagem no seguinte tópico do MQTT quando um cliente se inscreve ou cancela a assinatura de um tópico do MQTT:

```
$aws/events/subscriptions/subscribed/clientId
```

or

```
$aws/events/subscriptions/unsubscribed/clientId
```

Onde `clientId` é o ID do cliente MQTT que se conecta ao agente de mensagens da AWS IoT .

A mensagem publicada para este tópico tem a seguinte estrutura:

```
{
    "clientId": "186b5",
    "thingName": "exampleThing",
    "timestamp": 1460065214626,
    "eventType": "subscribed" | "unsubscribed",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "principalIdentifier": "12345678901234567890123456789012",
    "topics" : ["foo/bar","device/data","dog/cat"]
}
```

Veja a seguir uma lista de elementos JSON que estão contidos nas mensagens assinadas e não assinadas publicadas nos tópicos `$aws/events/subscriptions/subscribed/clientId` e `$aws/events/subscriptions/unsubscribed/clientId`.

clientId

O ID do cliente que assina ou que cancela a assinatura.

> **Note**
>
> Clientes IDs que contêm # ou + não recebem eventos de ciclo de vida.

thingName

O nome da sua coisa da IoT. `thingName` só será incluído se o cliente estiver se conectando usando o recurso de coisa exclusiva.

eventType

O tipo de evento. Os valores válidos são `subscribed` ou `unsubscribed`.

principalIdentifier

A credencial usada para autenticar. Para certificados de autenticação mútua TLS, a credencial é o ID do certificado. Para outras conexões, são as credenciais do IAM.

sessionIdentifier

Um identificador globalmente exclusivo AWS IoT que existe durante toda a sessão.

timestamp

Uma estimativa de quando o evento ocorreu.

tópicos

Uma série de tópicos MQTT nos quais o cliente se inscreveu.

> **Note**
>
> As mensagens de ciclo de vida podem ser enviadas fora de ordem. Você pode receber mensagens duplicadas.

Solução de problemas AWS IoT

Ajude-nos a melhorar este tópico

Conte para nós o que ajudaria a torná-lo melhor

As informações a seguir podem ajudar a solucionar problemas comuns no AWS IoT.

Tarefas

- AWS IoT Core guia de solução de problemas
- AWS IoT Device Management guia de solução de problemas
- AWS IoT Guia de solução de problemas do Device Advisor
- AWS IoT erros

AWS IoT Core guia de solução de problemas

Ajude-nos a melhorar este tópico

Conte para nós o que ajudaria a torná-lo melhor

Esta é a seção de solução de problemas do AWS IoT Core.

Tópicos

- Diagnóstico de problemas de conectividade
- Diagnosticar problemas de regras
- Diagnosticar problemas com shadows
- Diagnosticar problemas de ação no fluxo de entrada da Salesforce IoT
- Diagnosticando limites de fluxo
- Solução de problemas de desconexões da frota de dispositivos

Diagnóstico de problemas de conectividade

> Ajude-nos a melhorar este tópico
>
> Conte para nós o que ajudaria a torná-lo melhor

Uma conexão bem-sucedida com AWS IoT requer:

- Uma conexão válida
- Um certificado válido e ativo
- Uma política que permite a conexão e a operação desejadas

Conexão

Como faço para encontrar o endpoint correto?

- O `endpointAddress` retornado por `aws iot describe-endpoint --endpoint-type iot:Data-ATS`

  ou

- O `domainName` retornado por `aws iot describe-domain-configuration --domain-configuration-name "domain_configuration_name"`

Como faço para encontrar o valor correto da Server Name Indication (SNI — Indicação de nome de servidor)?

O valor correto do SNI é o `endpointAddress` retornado pelo describe-endpoint ou o `domainName` retornado pelos describe-domain-configuration comandos. É o mesmo endereço do endpoint na etapa anterior. Ao conectar dispositivos a AWS IoT Core, os clientes podem enviar a extensão de Indicação de Nome do Servidor (SNI), que não é obrigatória, mas é altamente recomendada. Para usar atributos como registro de várias contas, domínios personalizados e endpoints da VPC, é necessário usar a extensão SNI. Para obter mais informações, consulte Segurança de transporte em AWS IoT.

Como resolvo um problema de conectividade que persiste?

Você pode usar o AWS Device Advisor para ajudar a solucionar problemas. Os testes pré-criados do Device Advisor ajudam você a validar o software do dispositivo em relação às práticas recomendadas de uso de TLS, MQTT, Sombra do dispositivo AWS IoT e Tarefas de AWS IoT.

Aqui está um link para o conteúdo existente do Device Advisor.

Autenticação

Os dispositivos devem ser autenticados para se conectarem aos AWS IoT endpoints. Para dispositivos que usam Certificados do cliente X.509 para autenticação, os certificados devem estar registrados AWS IoT e estar ativos.

Como meus dispositivos autenticam AWS IoT endpoints?

Adicione o certificado AWS IoT CA ao armazenamento confiável do seu cliente. Consulte a documentação em Autenticação de servidor no AWS IoT Core e siga os links para fazer download do certificado apropriado da CA.

O que é verificado quando um dispositivo se conecta a AWS IoT?

Quando um dispositivo tenta se conectar ao AWS IoT:

1. AWS IoT verifica se há um certificado válido e um valor de Indicação de Nome de Servidor (SNI).
2. AWS IoT verifica se o certificado usado está registrado na AWS IoT Conta e se foi ativado.
3. Quando um dispositivo tenta realizar qualquer ação AWS IoT, como assinar ou publicar uma mensagem, a política anexada ao certificado usado para se conectar é verificada para confirmar se o dispositivo está autorizado a realizar essa ação.

Como posso validar um certificado configurado corretamente?

Use o comando `s_client` OpenSSL para testar uma conexão com o endpoint da AWS IoT :

```
openssl s_client -connect custom_endpoint.iot.aws-region.amazonaws.com:8443 -
CAfile CA.pem -cert cert.pem -key privateKey.pem
```

Para obter mais informações sobre como usar o `openssl s_client`, consulte a documentação do OpenSSL s_client.

Como faço para verificar o status de um certificado?

- Liste os certificados

  Se você não souber o ID do certificado, poderá ver o status de todos os seus certificados usando o comando `aws iot list-certificates`.
- Mostrar os detalhes de um certificado

Se você souber o ID do certificado, este comando mostrará informações mais detalhadas sobre o certificado.

```
aws iot describe-certificate --certificate-id "certificateId"
```

- Revise o certificado no AWS IoT console

  No console do AWS IoT, no menu esquerdo, selecione Segurança e, depois, selecione Certificados.

  Escolha o certificado que você está usando para se conectar na lista para abrir sua página de detalhes.

  Na página de detalhes do certificado, você pode ver seu status atual.

  O status do certificado pode ser alterado usando o menu Ações no canto superior direito da página de detalhes.

Autorização

AWS IoT recursos usados AWS IoT Core políticas para autorizar esses recursos a realizar ações. Para que uma ação seja autorizada, os AWS IoT recursos especificados devem ter um documento de política anexado que conceda permissão para realizar essa ação.

Recebi uma resposta PUBNACK ou SUBNACK do operador. O que devo fazer?

Verifique se há uma política anexada ao certificado que você está usando para ligar AWS IoT. Todas as publish/subscribe operações são negadas por padrão.

Certifique-se de que a política anexada autorize as ações que você está tentando realizar.

Certifique-se de que a política anexada autorize os recursos que estão tentando executar as ações autorizadas.

Eu tenho uma entrada AUTHORIZATION_FAILURE em meus logs.

Verifique se há uma política anexada ao certificado que você está usando para ligar AWS IoT. Todas as publish/subscribe operações são negadas por padrão.

Certifique-se de que a política anexada autorize as ações que você está tentando realizar.

Certifique-se de que a política anexada autorize os recursos que estão tentando executar as ações autorizadas.

Como faço para verificar o que a política autoriza?

No console do AWS IoT, no menu esquerdo, selecione Segurança e, depois, escolha Certificados.

Escolha o certificado que você está usando para se conectar na lista para abrir sua página de detalhes.

Na página de detalhes do certificado, você pode ver seu status atual.

No menu esquerdo da página de detalhes do certificado, selecione Políticas para visualizar as políticas que estão anexadas ao certificado.

Escolha a política desejada para ver sua página de detalhes.

Na página de detalhes da política, revise o Documento da política para ver o que ele autoriza.

Escolha Editar documento de política para fazer alterações no documento de política.

Segurança e identidade

Quando você fornece os certificados do servidor para configuração de domínio AWS IoT personalizada, os certificados têm no máximo quatro nomes de domínio.

Para saber mais, consulte Endpoints e cotas do AWS IoT Core.

Diagnosticar problemas de regras

Ajude-nos a melhorar este tópico

Conte para nós o que ajudaria a torná-lo melhor

Esta seção descreve algumas objetos a serem verificadas quando você encontra um problema com a regra.

Configurando CloudWatch registros para solução de problemas

A melhor maneira de depurar problemas que você está tendo com as regras é usar o CloudWatch Logs. Ao ativar CloudWatch os registros para AWS IoT, você pode ver quais regras são acionadas e seu sucesso ou fracasso. Você também obtém informações que indicam se as condições da cláusula WHERE são correspondentes. Para obter mais informações, consulte Monitore AWS IoT usando CloudWatch registros.

O problema mais comum de regras é a autorização. Os registros mostram se sua função não está autorizada a atuar AssumeRole no recurso. Este é um log de exemplo gerado por log refinado:

```
{
    "timestamp": "2017-12-09 22:49:17.954",
    "logLevel": "ERROR",
    "traceId": "ff563525-6469-506a-e141-78d40375fc4e",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleExecution",
    "clientId": "iotconsole-123456789012-3",
    "topicName": "test-topic",
    "ruleName": "rule1",
    "ruleAction": "DynamoAction",
    "resources": {
        "ItemHashKeyField": "id",
        "Table": "trashbin",
        "Operation": "Insert",
        "ItemHashKeyValue": "id",
        "IsPayloadJSON": "true"
    },
    "principalId": "ABCDEFG1234567ABCD890:outis",
    "details": "User: arn:aws:sts::123456789012:assumed-role/dynamo-
testbin/5aUMInJH is not authorized to perform: dynamodb:PutItem on
 resource: arn:aws:dynamodb:us-east-1:123456789012:table/testbin (Service:
 AmazonDynamoDBv2; Status Code: 400; Error Code: AccessDeniedException; Request ID:
 AKQJ987654321AKQJ123456789AKQJ987654321AKQJ987654321)"
}
```

Este é um log de exemplo semelhante gerado por log global:

```
2017-12-09 22:49:17.954 TRACEID:ff562535-6964-506a-e141-78d40375fc4e
PRINCIPALID:ABCDEFG1234567ABCD890:outis [ERROR] EVENT:DynamoActionFailure
TOPICNAME:test-topic CLIENTID:iotconsole-123456789012-3
```

```
MESSAGE:Dynamo Insert record failed. The error received was User:
 arn:aws:sts::123456789012:assumed-role/dynamo-testbin/5aUMInJI is not authorized to
 perform: dynamodb:PutItem on resource: arn:aws:dynamodb:us-east-1:123456789012:table/
testbin
(Service: AmazonDynamoDBv2; Status Code: 400; Error Code: AccessDeniedException;
 Request ID: AKQJ987654321AKQJ987654321AKQJ987654321AKQJ987654321).
Message arrived on: test-topic, Action: dynamo, Table: trashbin, HashKeyField: id,
 HashKeyValue: id, RangeKeyField: None, RangeKeyValue: 123456789012
No newer events found at the moment. Retry.
```

Para obter mais informações, consulte the section called “Visualizando AWS IoT registros no CloudWatch console”.

Diagnosticando serviços externos

Os serviços externos são controlados pelo usuário final. Antes da execução da regra, certifique-se de que os serviços externos vinculados à sua regra estejam configurados e tenham unidades de throughput e capacidade suficientes para seu aplicativo.

Diagnosticar problemas de SQL

Se sua consulta SQL não estiver retornando os dados esperados:

- Examine os logs em busca de mensagens de erro.
- Confirme se sua sintaxe SQL corresponde ao documento JSON na mensagem.

  Analise os nomes de objetos e propriedades usados na consulta com aqueles usados no documento JSON da carga útil da mensagem do tópico. Para obter mais informações sobre a formatação JSON em consultas SQL, consulte Extensões JSON.
- Verifique se os nomes do objeto ou da propriedade JSON incluem caracteres reservados ou numéricos.

  Para obter mais informações sobre caracteres reservados em referências de objetos JSON em consultas SQL, consulte Extensões JSON.

Diagnosticar problemas com shadows

Ajude-nos a melhorar este tópico

Conte para nós o que ajudaria a torná-lo melhor

Diagnosticar shadows

Problema	Diretrizes de solução de problemas
O documento de shadow de um dispositivo é rejeitado com `Invalid JSON document`.	Se você não está familiarizado com o JSON, modifique os exemplos fornecidos neste guia para o seu próprio uso. Para obter mais informações, consulte Exemplos de documentos de sombra.
Enviei o JSON correto, mas apenas partes dele (ou nenhuma parte) estão armazenadas no documento de shadow de dispositivo.	Verifique se você está seguindo as diretrizes de formatação JSON. Somente os campos JSON nas seções `desired` e `reported` são armazenados. Conteúdo JSON (mesmo que formalmente correto) fora dessas seções é ignorado.
Eu recebi uma mensagem de erro informando que a shadow de dispositivo excede o tamanho permitido.	A Sombra do Dispositivo oferece suporte somente a 8 KB de dados. Tente reduzir os nomes de campos dentro do seu documento JSON ou simplesmente criar mais shadows criando mais objetos. Um dispositivo pode ter um número ilimitado de things/shadows associados a ele. O único requisito é que cada nome de objeto seja exclusivo na sua conta.
Quando eu recebo uma shadow de dispositivo, ela é maior que 8 KB. Como isso pode acontecer?	Após o recebimento, o AWS IoT serviço adiciona metadados à sombra do dispositivo. O serviço inclui esses dados em sua resposta, mas não é contabilizado para o limite de 8 KB. Somente os dados dos estados `desired` e

Problema	Diretrizes de solução de problemas
	`reported` dentro do documento de estado enviado à shadow do dispositivo são contabilizados para o limite.
Minha solicitação foi rejeitada devido à versão incorreta. O que devo fazer?	Execute uma operação GET para sincronizar com a versão mais recente do documento de estado. Ao usar MQTT, assine o tópico /update/accepted para ser notificado sobre alterações no estado e receber a versão mais recente do documento JSON.
O carimbo de data e hora é desativado por alguns segundos.	O carimbo de data/hora para campos individuais e todo o documento JSON é atualizado quando o documento é recebido pelo AWS IoT serviço ou quando o documento estadual é publicado no. /update/accepted and ./update/deltamensagem. As mensagens podem ser atrasadas na rede, o que pode fazer com que o carimbo de data e hora fique desativado por alguns segundos.
Meu dispositivo pode publicar e se inscrever nos tópicos correspondentes do Shadow, mas quando eu tento atualizar o documento do Shadow pela API REST HTTP, recebo o erro HTTP 403.	Certifique-se de ter criado políticas no IAM para permitir o acesso a esses tópicos e à ação (UPDATE/GET/DELETE) correspondente às credenciais que você está usando. As políticas do IAM e as políticas de certificado são independentes.
Outros problemas.	O serviço Device Shadow registra erros no CloudWatch Logs. Para identificar problemas de dispositivo e configuração, ative CloudWatch os registros e visualize os registros para obter informações de depuração.

Diagnosticar problemas de ação no fluxo de entrada da Salesforce IoT

Ajude-nos a melhorar este tópico

Conte para nós o que ajudaria a torná-lo melhor

Rastreamento da execução

Como posso ver o rastreamento da execução de uma ação da Salesforce?

Consulte a seção Monitore AWS IoT usando CloudWatch registros. Depois que os logs forem ativados, você poderá ver o rastreamento da execução da ação da Salesforce.

Sucesso e falha da ação

Como faço para verificar se as mensagens foram enviadas com sucesso para um fluxo de entrada da Salesforce IoT?

Visualize os registros gerados pela execução da ação do Salesforce em CloudWatch Registros. Se você ver`Action executed successfully`, isso significa que o mecanismo de AWS IoT regras recebeu a confirmação do Salesforce IoT de que a mensagem foi enviada com sucesso para o fluxo de entrada de destino.

Se houver problemas com a plataforma da Salesforce IoT, entre em contato com o suporte da Salesforce IoT.

O que eu faço se as mensagens não foram enviadas com sucesso para o fluxo de entrada da Salesforce IoT?

Visualize os registros gerados pela execução da ação do Salesforce em CloudWatch Registros. Dependendo da entrada do log, você pode tentar as seguintes ações:

`Failed to locate the host`

Verifique se o parâmetro `url` da ação está correto e se o fluxo de entrada da Salesforce IoT existe.

`Received Internal Server Error from Salesforce`

Tentar novamente. Se o problema persistir, entre em contato com o suporte da Salesforce IoT.

`Received Bad Request Exception from Salesforce`

Verifique se há erros na carga que você está enviando.

`Received Unsupported Media Type Exception from Salesforce`

A Salesforce IoT não oferece suporte a uma carga binária no momento. Verifique se você está enviando uma carga JSON.

`Received Unauthorized Exception from Salesforce`

Verifique se o parâmetro `token` da ação está correto e se o token ainda é válido.

`Received Not Found Exception from Salesforce`

Verifique se o parâmetro `url` da ação está correto e se o fluxo de entrada da Salesforce IoT existe.

Se você receber um erro que não esteja listado aqui, entre em contato com o AWS IoT Support.

Diagnosticando limites de fluxo

Solução de problemas "Limite de transmissão excedido para sua conta AWS "

Se você vir `"Error: You have exceeded the limit for the number of streams in your AWS account."`, poderá limpar os fluxos não utilizados em sua conta em vez de solicitar um aumento de limite.

Para limpar um stream não utilizado que você criou usando o AWS CLI ou SDK:

```
aws iot delete-stream –stream-id value
```

Para obter mais detalhes, consulte delete-stream.

> **Note**
>
> Você pode usar o `list-streams` comando para encontrar o stream IDs.

Solução de problemas de desconexões da frota de dispositivos

Ajude-nos a melhorar este tópico

Conte para nós o que ajudaria a torná-lo melhor

AWS IoT as desconexões da frota de dispositivos podem ocorrer por vários motivos. Este artigo explica como diagnosticar um motivo de desconexão e como lidar com desconexões causadas pela manutenção regular do AWS IoT serviço ou por um limite de limitação.

Para diagnosticar o motivo da desconexão

Você pode verificar o grupo de registros AWSIotLogSv2 CloudWatchpara identificar o motivo da desconexão no `disconnectReason` campo da entrada de registro.

Você também pode usar o recurso de eventos AWS IoT do ciclo de vida para identificar o motivo da desconexão. Se você se inscreveu no evento de desconexão (`$aws/events/presence/disconnected/clientId`) do ciclo de vida, receberá uma notificação AWS IoT quando a desconexão ocorrer. Você pode identificar o motivo da desconexão no campo `disconnectReason` da notificação.

Para obter mais informações, consulte entradas de CloudWatch AWS IoT registro e eventos do ciclo de vida.

Para solucionar problemas de desconexões devido à AWS IoT manutenção do serviço

As desconexões causadas pela manutenção AWS IoT do serviço são registradas como `SERVER_INITIATED_DISCONNECT` no evento AWS IoT do ciclo de vida e. CloudWatch Para lidar com essas desconexões, ajuste a configuração do lado do cliente para garantir que seus dispositivos possam ser reconectados automaticamente à plataforma. AWS IoT

Para solucionar problemas de desconexões devido a um limite de controle de utilização

As desconexões causadas por um limite de limitação são registradas como `THROTTLED` no evento do ciclo AWS IoT de vida e. CloudWatch Para lidar com essas desconexões, você pode solicitar aumentos no limite do agente de mensagens à medida que a contagem de dispositivos aumenta.

Para obter mais informações, consulte Agente de mensagens Core AWS IoT.

AWS IoT Device Management guia de solução de problemas

> Ajude-nos a melhorar este tópico
>
> Conte para nós o que ajudaria a torná-lo melhor

Esta é a seção de solução de problemas do AWS IoT Device Management.

Tópicos

- AWS IoT Solução de problemas de trabalhos
- Solução de problemas de indexação de frota
- AWS IoT Solução de problemas do Device Management Software Package Catalog

AWS IoT Solução de problemas de trabalhos

Esta é a seção de solução de problemas para AWS IoT trabalhos.

Como localizo um endpoint do AWS IoT Jobs?

Como localizo o endpoint do plano AWS IoT de controle de Jobs?

AWS IoT O Jobs oferece suporte a operações de API do plano de controles usando o protocolo HTTPS. Verifique se você se conectou ao endpoint do ambiente de gerenciamento correto usando o protocolo HTTPS.

Para obter uma lista de endpoints AWS específicos da região, consulte AWS IoT Core - endpoints do plano de controle.

Para obter uma lista de endpoints do ambiente de gerenciamento de trabalhos de AWS IoT compatíveis com FIPS, consulte Endpoints FIPS por serviço

> Note
>
> AWS IoT Trabalhe e AWS IoT Core compartilhe os mesmos endpoints AWS específicos da região.

Como localizo o endpoint do plano AWS IoT de dados do Jobs?

AWS IoT O Jobs suporta operações de API de plano de dados usando os protocolos HTTPS e MQTT. Verifique se você se conectou ao endpoint correto do plano de dados usando o protocolo HTTPS ou MQTT.

- Protocolo HTTPS
  - Use o seguinte comando da describe-endpointCLI mostrado abaixo ou a API `DescribeEndpoint`REST. Para o tipo de endpoint, use `iot:Jobs`.

    ```
    aws iot describe-endpoint --endpoint-type iot:Jobs
    ```

- Protocolo MQTT
  - Use o seguinte comando da describe-endpointCLI mostrado abaixo ou a API `DescribeEndpoint`REST. Para o tipo de endpoint, use `iot:Data-ATS`.

    ```
    aws iot describe-endpoint --endpoint-type iot:Data-ATS
    ```

Para obter uma lista de endpoints do plano de dados de trabalhos de AWS IoT compatíveis com FIPS, consulte Endpoints FIPS por serviço

Como faço para monitorar a atividade do AWS IoT Jobs e fornecer métricas?

O monitoramento da atividade de AWS IoT trabalhos usando a Amazon CloudWatch fornece visibilidade em tempo real das operações contínuas de AWS IoT trabalhos e ajuda a controlar os custos com CloudWatch alarmes por meio de AWS IoT regras. Você deve configurar o registro antes de poder monitorar a atividade do AWS IoT Jobs e configurar CloudWatch alarmes. Para obter mais informações sobre como configurar o log, consulte Configurar o AWS IoT registro.

Para obter mais informações sobre a Amazon CloudWatch e como configurar a permissão por meio de uma função de usuário do IAM para usar CloudWatch recursos, consulte Gerenciamento de identidade e acesso para a Amazon CloudWatch.

Como faço para configurar as métricas e o monitoramento de AWS IoT empregos usando a Amazon CloudWatch?

Para configurar o AWS IoT registro, siga as etapas descritas em Configurar o AWS IoT registro. AWS IoT a configuração do registro pode ser feita na API Console de gerenciamento da AWS

AWS CLI, ou. AWS IoT o registro configurado para grupos de coisas específicos deve ser feito somente na API AWS CLI ou.

A seção Métricas de AWS IoT trabalhos contém as métricas de AWS IoT trabalhos usadas para monitorar a atividade de AWS IoT trabalhos. Ele explica como visualizar as métricas no Console de gerenciamento da AWS AWS CLI e.

Além disso, você pode configurar CloudWatch alarmes para alertá-lo sobre métricas específicas que você deseja monitorar de perto. Para obter orientação sobre a configuração de alarmes, consulte Como usar CloudWatch alarmes da Amazon.

Frotas de dispositivos e solução de problemas de dispositivos únicos

A execução de um trabalho mantém um status **QUEUED** indefinidamente

Quando uma execução de trabalho com um estado de status `QUEUED` não prossegue para o próximo estado de status lógico, como `IN_PROGRESS`, `FAILED`, ou `TIMED_OUT` um dos seguintes cenários pode ser a causa:

- Revise a atividade do seu dispositivo nos CloudWatch registros localizados no CloudWatch console. Para obter mais informações, consulte Monitorar AWS IoT usando CloudWatch registros.
- O perfil do IAM associado ao trabalho e à execução subsequente do trabalho pode não ter as permissões corretas listadas em uma das declarações de política do IAM anexadas a esse perfil do IAM. Use a API `describe-job` para identificar o perfil do IAM vinculado a esse trabalho e à execução subsequente do trabalho e revise a política do IAM para obter as permissões corretas. Depois que as declarações de permissão da política forem atualizadas, você poderá executar o comando da API `AssumeRole` no recurso.

Uma execução de trabalho não foi criada para meu objeto ou grupo de objetos

Quando um trabalho atualiza seu status para `IN_PROGRESS`, a distribuição do documento do trabalho será iniciada em todos os dispositivos do seu grupo de destino. Essa atualização do estado de status criará uma execução de trabalho para cada dispositivo de destino. Se a execução de uma tarefa não tiver sido criada para um dos dispositivos de destino, consulte as diretrizes a seguir:

- A `thing` é direcionada diretamente pelo trabalho, o trabalho tem um estado de status `IN_PROGRESS` e o trabalho é simultâneo? Se todas as três condições forem atendidas, o

trabalho ainda estará enviando execuções de trabalho para todos os dispositivos do seu grupo de destino e essa `thing` específica ainda não recebeu sua execução de trabalho.

- Analise os dispositivos em seu grupo-alvo para o trabalho e o estado do status do trabalho no AWS Management Console ou use o comando da `describe-job`API.
- Use o comando da API `describe-job` para verificar se o trabalho tem a propriedade `IsConcurrent` definida como verdadeira ou falsa. Para obter mais informações, consulte Job limits.

- A `thing` não é alvo direto do trabalho.
  - Se a `Thing` foi adicionada a um `ThingGroup` e o trabalho foi direcionado ao `ThingGroup`, verifique se a `Thing` faz parte do `ThingGroup`.
  - Se o trabalho for um trabalho de instantâneo com um estado de status `IN_PROGRESS` e for simultâneo, o trabalho ainda estará enviando execuções de trabalho para todos os dispositivos em seu grupo de destino e essa `Thing` específica ainda não recebeu sua execução de trabalho.
  - Se o trabalho for um trabalho contínuo com um estado de status `IN_PROGRESS` e for simultâneo, o trabalho ainda estará enviando execuções de trabalho para todos os dispositivos em seu grupo de destino e essa `Thing` específica ainda não recebeu sua execução de trabalho. Somente para trabalhos contínuos, você também pode remover a `Thing` do `ThingGroup` e depois adicionar a `Thing` de volta ao `ThingGroup`.
  - Se o trabalho for um trabalho instantâneo com um estado de status `IN_PROGRESS` e não for simultâneo, é provável que a relação de `ThingGroup` associação `Thing` ou associação não seja reconhecida por Jobs. AWS IoT É recomendável adicionar alguns segundos de tempo de espera após a chamada `AddThingToThingGroup` antes de criar seu `Job`. Como alternativa, você pode alternar a seleção do alvo para `Continuous`, fazendo com que o serviço preencha o evento de anexo de associação `Thing` e `ThingGroup` atrasado.

O novo trabalho falha devido a um erro **`LimitedExceededException`**

Se a criação do seu trabalho falhar com uma resposta de erro `LimitedExceededException`, chame a API `list-jobs` e revise todos os trabalhos com `isConcurrent=true` para determinar se você está no limite de simultaneidade do trabalho. Consulte Limites de trabalho para obter informações adicionais sobre trabalhos simultâneos. Para ver os limites de simultaneidade de trabalho e solicitar um aumento de limite, consulte Limites de trabalhos AWS IoT Device Management e cotas.

Limite de tamanho do documento de trabalho

O tamanho do documento de trabalho é limitado pelo tamanho da carga útil do MQTT. Se você precisar de um documento de trabalho maior que 32 kB (kilobytes), 32.000 B (bytes), crie e armazene o documento de trabalho no Amazon S3 e adicione um URL de objeto do Amazon S3 no campo `documentSource` para a API `CreateJob` ou usando o AWS CLI. Para o Console de gerenciamento da AWS, adicione uma URL de objeto do Amazon S3 na caixa de texto URL do Amazon S3 ao criar um trabalho.

- Console de gerenciamento da AWS criar documentação do trabalho: Crie e gerencie trabalhos usando o Console de gerenciamento da AWS
- AWS CLI criar documentação do trabalho: Crie e gerencie trabalhos usando o AWS CLI
- `CreateJob`Documentação da API: CreateJob

A mensagem MQTT do lado do dispositivo solicita limites de controle de utilização

Se você receber um código de erro 400 `ThrottlingException`, a mensagem MQTT do lado do dispositivo falhou devido ao alcance do limite de solicitações simultâneas do lado do dispositivo. Consulte Limites e cotas de trabalhos AWS IoT Device Management para obter mais informações sobre limites de controle de utilização e se eles são ajustáveis.

Erro de tempo limite de conexão

Um código de erro 400 `RequestExpired` indica uma falha de conexão devido a alta latência ou baixos valores de tempo limite do lado do cliente.

- Consulte Teste de conectividade com o endpoint de dados do dispositivo para obter informações sobre como testar a conexão entre o lado do cliente e o lado do servidor.

Comando de API inválido

Confirme se o comando API correto foi inserido para evitar uma mensagem de erro informando que o comando API é inválido. Consulte a Referência da API AWS IoT para obter uma lista abrangente de todos os comandos da API AWS IoT .

Erro de conexão do lado do serviço

Um código de erro 503 `ServiceUnavailable` indica que o erro foi originado do lado do servidor.

- Consulte AWS Health Dashboard (todos os AWS serviços) para ver o status atual de todos os AWS serviços.
- Consulte AWS Health Dashboard (pessoal Conta da AWS) para obter o status atual de sua conta pessoal Conta da AWS.

Solução de problemas de indexação de frota

Solução de problemas de consultas de agregação para o serviço de indexação de frota

Se você estiver tendo erros de incompatibilidade de tipos, você pode usar o CloudWatch Logs para solucionar o problema. CloudWatch Os registros devem ser ativados antes que os registros sejam gravados pelo serviço de indexação de frotas. Para obter mais informações, consulte Monitore AWS IoT usando CloudWatch registros.

Para fazer consultas de agregação em campos não gerenciados, você deve especificar um campo definido no argumento `customFields` passado para `UpdateIndexingConfiguration` ou update-indexing-configuration. Se o valor do campo for inconsistente com o tipo de dados de campo configurado, esse valor será ignorado quando você executar uma consulta de agregação.

Se um campo não puder ser indexado devido a um tipo incompatível, o serviço de indexação de frotas envia um registro de erros para o Logs. CloudWatch O log de erros contém o nome do campo, o valor que não foi convertido e o nome do objeto para o dispositivo. Veja a seguir um exemplo de log de erros:

```
{
  "timestamp": "2017-02-20 20:31:22.932",
  "logLevel": "ERROR",
  "traceId": "79738924-1025-3a00-a669-7bec69f7f07a",
  "accountId": "000000000000",
  "status": "SucceededWithIssues",
  "eventType": "IndexingCustomFieldFailed",
  "thingName": "thing0",
  "failedCustomFields": [
    {
      "Name": "attributeName1",
      "Value": "apple",
      "ExpectedType": "String"
    },
```

```
    {
      "Name": "attributeName2",
      "Value": "2",
      "ExpectedType": "Boolean"
    }
  ]
}
```

Se um dispositivo foi desconectado por aproximadamente uma hora, o valor `timestamp` do status de conectividade pode estar ausente. Para sessões persistentes, o valor pode estar ausente após um cliente ter sido desconectado por mais tempo do que o configurado time-to-live (TTL) para a sessão persistente. Os dados de status de conectividade são indexados apenas para conexões em que o ID do cliente tem um nome de objeto correspondente. (O ID do cliente é o valor usado para conectar um dispositivo AWS IoT Core a.)

Solução de problemas de configuração de indexação de frota

Não é possível fazer o downgrade da configuração de indexação da frota

O downgrade da configuração de indexação de frota não é suportado quando você deseja remover as fontes de dados associadas a uma métrica de frota ou a um grupo dinâmico.

Por exemplo, se a sua configuração de indexação tiver dados de registro, dados de sombra e dados de conectividade, e existir uma métrica de frota com a consulta `thingName:TempSensor* AND shadow.desired.temperature>80`, atualizar a configuração de indexação para incluir apenas os dados de registro resultará em um erro.

A modificação de campos personalizados usados pelas métricas de frota existentes não é suportada.

Não é possível atualizar sua configuração de indexação devido a métricas de frota ou grupos dinâmicos incompatíveis

Se você não conseguir atualizar sua configuração de indexação devido a métricas de frota ou grupos dinâmicos incompatíveis, exclua as métricas de frota ou grupos dinâmicos incompatíveis antes de atualizar a configuração de indexação.

Solução de problemas de indexação de localização e consultas geográficas

Para solucionar erros de tipos incompatíveis na indexação de localização e nas consultas geográficas, você pode ativar os registros. CloudWatch Para obter mais informações sobre como monitorar AWS IoT o uso CloudWatch, siga o step-by-step guia.

Quando você indexa dados de localização usando consultas geográficas, os campos de localização especificados em `geoLocations` devem corresponder aos campos de localização para os quais você passa para `UpdateIndexingConfiguration`. Se houver uma incompatibilidade, a indexação da frota envia um erro de tipo incompatível para. CloudWatch O log de erros contém o nome do campo, o valor que não foi convertido e o nome do objeto para o dispositivo.

Veja a seguir um exemplo de log de erros:

```
{
"timestamp": "2023-11-09 01:39:43.466",
    "logLevel": "ERROR",
    "traceId": "79738924-1025-3a00-a669-7bec69f7f07a",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "IndexingGeoLocationFieldFailed",
    "thingName": "thing0",
    "failedGeolocationFields": [
        {
"Name": "attributeName1",
            "Value": "apple",
            "ExpectedType": "Geopoint"
        }
    ],
    "reason": "failed to index the field because it could not be converted to one of
 the expected geoLocation formats."
}
```

Para obter mais informações, consulte Indexação de dados de localização.

Solução de problemas de métricas de frota

Não consigo ver pontos de dados em CloudWatch

Se você conseguir criar uma métrica de frota, mas não conseguir ver os pontos de dados CloudWatch, é provável que não tenha nada que atenda aos critérios da sequência de caracteres de consulta.

Veja este exemplo de comando de como criar uma métrica de frota:

```
aws iot create-fleet-metric --metric-name "example_FM" --query-string
 "thingName:TempSensor* AND attributes.temperature>80" --period 60 --aggregation-field
 "attributes.temperature" --aggregation-type name=Statistics,values=count
```

Se você não tiver nada que atenda aos critérios da string de consulta de consulta `--query-string "thingName:TempSensor* AND attributes.temperature>80"`:

- Com isso`values=count`, você poderá criar uma métrica de frota e haverá pontos de dados para mostrar CloudWatch. Os pontos de dados do valor `count` são sempre 0.
- Com `values` outras opções`count`, você poderá criar uma métrica de frota, mas não verá a métrica da frota CloudWatch e não haverá pontos de dados para mostrar CloudWatch.

AWS IoT Solução de problemas do Device Management Software Package Catalog

Esta é a seção de solução de problemas do AWS IoT Device Management Software Package Catalog.

Mensagens de erro de solução de problemas geral

Esta seção lista os erros comuns observados em todo o ciclo de vida da versão do pacote de software.

Erros de **`HeadBucket`**

As mensagens de erro a seguir aparecem ao chamar a operação da API `HeadBucket` ou o comando da CLI `head-bucket` para validar o bucket do Amazon S3 usado para upload de arquivos durante a implantação de um trabalho.

Para obter mais informações sobre como usar um bucket do Amazon S3 para fazer o upload de arquivos durante uma implantação de trabalho, consulte URL pré-assinada para upload de arquivo.

`InvalidRoleException`

```
    "Permission denied when attempting to use role %s to access bucket %s."
```

`InvalidRequestException`

```
    "Cross region S3 bucket is not supported for presigned url upload placeholder"
```

`InvalidRequestException`

```
    "S3 bucket in job document presigned url upload placeholder not found"
```

`InvalidRequestException`

```
    "Given S3 bucket name is invalid."
```

```
InvalidRequestException
    "Provided S3 bucket is not valid: %s. Error: %s"
```

Amazon S3 GetObject

A mensagem de erro a seguir ocorre quando um argumento inválido é fornecido, fazendo com que a operação da API `GetObject` do Amazon S3 falhe.

```
InvalidRequestException
    "Provided argument for presigned url is invalid"
```

Suporte ao ID de versão do Amazon S3

Ao solicitar acesso a um bucket do Amazon S3 usando o versionamento, inclua o `versionId` ou o erro abaixo poderá ser preenchido.

Para obter mais informações sobre os buckets do Amazon S3 usando versionamento, consulte Usar versionamento em buckets do Amazon S3

```
InvalidRequestException
    "VersionId not found when attempting to access s3 url"
```

Espaços reservados dentro de um URL pré-assinado para upload de arquivo

As mensagens de erro a seguir aparecem ao encontrar problemas com um espaço reservado dentro de um URL pré-assinado usado para carregar arquivos em um bucket do Amazon S3 de destino durante a implantação de um trabalho. Para obter mais informações sobre como usar um bucket do Amazon S3 para fazer o upload de arquivos durante uma implantação de trabalho e sobre o que é um espaço reservado, consulte URL pré-assinada para upload de arquivo.

A mensagem de erro abaixo aparece quando o espaço reservado local não é reconhecido.

```
InvalidJobDocumentException
    "Undefined placeholder, ${...}, inside of presign url upload parameter"
```

A mensagem de erro abaixo aparece ao tentar usar o espaço reservado local em um URL pré-assinado não destinado ao upload de arquivo.

```
InvalidJobDocumentException
```

```
    "Local placeholder, ${...}, is only valid inside of presign url upload"
```

URL do Amazon S3 aninhado incorretamente

A mensagem de erro a seguir aparece quando a URL do Amazon S3 está incorretamente aninhada em outro espaço reservado.

```
InvalidJobDocumentException
    "${aws:%s[...]} should not be the second layer pattern."
```

Aninhamento de artefato de versão do pacote

A mensagem de erro a seguir aparece quando o URL pré-assinado de artefato de versão do pacote está incorretamente aninhada em outro espaço reservado.

```
InvalidJobDocumentException
    "${aws:iot:package:[...]:artifact:s3-presigned-url} cannot be nested inside another
 placeholder."
```

Artefato de versão do pacote ausente

A mensagem de erro a seguir aparece quando o artefato da versão do pacote referenciado não é encontrado.

```
InvalidJobDocumentException
    "Package %s version %s does not have an associated artifact to generate an S3
 presigned url."
```

Espaços reservados para pacote de software e versão do pacote

A mensagem de erro a seguir aparece quando o espaço reservado do documento de trabalho para o pacote de software e a versão do pacote não consegue resolver os valores válidos desejados para a implantação do trabalho porque vários pacotes de software e versões de pacotes são referenciados no parâmetro `destinationPackageVersions` ou na guia ARN da versão na página de detalhes de Versão do pacote.

```
InvalidJobDocumentException
    "Cannot resolve empty package name and version name given multiple elements in
 destination package versions."
```

Usar o pacote de software vazio e a versão do pacote

A mensagem de erro a seguir aparece quando você tenta usar um pacote vazio ou uma versão de pacote sem a outra em um documento de trabalho.

```
InvalidJobDocumentException
    "Empty package name and version name have to be used in pair."
```

Uso de Null no documento de trabalho

A mensagem de erro a seguir aparece quando você tenta especificar `$null` como uma versão do pacote no documento de trabalho. `$null` só pode ser usado dentro do parâmetro `destinationPackageVersions` ao usar a operação da API `CreateJob`.

```
InvalidJobDocumentException
    "$null is not allowed to be referenced as a package version in job documents."
```

Todos os atributos em uma versão do pacote

A mensagem de erro a seguir aparece quando você tenta usar todos os atributos em uma versão do pacote e envolvê-la com espaços reservados ou textos adicionais.

Para obter mais informações sobre o uso de todos os atributos em uma versão de pacote de software, consulte Parâmetros de substituição para trabalhos AWS IoT

```
InvalidJobDocumentException
    "The package version attribute placeholder for all attributes has to be a
 json value by itself and not appended with other strings or nested with other
 placeholders."
```

Limite de espaço reservado local em URL pré-assinado para upload de arquivo

A mensagem de erro a seguir aparece quando você excede o limite do número de espaços reservados locais usados em um URL pré-assinado para upload de arquivos durante a implantação de uma tarefa.

Para obter mais informações sobre o uso de um URL pré-assinado para upload de arquivos durante a implantação de uma tarefa, consulte URL pré-assinada para upload de arquivo

```
InvalidJobDocumentException
    "The occurrence of local placeholder %s within S3 presigned url upload placeholder
 exceeds limit of %d."
```

Espaços reservados locais em um bucket do Amazon S3

A mensagem de erro a seguir aparece quando você tenta colocar um URL de espaço reservado local no nome do bucket do Amazon S3 para um espaço reservado de URL pré-assinado usado para upload de arquivos durante a implantação de um trabalho.

Para obter mais informações sobre o uso de um URL pré-assinado para upload de arquivos durante a implantação de uma tarefa, consulte URL pré-assinada para upload de arquivo

```
InvalidJobDocumentException
    "S3 bucket name in presigned url upload is not allowed to contain any placeholders"
```

Suportes de abertura e fechamento

A mensagem de erro a seguir aparece quando você adiciona um parâmetro ou espaço reservado a um documento de trabalho sem a chave de fechamento “}”.

```
InvalidJobDocumentException
    "One or more parameters or placeholders are not terminated."
```

Perfil do IAM com o URL pré-assinado do Amazon S3

A mensagem de erro a seguir aparece quando você tenta usar um URL pré-assinado do Amazon S3 em um documento de trabalho sem um perfil do IAM.

Para obter mais informações sobre o Amazon S3 pré-assinado URLs, consulte Como trabalhar com pré-assinado. URLs

```
InvalidRequestException
    "presignedUrlConfig role ARN is required to generate an S3 presigned url in job
 document."
```

Perfil do IAM com o URL pré-assinado do Amazon S3 para artefato de versão do pacote

A mensagem de erro a seguir aparece quando você tenta usar um URL pré-assinado do Amazon S3 representando um artefato de versão do pacote em um documento de trabalho sem um perfil do IAM.

```
InvalidRequestException
    "presignedUrlConfig role ARN is required to generate an S3 presigned url in job
 document for package %s version %s artifact."
```

Mensagens de erro da lista de materiais de software

Esta seção lista os erros comuns associados a uma lista de materiais de software (SBOM) vinculada a uma versão do pacote.

Validação de entrada para solicitação de associação da SBOM

A mensagem de erro a seguir aparece ao usar a operação `AssociateSbomWithPackageVersion` da API e o parâmetro `s3Location` é nulo.

```
InvalidRequestException "Associate request needs to include SBOM reference"
```

Para obter mais informações sobre a operação `AssociateSbomWithPackageVersion` da API, consulte AssociateSbomWithPackageVersion.

Erros de validação da SBOM

Esta seção lista os erros comuns observados durante a validação inicial da lista de materiais do software (SBOM) quando associada a uma versão do pacote de software.

A mensagem de erro a seguir aparece ao usar a operação da API `AssociateSbomWithPackageVersion` e `bucket` no parâmetro `s3Location` é nulo.

```
InvalidRequestException "S3 bucket name for SBOM cannot be null"
```

A mensagem de erro a seguir aparece quando a sequência de caracteres `bucket` no parâmetro `s3Location` da operação da API `AssociateSbomWithPackageVersion` é longa demais.

```
InvalidRequestException "S3 bucket name for SBOM is illegal. String length exceeds
 limit"
```

A mensagem de erro a seguir aparece quando o parâmetro `key` é nulo.

```
InvalidRequestException "S3 key name for SBOM cannot be null"
```

A mensagem de erro a seguir aparece quando a sequência de caracteres `key` no parâmetro `s3Location` da operação da API `AssociateSbomWithPackageVersion` é longa demais.

```
InvalidRequestException "S3 key name for SBOM is illegal. String length exceeds limit"
```

A mensagem de erro a seguir aparece quando a sequência de caracteres `version` no parâmetro `s3Location` da operação da API `AssociateSbomWithPackageVersion` é nula.

```
InvalidRequestException "S3 object version for SBOM cannot be null"
```

A mensagem de erro a seguir aparece quando a sequência de caracteres `version` no parâmetro `s3Location` da operação da API `AssociateSbomWithPackageVersion` é longa demais.

```
InvalidRequestException "S3 object version for SBOM is illegal. String length exceeds
 limit"
```

A mensagem de erro a seguir aparece quando o tamanho do arquivo zip da SBOM armazenado no bucket do Amazon S3 é grande demais.

```
InvalidRequestException "S3 object file size exceeds limit"
```

A mensagem de erro a seguir aparece quando você usa a operação da API `AssociateSbomWithPackageVersion` e o número atual de validações da SBOM em andamento já está no limite máximo.

```
LimitExceededException "Too many ongoing SBOM validation workflows. Please wait and
 retry"
```

Problemas de acesso com o arquivo da SBOM no bucket do Amazon S3

A mensagem de erro a seguir aparece quando outra entidade não consegue acessar o bucket do Amazon S3 porque o bucket do Amazon S3 não existe ou as permissões adequadas não foram concedidas para acessar o bucket do Amazon S3.

Para obter mais informações sobre a política de permissões necessárias para acessar um bucket do Amazon S3, consulte Armazenamento da lista de materiais de software.

```
InvalidRequestException "SBOM not accessible by the service. Please make sure the
 bucket exists and S3 permission is granted."
```

A mensagem de erro a seguir aparece quando outra entidade não consegue acessar o arquivo zip da SBOM no parâmetro `key` porque o bucket do Amazon S3 não existe ou as permissões adequadas não foram concedidas para acessar o conteúdo armazenado no bucket do Amazon S3.

```
InvalidRequestException "SBOM not accessible by the service. Please make sure the key
 exists and S3 permission is granted."
```

A mensagem de erro a seguir aparece quando outra entidade não consegue acessar o bucket do Amazon S3 porque o bucket, a chave e o ID da versão não existem ou as permissões adequadas não foram concedidas para acessar o bucket do Amazon S3. Além disso, essa mensagem de erro poderá aparecer se as permissões concedidas forem insuficientes para acessar o arquivo zip da SBOM no bucket do Amazon S3.

```
InvalidRequestException "SBOM not accessible by the service. Please make sure the
 bucket/key/version exists and S3 permission is granted."
```

A mensagem de erro a seguir aparece quando outra entidade não consegue acessar o bucket do Amazon S3 porque o bucket está localizado em outra região.

```
InvalidRequestException "Cross-region S3 bucket for %s is not supported."
```

A mensagem de erro a seguir aparece quando outra entidade não consegue acessar o bucket do Amazon S3 porque os parâmetros `bucket`, `key` ou `version` foram inseridos incorretamente ao usar a operação da API `AssociateSbomWithPackageVersion`.

```
InvalidRequestException "Please make sure SBOM S3 bucket name/key length/version is
 valid"
```

AWS IoT Guia de solução de problemas do Device Advisor

Ajude-nos a melhorar este tópico

Conte para nós o que ajudaria a torná-lo melhor

Geral

P: Posso executar vários conjuntos de testes em paralelo?

R: Sim. O Device Advisor agora oferece suporte à execução de vários conjuntos de testes em diferentes dispositivos usando um endpoint no nível do dispositivo. Se você usar o endpoint no

nível da conta, poderá executar um conjunto por vez porque um endpoint do Device Advisor está disponível por conta. Para obter mais informações, consulte Configure seu dispositivo.

P: Vi no meu dispositivo que a conexão TLS foi negada pelo Device Advisor. Isso é esperado?

R: Sim. O Device Advisor nega a conexão TLS antes e depois de cada execução de teste. Recomendamos que os usuários implementem um mecanismo de nova tentativa de dispositivo para ter uma experiência de teste totalmente automatizada com o Device Advisor. Se você executar um conjunto de testes com mais de um caso de teste, por exemplo, conexão TLS, conexão MQTT e publicação MQTT, recomendamos que você tenha um mecanismo criado para seu dispositivo. O mecanismo pode tentar se conectar ao nosso endpoint de teste a cada 5 segundos por um ou dois minutos. Dessa forma, você pode executar vários casos de teste em sequência de maneira automatizada.

P: Posso obter um histórico de chamadas de API do Device Advisor feitas em minha conta para fins de análise de segurança e solução de problemas operacionais?

R: Sim. Para receber um histórico das chamadas da API Device Advisor feitas em sua conta, basta ativar CloudTrail o AWS IoT Management Console e filtrar a origem do evento`iotdeviceadvisor.amazonaws.com`.

P: Como faço para ver os logins do Device Advisor CloudWatch?

R: Os registros gerados durante a execução de uma suíte de testes serão enviados CloudWatch se você adicionar a política necessária (por exemplo, CloudWatchFullAccess) à sua função de serviço (consulteConfigurar). Se houver pelo menos um caso de teste na suíte de testes, um grupo de log "aws/iot/deviceadvisor/$testSuiteId" será criado com dois fluxos de log. Um stream é o “$testRunId" e inclui registros de ações realizadas antes e depois da execução dos casos de teste em sua suíte de testes, como etapas de configuração e limpeza. O outro fluxo de log é “$ suiteRunId _$”testRunId, que é específico para a execução de uma suíte de testes. Eventos enviados de dispositivos e AWS IoT Core serão registrados nesse fluxo de log.

P: Qual é a finalidade da função de permissão do dispositivo?

R: O Device Advisor fica entre seu dispositivo de teste e AWS IoT Core simula cenários de teste. Ele aceita conexões e mensagens dos seus dispositivos de teste e as encaminha para o AWS IoT Core , assumindo a função de permissão do seu dispositivo e iniciando uma conexão em seu nome. É importante garantir que as permissões da função do dispositivo sejam as mesmas do certificado que você usa para executar testes. AWS IoT as políticas de certificado não são aplicadas quando o Device Advisor inicia uma conexão AWS IoT Core em seu nome usando a função de permissão do dispositivo. No entanto, as permissões da função de permissão do dispositivo definidas são aplicadas.

P: Em quais regiões o Device Advisor é suportado?

R: O Device Advisor é compatível com as regiões us-east-1, us-west-2, ap-northeast-1 e eu-west-1.

P: Por que vejo resultados inconsistentes?

R: Uma das principais causas de resultados inconsistentes é definir um teste de `EXECUTION_TIMEOUT` com um valor muito baixo. Para obter mais informações sobre valores de `EXECUTION_TIMEOUT` recomendados e padrão, consulte Casos de teste do Device Advisor.

P: Qual protocolo MQTT é compatível com o Device Advisor?

R: O Device Advisor suporta a versão 3.1.1 do MQTT com certificados de cliente X509.

P: E se meu caso de teste falhar com uma mensagem de tempo limite de execução, mesmo que eu tenha tentado conectar meu dispositivo ao endpoint de teste?

R: Valide todas as etapas em Criar um perfil do IAM para ser usada como sua função de dispositivo. Se o teste ainda falhar, pode ser que o dispositivo não esteja enviando a extensão SNI (Server Name Indication) correta, necessária para o funcionamento do Device Advisor. O valor correto do SNI é o endereço do endpoint retornado ao seguir a seção Configurar seu dispositivo. AWS IoT também exige que os dispositivos enviem a extensão Server Name Indication (SNI) para o protocolo Transport Layer Security (TLS). Para obter mais informações, consulte Segurança de transporte em AWS IoT.

P: Minha conexão MQTT falha com um erro "libaws-c-mqtt: AWS_ERROR _MQTT_UNEXPECTED_HANGUP” (ou) a conexão MQTT do meu dispositivo está sendo automaticamente desconectada do endpoint do Device Advisor. Como esse erro pode ser resolvido?

R: Esse código de erro específico e desconexões inesperadas podem ser causados por muitas objetos diferentes, mas provavelmente estão relacionados à função do dispositivo associada ao dispositivo. Os pontos de verificação abaixo (em ordem de prioridade) resolverão esse problema.

- A função de dispositivo anexada ao dispositivo deve ter as permissões mínimas de IAM necessárias para executar os testes. O Device Advisor usará a função de dispositivo anexada para executar ações AWS IoT MQTT em nome do dispositivo de teste. Se as permissões necessárias estiverem ausentes, o erro `AWS_ERROR_MQTT_UNEXPECTED_HANGUP` será visto ou ocorrerão desconexões inesperadas enquanto o dispositivo tenta se conectar ao endpoint do Device Advisor. Por exemplo, se você optou por executar o caso de teste do MQTT Publish, as ações Connect e Publish devem ser incluídas na função com o correspondente ClientId e o Topic (você pode fornecer vários valores usando vírgulas para separar os valores

e fornecer valores de prefixo usando um caractere curinga (*). Por exemplo: para fornecer permissões para publicar em qualquer tópico que comece com `TestTopic`, você pode fornecer "`TestTopic*`" como valor do recurso. Aqui estão alguns exemplos de políticas.

- Incompatibilidade entre os valores definidos na função do dispositivo para seus tipos de recursos e os valores reais usados no código. Por exemplo: uma incompatibilidade entre a ClientId definição da função e a real ClientId usada no código do seu dispositivo. Valores como ClientId, Tópico e TopicFilter devem ser idênticos na função e no código do dispositivo.
- O certificado do dispositivo anexado ao seu dispositivo deve estar ativo e ter uma política anexada a ele com as permissões de ação necessárias para os recursos. Observe que a política de certificado do dispositivo concede ou nega acesso aos AWS IoT recursos e às operações do plano de AWS IoT Core dados. O Device Advisor exige que você tenha um certificado de dispositivo ativo anexado ao seu dispositivo, que conceda as permissões de ação usadas durante um caso de teste.

AWS IoT erros

Ajude-nos a melhorar este tópico

Conte para nós o que ajudaria a torná-lo melhor

Esta seção lista os códigos de erro enviados por AWS IoT.

Códigos de erro do agente de mensagens

Código de erro	Descrição do erro
400	Solicitação inválida.
401	Não autorizado.
403	Proibido.
426	É necessário fazer upgrade.
503	Serviço indisponível.

Códigos de erro de segurança e identidade

Código de erro	Descrição do erro
401	Não autorizado.

Códigos de erro da Sombra do Dispositivo

Código de erro	Descrição do erro
400	Solicitação inválida.
401	Não autorizado.
403	Proibido.
404	Não encontrado.
409	Conflito.
413	Solicitação muito grande.
422	Falha ao processar a solicitação.
429	Muitas solicitações.
500	Erro interno.
503	Serviço indisponível.

AWS IoT Device SDKs, Mobile SDKs e AWS IoT Device Client

Esta página resume os AWS IoT Device SDKs, bibliotecas de código aberto, guias para desenvolvedores, exemplos de aplicações e guias de portabilidade para ajudar você a criar soluções inovadoras de IoT com AWS IoT e plataformas de hardware de sua escolha.

Esses SDKs são para uso no dispositivo de IoT. Se você estiver desenvolvendo uma aplicação de IoT para uso em um dispositivo móvel, consulte os AWSSDKs móveis do. Se você estiver desenvolvendo uma aplicação de IoT ou um programa do lado do servidor, consulte os AWS SDKs.

AWS IoTSDKs de dispositivo da

Os SDKs de dispositivos da AWS IoT incluem bibliotecas de código aberto, guias de desenvolvedor com exemplos e guias de portabilidade para que você possa criar produtos ou soluções inovadoras da IoT nas plataformas de hardware de sua preferência.

> **Note**
>
> Os SDKs do Dispositivo AWS IoT lançaram um cliente MQTT 5. Os AWS IoT Device SDKs não oferecem suporte ao uso do TLS 1.3 no macOS.

Esses SDKs ajudam você a conectar dispositivos de IoT à AWS IoT usando os protocolos MQTT e WSS.

C++

SDK do dispositivo C++ da AWS IoT

O AWS IoT C++ Device SDK permite que os desenvolvedores compilem aplicações conectadas usando a AWS e as APIs da AWS IoT. Esse SDK foi especificamente projetado para dispositivos que não têm restrições de recursos e exigem recursos avançados, como enfileiramento de mensagens, suporte a vários threads e os mais recentes recursos de linguagem. Para obter mais informações, consulte:

- SDK do dispositivo C++ da AWS IoT v2 no GitHub

- Arquivo Leia-me do SDK do dispositivo C++ da AWS IoT v2
- Exemplos do AWS IoT Device SDK C++ v2
- Documentação da API do SDK do dispositivo C++ da AWS IoT v2

Python

SDK do dispositivo de AWS IoT para Python

O SDK do dispositivo da AWS IoT para Python permite que os desenvolvedores escrevam scripts Python para usar seus dispositivos para acessar a plataforma da AWS IoT por meio de MQTT ou MQTT por meio do protocolo WebSocket. Ao conectar os dispositivos à AWS IoT, os usuários podem trabalhar de modo seguro com o agente de mensagens, as regras e os shadows fornecidos pela AWS IoT e com outros serviços da AWS, como AWS Lambda, Kinesis, Amazon S3 e muito mais.

- SDK do dispositivo AWS IoT para Python v2 no GitHub
- Leia-me do SDK do dispositivo AWS IoT para Python v2
- Exemplos da API do SDK do dispositivo de AWS IoT para Python v2
- Documentação da API do SDK do dispositivo de AWS IoT para Python v2

JavaScript

SDK do dispositivo de AWS IoT para JavaScript

O pacote aws-iot-device-sdk.js permite que os desenvolvedores escrevam aplicativos JavaScript que acessam a AWS IoT usando MQTT ou MQTT sobre o protocolo WebSocket. Ele pode ser usado em ambientes Node.js e aplicações de navegador. Para obter mais informações, consulte:

- SDK do dispositivo de AWS IoT para JavaScript v2 no GitHub
- Leia-me do SDK do dispositivo AWS IoT para JavaScript v2
- Exemplos de SDK do dispositivo de AWS IoT para JavaScript v2
- Documentação da API do SDK do dispositivo de AWS IoT para JavaScript v2

Java

SDK do dispositivo de AWS IoT para Java

O SDK do dispositivo da AWS IoT para Java permite que os desenvolvedores de Java acessem a plataforma da AWS IoT por meio do MQTT ou do MQTT sobre o protocolo WebSocket. O SDK é criado com suporte a sombras. Você pode acessar as sombras usando métodos HTTP, inclusive GET, UPDATE e DELETE. O SDK também oferece suporte a um modelo simplificado de acesso a sombras, o que permite que os desenvolvedores troquem dados com as sombras usando apenas os métodos getter e setter, sem necessidade de serializar ou desserializar nenhum documento JSON.

> **Note**
>
> O SDK do Dispositivo AWS IoT para Java v2 agora é compatível com o desenvolvimento do Android. Para obter mais informações, consulte SDK de dispositivos AWS IoT para Android.

Para obter mais informações, consulte:

- AWS IoT SDK do dispositivo para Java v2 no GitHub
- AWS IoT Leia-me do SDK do dispositivo para Java v
- AWS IoT Exemplos de SDK do dispositivo de para Java v
- AWS IoT Documentação da API do SDK do dispositivo de para Java v

Swift

AWS IoT Device SDK para Swift

O AWS IoT Device SDK para Swift possibilita que os desenvolvedores do Swift criem aplicações AWS IoT para plataformas Linux e Apple macOS, iOS e tvOS usando o protocolo MQTT 5.

Para obter mais informações, consulte:

- AWS IoT Device SDK para Swift no GitHub
- Leiame do AWS IoT Device SDK para Swift
- Exemplos do AWS IoT Device SDK para Swift

SDK do dispositivo de AWS IoT para C incorporado

> **Note**
>
> Esse SDK é destinado ao uso por desenvolvedores de experientes de software incorporado.

O AWS IoT Device SDK para C incorporado (C-SDK) é um conjunto de arquivos de origem C na licença de código aberto do MIT que pode ser usado em aplicações incorporadas para conectar dispositivos IoT à AWS IoT Core com segurança. Ele inclui um cliente MQTT, analisador JSON e sombra do dispositivo da AWS IoT, trabalhos da AWS IoT, provisionamento de frota da AWS IoT e bibliotecas do AWS IoT Device Defender. Esse SDK é distribuído na forma de origem e pode ser compilado no firmware do cliente junto ao código da aplicação, outras bibliotecas e um sistema operacional (SO) de sua preferência.

Em geral, o AWS IoT Device SDK para C incorporado destina-se a dispositivos com restrição de recursos que exigem um runtime de linguagem C otimizado. É possível usar o SDK em qualquer sistema operacional e hospedá-lo em qualquer tipo de processador (p. ex., MCUs e MPUs).

Para obter mais informações, consulte:

- AWS IoT SDK do dispositivo para C incorporado no GitHub
- AWS IoT Arquivo leia-me do SDK do dispositivo da para C incorporado
- AWS IoT Exemplos de SDK do dispositivo de para C incorporado

AWSSDKs móveis do

Os SDKs móveis da AWS fornecem aos desenvolvedores de aplicações móveis suporte específico da plataforma para as APIs dos serviços do AWS IoT Core, a comunicação de dispositivos de IoT usando o MQTT e as APIs de outros serviços da AWS.

Android

AWS Mobile SDK para Android

O AWS Mobile SDK para Android contém uma biblioteca, exemplos e documentação para que os desenvolvedores criem aplicações móveis conectadas usando a AWS. Esse SDK também inclui suporte para comunicações de dispositivos MQTT e chamadas de APIs dos serviços do AWS IoT Core. Para obter mais informações, consulte:

- AWS Mobile SDK para Android no GitHub
- AWS Mobile SDK para Android Leia-me do
- AWS Mobile SDK para Android Exemplos da
- AWS Mobile SDK para Android Referência de API do
- Documentação de referência da classe AWSIoTClient

iOS

AWS Mobile SDK for iOS

O AWS Mobile SDK for iOS é um kit de desenvolvimento de software de código aberto distribuído em uma licença de código aberto Apache. O AWS Mobile SDK for iOS fornece uma biblioteca, exemplos de código e documentação para ajudar os desenvolvedores a criar aplicações móveis conectadas usando a AWS. Esse SDK também inclui suporte para comunicações de dispositivos MQTT e chamadas de APIs dos serviços do AWS IoT Core. Para obter mais informações, consulte:

- AWS Mobile SDK for iOS no GitHub
- AWS Mobile SDK for iOS Leia-me do
- AWS Mobile SDK for iOS Exemplos da
- Documentos de referência da classe AWSIoT no AWS Mobile SDK for iOS

AWS IoT Device Client

O AWS IoT Device Client fornece código para ajudar o dispositivo a se conectar à AWS IoT, realizar tarefas de provisionamento de frota, oferecer suporte às políticas de segurança do dispositivo, conectar-se usando tunelamento seguro e processar tarefas no dispositivo. Você pode instalar esse software no dispositivo para lidar com essas tarefas rotineiras do dispositivo, para que você possa se concentrar na solução específica.

> **Note**
>
> O AWS IoT Device Client funciona com dispositivos IoT baseados em microprocessadores com processadores x86_64 ou ARM e sistemas operacionais Linux comuns.

C++

AWS IoT Device Client

Para obter mais informações sobre o AWS IoT Device Client em C++, consulte o seguinte:

- AWS IoT Device Client em código-fonte C++ no GitHub
- AWS IoT Leia-me do Device Client em C

Versões anteriores de AWS IoT Device SDKs

Essas são versões anteriores dos AWS IoT Device SDKs que foram substituídas pelas versões mais recentes listadas acima. Esses SDKs estão recebendo somente atualizações de manutenção e segurança. Eles não serão atualizados para incluir novos recursos e não devem ser usados em novos projetos.

- AWS IoT SDK do dispositivo C++ no GitHub
- AWS IoT Arquivo leia-me do SDK do dispositivo C++ da
- AWS IoT SDK do dispositivo para Python v1 no GitHub
- AWS IoT Leia-me do SDK do dispositivo para Python v
- AWS IoT SDK do dispositivo de para Java no GitHub
- AWS IoT Leiame do SDK para Java do dispositivo do
- AWS IoT SDK do dispositivo de para JavaScript no GitHub
- AWS IoT Leia-me do SDK para JavaScript do dispositivo do
- Arduino Yún SDK no GitHub
- Arquivo Leia-me do Arduino Yún SDK

Exemplos de código para AWS IoT usar AWS SDKs

Os exemplos de código a seguir mostram como usar AWS IoT com um kit AWS de desenvolvimento de software (SDK).

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Exemplos de código

- Exemplos básicos de AWS IoT uso AWS SDKs
  - Olá AWS IoT
  - Aprenda o básico AWS IoT com um SDK AWS
  - Ações para AWS IoT usar AWS SDKs
    - Use AttachThingPrincipal com um AWS SDK ou CLI
    - Use CreateKeysAndCertificate com um AWS SDK ou CLI
    - Use CreateThing com um AWS SDK ou CLI
    - Use CreateTopicRule com um AWS SDK ou CLI
    - Use DeleteCertificate com um AWS SDK ou CLI
    - Use DeleteThing com um AWS SDK ou CLI
    - Use DeleteTopicRule com um AWS SDK ou CLI
    - Use DescribeEndpoint com um AWS SDK ou CLI
    - Use DescribeThing com um AWS SDK ou CLI
    - Use DetachThingPrincipal com um AWS SDK ou CLI
    - Use ListCertificates com um AWS SDK ou CLI
    - Use ListThings com um AWS SDK ou CLI
    - Use SearchIndex com um AWS SDK ou CLI

- Use UpdateIndexingConfiguration com um AWS SDK ou CLI
- Use UpdateThing com um AWS SDK ou CLI

Exemplos básicos de AWS IoT uso AWS SDKs

Os exemplos de código a seguir mostram como usar o básico do AWS IoT with AWS SDKs.

Exemplos

- Olá AWS IoT
- Aprenda o básico AWS IoT com um SDK AWS
- Ações para AWS IoT usar AWS SDKs
  - Use AttachThingPrincipal com um AWS SDK ou CLI
  - Use CreateKeysAndCertificate com um AWS SDK ou CLI
  - Use CreateThing com um AWS SDK ou CLI
  - Use CreateTopicRule com um AWS SDK ou CLI
  - Use DeleteCertificate com um AWS SDK ou CLI
  - Use DeleteThing com um AWS SDK ou CLI
  - Use DeleteTopicRule com um AWS SDK ou CLI
  - Use DescribeEndpoint com um AWS SDK ou CLI
  - Use DescribeThing com um AWS SDK ou CLI
  - Use DetachThingPrincipal com um AWS SDK ou CLI
  - Use ListCertificates com um AWS SDK ou CLI
  - Use ListThings com um AWS SDK ou CLI
  - Use SearchIndex com um AWS SDK ou CLI
  - Use UpdateIndexingConfiguration com um AWS SDK ou CLI
  - Use UpdateThing com um AWS SDK ou CLI

Olá AWS IoT

O exemplo de código a seguir mostra como começar a usar o AWS IoT.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Hello AWS IoT example.
/// </summary>
public class HelloIoT
{
    /// <summary>
    /// Main method to run the Hello IoT example.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        var iotClient = new AmazonIoTClient();

        try
        {
            Console.WriteLine("Hello AWS IoT! Let's list your IoT Things:");
            Console.WriteLine(new string('-', 80));

            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await iotClient.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
 paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
```

```
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the
next page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await iotClient.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
            }

            if (things is { Count: > 0 })
            {
                Console.WriteLine($"Found {things.Count} IoT Things:");
                foreach (var thing in things)
                {
                    Console.WriteLine($"- Thing Name: {thing.ThingName}");
                    Console.WriteLine($"  Thing ARN: {thing.ThingArn}");
                    Console.WriteLine($"  Thing Type: {thing.ThingTypeName ?? "No
type specified"}");
                    Console.WriteLine($"  Version: {thing.Version}");

                    if (thing.Attributes?.Count > 0)
                    {
                        Console.WriteLine("  Attributes:");
                        foreach (var attr in thing.Attributes)
                        {
                            Console.WriteLine($"    {attr.Key}: {attr.Value}");
                        }
                    }
                    Console.WriteLine();
                }
            }
            else
            {
                Console.WriteLine("No IoT Things found in your account.");
                Console.WriteLine("You can create IoT Things using the IoT Basics
scenario example.");
            }

            Console.WriteLine("Hello IoT completed successfully.");
```

```
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            Console.WriteLine($"Request throttled, please try again later:
 {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list Things. Here's why: {ex.Message}");
        }
    }
}
```

- Consulte detalhes da API em listThings na Referência da API do AWS SDK para .NET .

C++

SDK para C++

Código para o CMake arquivo CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iot)

# Set this project's name.
project("hello_iot")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
```

```
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_iot.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem hello_iot.cpp.

```
#include <aws/core/Aws.h>
#include <aws/iot/IoTClient.h>
#include <aws/iot/model/ListThingsRequest.h>
#include <iostream>

/*
 *  A "Hello IoT" starter application which initializes an AWS IoT client and
 *  lists the AWS IoT topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_iot'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
```

```
    //  Optional: change the log level for debugging.
    //  options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IoT::IoTClient iotClient(clientConfig);
        // List the things in the current account.
        Aws::IoT::Model::ListThingsRequest listThingsRequest;

        Aws::String nextToken; // Used for pagination.
        Aws::Vector<Aws::IoT::Model::ThingAttribute> allThings;

        do {
            if (!nextToken.empty()) {
                listThingsRequest.SetNextToken(nextToken);
            }

            Aws::IoT::Model::ListThingsOutcome listThingsOutcome =
iotClient.ListThings(
                    listThingsRequest);
            if (listThingsOutcome.IsSuccess()) {
                const Aws::Vector<Aws::IoT::Model::ThingAttribute> &things =
listThingsOutcome.GetResult().GetThings();
                allThings.insert(allThings.end(), things.begin(), things.end());
                nextToken = listThingsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "List things failed"
                          << listThingsOutcome.GetError().GetMessage() <<
std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << allThings.size() << " thing(s) found." << std::endl;
        for (auto const &thing: allThings) {
            std::cout << thing.GetThingName() << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
```

```
    return 0;
}
```

- Consulte detalhes da API em listThings na Referência da API do AWS SDK para C++ .

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
import software.amazon.awssdk.services.iot.paginators.ListThingsIterable;

import java.util.List;

public class HelloIoT {
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
 Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllThings(iotClient);
    }
```

```
    public static void listAllThings(IotClient iotClient) {
        iotClient.listThingsPaginator(ListThingsRequest.builder()
                .maxResults(10)
                .build())
            .stream()
            .flatMap(response -> response.things().stream())
            .forEach(attribute -> {
                System.out.println("Thing name: " + attribute.thingName());
                System.out.println("Thing ARN: " + attribute.thingArn());
            });
    }
}
```

- Consulte detalhes da API em listThings na Referência da API do AWS SDK for Java 2.x .

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.ListThingsRequest

suspend fun main() {
    println("A listing of your AWS IoT Things:")
    listAllThings()
}

suspend fun listAllThings() {
    val thingsRequest =
        ListThingsRequest {
            maxResults = 10
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
```

```
        val response = iotClient.listThings(thingsRequest)
        val thingList = response.things
        if (thingList != null) {
            for (attribute in thingList) {
                println("Thing name ${attribute.thingName}")
                println("Thing ARN: ${attribute.thingArn}")
            }
        }
    }
}
```

- Para detalhes da API, consulte listThings em Referência de API AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
def hello_iot():
    """
    Use the AWS SDK for Python (Boto3) to create an AWS IoT client and list
    up to 10 things in your AWS IoT account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """
    try:
        iot_client = boto3.client("iot")
        response = iot_client.list_things(maxResults=10)
        things = response.get("things", [])

        print("Hello, AWS IoT! Here are your things:")
        if things:
            for i, thing in enumerate(things, 1):
                print(f"{i}. {thing['thingName']}")
        else:
            print("No things found in your AWS IoT account.")
```

```
    except ClientError as e:
        if e.response["Error"]["Code"] == "UnauthorizedException":
            print("You don't have permission to access AWS IoT.")
        else:
            print(f"Couldn't access AWS IoT. Error: {e}")
    except NoCredentialsError:
        print("No AWS credentials found. Please configure your credentials.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
```

- Para obter detalhes da API, consulte listThings na Referência da API AWS SDK para Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Aprenda o básico AWS IoT com um SDK AWS

Os exemplos de código a seguir mostram como:

- Crie qualquer AWS IoT coisa.
- Gerar um certificado de dispositivo.
- Atualize AWS IoT qualquer coisa com atributos.
- Exibir um endpoint exclusivo.
- Liste seus AWS IoT certificados.
- Atualize uma AWS IoT sombra.
- Gravar informações do estado.
- Cria uma regra.
- Listar suas regras.
- Pesquisar coisas usando o nome da coisa.
- Exclua qualquer AWS IoT coisa.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando AWS IoT recursos.

```
/// <summary>
/// Scenario class for AWS IoT basics.
/// </summary>
public class IoTBasics
{
    public static bool IsInteractive = true;
    public static IoTWrapper? Wrapper = null;
    public static IAmazonCloudFormation? CloudFormationClient = null;
    public static ILogger<IoTBasics> logger = null!;
    private static IoTWrapper _iotWrapper = null!;
    private static IAmazonCloudFormation _amazonCloudFormation = null!;
    private static ILogger<IoTBasics> _logger = null!;

    private static string _stackName = "IoTBasicsStack";
    private static string _stackResourcePath = "../../../../../../scenarios/
basics/iot/iot_usecase/resources/cfn_template.yaml";

    /// <summary>
    /// Main method for the IoT Basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIoT>(new AWSOptions() { Region =
 RegionEndpoint.USEast1 })
                    .AddAWSService<IAmazonCloudFormation>()
                        .AddTransient<IoTWrapper>()
```

```
                        .AddLogging(builder => builder.AddConsole())
                        .AddSingleton<IAmazonIotData>(sp =>
                        {
                            var iotService = sp.GetRequiredService<IAmazonIoT>();
                            var request = new DescribeEndpointRequest
                            {
                                EndpointType = "iot:Data-ATS"
                            };
                            var response =
 iotService.DescribeEndpointAsync(request).Result;
                            return new AmazonIotDataClient($"https://
{response.EndpointAddress}/");
                        })
            )
            .Build();

        logger = LoggerFactory.Create(builder => builder.AddConsole())
            .CreateLogger<IoTBasics>();

        Wrapper = host.Services.GetRequiredService<IoTWrapper>();
        CloudFormationClient =
 host.Services.GetRequiredService<IAmazonCloudFormation>();

        // Set the private fields for backwards compatibility
        _logger = logger;
        _iotWrapper = Wrapper;
        _amazonCloudFormation = CloudFormationClient;

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the AWS IoT example scenario.");
        Console.WriteLine("This example program demonstrates various interactions
 with the AWS Internet of Things (IoT) Core service.");
        Console.WriteLine();
        if (IsInteractive)
        {
            Console.WriteLine("Press Enter to continue...");
            Console.ReadLine();
        }
        Console.WriteLine(new string('-', 80));

        try
        {
            await RunScenarioAsync();
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem running the scenario.");
            Console.WriteLine($"\nAn error occurred: {ex.Message}");
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The AWS IoT scenario has successfully completed.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Run the IoT Basics scenario.
    /// </summary>
    /// <returns>A Task object.</returns>
    public static async Task RunScenarioAsync()
    {
        // Use static properties if available, otherwise use private fields
        var iotWrapper = Wrapper ?? _iotWrapper;
        var cloudFormationClient = CloudFormationClient ?? _amazonCloudFormation;
        var scenarioLogger = logger ?? _logger;

        await RunScenarioInternalAsync(iotWrapper, cloudFormationClient,
 scenarioLogger);
    }

    /// <summary>
    /// Internal method to run the IoT Basics scenario with injected
 dependencies.
    /// </summary>
    /// <param name="iotWrapper">The IoT wrapper instance.</param>
    /// <param name="cloudFormationClient">The CloudFormation client instance.</
param>
    /// <param name="scenarioLogger">The logger instance.</param>
    /// <returns>A Task object.</returns>
    private static async Task RunScenarioInternalAsync(IoTWrapper iotWrapper,
 IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        string thingName = $"iot-thing-{Guid.NewGuid():N}";
        string certificateArn = "";
        string certificateId = "";
        string ruleName = $"iotruledefault";
        string snsTopicArn = "";
```

```
        try
        {
            // Step 1: Create an AWS IoT Thing
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("1. Create an AWS IoT Thing.");
            Console.WriteLine("An AWS IoT Thing represents a virtual entity in
the AWS IoT service that can be associated with a physical device.");
            Console.WriteLine();

            if (IsInteractive)
            {
                Console.Write("Enter Thing name: ");
                var userInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(userInput))
                    thingName = userInput;
            }
            else
            {
                Console.WriteLine($"Using default Thing name: {thingName}");
            }

            var thingArn = await iotWrapper.CreateThingAsync(thingName);
            Console.WriteLine($"{thingName} was successfully created. The ARN
value is {thingArn}");
            Console.WriteLine(new string('-', 80));

            // Step 1.1: List AWS IoT Things
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("2. List AWS IoT Things.");
            Console.WriteLine("Now let's list the IoT Things to see the Thing we
just created.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var things = await iotWrapper.ListThingsAsync();
            Console.WriteLine($"Found {things.Count} IoT Things:");
            foreach (var thing in things.Take(10)) // Show first 10 things
            {
                Console.WriteLine($"Thing Name: {thing.ThingName}");
                Console.WriteLine($"Thing ARN: {thing.ThingArn}");
```

```
                if (thing.Attributes != null && thing.Attributes.Any())
                {
                    Console.WriteLine("Attributes:");
                    foreach (var attr in thing.Attributes)
                    {
                        Console.WriteLine($"  {attr.Key}: {attr.Value}");
                    }
                }
                Console.WriteLine("--------------");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 2: Generate a Device Certificate
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("3. Generate a device certificate.");
            Console.WriteLine("A device certificate performs a role in securing
the communication between devices (Things) and the AWS IoT platform.");
            Console.WriteLine();

            var createCert = "y";
            if (IsInteractive)
            {
                Console.Write($"Do you want to create a certificate for
{thingName}? (y/n)");
                createCert = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Creating certificate for {thingName}...");
            }

            if (createCert?.ToLower() == "y")
            {
                var certificateResult = await
iotWrapper.CreateKeysAndCertificateAsync();
                if (certificateResult.HasValue)
                {
                    var (certArn, certPem, certId) = certificateResult.Value;
                    certificateArn = certArn;
                    certificateId = certId;

                    Console.WriteLine($"\nCertificate:");
                    // Show only first few lines of certificate for brevity
```

```
                    var lines = certPem.Split('\n');
                    for (int i = 0; i < Math.Min(lines.Length, 5); i++)
                    {
                        Console.WriteLine(lines[i]);
                    }
                    if (lines.Length > 5)
                    {
                        Console.WriteLine("...");
                    }

                    Console.WriteLine($"\nCertificate ARN:");
                    Console.WriteLine(certificateArn);

                    // Step 3: Attach the Certificate to the AWS IoT Thing
                    Console.WriteLine("Attach the certificate to the AWS IoT
Thing.");
                    var attachResult = await
iotWrapper.AttachThingPrincipalAsync(thingName, certificateArn);
                    if (attachResult)
                    {
                        Console.WriteLine("Certificate attached to Thing
successfully.");
                    }
                    else
                    {
                        Console.WriteLine("Failed to attach certificate to
Thing.");
                    }

                    Console.WriteLine("Thing Details:");
                    Console.WriteLine($"Thing Name: {thingName}");
                    Console.WriteLine($"Thing ARN: {thingArn}");
                }
                else
                {
                    Console.WriteLine("Failed to create certificate.");
                }
            }
            Console.WriteLine(new string('-', 80));

            // Step 4: Update an AWS IoT Thing with Attributes
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("4. Update an AWS IoT Thing with Attributes.");
```

```
            Console.WriteLine("IoT Thing attributes, represented as key-value
pairs, offer a pivotal advantage in facilitating efficient data");
            Console.WriteLine("management and retrieval within the AWS IoT
ecosystem.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var attributes = new Dictionary<string, string>
            {
                { "Location", "Seattle" },
                { "DeviceType", "Sensor" },
                { "Firmware", "1.2.3" }
            };

            await iotWrapper.UpdateThingAsync(thingName, attributes);
            Console.WriteLine("Thing attributes updated successfully.");
            Console.WriteLine(new string('-', 80));

            // Step 5: Return a unique endpoint specific to the Amazon Web
Services account
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("5. Return a unique endpoint specific to the Amazon
Web Services account.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var endpoint = await iotWrapper.DescribeEndpointAsync();
            if (endpoint != null)
            {
                var subdomain = endpoint.Split('.')[0];
                Console.WriteLine($"Extracted subdomain: {subdomain}");
                Console.WriteLine($"Full Endpoint URL: https://{endpoint}");
            }
            else
            {
                Console.WriteLine("Failed to retrieve endpoint.");
```

```
            }
            Console.WriteLine(new string('-', 80));

            // Step 6: List your AWS IoT certificates
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("6. List your AWS IoT certificates");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var certificates = await iotWrapper.ListCertificatesAsync();
            foreach (var cert in certificates.Take(5)) // Show first 5
certificates
            {
                Console.WriteLine($"Cert id: {cert.CertificateId}");
                Console.WriteLine($"Cert Arn: {cert.CertificateArn}");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 7: Create an IoT shadow
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("7. Update an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var shadowPayload = JsonSerializer.Serialize(new
            {
                state = new
                {
                    desired = new
                    {
                        temperature = 25,
                        humidity = 50
                    }
                }
            });
```

```
            await iotWrapper.UpdateThingShadowAsync(thingName, shadowPayload);
            Console.WriteLine("Thing Shadow updated successfully.");
            Console.WriteLine(new string('-', 80));

            // Step 8: Write out the state information, in JSON format
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("8. Write out the state information, in JSON
format.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var shadowData = await iotWrapper.GetThingShadowAsync(thingName);
            Console.WriteLine($"Received Shadow Data: {shadowData}");
            Console.WriteLine(new string('-', 80));

            // Step 9: Set up resources (SNS topic and IAM role) and create a
rule
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("9. Set up resources and create a rule");
            Console.WriteLine();

            // Deploy CloudFormation stack to create SNS topic and IAM role
            Console.WriteLine("Deploying CloudFormation stack to create SNS topic
and IAM role...");

            var deployStack = !IsInteractive || GetYesNoResponse("Would you like
to deploy the CloudFormation stack? (y/n) ");
            if (deployStack)
            {
                if (IsInteractive)
                {
                    Console.Write(
                        $"Enter stack resource file path (or press Enter for
default '{_stackResourcePath}'): ");
                    var userResourcePath = Console.ReadLine();
                    if (!string.IsNullOrEmpty(userResourcePath))
                        _stackResourcePath = userResourcePath;
                }

                _stackName = PromptUserForStackName();
```

```
                var deploySuccess = await DeployCloudFormationStack(_stackName,
cloudFormationClient, scenarioLogger);

                if (deploySuccess)
                {
                    // Get stack outputs
                    var stackOutputs = await GetStackOutputs(_stackName,
cloudFormationClient, scenarioLogger);
                    if (stackOutputs != null)
                    {
                        snsTopicArn = stackOutputs["SNSTopicArn"];
                        string roleArn = stackOutputs["RoleArn"];

                        Console.WriteLine($"Successfully deployed stack. SNS
topic: {snsTopicArn}");
                        Console.WriteLine($"Successfully deployed stack. IAM
role: {roleArn}");

                        if (IsInteractive)
                        {
                            Console.Write($"Enter Rule name (press Enter for
default '{ruleName}'): ");
                            var userRuleName = Console.ReadLine();
                            if (!string.IsNullOrEmpty(userRuleName))
                                ruleName = userRuleName;
                        }
                        else
                        {
                            Console.WriteLine($"Using default rule name:
{ruleName}");
                        }

                        // Now create the IoT rule with the CloudFormation
outputs
                        var ruleResult = await
iotWrapper.CreateTopicRuleAsync(ruleName, snsTopicArn, roleArn);
                        if (ruleResult)
                        {
                            Console.WriteLine("IoT Rule created successfully.");
                        }
                        else
                        {
                            Console.WriteLine("Failed to create IoT rule.");
```

```
                    }
                }
                else
                {
                    Console.WriteLine("Failed to get stack outputs. Skipping
rule creation.");
                }
            }
            else
            {
                Console.WriteLine("Failed to deploy CloudFormation stack.
Skipping rule creation.");
            }
        }
        else
        {
            Console.WriteLine("Skipping CloudFormation stack deployment and
rule creation.");
        }
        Console.WriteLine(new string('-', 80));

        // Step 10: List your rules
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("10. List your rules.");
        if (IsInteractive)
        {
            Console.WriteLine("Press Enter to continue...");
            Console.ReadLine();
        }

        var rules = await iotWrapper.ListTopicRulesAsync();
        Console.WriteLine("List of IoT Rules:");
        foreach (var rule in rules.Take(5)) // Show first 5 rules
        {
            Console.WriteLine($"Rule Name: {rule.RuleName}");
            Console.WriteLine($"Rule ARN: {rule.RuleArn}");
            Console.WriteLine("--------------");
        }
        Console.WriteLine();
        Console.WriteLine(new string('-', 80));

        // Step 11: Search things using the Thing name
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("11. Search things using the Thing name.");
```

```
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var searchResults = await iotWrapper.SearchIndexAsync($"thingName:
{thingName}");
            if (searchResults.Any())
            {
                Console.WriteLine($"Thing id found using search is
 {searchResults.First().ThingId}");
            }
            else
            {
                Console.WriteLine($"No search results found for Thing:
 {thingName}");
            }
            Console.WriteLine(new string('-', 80));

            // Step 12: Cleanup - Detach and delete certificate
            if (!string.IsNullOrEmpty(certificateArn))
            {
                Console.WriteLine(new string('-', 80));
                var deleteCert = "y";
                if (IsInteractive)
                {
                    Console.Write($"Do you want to detach and delete the
 certificate for {thingName}? (y/n)");
                    deleteCert = Console.ReadLine();
                }
                else
                {
                    Console.WriteLine($"Detaching and deleting certificate for
 {thingName}...");
                }

                if (deleteCert?.ToLower() == "y")
                {
                    Console.WriteLine("12. You selected to detach and delete the
 certificate.");
                    if (IsInteractive)
                    {
                        Console.WriteLine("Press Enter to continue...");
```

```
                    Console.ReadLine();
                }

                await iotWrapper.DetachThingPrincipalAsync(thingName,
certificateArn);
                Console.WriteLine($"{certificateArn} was successfully removed
from {thingName}");

                await iotWrapper.DeleteCertificateAsync(certificateId);
                Console.WriteLine($"{certificateArn} was successfully
deleted.");
            }
            Console.WriteLine(new string('-', 80));
        }

        // Step 13: Delete the AWS IoT Thing
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("13. Delete the AWS IoT Thing.");
        var deleteThing = "y";
        if (IsInteractive)
        {
            Console.Write($"Do you want to delete the IoT Thing? (y/n)");
            deleteThing = Console.ReadLine();
        }
        else
        {
            Console.WriteLine($"Deleting IoT Thing {thingName}...");
        }

        if (deleteThing?.ToLower() == "y")
        {
            await iotWrapper.DeleteThingAsync(thingName);
            Console.WriteLine($"Deleted Thing {thingName}");
        }
        Console.WriteLine(new string('-', 80));

        // Step 14: Clean up CloudFormation stack
        if (!string.IsNullOrEmpty(snsTopicArn))
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("14. Clean up CloudFormation stack.");
            Console.WriteLine("Deleting the CloudFormation stack and all
resources...");
```

```
                var cleanup = !IsInteractive || GetYesNoResponse("Do you want to
 delete the CloudFormation stack and all resources? (y/n) ");
                if (cleanup)
                {
                    var ruleCleanupSuccess = await
 iotWrapper.DeleteTopicRuleAsync(ruleName);

                    var stackCleanupSuccess = await
 DeleteCloudFormationStack(_stackName, cloudFormationClient, scenarioLogger);
                    if (ruleCleanupSuccess && stackCleanupSuccess)
                    {
                        Console.WriteLine("Successfully cleaned up CloudFormation
 stack and all resources.");
                    }
                    else
                    {
                        Console.WriteLine("Some cleanup operations failed. Check
 the logs for details.");
                    }
                }
                else
                {
                    Console.WriteLine($"Resources will remain. Stack name:
 {_stackName}");
                }
                Console.WriteLine(new string('-', 80));
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, "Error occurred during scenario
 execution.");

            // Cleanup on error
            if (!string.IsNullOrEmpty(certificateArn) && !
string.IsNullOrEmpty(thingName))
            {
                try
                {
                    await iotWrapper.DetachThingPrincipalAsync(thingName,
 certificateArn);
                    await iotWrapper.DeleteCertificateAsync(certificateId);
                }
                catch (Exception cleanupEx)
```

```
                {
                    scenarioLogger.LogError(cleanupEx, "Error during cleanup.");
                }
            }

            if (!string.IsNullOrEmpty(thingName))
            {
                try
                {
                    await iotWrapper.DeleteThingAsync(thingName);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during Thing
cleanup.");
                }
            }

            // Clean up CloudFormation stack on error
            if (!string.IsNullOrEmpty(snsTopicArn))
            {
                try
                {
                    await _iotWrapper.DeleteTopicRuleAsync(ruleName);
                    await DeleteCloudFormationStack(_stackName,
cloudFormationClient, scenarioLogger);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during
CloudFormation stack cleanup.");
                }
            }

            throw;
        }
    }

    /// <summary>
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
```

```
   /// <returns>True if the stack was deployed successfully.</returns>
   private static async Task<bool> DeployCloudFormationStack(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
   {
       Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");

       try
       {
           var request = new CreateStackRequest
           {
               StackName = stackName,
               TemplateBody = await File.ReadAllTextAsync(_stackResourcePath),
               Capabilities = new List<string>
{ Capability.CAPABILITY_NAMED_IAM }
           };

           var response = await cloudFormationClient.CreateStackAsync(request);

           if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
           {
               Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

               bool stackCreated = await
WaitForStackCompletion(response.StackId, cloudFormationClient, scenarioLogger);

               if (stackCreated)
               {
                   Console.WriteLine("CloudFormation stack created
successfully.");
                   return true;
               }
               else
               {
                   scenarioLogger.LogError($"CloudFormation stack creation
failed: {stackName}");
                   return false;
               }
           }
           else
           {
               scenarioLogger.LogError($"Failed to create CloudFormation stack:
{stackName}");
               return false;
```

```
            }
        }
        catch (AlreadyExistsException)
        {
            scenarioLogger.LogWarning($"CloudFormation stack '{stackName}'
already exists. Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName,
cloudFormationClient, scenarioLogger);
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackId
            };

            var describeStacksResponse = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
```

```
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
                {
                    return false;
                }
            }

            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        scenarioLogger.LogError("Timed out waiting for CloudFormation stack
creation to complete.");
        return false;
    }

    /// <summary>
    /// Gets the outputs from the CloudFormation stack.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>A dictionary of stack outputs.</returns>
    private static async Task<Dictionary<string, string>?> GetStackOutputs(string
stackName, IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics>
scenarioLogger)
    {
        try
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };
```

```
            var response = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

            if (response.Stacks.Count > 0)
            {
                var outputs = new Dictionary<string, string>();
                foreach (var output in response.Stacks[0].Outputs)
                {
                    outputs[output.OutputKey] = output.OutputValue;
                }
                return outputs;
            }

            return null;
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"Failed to get stack outputs for
{stackName}");
            return null;
        }
    }

    /// <summary>
    /// Deletes the CloudFormation stack and waits for confirmation.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deleted successfully.</returns>
    private static async Task<bool> DeleteCloudFormationStack(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        try
        {
            var request = new DeleteStackRequest
            {
                StackName = stackName
            };

            await cloudFormationClient.DeleteStackAsync(request);
            Console.WriteLine($"CloudFormation stack '{stackName}' is being
deleted. This may take a few minutes.");
```

```
            bool stackDeleted = await WaitForStackDeletion(stackName,
cloudFormationClient, scenarioLogger);

            if (stackDeleted)
            {
                Console.WriteLine($"CloudFormation stack '{stackName}' has been
deleted.");
                return true;
            }
            else
            {
                scenarioLogger.LogError($"Failed to delete CloudFormation stack
'{stackName}'.");
                return false;
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"An error occurred while deleting the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the stack to be deleted.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deleted successfully.</returns>
    private static async Task<bool> WaitForStackDeletion(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };
```

```
            try
            {
                var describeStacksResponse = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_COMPLETE)
                {
                    return true;
                }
            }
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
                return true;
            }

            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to
be deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        scenarioLogger.LogError($"Timed out waiting for CloudFormation stack
'{stackName}' to be deleted.");
        return false;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Prompts the user for a stack name.
```

```
    /// </summary>
    private static string PromptUserForStackName()
    {
        if (IsInteractive)
        {
            Console.Write($"Enter a name for the CloudFormation stack (press
 Enter for default '{_stackName}'): ");
            string? input = Console.ReadLine();
            if (!string.IsNullOrWhiteSpace(input))
            {
                var regex = new System.Text.RegularExpressions.Regex("[a-zA-Z][-
a-zA-Z0-9]*");
                if (!regex.IsMatch(input))
                {
                    Console.WriteLine($"Invalid stack name. Using default:
 {_stackName}");
                    return _stackName;
                }
                return input;
            }
        }
        return _stackName;
    }
}
```

Uma classe de invólucro para métodos do AWS IoT SDK.

```
/// <summary>
/// Wrapper methods to use Amazon IoT Core with .NET.
/// </summary>
public class IoTWrapper
{
    private readonly IAmazonIoT _amazonIoT;
    private readonly IAmazonIotData _amazonIotData;
    private readonly ILogger<IoTWrapper> _logger;

    /// <summary>
    /// Constructor for the IoT wrapper.
    /// </summary>
    /// <param name="amazonIoT">The injected IoT client.</param>
    /// <param name="amazonIotData">The injected IoT Data client.</param>
    /// <param name="logger">The injected logger.</param>
```

```
    public IoTWrapper(IAmazonIoT amazonIoT, IAmazonIotData amazonIotData,
 ILogger<IoTWrapper> logger)
    {
        _amazonIoT = amazonIoT;
        _amazonIotData = amazonIotData;
        _logger = logger;
    }

    /// <summary>
    /// Creates an AWS IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to create.</param>
    /// <returns>The ARN of the Thing created, or null if creation failed.</
returns>
    public async Task<string?> CreateThingAsync(string thingName)
    {
        try
        {
            var request = new CreateThingRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIoT.CreateThingAsync(request);
            _logger.LogInformation($"Created Thing {thingName} with ARN
 {response.ThingArn}");
            return response.ThingArn;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Thing {thingName} already exists:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create Thing {thingName}. Here's why:
 {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Creates a device certificate for AWS IoT.
```

```
    /// </summary>
    /// <returns>The certificate details including ARN and certificate PEM, or
 null if creation failed.</returns>
    public async Task<(string CertificateArn, string CertificatePem, string
 CertificateId)?> CreateKeysAndCertificateAsync()
    {
        try
        {
            var request = new CreateKeysAndCertificateRequest
            {
                SetAsActive = true
            };

            var response = await
 _amazonIoT.CreateKeysAndCertificateAsync(request);
            _logger.LogInformation($"Created certificate with ARN
 {response.CertificateArn}");
            return (response.CertificateArn, response.CertificatePem,
 response.CertificateId);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create certificate. Here's why:
 {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Attaches a certificate to an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to attach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> AttachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
```

```
        try
        {
            var request = new AttachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.AttachThingPrincipalAsync(request);
            _logger.LogInformation($"Attached certificate {certificateArn} to
Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot attach certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't attach certificate to Thing. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Updates an IoT Thing with attributes.
    /// </summary>
    /// <param name="thingName">The name of the Thing to update.</param>
    /// <param name="attributes">Dictionary of attributes to add.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingAsync(string thingName, Dictionary<string,
string> attributes)
    {
        try
        {
            var request = new UpdateThingRequest
            {
                ThingName = thingName,
                AttributePayload = new AttributePayload
                {
                    Attributes = attributes,
```

```
                    Merge = true
                }
            };

            await _amazonIoT.UpdateThingAsync(request);
            _logger.LogInformation($"Updated Thing {thingName} with attributes");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing attributes. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Gets the AWS IoT endpoint URL.
    /// </summary>
    /// <returns>The endpoint URL, or null if retrieval failed.</returns>
    public async Task<string?> DescribeEndpointAsync()
    {
        try
        {
            var request = new DescribeEndpointRequest
            {
                EndpointType = "iot:Data-ATS"
            };

            var response = await _amazonIoT.DescribeEndpointAsync(request);
            _logger.LogInformation($"Retrieved endpoint:
{response.EndpointAddress}");
            return response.EndpointAddress;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
```

```
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't describe endpoint. Here's why:
{ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Lists all certificates associated with the account.
    /// </summary>
    /// <returns>List of certificate information, or empty list if listing
failed.</returns>
    public async Task<List<Certificate>> ListCertificatesAsync()
    {
        try
        {
            var request = new ListCertificatesRequest();
            var response = await _amazonIoT.ListCertificatesAsync(request);

            _logger.LogInformation($"Retrieved {response.Certificates.Count}
certificates");
            return response.Certificates;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<Certificate>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list certificates. Here's why:
{ex.Message}");
            return new List<Certificate>();
        }
    }

    /// <summary>
    /// Updates the Thing's shadow with new state information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
```

```
    /// <param name="shadowPayload">The shadow payload in JSON format.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingShadowAsync(string thingName, string
 shadowPayload)
    {
        try
        {
            var request = new UpdateThingShadowRequest
            {
                ThingName = thingName,
                Payload = new
 MemoryStream(System.Text.Encoding.UTF8.GetBytes(shadowPayload))
            };

            await _amazonIotData.UpdateThingShadowAsync(request);
            _logger.LogInformation($"Updated shadow for Thing {thingName}");
            return true;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing shadow - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing shadow. Here's why:
 {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Gets the Thing's shadow information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <returns>The shadow data as a string, or null if retrieval failed.</
returns>
    public async Task<string?> GetThingShadowAsync(string thingName)
    {
        try
        {
            var request = new GetThingShadowRequest
            {
```

```
                ThingName = thingName
            };

            var response = await _amazonIotData.GetThingShadowAsync(request);
            using var reader = new StreamReader(response.Payload);
            var shadowData = await reader.ReadToEndAsync();

            _logger.LogInformation($"Retrieved shadow for Thing {thingName}");
            return shadowData;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot get Thing shadow - resource not found:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't get Thing shadow. Here's why:
 {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Creates an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="snsTopicArn">The ARN of the SNS topic for the action.</
param>
    /// <param name="roleArn">The ARN of the IAM role.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> CreateTopicRuleAsync(string ruleName, string
 snsTopicArn, string roleArn)
    {
        try
        {
            var request = new CreateTopicRuleRequest
            {
                RuleName = ruleName,
                TopicRulePayload = new TopicRulePayload
                {
                    Sql = "SELECT * FROM 'topic/subtopic'",
                    Description = $"Rule created by .NET example: {ruleName}",
```

```
                    Actions = new List<Amazon.IoT.Model.Action>
                    {
                        new Amazon.IoT.Model.Action
                        {
                            Sns = new SnsAction
                            {
                                TargetArn = snsTopicArn,
                                RoleArn = roleArn
                            }
                        }
                    },
                    RuleDisabled = false
                }
            };

            await _amazonIoT.CreateTopicRuleAsync(request);
            _logger.LogInformation($"Created IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Rule {ruleName} already exists: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteTopicRuleAsync(string ruleName)
    {
        try
        {
            var request = new DeleteTopicRuleRequest
            {
                RuleName = ruleName,
```

```
            };

            await _amazonIoT.DeleteTopicRuleAsync(request);
            _logger.LogInformation($"Deleted IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogWarning($"Rule {ruleName} not found: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Lists all IoT topic rules.
    /// </summary>
    /// <returns>List of topic rules, or empty list if listing failed.</returns>
    public async Task<List<TopicRuleListItem>> ListTopicRulesAsync()
    {
        try
        {
            var request = new ListTopicRulesRequest();
            var response = await _amazonIoT.ListTopicRulesAsync(request);

            _logger.LogInformation($"Retrieved {response.Rules.Count} IoT
rules");
            return response.Rules;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<TopicRuleListItem>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list topic rules. Here's why:
{ex.Message}");
```

```
            return new List<TopicRuleListItem>();
        }
    }

    /// <summary>
    /// Searches for IoT Things using the search index.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
search failed.</returns>
    public async Task<List<ThingDocument>> SearchIndexAsync(string queryString)
    {
        try
        {
            // First, try to perform the search
            var request = new SearchIndexRequest
            {
                QueryString = queryString
            };

            var response = await _amazonIoT.SearchIndexAsync(request);
            _logger.LogInformation($"Search found {response.Things.Count}
Things");
            return response.Things;
        }
        catch (Amazon.IoT.Model.IndexNotReadyException ex)
        {
            _logger.LogWarning($"Search index not ready, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
        {
            _logger.LogWarning($"Search index not configured, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingDocument>();
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't search index. Here's why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Sets up the indexing configuration and retries the search after waiting
for the index to be ready.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
setup/search failed.</returns>
    private async Task<List<ThingDocument>> SetupIndexAndRetrySearchAsync(string
queryString)
    {
        try
        {
            // Update indexing configuration to REGISTRY mode
            _logger.LogInformation("Setting up IoT search indexing
configuration...");
            await _amazonIoT.UpdateIndexingConfigurationAsync(
                new UpdateIndexingConfigurationRequest()
                {
                    ThingIndexingConfiguration = new ThingIndexingConfiguration()
                    {
                        ThingIndexingMode = ThingIndexingMode.REGISTRY
                    }
                });

            _logger.LogInformation("Indexing configuration updated. Waiting for
index to be ready...");

            // Wait for the index to be set up - this can take some time
            const int maxRetries = 10;
            const int retryDelaySeconds = 10;

            for (int attempt = 1; attempt <= maxRetries; attempt++)
            {
                try
                {
                    _logger.LogInformation($"Waiting for index to be ready
(attempt {attempt}/{maxRetries})...");
```

```
                    await Task.Delay(TimeSpan.FromSeconds(retryDelaySeconds));

                    // Try to get the current indexing configuration to see if
it's ready
                    var configResponse = await
_amazonIoT.GetIndexingConfigurationAsync(new GetIndexingConfigurationRequest());
                    if
(configResponse.ThingIndexingConfiguration?.ThingIndexingMode ==
ThingIndexingMode.REGISTRY)
                    {
                        // Try the search again
                        var request = new SearchIndexRequest
                        {
                            QueryString = queryString
                        };

                        var response = await
_amazonIoT.SearchIndexAsync(request);
                        _logger.LogInformation($"Search found
{response.Things.Count} Things after index setup");
                        return response.Things;
                    }
                }
                catch (Amazon.IoT.Model.IndexNotReadyException)
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
                catch (Amazon.IoT.Model.InvalidRequestException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
            }

            _logger.LogWarning("Timeout waiting for search index to be ready
after configuration update");
            return new List<ThingDocument>();
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't set up search index configuration. Here's
 why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Detaches a certificate from an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to detach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DetachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
        try
        {
            var request = new DetachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.DetachThingPrincipalAsync(request);
            _logger.LogInformation($"Detached certificate {certificateArn} from
 Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot detach certificate - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't detach certificate from Thing. Here's
 why: {ex.Message}");
            return false;
        }
    }
```

```
    /// <summary>
    /// Deletes an IoT certificate.
    /// </summary>
    /// <param name="certificateId">The ID of the certificate to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteCertificateAsync(string certificateId)
    {
        try
        {
            // First, update the certificate to inactive state
            var updateRequest = new UpdateCertificateRequest
            {
                CertificateId = certificateId,
                NewStatus = CertificateStatus.INACTIVE
            };
            await _amazonIoT.UpdateCertificateAsync(updateRequest);

            // Then delete the certificate
            var deleteRequest = new DeleteCertificateRequest
            {
                CertificateId = certificateId
            };

            await _amazonIoT.DeleteCertificateAsync(deleteRequest);
            _logger.LogInformation($"Deleted certificate {certificateId}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete certificate. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT Thing.
```

```
    /// </summary>
    /// <param name="thingName">The name of the Thing to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteThingAsync(string thingName)
    {
        try
        {
            var request = new DeleteThingRequest
            {
                ThingName = thingName
            };

            await _amazonIoT.DeleteThingAsync(request);
            _logger.LogInformation($"Deleted Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete Thing. Here's why: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Lists IoT Things with pagination support.
    /// </summary>
    /// <returns>List of Things, or empty list if listing failed.</returns>
    public async Task<List<ThingAttribute>> ListThingsAsync()
    {
        try
        {
            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await _amazonIoT.ListThingsAsync(request);
```

```
            // Since there is not a built-in paginator, use the NextMarker to
 paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the
 next page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await _amazonIoT.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
            }

            _logger.LogInformation($"Retrieved {things.Count} Things");
            return things;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
 {ex.Message}");
            return new List<ThingAttribute>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list Things. Here's why: {ex.Message}");
            return new List<ThingAttribute>();
        }
    }

}
```

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie qualquer AWS IoT coisa.

```
    Aws::String thingName = askQuestion("Enter a thing name: ");

    if (!createThing(thingName, clientConfiguration)) {
        std::cerr << "Exiting because createThing failed." << std::endl;
        cleanup("", "", "", "", "", false, clientConfiguration);
        return false;
    }
```

```
//! Create an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
            createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
```

```
    }

    return outcome.IsSuccess();
}
```

Gerar e anexar um certificado de dispositivo.

```
    Aws::String certificateARN;
    Aws::String certificateID;
    if (askYesNoQuestion("Would you like to create a certificate for your thing?
(y/n) ")) {
        Aws::String outputFolder;
        if (askYesNoQuestion(
                "Would you like to save the certificate and keys to file? (y/n)
")) {
            outputFolder = std::filesystem::current_path();
            outputFolder += "/device_keys_and_certificates";

            std::filesystem::create_directories(outputFolder);

            std::cout << "The certificate and keys will be saved to the folder: "
                      << outputFolder << std::endl;
        }

        if (!createKeysAndCertificate(outputFolder, certificateARN,
certificateID,
                                      clientConfiguration)) {
            std::cerr << "Exiting because createKeysAndCertificate failed."
                      << std::endl;
            cleanup(thingName, "", "", "", "", false, clientConfiguration);
            return false;
        }

        std::cout << "\nNext, the certificate will be attached to the thing.\n"
                  << std::endl;
        if (!attachThingPrincipal(certificateARN, thingName,
clientConfiguration)) {
            std::cerr << "Exiting because attachThingPrincipal failed." <<
std::endl;
            cleanup(thingName, certificateARN, certificateID, "", "",
                    false,
                    clientConfiguration);
```

```
            return false;
        }
    }
```

```
//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if
 provided.
/*!
  \param outputFolder: Location for storing output in files, ignored when string
 is empty.
  \param certificateARNResult: A string to receive the ARN of the created
 certificate.
  \param certificateID: A string to receive the ID of the created certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
 createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
            client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ",
 certificate ID: "
                  << certificateID << std::endl;

        if (!outputFolder.empty()) {
            std::cout << "Writing certificate and keys to the folder '" <<
 outputFolder
                      << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;
```

```
            Aws::String certificateFilePath = outputFolder + "/
certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
 certificateFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            certificateFile << outcome.GetResult().GetCertificatePem();
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
 outcome.GetResult().GetKeyPair();

            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);
            if (!privateKeyFile.is_open()) {
                std::cerr << "Error opening private key file, '" <<
 privateKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            privateKeyFile << keyPair.GetPrivateKey();
            privateKeyFile.close();

            Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
            std::ofstream publicKeyFile(publicKeyFilePath);
            if (!publicKeyFile.is_open()) {
                std::cerr << "Error opening public key file, '" <<
 publicKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            publicKeyFile << keyPair.GetPublicKey();
        }
    }
    else {
        std::cerr << "Error creating keys and certificate: "
                  << outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}

//! Attach a principal to an AWS IoT thing.
/*!
  \param principal: A principal to attach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
 client.AttachThingPrincipal(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Execute várias operações na AWS IoT coisa.

```
    if (!updateThing(thingName, { {"location", "Office"}, {"firmwareVersion",
 "v2.0"} }, clientConfiguration)) {
        std::cerr << "Exiting because updateThing failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
```

```
   printAsterisksLine();

   std::cout << "Now an endpoint will be retrieved for your account.\n" <<
std::endl;
   std::cout << "An IoT Endpoint refers to a specific URL or Uniform Resource
Locator that serves as the entry point\n"
   << "for communication between IoT devices and the AWS IoT service." <<
std::endl;

   askQuestion("Press Enter to continue:", alwaysTrueTest);

   Aws::String endpoint;
   if (!describeEndpoint(endpoint, clientConfiguration)) {
       std::cerr << "Exiting because getEndpoint failed." << std::endl;
       cleanup(thingName, certificateARN, certificateID, "", "", false,
               clientConfiguration);
       return false;
   }
   std::cout <<"Your endpoint is " << endpoint << "." << std::endl;
   printAsterisksLine();

   std::cout << "Now the certificates in your account will be listed." <<
std::endl;
   askQuestion("Press Enter to continue:", alwaysTrueTest);

   if (!listCertificates(clientConfiguration)) {
       std::cerr << "Exiting because listCertificates failed." << std::endl;
       cleanup(thingName, certificateARN, certificateID, "", "", false,
               clientConfiguration);
       return false;
   }

   printAsterisksLine();

   std::cout << "Now the shadow for the thing will be updated.\n" << std::endl;
   std::cout << "A thing shadow refers to a feature that enables you to create a
virtual representation, or \"shadow,\"\n"
   << "of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between\n"
   << "the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a thing shadow." << std::endl;
   askQuestion("Press Enter to continue:", alwaysTrueTest);
```

```
    if (!updateThingShadow(thingName, R"({"state":{"reported":
{"temperature":25,"humidity":50}}})", clientConfiguration)) {
        std::cerr << "Exiting because updateThingShadow failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now, the state information for the shadow will be retrieved.\n"
 << std::endl;
    askQuestion("Press Enter to continue:", alwaysTrueTest);

    Aws::String shadowState;
    if (!getThingShadow(thingName, shadowState, clientConfiguration)) {
        std::cerr << "Exiting because getThingShadow failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
    std::cout << "The retrieved shadow state is: " << shadowState << std::endl;

    printAsterisksLine();

    std::cout << "A rule with now be added to to the thing.\n" << std::endl;
    std::cout << "Any user who has permission to create rules will be able to
 access data processed by the rule." << std::endl;
    std::cout << "In this case, the rule will use an Simple Notification Service
(SNS) topic and an IAM rule." << std::endl;
    std::cout << "These resources will be created using a CloudFormation
 template." << std::endl;
    std::cout << "Stack creation may take a few minutes." << std::endl;

    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    Aws::Map<Aws::String, Aws::String> outputs
 =createCloudFormationStack(STACK_NAME,clientConfiguration);
    if (outputs.empty()) {
        std::cerr << "Exiting because createCloudFormationStack failed." <<
 std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
```

```
    // Retrieve the topic ARN and role ARN from the CloudFormation stack outputs.
    auto topicArnIter = outputs.find(SNS_TOPIC_ARN_OUTPUT);
    auto roleArnIter = outputs.find(ROLE_ARN_OUTPUT);
    if ((topicArnIter == outputs.end()) || (roleArnIter == outputs.end())) {
        std::cerr << "Exiting because output '" << SNS_TOPIC_ARN_OUTPUT <<
        "' or '" << ROLE_ARN_OUTPUT << "'not found in the CloudFormation stack."
 << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
                false,
                clientConfiguration);
        return false;
    }

    Aws::String topicArn = topicArnIter->second;
    Aws::String roleArn = roleArnIter->second;
    Aws::String sqlStatement = "SELECT * FROM '";
    sqlStatement += MQTT_MESSAGE_TOPIC_FILTER;
    sqlStatement += "'";

    printAsterisksLine();

    std::cout << "Now a rule will be created.\n" << std::endl;
    std::cout << "Rules are an administrator-level action. Any user who has
 permission\n"
                 << "to create rules will be able to access data processed by the
 rule." << std::endl;
    std::cout << "In this case, the rule will use an SNS topic" << std::endl;
    std::cout << "and the following SQL statement '" << sqlStatement << "'." <<
 std::endl;
    std::cout << "For more information on IoT SQL, see https://
docs.aws.amazon.com/iot/latest/developerguide/iot-sql-reference.html" <<
 std::endl;
    Aws::String ruleName = askQuestion("Enter a rule name: ");
    if (!createTopicRule(ruleName, topicArn, sqlStatement, roleArn,
 clientConfiguration)) {
        std::cerr << "Exiting because createRule failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
                false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();
```

```
    std::cout << "Now your rules will be listed.\n" << std::endl;
    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    if (!listTopicRules(clientConfiguration)) {
        std::cerr << "Exiting because listRules failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
                false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();
    Aws::String queryString = "thingName:" + thingName;
    std::cout << "Now the AWS IoT fleet index will be queried with the query\n'"
    << queryString << "'.\n" << std::endl;
    std::cout << "For query information, see https://docs.aws.amazon.com/iot/
latest/developerguide/query-syntax.html" << std::endl;

    std::cout << "For this query to work, thing indexing must be enabled in your
 account.\n"
    << "This can be done with the awscli command line by calling 'aws iot update-
indexing-configuration'\n"
       << "or it can be done programmatically." << std::endl;
    std::cout << "For more information, see https://docs.aws.amazon.com/iot/
latest/developerguide/managing-index.html" << std::endl;
    if (askYesNoQuestion("Do you want to enable thing indexing in your account?
 (y/n) "))
    {
        Aws::IoT::Model::ThingIndexingConfiguration thingIndexingConfiguration;

 thingIndexingConfiguration.SetThingIndexingMode(Aws::IoT::Model::ThingIndexingMode::REGI

 thingIndexingConfiguration.SetThingConnectivityIndexingMode(Aws::IoT::Model::ThingConnec
        // The ThingGroupIndexingConfiguration object is ignored if not set.
        Aws::IoT::Model::ThingGroupIndexingConfiguration
 thingGroupIndexingConfiguration;
        if (!updateIndexingConfiguration(thingIndexingConfiguration,
 thingGroupIndexingConfiguration, clientConfiguration)) {
            std::cerr << "Exiting because updateIndexingConfiguration failed." <<
 std::endl;
            cleanup(thingName, certificateARN, certificateID, STACK_NAME,
                    ruleName, false,
                    clientConfiguration);
            return false;
```

```
        }
    }

    if (!searchIndex(queryString, clientConfiguration)) {

        std::cerr << "Exiting because searchIndex failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
                false,
                clientConfiguration);
        return false;
    }
```

```
//! Update an AWS IoT thing with attributes.
/*!
  \param thingName: The name for the thing.
  \param attributeMap: A map of key/value attributes/
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
 &attributeMap,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}

//! Describe the endpoint specific to the AWS account making the call.
/*!
  \param endpointResult: String to receive the endpoint result.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
            "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome =
 iotClient.DescribeEndpoint(
            describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

//! List certificates registered in the AWS account making the call.
/*!
   \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;
```

```
    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
 iotClient.ListCertificates(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
 outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                   result.GetCertificates().begin(),
                                   result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
 std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << std::endl;
    }

    return true;
}

//! Update the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param document: The state information, in JSON format.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                    const Aws::String &document,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient
 iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
 std::make_shared<std::stringstream>(
            document);
    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
 iotDataPlaneClient.UpdateThingShadow(
            updateThingShadowRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Get the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param documentResult: String to receive the state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
                                 Aws::String &documentResult,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
```

```
    if (outcome.IsSuccess()) {
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
        std::cerr << "Error getting thing shadow: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Create an AWS IoT rule with an SNS topic as the target.
/*!
  \param ruleName: The name for the rule.
  \param snsTopic: The SNS topic ARN for the action.
  \param sql: The SQL statement used to query the topic.
  \param roleARN: The IAM role ARN for the action.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String
 &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});
```

```
    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
 std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}

//! Lists the AWS IoT topic rules.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listTopicRules(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListTopicRulesRequest request;

    Aws::Vector<Aws::IoT::Model::TopicRuleListItem> allRules;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::ListTopicRulesOutcome outcome =
 iotClient.ListTopicRules(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListTopicRulesResult &result =
 outcome.GetResult();
            allRules.insert(allRules.end(),
                            result.GetRules().cbegin(),
                            result.GetRules().cend());

            nextToken = result.GetNextToken();
        }
```

```
        else {
            std::cerr << "ListTopicRules error: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

    } while (!nextToken.empty());

    std::cout << "ListTopicRules: " << allRules.size() << " rule(s) found."
              << std::endl;
    for (auto &rule: allRules) {
        std::cout << "  Rule name: " << rule.GetRuleName() << ", rule ARN: "
                  << rule.GetRuleArn() << "." << std::endl;
    }

    return true;
}

//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
/*!
  \param: query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
 iotClient.SearchIndex(request);
```

```
        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result =
 outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
            nextToken = result.GetNextToken();

        }
        else {
            std::cerr << "Error in SearchIndex: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
 std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
                  << std::endl;
    }
    return true;
}
```

Limpe recursos.

```
bool
AwsDoc::IoT::cleanup(const Aws::String &thingName, const Aws::String
 &certificateARN,
                     const Aws::String &certificateID, const Aws::String
 &stackName,
                     const Aws::String &ruleName, bool askForConfirmation,
                     const Aws::Client::ClientConfiguration &clientConfiguration)
 {
    bool result = true;

    if (!ruleName.empty() && (!askForConfirmation ||
                              askYesNoQuestion("Delete the rule '" + ruleName +
                                               "'? (y/n) "))) {
        result &= deleteTopicRule(ruleName, clientConfiguration);
```

```
    }

    Aws::CloudFormation::CloudFormationClient
 cloudFormationClient(clientConfiguration);

    if (!stackName.empty() && (!askForConfirmation ||
                               askYesNoQuestion(
                                       "Delete the CloudFormation stack '" +
 stackName +
                                       "'? (y/n) "))) {
        result &= deleteStack(stackName, clientConfiguration);
    }

    if (!certificateARN.empty() && (!askForConfirmation ||
                                    askYesNoQuestion("Delete the certificate '" +
                                                     certificateARN + "'? (y/n)
 "))) {
        result &= detachThingPrincipal(certificateARN, thingName,
 clientConfiguration);
        result &= deleteCertificate(certificateID, clientConfiguration);
    }

    if (!thingName.empty() && (!askForConfirmation ||
                               askYesNoQuestion("Delete the thing '" + thingName
 +
                                                "'? (y/n) "))) {
        result &= deleteThing(thingName, clientConfiguration);
    }

    return result;
}
```

```
//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
```

```
                                        const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
 iotClient.DetachThingPrincipal(
           detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from
 thing "
                  << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                  << thingName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome =
 iotClient.DeleteCertificate(
           request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT rule.
/*!
  \param ruleName: The name for the rule.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Execute um cenário interativo demonstrando AWS IoT recursos.

```
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
```

```
 * This Java example performs these tasks:
 *
 * 1. Creates an AWS IoT Thing.
 * 2. Generate and attach a device certificate.
 * 3. Update an AWS IoT Thing with Attributes.
 * 4. Get an AWS IoT Endpoint.
 * 5. List your certificates.
 * 6. Updates the shadow for the specified thing..
 * 7. Write out the state information, in JSON format
 * 8. Creates a rule
 * 9. List rules
 * 10. Search things
 * 11. Detach amd delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <roleARN> <snsAction>

                Where:
                    roleARN - The ARN of an IAM role that has permission to work
 with AWS IOT.
                    snsAction  - An ARN of an SNS topic.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        IotActions iotActions = new IotActions();
        String thingName;
        String ruleName;
        String roleARN = args[0];
        String snsAction = args[1];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
```

```
        System.out.println("Welcome to the AWS IoT basics scenario.");
        System.out.println("""
            This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service. The program guides you through a series
of steps,
            including creating an IoT Thing, generating a device certificate,
updating the Thing with attributes, and so on.
            It utilizes the AWS SDK for Java V2 and incorporates functionality
for creating and managing IoT Things, certificates, rules,
            shadows, and performing searches. The program aims to showcase AWS
IoT capabilities and provides a comprehensive example for
            developers working with AWS IoT in a Java environment.

            Let's get started...

            """);
        System.out.println(DASHES);

        System.out.println("1. Create an AWS IoT Thing.");
        System.out.println("""
            An AWS IoT Thing represents a virtual entity in the AWS IoT service
that can be associated with
            a physical device.
            """);
        // Prompt the user for input.
        System.out.print("Enter Thing name: ");
        thingName = scanner.nextLine();
        iotActions.createIoTThing(thingName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Generate a device certificate.");
        System.out.println("""
            A device certificate performs a role in securing the communication
between devices (Things)
            and the AWS IoT platform.
            """);

        System.out.print("Do you want to create a certificate for " +thingName
+"? (y/n)");
        String certAns = scanner.nextLine();
        String certificateArn="" ;
        if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
            certificateArn = iotActions.createCertificate();
```

```
            System.out.println("Attach the certificate to the AWS IoT Thing.");
            iotActions.attachCertificateToThing(thingName, certificateArn);
        } else {
            System.out.println("A device certificate was not created.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Update an AWS IoT Thing with Attributes.");
        System.out.println("""
             IoT Thing attributes, represented as key-value pairs, offer a
pivotal advantage in facilitating efficient data
             management and retrieval within the AWS IoT ecosystem.
            """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Return a unique endpoint specific to the Amazon
Web Services account.");
        System.out.println("""
            An IoT Endpoint refers to a specific URL or Uniform Resource Locator
that serves as the entry point for communication between IoT devices and the AWS
IoT service.
           """);
        waitForInputToContinue(scanner);
        String endpointUrl = iotActions.describeEndpoint();
        System.out.println("The endpoint is "+endpointUrl);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List your AWS IoT certificates");
        waitForInputToContinue(scanner);
        if (certificateArn.length() > 0) {
            iotActions.listCertificates();
        } else {
            System.out.println("You did not create a certificates. Skipping this
step.");
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
        System.out.println("""
            A Thing Shadow refers to a feature that enables you to create a
virtual representation, or "shadow,"
            of a physical device or thing. The Thing Shadow allows you to
synchronize and control the state of a device between
            the cloud and the device itself. and the AWS IoT service. For
example, you can write and retrieve JSON data from a Thing Shadow.
           """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Write out the state information, in JSON
format.");
        waitForInputToContinue(scanner);
        iotActions.getPayload(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Creates a rule");
        System.out.println("""
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """);
        System.out.print("Enter Rule name: ");
        ruleName = scanner.nextLine();
        iotActions.createIoTRule(roleARN, ruleName, snsAction);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List your rules.");
        waitForInputToContinue(scanner);
        iotActions.listIoTRules();
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("10. Search things using the Thing name.");
        waitForInputToContinue(scanner);
        String queryString = "thingName:"+thingName ;
        iotActions.searchThings(queryString);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        if (certificateArn.length() > 0) {
            System.out.print("Do you want to detach and delete the certificate
for " +thingName +"? (y/n)");
            String delAns = scanner.nextLine();
            if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
                System.out.println("11. You selected to detach amd delete the
certificate.");
                waitForInputToContinue(scanner);
                iotActions.detachThingPrincipal(thingName, certificateArn);
                iotActions.deleteCertificate(certificateArn);
                waitForInputToContinue(scanner);
            } else {
                System.out.println("11. You selected not to delete the
certificate.");
            }
        } else {
            System.out.println("11. You did not create a certificate so there is
nothing to delete.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete the AWS IoT Thing.");
        System.out.print("Do you want to delete the IoT Thing? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            iotActions.deleteIoTThing(thingName);
        } else {
            System.out.println("The IoT Thing was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS IoT workflow has successfully completed.");
```

```
        System.out.println(DASHES);
    }


    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

Uma classe de invólucro para métodos do AWS IoT SDK.

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotAsyncClient;
import software.amazon.awssdk.services.iot.model.Action;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.Certificate;
import
 software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
```

```
import software.amazon.awssdk.services.iot.model.CreateTopicRuleResponse;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DeleteThingResponse;
import software.amazon.awssdk.services.iot.model.DescribeEndpointRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointResponse;
import software.amazon.awssdk.services.iot.model.DescribeThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeThingResponse;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.IotException;
import software.amazon.awssdk.services.iot.model.ListCertificatesResponse;
import software.amazon.awssdk.services.iot.model.ListTopicRulesRequest;
import software.amazon.awssdk.services.iot.model.ListTopicRulesResponse;
import software.amazon.awssdk.services.iot.model.SearchIndexRequest;
import software.amazon.awssdk.services.iot.model.SearchIndexResponse;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowResponse;
import
 software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowRequest;
import
 software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowResponse;
import java.nio.charset.StandardCharsets;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class IotActions {

    private static IotAsyncClient iotAsyncClient;

    private static IotDataPlaneAsyncClient iotAsyncDataPlaneClient;

    private static final String TOPIC = "your-iot-topic";

    private static IotDataPlaneAsyncClient getAsyncDataPlaneClient() {
```

```
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        if (iotAsyncDataPlaneClient == null) {
            iotAsyncDataPlaneClient = IotDataPlaneAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncDataPlaneClient;
    }


    private static IotAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();
```

```
        if (iotAsyncClient == null) {
            iotAsyncClient = IotAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncClient;
    }

    /**
     * Creates an IoT certificate asynchronously.
     *
     * @return The ARN of the created certificate.
     * <p>
     * This method initiates an asynchronous request to create an IoT
certificate.
     * If the request is successful, it prints the certificate details and
returns the certificate ARN.
     * If an exception occurs, it prints the error message.
     */
    public String createCertificate() {
        CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
        final String[] certificateArn = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String certificatePem = response.certificatePem();
                certificateArn[0] = response.certificateArn();

                // Print the details.
                System.out.println("\nCertificate:");
                System.out.println(certificatePem);
                System.out.println("\nCertificate ARN:");
                System.out.println(certificateArn[0]);

            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
```

```
                    System.err.println("Unexpected error: " +
cause.getMessage());
                }
            }
        });

        future.join();
        return certificateArn[0];
    }

    /**
     * Creates an IoT Thing with the specified name asynchronously.
     *
     * @param thingName The name of the IoT Thing to create.
     *
     * This method initiates an asynchronous request to create an IoT Thing with
the specified name.
     * If the request is successful, it prints the name of the thing and its ARN
value.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTThing(String thingName) {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
        future.whenComplete((createThingResponse, ex) -> {
            if (createThingResponse != null) {
                System.out.println(thingName + " was successfully created. The
ARN value is " + createThingResponse.thingArn());
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                }
            }
        });
```

```
        future.join();
    }

    /**
     * Attaches a certificate to an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to attach.
     *
     * This method initiates an asynchronous request to attach a certificate to
an IoT Thing.
     * If the request is successful, it prints a confirmation message and
additional information about the Thing.
     * If an exception occurs, it prints the error message.
     */
    public void attachCertificateToThing(String thingName, String certificateArn)
{
        AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
            .thingName(thingName)
            .principal(certificateArn)
            .build();

        CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
        future.whenComplete((attachResponse, ex) -> {
            if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
                System.out.println("Certificate attached to Thing
successfully.");

                // Print additional information about the Thing.
                describeThing(thingName);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to attach certificate to Thing.
HTTP Status Code: " +
```

```
                        attachResponse.sdkHttpResponse().statusCode());
                }
            }
        });

        future.join();
    }

    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " +
describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });
```

```
        future.join();
    }

    /**
     * Updates the shadow of an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to update the shadow of an
IoT Thing.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void updateShadowThing(String thingName) {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}";
        SdkBytes data = SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8);
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
        future.whenComplete((updateResponse, ex) -> {
            if (updateResponse != null) {
                System.out.println("Thing Shadow updated successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });
```

```
        future.join();
    }

    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of
 the IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Da
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }
```

```
    /**
     * Extracts a specific value from the endpoint URL.
     *
     * @param input The endpoint URL to process.
     * @return The extracted value from the endpoint URL.
     */
    private static String getValue(String input) {
        // Define a regular expression pattern for extracting the subdomain.
        Pattern pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\
\.com");

        // Match the pattern against the input string.
        Matcher matcher = pattern.matcher(input);

        // Check if a match is found.
        if (matcher.find()) {
            // Extract the subdomain from the first capturing group.
            String subdomain = matcher.group(1);
            System.out.println("Extracted subdomain: " + subdomain);
            return subdomain ;
        } else {
            System.out.println("No match found");
        }
        return "" ;
    }

    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
 getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
```

```
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
    }

    /**
     * Retrieves the payload of a Thing's shadow asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to get the payload of a
Thing's shadow.
     * If the request is successful, it prints the shadow data.
     * If an exception occurs, it prints the error message.
     */
    public void getPayload(String thingName) {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
        future.whenComplete((getThingShadowResponse, ex) -> {
            if (getThingShadowResponse != null) {
                // Extracting payload from response.
                SdkBytes payload = getThingShadowResponse.payload();
                String payloadString = payload.asUtf8String();
                System.out.println("Received Shadow Data: " + payloadString);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
```

```
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to get Thing Shadow payload.");
                }
            }
        });

        future.join();
    }

    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();
```

```
        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }

    /**
     * Lists IoT rules asynchronously.
     *
     * This method initiates an asynchronous request to list IoT rules.
     * If the request is successful, it prints the names and ARNs of the rules.
     * If an exception occurs, it prints the error message.
     */
    public void listIoTRules() {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        CompletableFuture<ListTopicRulesResponse> future =
getAsyncClient().listTopicRules(listTopicRulesRequest);
        future.whenComplete((listTopicRulesResponse, ex) -> {
            if (listTopicRulesResponse != null) {
                System.out.println("List of IoT Rules:");
```

```
                List<TopicRuleListItem> ruleList =
listTopicRulesResponse.rules();
                for (TopicRuleListItem rule : ruleList) {
                    System.out.println("Rule Name: " + rule.ruleName());
                    System.out.println("Rule ARN: " + rule.ruleArn());
                    System.out.println("--------------");
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to list IoT Rules.");
                }
            }
        });

        future.join();
    }

    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
```

```
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }

    /**
     * Detaches a principal (certificate) from an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to detach.
     *
     * This method initiates an asynchronous request to detach a certificate from
an IoT Thing.
     * If the detachment is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void detachThingPrincipal(String thingName, String certificateArn) {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
        future.whenComplete((voidResult, ex) -> {
```

```
            if (ex == null) {
                System.out.println(certificateArn + " was successfully removed
from " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes a certificate asynchronously.
     *
     * @param certificateArn The ARN of the certificate to delete.
     *
     * This method initiates an asynchronous request to delete a certificate.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteCertificate(String certificateArn) {
        DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
            .certificateId(extractCertificateId(certificateArn))
            .build();

        CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully
deleted.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
```

```
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    // Get the cert Id  from the Cert ARN value.
    private String extractCertificateId(String certificateArn) {
        // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
```

```
        String[] arnParts = certificateArn.split(":");
        String certificateIdPart = arnParts[arnParts.length - 1];
        return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") +
 1);
    }
}
```

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.Action
import aws.sdk.kotlin.services.iot.model.AttachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.AttributePayload
import aws.sdk.kotlin.services.iot.model.CreateThingRequest
import aws.sdk.kotlin.services.iot.model.CreateTopicRuleRequest
import aws.sdk.kotlin.services.iot.model.DeleteCertificateRequest
import aws.sdk.kotlin.services.iot.model.DeleteThingRequest
import aws.sdk.kotlin.services.iot.model.DescribeEndpointRequest
import aws.sdk.kotlin.services.iot.model.DescribeThingRequest
import aws.sdk.kotlin.services.iot.model.DetachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.ListTopicRulesRequest
import aws.sdk.kotlin.services.iot.model.SearchIndexRequest
import aws.sdk.kotlin.services.iot.model.SnsAction
import aws.sdk.kotlin.services.iot.model.TopicRulePayload
import aws.sdk.kotlin.services.iot.model.UpdateThingRequest
import aws.sdk.kotlin.services.iotdataplane.IotDataPlaneClient
import aws.sdk.kotlin.services.iotdataplane.model.GetThingShadowRequest
import aws.sdk.kotlin.services.iotdataplane.model.UpdateThingShadowRequest
import aws.smithy.kotlin.runtime.content.ByteStream
import aws.smithy.kotlin.runtime.content.toByteArray
import java.util.Scanner
import java.util.regex.Pattern
```

```
import kotlin.system.exitProcess

/**
 * Before running this Kotlin code example, ensure that your development
 environment
 * is set up, including configuring your credentials.
 *
 * For detailed instructions, refer to the following documentation topic:
 * [Setting Up Your Development Environment](https://docs.aws.amazon.com/sdk-for-
kotlin/latest/developer-guide/setup.html)
 *
 * This code example requires an SNS topic and an IAM Role.
 * Follow the steps in the documentation to set up these resources:
 *
 * - [Creating an SNS Topic](https://docs.aws.amazon.com/sns/latest/dg/sns-
getting-started.html#step-create-topic)
 * - [Creating an IAM Role](https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html)
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")
val TOPIC = "your-iot-topic"

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage:
            <roleARN> <snsAction>

        Where:
            roleARN - The ARN of an IAM role that has permission to work with AWS
 IOT.
            snsAction  - An ARN of an SNS topic.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    var thingName: String
    val roleARN = args[0]
    val snsAction = args[1]
```

```
    val scanner = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the AWS IoT example scenario.")
    println(
        """
        This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service.
        The program guides you through a series of steps, including creating an
IoT thing, generating a device certificate,
        updating the thing with attributes, and so on.

        It utilizes the AWS SDK for Kotlin and incorporates functionality for
creating and managing IoT things, certificates, rules,
        shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
        developers working with AWS IoT in a Kotlin environment.
        """.trimIndent(),
    )

    print("Press Enter to continue...")
    scanner.nextLine()
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS IoT thing.")
    println(
        """
        An AWS IoT thing represents a virtual entity in the AWS IoT service that
can be associated with a physical device.
        """.trimIndent(),
    )
    // Prompt the user for input.
    print("Enter thing name: ")
    thingName = scanner.nextLine()
    createIoTThing(thingName)
    describeThing(thingName)
    println(DASHES)

    println(DASHES)
    println("2. Generate a device certificate.")
    println(
        """
```

```
        A device certificate performs a role in securing the communication
between devices (things) and the AWS IoT platform.
        """.trimIndent(),
    )

    print("Do you want to create a certificate for $thingName? (y/n)")
    val certAns = scanner.nextLine()
    var certificateArn: String? = ""
    if (certAns != null && certAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
        certificateArn = createCertificate()
        println("Attach the certificate to the AWS IoT thing.")
        attachCertificateToThing(thingName, certificateArn)
    } else {
        println("A device certificate was not created.")
    }
    println(DASHES)

    println(DASHES)
    println("3. Update an AWS IoT thing with Attributes.")
    println(
        """
        IoT thing attributes, represented as key-value pairs, offer a pivotal
advantage in facilitating efficient data
        management and retrieval within the AWS IoT ecosystem.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateThing(thingName)
    println(DASHES)

    println(DASHES)
    println("4. Return a unique endpoint specific to the Amazon Web Services
account.")
    println(
        """
        An IoT Endpoint refers to a specific URL or Uniform Resource Locator that
serves as the entry point for communication between IoT devices and the AWS IoT
service.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
```

```
    val endpointUrl = describeEndpoint()
    println(DASHES)

    println(DASHES)
    println("5. List your AWS IoT certificates")
    print("Press Enter to continue...")
    scanner.nextLine()
    if (certificateArn!!.isNotEmpty()) {
        listCertificates()
    } else {
        println("You did not create a certificates. Skipping this step.")
    }
    println(DASHES)

    println(DASHES)
    println("6. Create an IoT shadow that refers to a digital representation or
virtual twin of a physical IoT device")
    println(
        """
        A thing shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
        of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between
        the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a thing shadow.

        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateShawdowThing(thingName)
    println(DASHES)

    println(DASHES)
    println("7. Write out the state information, in JSON format.")
    print("Press Enter to continue...")
    scanner.nextLine()
    getPayload(thingName)
    println(DASHES)

    println(DASHES)
    println("8. Creates a rule")
    println(
        """
```

```
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """.trimIndent(),
    )
    print("Enter Rule name: ")
    val ruleName = scanner.nextLine()
    createIoTRule(roleARN, ruleName, snsAction)
    println(DASHES)

    println(DASHES)
    println("9. List your rules.")
    print("Press Enter to continue...")
    scanner.nextLine()
    listIoTRules()
    println(DASHES)

    println(DASHES)
    println("10. Search things using the name.")
    print("Press Enter to continue...")
    scanner.nextLine()
    val queryString = "thingName:$thingName"
    searchThings(queryString)
    println(DASHES)

    println(DASHES)
    if (certificateArn.length > 0) {
        print("Do you want to detach and delete the certificate for $thingName?
(y/n)")
        val delAns = scanner.nextLine()
        if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
            println("11. You selected to detach amd delete the certificate.")
            print("Press Enter to continue...")
            scanner.nextLine()
            detachThingPrincipal(thingName, certificateArn)
            deleteCertificate(certificateArn)
        } else {
            println("11. You selected not to delete the certificate.")
        }
    } else {
        println("11. You did not create a certificate so there is nothing to
delete.")
    }
```

```
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS IoT thing.")
    print("Do you want to delete the IoT thing? (y/n)")
    val delAns = scanner.nextLine()
    if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
 true)) {
        deleteIoTThing(thingName)
    } else {
        println("The IoT thing was not deleted.")
    }
    println(DASHES)

    println(DASHES)
    println("The AWS IoT workflow has successfully completed.")
    println(DASHES)
}

suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}

suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}

private fun extractCertificateId(certificateArn: String): String? {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
```

```
    val arnParts = certificateArn.split(":".toRegex()).dropLastWhile
 { it.isEmpty() }.toTypedArray()
    val certificateIdPart = arnParts[arnParts.size - 1]
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1)
}

suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}

suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}

suspend fun listIoTRules() {
    val listTopicRulesRequest = ListTopicRulesRequest {}

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
```

```
        val listTopicRulesResponse =
 iotClient.listTopicRules(listTopicRulesRequest)
        println("List of IoT rules:")
        val ruleList = listTopicRulesResponse.rules
        ruleList?.forEach { rule ->
            println("Rule name: ${rule.ruleName}")
            println("Rule ARN: ${rule.ruleArn}")
            println("--------------")
        }
    }
}

suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
```

```
    }
}

suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use
 { iotPlaneClient ->
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}

suspend fun listCertificates() {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}

suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}

private fun getValue(input: String?): String {
    // Define a regular expression pattern for extracting the subdomain.
```

```
    val pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\\.com")

    // Match the pattern against the input string.
    val matcher = pattern.matcher(input)

    // Check if a match is found.
    if (matcher.find()) {
        val subdomain = matcher.group(1)
        println("Extracted subdomain: $subdomain")
        return subdomain
    } else {
        println("No match found")
    }
    return ""
}

suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}

suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
```

```
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use
 { iotPlaneClient ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}

suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}

suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
```

```
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}

suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}

suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

Crie uma classe de invólucro de IoT para gerenciar as operações.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)


    def create_thing(self, thing_name):
        """
        Creates an AWS IoT thing.

        :param thing_name: The name of the thing to create.
        :return: The name and ARN of the created thing.
        """
        try:
            response = self.iot_client.create_thing(thingName=thing_name)
            logger.info("Created thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Thing %s already exists. Skipping creation.",
 thing_name)
                return None
            logger.error(
                "Couldn't create thing %s. Here's why: %s: %s",
                thing_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

```
    def list_things(self):
        """
        Lists AWS IoT things.

        :return: The list of things.
        """
        try:
            things = []
            paginator = self.iot_client.get_paginator("list_things")
            for page in paginator.paginate():
                things.extend(page["things"])
            logger.info("Retrieved %s things.", len(things))
            return things
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list things. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise



    def create_keys_and_certificate(self):
        """
        Creates keys and a certificate for an AWS IoT thing.

        :return: The certificate ID, ARN, and PEM.
        """
        try:
            response =
self.iot_client.create_keys_and_certificate(setAsActive=True)
            logger.info("Created certificate %s.", response["certificateId"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't create keys and certificate. Here's why: %s: %s",
```

```
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response


    def attach_thing_principal(self, thing_name, principal):
        """
        Attaches a certificate to an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.attach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Attached principal %s to thing %s.", principal,
thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot attach principal. Resource not found.")
                return
            logger.error(
                "Couldn't attach principal to thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_endpoint(self, endpoint_type="iot:Data-ATS"):
        """
        Gets the AWS IoT endpoint.

        :param endpoint_type: The endpoint type.
        :return: The endpoint.
        """
        try:
            response =
self.iot_client.describe_endpoint(endpointType=endpoint_type)
            logger.info("Retrieved endpoint %s.", response["endpointAddress"])
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't describe endpoint. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["endpointAddress"]


    def list_certificates(self):
        """
        Lists AWS IoT certificates.

        :return: The list of certificates.
        """
        try:
            certificates = []
            paginator = self.iot_client.get_paginator("list_certificates")
            for page in paginator.paginate():
                certificates.extend(page["certificates"])
            logger.info("Retrieved %s certificates.", len(certificates))
            return certificates
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list certificates. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def detach_thing_principal(self, thing_name, principal):
        """
        Detaches a certificate from an AWS IoT thing.

        :param thing_name: The name of the thing.
```

```
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.detach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Detached principal %s from thing %s.", principal,
thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot detach principal. Resource not found.")
                return
            logger.error(
                "Couldn't detach principal from thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_certificate(self, certificate_id):
        """
        Deletes an AWS IoT certificate.

        :param certificate_id: The ID of the certificate to delete.
        """
        try:
            self.iot_client.update_certificate(
                certificateId=certificate_id, newStatus="INACTIVE"
            )
            self.iot_client.delete_certificate(certificateId=certificate_id)
            logger.info("Deleted certificate %s.", certificate_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete certificate. Resource not found.")
                return
            logger.error(
                "Couldn't delete certificate. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def create_topic_rule(self, rule_name, topic, sns_action_arn, role_arn):
        """
        Creates an AWS IoT topic rule.

        :param rule_name: The name of the rule.
        :param topic: The MQTT topic to subscribe to.
        :param sns_action_arn: The ARN of the SNS topic to publish to.
        :param role_arn: The ARN of the IAM role.
        """
        try:
            self.iot_client.create_topic_rule(
                ruleName=rule_name,
                topicRulePayload={
                    "sql": f"SELECT * FROM '{topic}'",
                    "actions": [
                        {"sns": {"targetArn": sns_action_arn, "roleArn":
role_arn}}
                    ],
                },
            )
            logger.info("Created topic rule %s.", rule_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Topic rule %s already exists. Skipping creation.",
rule_name)
                return
            logger.error(
                "Couldn't create topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def list_topic_rules(self):
        """
        Lists AWS IoT topic rules.

        :return: The list of topic rules.
        """
        try:
            rules = []
            paginator = self.iot_client.get_paginator("list_topic_rules")
            for page in paginator.paginate():
```

```
                rules.extend(page["rules"])
            logger.info("Retrieved %s topic rules.", len(rules))
            return rules
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list topic rules. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise



    def search_index(self, query):
        """
        Searches the AWS IoT index.

        :param query: The search query.
        :return: The list of things found.
        """
        try:
            response = self.iot_client.search_index(queryString=query)
            logger.info("Found %s things.", len(response.get("things", [])))
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't search index. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response.get("things", [])


    def update_indexing_configuration(self):
        """
        Updates the AWS IoT indexing configuration to enable thing indexing.
        """
```

```
        try:
            self.iot_client.update_indexing_configuration(
                thingIndexingConfiguration={"thingIndexingMode": "REGISTRY"}
            )
            logger.info("Updated indexing configuration.")
        except ClientError as err:
            logger.error(
                "Couldn't update indexing configuration. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_thing(self, thing_name):
        """
        Deletes an AWS IoT thing.

        :param thing_name: The name of the thing to delete.
        """
        try:
            self.iot_client.delete_thing(thingName=thing_name)
            logger.info("Deleted thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete thing. Resource not found.")
                return
            logger.error(
                "Couldn't delete thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_topic_rule(self, rule_name):
        """
        Deletes an AWS IoT topic rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.iot_client.delete_topic_rule(ruleName=rule_name)
            logger.info("Deleted topic rule %s.", rule_name)
```

```
        except ClientError as err:
            logger.error(
                "Couldn't delete topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update_thing_shadow(self, thing_name, shadow_state):
        """
        Updates the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param shadow_state: The shadow state as a dictionary.
        """
        import json
        try:
            self.iot_data_client.update_thing_shadow(
                thingName=thing_name, payload=json.dumps(shadow_state)
            )
            logger.info("Updated shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot update thing shadow. Resource not found.")
                return
            logger.error(
                "Couldn't update thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_thing_shadow(self, thing_name):
        """
        Gets the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :return: The shadow state as a dictionary.
        """
        import json
        try:
```

```
            response =
 self.iot_data_client.get_thing_shadow(thingName=thing_name)
            shadow = json.loads(response["payload"].read())
            logger.info("Retrieved shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot get thing shadow. Resource not found.")
                return None
            logger.error(
                "Couldn't get thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return shadow
```

Execute um cenário interativo demonstrando os fundamentos da IoT.

```
class IoTScenario:
    """Runs an interactive scenario that shows how to use AWS IoT."""

    is_interactive = True

    def __init__(self, iot_wrapper, iot_data_client, cfn_client,
 stack_name="IoTBasicsStack", template_path=None):
        """
        :param iot_wrapper: An instance of the IoTWrapper class.
        :param iot_data_client: A Boto3 IoT Data Plane client.
        :param cfn_client: A Boto3 CloudFormation client.
        :param stack_name: Name for the CloudFormation stack.
        :param template_path: Path to the CloudFormation template file.
        """
        self.iot_wrapper = iot_wrapper
        self.iot_data_client = iot_data_client
        self.cfn_client = cfn_client
        self.thing_name = None
        self.certificate_arn = None
        self.certificate_id = None
        self.rule_name = None
        self.stack_name = stack_name
```

```
        self.template_path = template_path or "../../../scenarios/basics/iot/
iot_usecase/resources/cfn_template.yaml"

    def _deploy_stack(self):
        """Deploy CloudFormation stack and return outputs."""
        with open(self.template_path, "r") as f:
            template_body = f.read()

        try:
            self.cfn_client.create_stack(
                StackName=self.stack_name,
                TemplateBody=template_body,
                Capabilities=["CAPABILITY_NAMED_IAM"]
            )

            waiter = self.cfn_client.get_waiter("stack_create_complete")
            waiter.wait(StackName=self.stack_name)

            response = self.cfn_client.describe_stacks(StackName=self.stack_name)
            outputs = {output["OutputKey"]: output["OutputValue"]
                       for output in response["Stacks"][0]["Outputs"]}
            return outputs["SNSTopicArn"], outputs["RoleArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                response =
 self.cfn_client.describe_stacks(StackName=self.stack_name)
                outputs = {output["OutputKey"]: output["OutputValue"]
                           for output in response["Stacks"][0]["Outputs"]}
                return outputs["SNSTopicArn"], outputs["RoleArn"]
            raise

    def _cleanup_stack(self):
        """Delete CloudFormation stack."""
        try:
            self.cfn_client.delete_stack(StackName=self.stack_name)
            waiter = self.cfn_client.get_waiter("stack_delete_complete")
            waiter.wait(StackName=self.stack_name)
            print("CloudFormation stack deleted successfully.")
        except ClientError as err:
            logger.error(f"Failed to delete stack: {err}")

    def run_scenario(self, thing_name, rule_name):
        """
        Runs the IoT basics scenario.
```

```
        :param thing_name: The name of the thing to create.
        :param rule_name: The name of the topic rule to create.
        """
        print("-" * 88)
        print("Welcome to the AWS IoT basics scenario!")
        print("-" * 88)
        print(
            "This scenario demonstrates how to interact with AWS IoT using the
AWS SDK for Python (Boto3).\n"
            "AWS IoT provides secure, bi-directional communication between
Internet-connected devices\n"
            "and the AWS cloud. You can manage device connections, process device
data, and build IoT applications.\n"
        )

        self.thing_name = thing_name
        self.rule_name = rule_name

        try:
            print("\nDeploying CloudFormation stack...")
            sns_topic_arn, role_arn = self._deploy_stack()
            print(f"Stack deployed. SNS Topic: {sns_topic_arn}")

            input("\nNext, we'll create an AWS IoT thing. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("1. Create an AWS IoT thing")
            print("-" * 88)
            response = self.iot_wrapper.create_thing(thing_name)
            print(f"Created thing: {response['thingName']}")
            print(f"Thing ARN: {response['thingArn']}")

            input("\nNext, we'll list things. Press Enter to continue...") if
self.is_interactive else None
            print("\n" + "-" * 88)
            print("2. List things")
            print("-" * 88)
            things = self.iot_wrapper.list_things()
            print(f"Found {len(things)} thing(s) in your account")
            for thing in things[:5]:  # Show first 5
                print(f"  Thing name: {thing['thingName']}")
```

```
            input("\nNext, we'll generate a device certificate. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("3. Generate a device certificate")
            print("-" * 88)
            cert_response = self.iot_wrapper.create_keys_and_certificate()
            self.certificate_arn = cert_response["certificateArn"]
            self.certificate_id = cert_response["certificateId"]
            print(f"Created certificate: {self.certificate_id}")

            input("\nNext, we'll attach the certificate to the thing. Press Enter
to continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("4. Attach the certificate to the thing")
            print("-" * 88)
            self.iot_wrapper.attach_thing_principal(thing_name,
self.certificate_arn)
            print(f"Attached certificate to thing: {thing_name}")

            input("\nNext, we'll update the thing shadow. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("5. Update the thing shadow")
            print("-" * 88)
            shadow_state = {"state": {"reported": {"temperature": 25, "humidity":
50}}}
            self.iot_wrapper.update_thing_shadow(thing_name, shadow_state)
            print(f"Updated shadow for thing: {thing_name}")

            input("\nNext, we'll get the thing shadow. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("6. Get the thing shadow")
            print("-" * 88)
            shadow = self.iot_wrapper.get_thing_shadow(thing_name)
            print(f"Shadow state: {json.dumps(shadow['state'], indent=2)}")

            input("\nNext, we'll get the AWS IoT endpoint. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("7. Get the AWS IoT endpoint")
            print("-" * 88)
            endpoint = self.iot_wrapper.describe_endpoint()
            print(f"IoT endpoint: {endpoint}")
```

```
            input("\nNext, we'll list certificates. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("8. List certificates")
            print("-" * 88)
            certificates = self.iot_wrapper.list_certificates()
            print(f"Found {len(certificates)} certificate(s)")
            for cert in certificates:
                print(f"  Certificate ID: {cert['certificateId']}")
                print(f"  Certificate ARN: {cert['certificateArn']}")
                print(f"  Status: {cert['status']}")

            input("\nNext, we'll create a topic rule. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("9. Create a topic rule")
            print("-" * 88)
            self.iot_wrapper.create_topic_rule(
                rule_name, f"device/{thing_name}/data", sns_topic_arn, role_arn
            )
            print(f"Created topic rule: {rule_name}")

            input("\nNext, we'll list topic rules. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("10. List topic rules")
            print("-" * 88)
            rules = self.iot_wrapper.list_topic_rules()
            print(f"Found {len(rules)} topic rule(s)")
            for rule in rules:
                print(f"  Rule name: {rule['ruleName']}")
                print(f"  Rule ARN: {rule['ruleArn']}")

            input("\nNext, we'll configure thing indexing. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("11. Configure thing indexing")
            print("-" * 88)
            self.iot_wrapper.update_indexing_configuration()
            print("Enabled thing indexing")
            print("Waiting for indexing to be ready...")
            time.sleep(10)
```

```
            input("\nNext, we'll search for things. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("12. Search for things")
            print("-" * 88)
            try:
                things = self.iot_wrapper.search_index(f"thingName:{thing_name}")
                if things:
                    print(f"Found {len(things)} thing(s) matching the query")
                    for thing in things:
                        print(f"  Thing name: {thing.get('thingName', 'N/A')}")
                        print(f"  Thing ID: {thing.get('thingId', 'N/A')}")
                else:
                    print("No things found. Indexing may take a few minutes.")
            except ClientError as err:
                if err.response["Error"]["Code"] in [
                    "IndexNotReadyException",
                    "InvalidRequestException",
                ]:
                    print("Search index not ready yet. This is expected.")
                else:
                    raise

        except ClientError as err:
            logger.error(
                "Scenario failed: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        finally:
            self._cleanup()

    def _cleanup(self):
        """Cleans up resources created during the scenario."""
        if not self.thing_name:
            return

        print("\n" + "-" * 88)
        print("Cleanup")
        print("-" * 88)

        if q.ask("Do you want to delete the resources? (y/n) ", q.is_yesno):
            try:
```

```
                if self.certificate_arn:
                    print(f"Detaching certificate from thing: {self.thing_name}")
                    self.iot_wrapper.detach_thing_principal(
                        self.thing_name, self.certificate_arn
                    )

                if self.certificate_id:
                    print(f"Deleting certificate: {self.certificate_id}")
                    self.iot_wrapper.delete_certificate(self.certificate_id)

                if self.thing_name:
                    print(f"Deleting thing: {self.thing_name}")
                    self.iot_wrapper.delete_thing(self.thing_name)

                if self.rule_name:
                    print(f"Deleting topic rule: {self.rule_name}")
                    self.iot_wrapper.delete_topic_rule(self.rule_name)

                self._cleanup_stack()
                print("Resources deleted successfully.")
            except ClientError as err:
                logger.error(
                    "Cleanup failed: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
        else:
            print("Resources will remain in your account.")

        print("\n" + "-" * 88)
        print("Thanks for using AWS IoT!")
        print("-" * 88)
```

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Ações para AWS IoT usar AWS SDKs

Os exemplos de código a seguir demonstram como realizar AWS IoT ações individuais com AWS SDKs. Cada exemplo inclui um link para GitHub, onde você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a Referência de APIs do AWS IoT.

Exemplos

- Use AttachThingPrincipal com um AWS SDK ou CLI
- Use CreateKeysAndCertificate com um AWS SDK ou CLI
- Use CreateThing com um AWS SDK ou CLI
- Use CreateTopicRule com um AWS SDK ou CLI
- Use DeleteCertificate com um AWS SDK ou CLI
- Use DeleteThing com um AWS SDK ou CLI
- Use DeleteTopicRule com um AWS SDK ou CLI
- Use DescribeEndpoint com um AWS SDK ou CLI
- Use DescribeThing com um AWS SDK ou CLI
- Use DetachThingPrincipal com um AWS SDK ou CLI
- Use ListCertificates com um AWS SDK ou CLI
- Use ListThings com um AWS SDK ou CLI
- Use SearchIndex com um AWS SDK ou CLI
- Use UpdateIndexingConfiguration com um AWS SDK ou CLI
- Use UpdateThing com um AWS SDK ou CLI

Use **`AttachThingPrincipal`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `AttachThingPrincipal`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Attaches a certificate to an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to attach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> AttachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
        try
        {
            var request = new AttachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.AttachThingPrincipalAsync(request);
            _logger.LogInformation($"Attached certificate {certificateArn} to
 Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot attach certificate - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError($"Couldn't attach certificate to Thing. Here's why:
 {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte AttachThingPrincipala Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Attach a principal to an AWS IoT thing.
/*!
  \param principal: A principal to attach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
 client.AttachThingPrincipal(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
```

```
    else {
        std::cerr << "Failed to attach principal to thing." <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte AttachThingPrincipala Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para anexar um certificado a um item

O `attach-thing-principal` exemplo a seguir anexa um certificado à MyTemperatureSensor coisa. O certificado é identificado por um ARN. Você pode encontrar o ARN de um certificado no console de IoT AWS .

```
aws iot attach-thing-principal \
    --thing-name MyTemperatureSensor \
    --principal arn:aws:iot:us-
west-2:123456789012:cert/2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8
```

Este comando não produz saída.

Para acessar mais informações, consulte Como gerenciar objetos com o registro no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte AttachThingPrincipalem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Attaches a certificate to an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to attach.
    *
    * This method initiates an asynchronous request to attach a certificate to
an IoT Thing.
    * If the request is successful, it prints a confirmation message and
additional information about the Thing.
    * If an exception occurs, it prints the error message.
    */
   public void attachCertificateToThing(String thingName, String certificateArn)
{
       AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
           .thingName(thingName)
           .principal(certificateArn)
           .build();

       CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
       future.whenComplete((attachResponse, ex) -> {
           if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
               System.out.println("Certificate attached to Thing
successfully.");

               // Print additional information about the Thing.
               describeThing(thingName);
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
```

```
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to attach certificate to Thing.
HTTP Status Code: " +
                        attachResponse.sdkHttpResponse().statusCode());
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte AttachThingPrincipala Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }
```

```
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

- Para obter detalhes da API, consulte a AttachThingPrincipalreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def attach_thing_principal(self, thing_name, principal):
        """
        Attaches a certificate to an AWS IoT thing.
```

```
        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.attach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Attached principal %s to thing %s.", principal,
thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot attach principal. Resource not found.")
                return
            logger.error(
                "Couldn't attach principal to thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a AttachThingPrincipalReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`CreateKeysAndCertificate`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateKeysAndCertificate`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates a device certificate for AWS IoT.
    /// </summary>
    /// <returns>The certificate details including ARN and certificate PEM, or
null if creation failed.</returns>
    public async Task<(string CertificateArn, string CertificatePem, string
CertificateId)?> CreateKeysAndCertificateAsync()
    {
        try
        {
            var request = new CreateKeysAndCertificateRequest
            {
                SetAsActive = true
            };

            var response = await
_amazonIoT.CreateKeysAndCertificateAsync(request);
            _logger.LogInformation($"Created certificate with ARN
{response.CertificateArn}");
            return (response.CertificateArn, response.CertificatePem,
response.CertificateId);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create certificate. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte CreateKeysAndCertificatea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if
 provided.
/*!
  \param outputFolder: Location for storing output in files, ignored when string
 is empty.
  \param certificateARNResult: A string to receive the ARN of the created
 certificate.
  \param certificateID: A string to receive the ID of the created certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
 createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
            client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ",
 certificate ID: "
                  << certificateID << std::endl;

        if (!outputFolder.empty()) {
```

```
            std::cout << "Writing certificate and keys to the folder '" <<
 outputFolder
                      << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;

            Aws::String certificateFilePath = outputFolder + "/
certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
 certificateFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            certificateFile << outcome.GetResult().GetCertificatePem();
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
 outcome.GetResult().GetKeyPair();

            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);
            if (!privateKeyFile.is_open()) {
                std::cerr << "Error opening private key file, '" <<
 privateKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            privateKeyFile << keyPair.GetPrivateKey();
            privateKeyFile.close();

            Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
            std::ofstream publicKeyFile(publicKeyFilePath);
            if (!publicKeyFile.is_open()) {
                std::cerr << "Error opening public key file, '" <<
 publicKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            publicKeyFile << keyPair.GetPublicKey();
        }
```

```
    }
    else {
        std::cerr << "Error creating keys and certificate: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateKeysAndCertificatea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para criar um par de chaves RSA e emitir um certificado X.509

O `create-keys-and-certificate` a seguir cria um par de chaves RSA de 2048 bits e emite um certificado X.509 usando a chave pública emitida. Como essa é a única vez que a AWS IoT fornece a chave privada para esse certificado, certifique-se de mantê-lo em um local seguro.

```
aws iot create-keys-and-certificate \
    --certificate-pem-outfile "myTest.cert.pem" \
    --public-key-outfile "myTest.public.key" \
    --private-key-outfile "myTest.private.key"
```

Saída:

```
{
    "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2
    "certificateId":
 "9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
    "certificatePem": "
-----BEGIN CERTIFICATE-----
MIICiTCCEXAMPLE6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgEXAMPLEAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSEXAMPLE2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYEXAMPLEb20wHhcNMTEwNDI1MjA0NTIxWhcN
```

```
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCEXAMPLEJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAEXAMPLEsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEXAMPLE25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+aEXAMPLE
EXAMPLEfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZEXAMPLELG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAEXAMPLEWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9qEXAMPLEyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDEXAMPLEBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----\n",
    "keyPair": {
        "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkEXAMPLEQEFAAOCAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h\nMMEXAMPLEuuN/
dMAS3fyce8DW/4+EXAMPLEyjmoF/YVF/gHr99VEEXAMPLE5VF13\n59VK7cEXAMPLE67GK+y
+jikqXOgHh/xJTwo
+sGpWEXAMPLEDz18xOd2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkR7qkEXAMPLE1DR6sx2HocliOOLtu6Fkw91swQ
\GB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEvp9mQOUXP6plfgxwKRX2fEXAMPLEDa
\nhJLXkX3rHU2xbxJSq7D+XEXAMPLEcw+LyFhI5mgFRl88eGdsAEXAMPLElnI9EesG\nFQIDAQAB
\n-----END PUBLIC KEY-----\n",
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\nkey omittted for security
 reasons\n-----END RSA PRIVATE KEY-----\n"
    }
}
```

Para obter mais informações, consulte Criar e registrar um certificado de dispositivo de AWS IoT no Guia do desenvolvedor de AWS IoT.

- Para obter detalhes da API, consulte CreateKeysAndCertificateem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
```

```
    * Creates an IoT certificate asynchronously.
    *
    * @return The ARN of the created certificate.
    * <p>
    * This method initiates an asynchronous request to create an IoT
certificate.
    * If the request is successful, it prints the certificate details and
returns the certificate ARN.
    * If an exception occurs, it prints the error message.
    */
   public String createCertificate() {
       CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
       final String[] certificateArn = {null};
       future.whenComplete((response, ex) -> {
           if (response != null) {
               String certificatePem = response.certificatePem();
               certificateArn[0] = response.certificateArn();

               // Print the details.
               System.out.println("\nCertificate:");
               System.out.println(certificatePem);
               System.out.println("\nCertificate ARN:");
               System.out.println(certificateArn[0]);

           } else {
               Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " +
cause.getMessage());
               }
           }
       });

       future.join();
       return certificateArn[0];
   }
```

- Para obter detalhes da API, consulte CreateKeysAndCertificatea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}
```

- Para obter detalhes da API, consulte a CreateKeysAndCertificatereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_keys_and_certificate(self):
        """
        Creates keys and a certificate for an AWS IoT thing.

        :return: The certificate ID, ARN, and PEM.
        """
        try:
            response =
 self.iot_client.create_keys_and_certificate(setAsActive=True)
            logger.info("Created certificate %s.", response["certificateId"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
```

```
                    "Couldn't create keys and certificate. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a CreateKeysAndCertificateReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`CreateThing`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateThing`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates an AWS IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to create.</param>
    /// <returns>The ARN of the Thing created, or null if creation failed.</
returns>
    public async Task<string?> CreateThingAsync(string thingName)
    {
        try
        {
```

```
            var request = new CreateThingRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIoT.CreateThingAsync(request);
            _logger.LogInformation($"Created Thing {thingName} with ARN
{response.ThingArn}");
            return response.ThingArn;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Thing {thingName} already exists:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create Thing {thingName}. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte CreateThinga Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
```

```
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
            createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte CreateThinga Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: criar um registro de item no registro

O `create-thing` exemplo a seguir cria uma entrada para um dispositivo no registro de coisas da AWS IoT.

```
aws iot create-thing \
    --thing-name SampleIoTThing
```

Saída:

```
{
    "thingName": "SampleIoTThing",
    "thingArn": "arn:aws:iot:us-west-2: 123456789012:thing/SampleIoTThing",
    "thingId": " EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE "
```

```
}
```

Exemplo 2: definir um item associado a um tipo de item

O exemplo `create-thing` a seguir cria um item com o tipo de item e seus atributos especificados.

```
aws iot create-thing \
    --thing-name "MyLightBulb" \
    --thing-type-name "LightBulb" \
    --attribute-payload "{"attributes": {"wattage":"75", "model":"123"}}"
```

Saída:

```
{
    "thingName": "MyLightBulb",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
    "thingId": "40da2e73-c6af-406e-b415-15acae538797"
}
```

Para obter mais informações, consulte Como gerenciar itens com o registro e Tipos de itens no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte CreateThingem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates an IoT Thing with the specified name asynchronously.
     *
     * @param thingName The name of the IoT Thing to create.
     *
```

```
    * This method initiates an asynchronous request to create an IoT Thing with
the specified name.
    * If the request is successful, it prints the name of the thing and its ARN
value.
    * If an exception occurs, it prints the error message.
    */
   public void createIoTThing(String thingName) {
       CreateThingRequest createThingRequest = CreateThingRequest.builder()
           .thingName(thingName)
           .build();

       CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
       future.whenComplete((createThingResponse, ex) -> {
           if (createThingResponse != null) {
               System.out.println(thingName + " was successfully created. The
ARN value is " + createThingResponse.thingArn());
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " +
cause.getMessage());
               }
           }
       });

       future.join();
   }
```

- Para obter detalhes da API, consulte CreateThinga Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- Para obter detalhes da API, consulte a CreateThingreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
```

```
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_thing(self, thing_name):
        """
        Creates an AWS IoT thing.

        :param thing_name: The name of the thing to create.
        :return: The name and ARN of the created thing.
        """
        try:
            response = self.iot_client.create_thing(thingName=thing_name)
            logger.info("Created thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Thing %s already exists. Skipping creation.",
thing_name)
                return None
            logger.error(
                "Couldn't create thing %s. Here's why: %s: %s",
                thing_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Para obter detalhes da API, consulte a CreateThingReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`CreateTopicRule`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `CreateTopicRule`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Creates an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="snsTopicArn">The ARN of the SNS topic for the action.</
param>
    /// <param name="roleArn">The ARN of the IAM role.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> CreateTopicRuleAsync(string ruleName, string
 snsTopicArn, string roleArn)
    {
        try
        {
            var request = new CreateTopicRuleRequest
            {
                RuleName = ruleName,
                TopicRulePayload = new TopicRulePayload
                {
                    Sql = "SELECT * FROM 'topic/subtopic'",
                    Description = $"Rule created by .NET example: {ruleName}",
                    Actions = new List<Amazon.IoT.Model.Action>
                    {
                        new Amazon.IoT.Model.Action
                        {
```

```
                            Sns = new SnsAction
                            {
                                TargetArn = snsTopicArn,
                                RoleArn = roleArn
                            }
                        }
                    },
                    RuleDisabled = false
                }
            };

            await _amazonIoT.CreateTopicRuleAsync(request);
            _logger.LogInformation($"Created IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Rule {ruleName} already exists: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte CreateTopicRulea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Create an AWS IoT rule with an SNS topic as the target.
/*!
  \param ruleName: The name for the rule.
  \param snsTopic: The SNS topic ARN for the action.
  \param sql: The SQL statement used to query the topic.
  \param roleARN: The IAM role ARN for the action.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String
 &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});

    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
 std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
```

```
}
```

- Para obter detalhes da API, consulte CreateTopicRulea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para criar uma regra que envie um alerta do Amazon SNS

O exemplo `create-topic-rule` a seguir cria uma regra que envia uma mensagem do Amazon SNS quando as leituras do nível de umidade do solo, encontradas em uma sombra do dispositivo, estão baixas.

```
aws iot create-topic-rule \
    --rule-name "LowMoistureRule" \
    --topic-rule-payload file://plant-rule.json
```

O exemplo exige que o código JSON a seguir seja salvo em um arquivo chamado `plant-rule.json`:

```
{
    "sql": "SELECT * FROM '$aws/things/MyRPi/shadow/update/accepted' WHERE
 state.reported.moisture = 'low'\n",
    "description": "Sends an alert whenever soil moisture level readings are too
 low.",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [{
            "sns": {
                "targetArn": "arn:aws:sns:us-
west-2:123456789012:MyRPiLowMoistureTopic",
                "roleArn": "arn:aws:iam::123456789012:role/service-role/
MyRPiLowMoistureTopicRole",
                "messageFormat": "RAW"
            }
    }]
}
```

Este comando não produz saída.

Para obter mais informações, consulte Criação de uma regra de AWS IoT no Guia para desenvolvedores de AWS IoT.

- Para obter detalhes da API, consulte CreateTopicRuleem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();
```

```
        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte CreateTopicRulea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
```

```
    }
}
```

- Para obter detalhes da API, consulte a CreateTopicRulereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_topic_rule(self, rule_name, topic, sns_action_arn, role_arn):
        """
        Creates an AWS IoT topic rule.

        :param rule_name: The name of the rule.
        :param topic: The MQTT topic to subscribe to.
        :param sns_action_arn: The ARN of the SNS topic to publish to.
```

```
        :param role_arn: The ARN of the IAM role.
        """
        try:
            self.iot_client.create_topic_rule(
                ruleName=rule_name,
                topicRulePayload={
                    "sql": f"SELECT * FROM '{topic}'",
                    "actions": [
                        {"sns": {"targetArn": sns_action_arn, "roleArn":
role_arn}}
                    ],
                },
            )
            logger.info("Created topic rule %s.", rule_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Topic rule %s already exists. Skipping creation.",
rule_name)
                return
            logger.error(
                "Couldn't create topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a CreateTopicRuleReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DeleteCertificate`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteCertificate`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Deletes an IoT certificate.
    /// </summary>
    /// <param name="certificateId">The ID of the certificate to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteCertificateAsync(string certificateId)
    {
        try
        {
            // First, update the certificate to inactive state
            var updateRequest = new UpdateCertificateRequest
            {
                CertificateId = certificateId,
                NewStatus = CertificateStatus.INACTIVE
            };
            await _amazonIoT.UpdateCertificateAsync(updateRequest);

            // Then delete the certificate
            var deleteRequest = new DeleteCertificateRequest
            {
                CertificateId = certificateId
            };

            await _amazonIoT.DeleteCertificateAsync(deleteRequest);
            _logger.LogInformation($"Deleted certificate {certificateId}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete certificate - resource not found:
{ex.Message}");
            return false;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete certificate. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteCertificatea Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome =
 iotClient.DeleteCertificate(
            request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteCertificatea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para excluir um certificado de um dispositivo

O exemplo de `delete-certificate` a seguir exclui o certificado do dispositivo com o ID especificado.

```
aws iot delete-certificate \
    --certificate-
id c0c57bbc8baaf4631a9a0345c957657f5e710473e3ddbee1428d216d54d53ac9
```

Este comando não produz saída.

Para obter mais informações, consulte a DeleteCertificateReferência da API de AWS IoT.

- Para obter detalhes da API, consulte DeleteCertificateem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /**
    * Deletes a certificate asynchronously.
    *
    * @param certificateArn The ARN of the certificate to delete.
    *
    * This method initiates an asynchronous request to delete a certificate.
    * If the deletion is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void deleteCertificate(String certificateArn) {
       DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
           .certificateId(extractCertificateId(certificateArn))
           .build();

       CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
       future.whenComplete((voidResult, ex) -> {
           if (ex == null) {
               System.out.println(certificateArn + " was successfully
deleted.");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " + ex.getMessage());
               }
           }
       });
```

```
        future.join();
    }
```

- Para obter detalhes da API, consulte DeleteCertificatea Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

- Para obter detalhes da API, consulte a DeleteCertificatereferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_certificate(self, certificate_id):
        """
        Deletes an AWS IoT certificate.

        :param certificate_id: The ID of the certificate to delete.
        """
        try:
            self.iot_client.update_certificate(
                certificateId=certificate_id, newStatus="INACTIVE"
            )
            self.iot_client.delete_certificate(certificateId=certificate_id)
            logger.info("Deleted certificate %s.", certificate_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete certificate. Resource not found.")
```

```
            return
        logger.error(
            "Couldn't delete certificate. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Para obter detalhes da API, consulte a DeleteCertificateReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DeleteThing`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteThing`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
/// <summary>
/// Deletes an IoT Thing.
/// </summary>
/// <param name="thingName">The name of the Thing to delete.</param>
/// <returns>True if successful, false otherwise.</returns>
public async Task<bool> DeleteThingAsync(string thingName)
{
    try
    {
        var request = new DeleteThingRequest
```

```
            {
                ThingName = thingName
            };

            await _amazonIoT.DeleteThingAsync(request);
            _logger.LogInformation($"Deleted Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete Thing - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete Thing. Here's why: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DeleteThinga Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
```

```
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DeleteThinga Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como exibir informações detalhadas sobre um item

O `delete-thing` exemplo a seguir exclui algo do registro de AWS IoT da sua conta. AWS

era delete-thing --thing-name” FourthBulb

Este comando não produz saída.

Para acessar mais informações, consulte Como gerenciar objetos com o registro no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte DeleteThingem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte DeleteThinga Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- Para obter detalhes da API, consulte a DeleteThingreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_thing(self, thing_name):
        """
        Deletes an AWS IoT thing.

        :param thing_name: The name of the thing to delete.
        """
        try:
            self.iot_client.delete_thing(thingName=thing_name)
            logger.info("Deleted thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete thing. Resource not found.")
                return
            logger.error(
                "Couldn't delete thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteThingReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DeleteTopicRule`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DeleteTopicRule`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Delete an AWS IoT rule.
/*!
  \param ruleName: The name for the rule.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- Para obter detalhes da API, consulte DeleteTopicRulea Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para excluir uma regra

O exemplo de `delete-topic-rule` a seguir exclui a regra especificada.

```
aws iot delete-topic-rule \
    --rule-name "LowMoistureRule"
```

Este comando não produz saída.

Para obter mais informações, consulte Excluir uma regra no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte DeleteTopicRuleem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
```

```
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_topic_rule(self, rule_name):
        """
        Deletes an AWS IoT topic rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.iot_client.delete_topic_rule(ruleName=rule_name)
            logger.info("Deleted topic rule %s.", rule_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DeleteTopicRuleReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DescribeEndpoint`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeEndpoint`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Gets the AWS IoT endpoint URL.
    /// </summary>
    /// <returns>The endpoint URL, or null if retrieval failed.</returns>
    public async Task<string?> DescribeEndpointAsync()
    {
        try
        {
            var request = new DescribeEndpointRequest
            {
                EndpointType = "iot:Data-ATS"
            };

            var response = await _amazonIoT.DescribeEndpointAsync(request);
            _logger.LogInformation($"Retrieved endpoint:
{response.EndpointAddress}");
            return response.EndpointAddress;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't describe endpoint. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- Para obter detalhes da API, consulte DescribeEndpointa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe the endpoint specific to the AWS account making the call.
/*!
  \param endpointResult: String to receive the endpoint result.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
            "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome =
 iotClient.DescribeEndpoint(
            describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeEndpointa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: Para obter seu AWS endpoint atual

O `describe-endpoint` exemplo a seguir recupera o AWS endpoint padrão ao qual todos os comandos são aplicados.

```
aws iot describe-endpoint
```

Saída:

```
{
    "endpointAddress": "abc123defghijk.iot.us-west-2.amazonaws.com"
}
```

Para obter mais informações, consulte o DescribeEndpointGuia do desenvolvedor de AWS IoT.

Exemplo 2: como obter o endpoint do ATS

O exemplo de `describe-endpoint` a seguir recupera o endpoint do Amazon Trust Services (ATS).

```
aws iot describe-endpoint \
    --endpoint-type iot:Data-ATS
```

Saída:

```
{
```

```
    "endpointAddress": "abc123defghijk-ats.iot.us-west-2.amazonaws.com"
}
```

Para obter mais informações, consulte Certificados X.509 e IoT AWS no Guia do desenvolvedor de AWS IoT.

- Para obter detalhes da API, consulte DescribeEndpointem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of
 the IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Da
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";
```

```
                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }
```

- Para obter detalhes da API, consulte DescribeEndpointa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
```

```
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}
```

- Para obter detalhes da API, consulte a DescribeEndpointreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def describe_endpoint(self, endpoint_type="iot:Data-ATS"):
        """
        Gets the AWS IoT endpoint.

        :param endpoint_type: The endpoint type.
```

```
        :return: The endpoint.
        """
        try:
            response =
self.iot_client.describe_endpoint(endpointType=endpoint_type)
            logger.info("Retrieved endpoint %s.", response["endpointAddress"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't describe endpoint. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["endpointAddress"]
```

- Para obter detalhes da API, consulte a DescribeEndpointReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_address(client: &Client, endpoint_type: &str) -> Result<(), Error>
 {
    let resp = client
        .describe_endpoint()
        .endpoint_type(endpoint_type)
        .send()
        .await?;
```

```
    println!("Endpoint address: {}", resp.endpoint_address.unwrap());

    println!();

    Ok(())
}
```

- Para obter detalhes da API, consulte a DescribeEndpointreferência da API AWS SDK for Rust.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DescribeThing`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DescribeThing`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Describe an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeThing(const Aws::String &thingName,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
```

```
    Aws::IoT::Model::DescribeThingRequest request;
    request.SetThingName(thingName);

    Aws::IoT::Model::DescribeThingOutcome outcome =
iotClient.DescribeThing(request);

    if (outcome.IsSuccess()) {
        const Aws::IoT::Model::DescribeThingResult &result = outcome.GetResult();
        std::cout << "Retrieved thing '" << result.GetThingName() << "'" <<
std::endl;
        std::cout << "thingArn: " << result.GetThingArn() << std::endl;
        std::cout << result.GetAttributes().size() << " attribute(s) retrieved"
                  << std::endl;
        for (const auto &attribute: result.GetAttributes()) {
            std::cout << "  attribute: " << attribute.first << "=" <<
attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error describing thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DescribeThinga Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como exibir informações detalhadas sobre um item

O `describe-thing` exemplo a seguir exibe informações sobre uma coisa (dispositivo) definida no registro de AWS IoT da sua AWS conta.

como descrever a coisa --nome da coisa” MyLightBulb

Saída:

```
{
    "defaultClientId": "MyLightBulb",
    "thingName": "MyLightBulb",
    "thingId": "40da2e73-c6af-406e-b415-15acae538797",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
    "thingTypeName": "LightBulb",
    "attributes": {
        "model": "123",
        "wattage": "75"
    },
    "version": 1
}
```

Para acessar mais informações, consulte Como gerenciar objetos com o registro no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte DescribeThingem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();
```

```
        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " +
describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte DescribeThinga Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun describeThing(thingNameVal: String) {
```

```
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}
```

- Para obter detalhes da API, consulte a DescribeThingreferência da API AWS SDK for Kotlin.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`DetachThingPrincipal`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `DetachThingPrincipal`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Detaches a certificate from an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to detach.</
param>
```

```
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DetachThingPrincipalAsync(string thingName, string
certificateArn)
    {
        try
        {
            var request = new DetachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.DetachThingPrincipalAsync(request);
            _logger.LogInformation($"Detached certificate {certificateArn} from
Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot detach certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't detach certificate from Thing. Here's
why: {ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte DetachThingPrincipala Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
 iotClient.DetachThingPrincipal(
            detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from
 thing "
                  << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                  << thingName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte DetachThingPrincipala Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para separar um certificate/principal de uma coisa

O exemplo `detach-thing-principal` a seguir remove um certificado que representa uma entidade principal do item especificado.

```
aws iot detach-thing-principal \
    --thing-name "MyLightBulb" \
    --principal "arn:aws:iot:us-
west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36
```

Este comando não produz saída.

Para acessar mais informações, consulte Como gerenciar objetos com o registro no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte DetachThingPrincipalem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
```

```
    * Detaches a principal (certificate) from an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to detach.
    *
    * This method initiates an asynchronous request to detach a certificate from
an IoT Thing.
    * If the detachment is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void detachThingPrincipal(String thingName, String certificateArn) {
       DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
           .principal(certificateArn)
           .thingName(thingName)
           .build();

       CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
       future.whenComplete((voidResult, ex) -> {
           if (ex == null) {
               System.out.println(certificateArn + " was successfully removed
from " + thingName);
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " + ex.getMessage());
               }
           }
       });

       future.join();
   }
```

- Para obter detalhes da API, consulte DetachThingPrincipala Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}
```

- Para obter detalhes da API, consulte a DetachThingPrincipalreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def detach_thing_principal(self, thing_name, principal):
        """
        Detaches a certificate from an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.detach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Detached principal %s from thing %s.", principal,
 thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot detach principal. Resource not found.")
                return
            logger.error(
                "Couldn't detach principal from thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a DetachThingPrincipalReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`ListCertificates`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListCertificates`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Lists all certificates associated with the account.
   /// </summary>
   /// <returns>List of certificate information, or empty list if listing
failed.</returns>
   public async Task<List<Certificate>> ListCertificatesAsync()
   {
       try
       {
           var request = new ListCertificatesRequest();
           var response = await _amazonIoT.ListCertificatesAsync(request);

           _logger.LogInformation($"Retrieved {response.Certificates.Count}
certificates");
           return response.Certificates;
       }
       catch (Amazon.IoT.Model.ThrottlingException ex)
       {
           _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
```

```
            return new List<Certificate>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list certificates. Here's why:
 {ex.Message}");
            return new List<Certificate>();
        }
    }
```

- Para obter detalhes da API, consulte ListCertificatesa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! List certificates registered in the AWS account making the call.
/*!
   \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
```

```
        Aws::IoT::Model::ListCertificatesOutcome outcome =
 iotClient.ListCertificates(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
 outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                   result.GetCertificates().begin(),
                                   result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
 std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << std::endl;
    }

    return true;
}
```

- Para obter detalhes da API, consulte ListCertificatesa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Exemplo 1: Para listar os certificados registrados em sua AWS conta

O exemplo de `list-certificates` a seguir lista os certificados registrados em sua conta. Se o limite de paginação for maior do que o padrão de 25, é possível usar o valor de resposta `nextMarker` desse comando e fornecê-lo ao próximo comando para obter o próximo lote de resultados. Repita até `nextMarker` retornar sem um valor.

```
aws iot list-certificates
```

Saída:

```
{
    "certificates": [
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36
            "certificateId":
 "604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36",
            "status": "ACTIVE",
            "creationDate": 1556810537.617
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/262a1ac8a7d8aa72f6e96e365480f7313aa9db74b8339ec65d34dc3074e1c31e
            "certificateId":
 "262a1ac8a7d8aa72f6e96e365480f7313aa9db74b8339ec65d34dc3074e1c31e",
            "status": "ACTIVE",
            "creationDate": 1546447050.885
        },
        {
            "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/
b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b",
            "certificateId":
 "b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b",
            "status": "ACTIVE",
            "creationDate": 1546292258.322
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/7aebeea3845d14a44ec80b06b8b78a89f3f8a706974b8b34d18f5adf0741db42
            "certificateId":
 "7aebeea3845d14a44ec80b06b8b78a89f3f8a706974b8b34d18f5adf0741db42",
            "status": "ACTIVE",
            "creationDate": 1541457693.453
```

```
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/54458aa39ebb3eb39c91ffbbdcc3a6ca1c7c094d1644b889f735a6fc2cd9a7e3
            "certificateId":
 "54458aa39ebb3eb39c91ffbbdcc3a6ca1c7c094d1644b889f735a6fc2cd9a7e3",
            "status": "ACTIVE",
            "creationDate": 1541113568.611
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e
            "certificateId":
 "4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e",
            "status": "ACTIVE",
            "creationDate": 1541022751.983
        }
    ]
}
```

- Para obter detalhes da API, consulte ListCertificatesem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
 getAsyncClient().listCertificates();
```

```
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte ListCertificatesa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun listCertificates() {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
```

```
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}
```

- Para obter detalhes da API, consulte a ListCertificatesreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def list_certificates(self):
        """
```

```
        Lists AWS IoT certificates.

        :return: The list of certificates.
        """
        try:
            certificates = []
            paginator = self.iot_client.get_paginator("list_certificates")
            for page in paginator.paginate():
                certificates.extend(page["certificates"])
            logger.info("Retrieved %s certificates.", len(certificates))
            return certificates
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list certificates. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a ListCertificatesReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`ListThings`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `ListThings`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Lists IoT Things with pagination support.
    /// </summary>
    /// <returns>List of Things, or empty list if listing failed.</returns>
    public async Task<List<ThingAttribute>> ListThingsAsync()
    {
        try
        {
            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await _amazonIoT.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the
next page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await _amazonIoT.ListThingsAsync(request);
                }
                else
```

```
                    hasMoreResults = false;
            }

            _logger.LogInformation($"Retrieved {things.Count} Things");
            return things;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingAttribute>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list Things. Here's why: {ex.Message}");
            return new List<ThingAttribute>();
        }
    }
```

- Para obter detalhes da API, consulte ListThingsa Referência AWS SDK para .NET da API.

CLI

AWS CLI

Exemplo 1: listar todas os itens do registro

O `list-things` exemplo a seguir lista as coisas (dispositivos) que estão definidas no registro de AWS IoT da sua AWS conta.

```
aws iot list-things
```

Saída:

```
{
    "things": [
        {
            "thingName": "ThirdBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/ThirdBulb",
            "attributes": {
```

```
                "model": "123",
                "wattage": "75"
            },
            "version": 2
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/
MyOtherLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 3
        },
        {
            "thingName": "MyLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1
        },
        {
        "thingName": "SampleIoTThing",
        "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/SampleIoTThing",
        "attributes": {},
        "version": 1
        }
    ]
}
```

Exemplo 2: como listar os itens definidos que possuem um atributo específico

O exemplo de `list-things` a seguir exibe uma lista dos itens que têm um atributo chamado `wattage`.

```
aws iot list-things \
    --attribute-name wattage
```

Saída:

```
{
    "things": [
        {
            "thingName": "MyLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/
MyOtherLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 3
        }
    ]
}
```

Para acessar mais informações, consulte Como gerenciar objetos com o registro no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte ListThingsem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def list_things(self):
        """
        Lists AWS IoT things.

        :return: The list of things.
        """
        try:
            things = []
            paginator = self.iot_client.get_paginator("list_things")
            for page in paginator.paginate():
                things.extend(page["things"])
            logger.info("Retrieved %s things.", len(things))
            return things
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list things. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Para obter detalhes da API, consulte a ListThingsReferência da API AWS SDK for Python (Boto3).

Rust

SDK para Rust

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
async fn show_things(client: &Client) -> Result<(), Error> {
    let resp = client.list_things().send().await?;

    println!("Things:");

    for thing in resp.things.unwrap() {
        println!(
            "  Name:  {}",
            thing.thing_name.as_deref().unwrap_or_default()
        );
        println!(
            "  Type:  {}",
            thing.thing_type_name.as_deref().unwrap_or_default()
        );
        println!(
            "  ARN:   {}",
            thing.thing_arn.as_deref().unwrap_or_default()
        );
        println!();
    }

    println!();

    Ok(())
}
```

- Para obter detalhes da API, consulte a ListThingsreferência da API AWS SDK for Rust.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`SearchIndex`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `SearchIndex`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
   /// <summary>
   /// Searches for IoT Things using the search index.
   /// </summary>
   /// <param name="queryString">The search query string.</param>
   /// <returns>List of Things that match the search criteria, or empty list if
search failed.</returns>
   public async Task<List<ThingDocument>> SearchIndexAsync(string queryString)
   {
       try
       {
           // First, try to perform the search
           var request = new SearchIndexRequest
           {
               QueryString = queryString
           };

           var response = await _amazonIoT.SearchIndexAsync(request);
           _logger.LogInformation($"Search found {response.Things.Count}
Things");
           return response.Things;
       }
       catch (Amazon.IoT.Model.IndexNotReadyException ex)
       {
```

```
            _logger.LogWarning($"Search index not ready, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
        {
            _logger.LogWarning($"Search index not configured, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't search index. Here's why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Sets up the indexing configuration and retries the search after waiting
for the index to be ready.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
setup/search failed.</returns>
    private async Task<List<ThingDocument>> SetupIndexAndRetrySearchAsync(string
queryString)
    {
        try
        {
            // Update indexing configuration to REGISTRY mode
            _logger.LogInformation("Setting up IoT search indexing
configuration...");
            await _amazonIoT.UpdateIndexingConfigurationAsync(
                new UpdateIndexingConfigurationRequest()
                {
                    ThingIndexingConfiguration = new ThingIndexingConfiguration()
                    {
```

```
                        ThingIndexingMode = ThingIndexingMode.REGISTRY
                    }
                });

            _logger.LogInformation("Indexing configuration updated. Waiting for
index to be ready...");

            // Wait for the index to be set up - this can take some time
            const int maxRetries = 10;
            const int retryDelaySeconds = 10;

            for (int attempt = 1; attempt <= maxRetries; attempt++)
            {
                try
                {
                    _logger.LogInformation($"Waiting for index to be ready
(attempt {attempt}/{maxRetries})...");
                    await Task.Delay(TimeSpan.FromSeconds(retryDelaySeconds));

                    // Try to get the current indexing configuration to see if
it's ready
                    var configResponse = await
_amazonIoT.GetIndexingConfigurationAsync(new GetIndexingConfigurationRequest());
                    if
(configResponse.ThingIndexingConfiguration?.ThingIndexingMode ==
ThingIndexingMode.REGISTRY)
                    {
                        // Try the search again
                        var request = new SearchIndexRequest
                        {
                            QueryString = queryString
                        };

                        var response = await
_amazonIoT.SearchIndexAsync(request);
                        _logger.LogInformation($"Search found
{response.Things.Count} Things after index setup");
                        return response.Things;
                    }
                }
                catch (Amazon.IoT.Model.IndexNotReadyException)
                {
                    // Index still not ready, continue waiting
```

```
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
                catch (Amazon.IoT.Model.InvalidRequestException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
            }

            _logger.LogWarning("Timeout waiting for search index to be ready
after configuration update");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't set up search index configuration. Here's
why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }
```

- Para obter detalhes da API, consulte SearchIndexa Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Query the AWS IoT fleet index.
```

```
//! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
/*!
  \param: query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
 iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result =
 outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
            nextToken = result.GetNextToken();

        }
        else {
            std::cerr << "Error in SearchIndex: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
 std::endl;
```

```
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
                  << std::endl;
    }
    return true;
}
```

- Para obter detalhes da API, consulte SearchIndexa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como consultar o índice do item

O exemplo de `search-index` a seguir consulta o índice `AWS_Things` em busca de itens que tenham um tipo de `LightBulb`.

```
aws iot search-index \
    --index-name "AWS_Things" \
    --query-string "thingTypeName:LightBulb"
```

Saída:

```
{
    "things": [
        {
            "thingName": "MyLightBulb",
            "thingId": "40da2e73-c6af-406e-b415-15acae538797",
            "thingTypeName": "LightBulb",
            "thingGroupNames": [
                "LightBulbs",
                "DeadBulbs"
            ],
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
```

```
        },
        {
            "thingName": "ThirdBulb",
            "thingId": "615c8455-33d5-40e8-95fd-3ee8b24490af",
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingId": "6dae0d3f-40c1-476a-80c4-1ed24ba6aa11",
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        }
    ]
}
```

Para obter mais informações, consulte Gerenciar indexações de itens no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte SearchIndexem Referência de AWS CLI Comandos.

Java

SDK para Java 2.x

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }
```

- Para obter detalhes da API, consulte SearchIndexa Referência AWS SDK for Java 2.x da API.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}
```

- Para obter detalhes da API, consulte a SearchIndexreferência da API AWS SDK for Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def search_index(self, query):
        """
        Searches the AWS IoT index.

        :param query: The search query.
        :return: The list of things found.
        """
        try:
            response = self.iot_client.search_index(queryString=query)
            logger.info("Found %s things.", len(response.get("things", [])))
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
```

```
                    "Couldn't search index. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response.get("things", [])
```

- Para obter detalhes da API, consulte a SearchIndexReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`UpdateIndexingConfiguration`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateIndexingConfiguration`.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Update the indexing configuration.
/*!
  \param thingIndexingConfiguration: A ThingIndexingConfiguration object which is
 ignored if not set.
  \param thingGroupIndexingConfiguration: A ThingGroupIndexingConfiguration
 object which is ignored if not set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::IoT::updateIndexingConfiguration(
        const Aws::IoT::Model::ThingIndexingConfiguration
 &thingIndexingConfiguration,
        const Aws::IoT::Model::ThingGroupIndexingConfiguration
 &thingGroupIndexingConfiguration,
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::UpdateIndexingConfigurationRequest request;

    if (thingIndexingConfiguration.ThingIndexingModeHasBeenSet()) {
        request.SetThingIndexingConfiguration(thingIndexingConfiguration);
    }

    if (thingGroupIndexingConfiguration.ThingGroupIndexingModeHasBeenSet()) {

 request.SetThingGroupIndexingConfiguration(thingGroupIndexingConfiguration);
    }

    Aws::IoT::Model::UpdateIndexingConfigurationOutcome outcome =
 iotClient.UpdateIndexingConfiguration(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "UpdateIndexingConfiguration succeeded." << std::endl;
    }
    else {
        std::cerr << "UpdateIndexingConfiguration failed."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte UpdateIndexingConfigurationa Referência AWS SDK para C++ da API.

CLI

AWS CLI

Para habilitar a indexação de itens

O `update-indexing-configuration` exemplo a seguir permite que a indexação de coisas ofereça suporte à pesquisa de dados de registro, dados de sombra e status de conectividade de AWS coisas usando o índice _Things.

```
aws iot update-indexing-configuration
    --thing-indexing-
configuration thingIndexingMode=REGISTRY_AND_SHADOW,thingConnectivityIndexingMode=STATUS
```

Este comando não produz saída.

Para obter mais informações, consulte Gerenciar indexações de itens no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte UpdateIndexingConfigurationem Referência de AWS CLI Comandos.

Python

SDK para Python (Boto3)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
```

```
        return cls(iot_client, iot_data_client)

    def update_indexing_configuration(self):
        """
        Updates the AWS IoT indexing configuration to enable thing indexing.
        """
        try:
            self.iot_client.update_indexing_configuration(
                thingIndexingConfiguration={"thingIndexingMode": "REGISTRY"}
            )
            logger.info("Updated indexing configuration.")
        except ClientError as err:
            logger.error(
                "Couldn't update indexing configuration. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obter detalhes da API, consulte a UpdateIndexingConfigurationReferência da API AWS SDK for Python (Boto3).

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Use **`UpdateThing`** com um AWS SDK ou CLI

Os exemplos de código a seguir mostram como usar o `UpdateThing`.

.NET

SDK para .NET (v4)

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
    /// <summary>
    /// Updates an IoT Thing with attributes.
    /// </summary>
    /// <param name="thingName">The name of the Thing to update.</param>
    /// <param name="attributes">Dictionary of attributes to add.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingAsync(string thingName, Dictionary<string,
string> attributes)
    {
        try
        {
            var request = new UpdateThingRequest
            {
                ThingName = thingName,
                AttributePayload = new AttributePayload
                {
                    Attributes = attributes,
                    Merge = true
                }
            };

            await _amazonIoT.UpdateThingAsync(request);
            _logger.LogInformation($"Updated Thing {thingName} with attributes");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing attributes. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- Para obter detalhes da API, consulte UpdateThinga Referência AWS SDK para .NET da API.

C++

SDK para C++

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
//! Update an AWS IoT thing with attributes.
/*!
  \param thingName: The name for the thing.
  \param attributeMap: A map of key/value attributes/
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
 &attributeMap,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obter detalhes da API, consulte UpdateThinga Referência AWS SDK para C++ da API.

CLI

AWS CLI

Como associar um item a um tipo de item

O `update-thing` exemplo a seguir associa uma coisa no registro de AWS IoT a um tipo de coisa. Ao fazer a associação, devem ser fornecidos valores para os atributos definidos pelo tipo de item.

```
aws iot update-thing \
    --thing-name "MyOtherLightBulb" \
    --thing-type-name "LightBulb" \
    --attribute-payload "{"attributes": {"wattage":"75", "model":"123"}}"
```

Esse comando não produz nenhuma saída. É possível usar o comando `describe-thing` para exibir o resultado.

Para obter mais informações, consulte Tipos de itens no Guia do desenvolvedor do AWS IoT.

- Para obter detalhes da API, consulte UpdateThingem Referência de AWS CLI Comandos.

Kotlin

SDK para Kotlin

> **Note**
>
> Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no AWS Code Examples Repository.

```
suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
```

```
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}
```

- Para obter detalhes da API, consulte a UpdateThingreferência da API AWS SDK for Kotlin.

Para obter uma lista completa dos guias do desenvolvedor do AWS SDK e exemplos de código, consulteUsando AWS IoT com um AWS SDK. Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Cotas do AWS IoT

Você pode encontrar informações sobre cotas do AWS IoT em Referência geral da AWS.

- Para acessar mais informações sobre cotas do AWS IoT Core, consulte Endpoints e cotas do AWS IoT Core.
- Para acessar mais informações sobre cotas do AWS IoT Device Management, consulte Endpoints e cotas do AWS IoT Device Management.
- Para acessar mais informações sobre cotas do AWS IoT Device Defender, consulte Endpoints e cotas do AWS IoT Device Defender.

Preços do AWS IoT Core

Você pode encontrar informações sobre preços do AWS IoT Core na página de Marketing da AWS e na Calculadora de preços da AWS.

- Para verificar as informações de preços do AWS IoT Core, consulte Preços do AWS IoT Core.
- Para estimar o custo de sua solução de arquiteto, consulte a Calculadora de preços da AWS.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.