



Guia do Desenvolvedor

AWS IoT Events



AWS IoT Events: 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 desprestige 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

.....	viii
O que AWS IoT Events é	1
Benefícios e recursos	1
Casos de uso	3
Monitore e mantenha dispositivos remotos	3
Gerencie robôs industriais	3
Rastreie sistemas de automação predial	3
AWS IoT Events fim do suporte	4
Considerações ao migrar para fora do AWS IoT Events	4
Modelos de detector	5
Comparando arquiteturas	5
Etapa 1: (opcional) exportar configurações do modelo do AWS IoT Events detector	7
Etapa 2: Criar um perfil do IAM	8
Etapa 3: Criar Amazon Kinesis Data Streams	10
Etapa 4: criar ou atualizar a regra de roteamento de mensagens do MQTT	11
Etapa 5: Obtenha o endpoint para o tópico MQTT de destino	13
Etapa 6: criar uma tabela do Amazon DynamoDB	13
Etapa 7: criar uma AWS Lambda função (console)	14
Etapa 8: Adicionar um gatilho do Amazon Kinesis Data Streams	22
Etapa 9: testar a funcionalidade de ingestão e saída de dados (AWS CLI)	23
Alarmes	24
Comparando arquiteturas	24
Etapa 1: ativar as notificações do MQTT na propriedade do ativo	25
Etapa 2: criar uma AWS Lambda função	26
Etapa 3: criar uma regra de roteamento de AWS IoT Core mensagens	27
Etapa 4: visualizar CloudWatch métricas	28
Etapa 5: criar CloudWatch alarmes	29
Etapa 6: (opcional) importar o CloudWatch alarme para AWS IoT SiteWise	29
Configurar	30
Configurando um Conta da AWS	30
Inscreva-se para um Conta da AWS	30
Criar um usuário com acesso administrativo	31
Configurando permissões para AWS IoT Events	32
Permissões de ação	33

Proteger dados de entrada	35
Política de função de CloudWatch registro da Amazon	36
Política de função de mensagens do Amazon SNS	38
Introdução	40
Pré-requisitos	42
Como criar uma entrada	43
Crie um arquivo de entrada JSON	43
Criar e configurar uma entrada	43
Crie uma entrada dentro do modelo do detector	44
Como criar um modelo de detector	45
Teste o modelo do detector	52
Práticas recomendadas	56
Ative o CloudWatch registro na Amazon ao desenvolver modelos AWS IoT Events de detectores	56
Publique regularmente para salvar seu modelo de detector ao trabalhar no AWS IoT Events console	57
Tutoriais	58
Usando AWS IoT Events para monitorar seus dispositivos de IoT	58
Como descobrir quais estados são necessários em um modelo de detector?	60
Como saber se você precisa de uma ou várias instâncias de um detector?	61
step-by-stepExemplo simples	62
Criar uma entrada para capturar dados do dispositivo	64
Criar um modelo de detector para representar os estados do dispositivo	65
Enviar mensagens como entradas para um detector	69
Restrições e limitações do modelo de detector	73
Um exemplo comentado: controle de temperatura com HVAC	76
Definições de entrada para modelos de detectores	77
Crie uma definição de modelo de detector	80
Usar o BatchUpdateDetector	101
Uso BatchPutMessage para entradas	103
Ingerir mensagens MQTT	105
Gere mensagens do Amazon SNS	107
Configurar a DescribeDetector API	108
Use o mecanismo de AWS IoT Core regras	110
Ações compatíveis	114
Use ações integradas	115

Definir ação do temporizador	115
Redefinir ação do temporizador	116
Apagar a ação do temporizador	116
Definir ação variável	117
Trabalhe com outros AWS serviços	117
AWS IoT Core	118
AWS IoT Events	119
AWS IoT SiteWise	120
Amazon DynamoDB	123
Amazon DynamoDB(v2)	125
Amazon Data Firehose	126
AWS Lambda	127
Amazon Simple Notification Service	128
Amazon Simple Queue Service	130
Expressões	132
Sintaxe para filtrar dados do dispositivo	132
Literais	132
Operadores	132
Funções para expressões	134
Referência para entradas e variáveis em expressões	139
Modelos de substituição	142
Usage	142
Escrevendo AWS IoT Events expressões	143
Exemplos de modelo de detector	145
Controle de temperatura HVAC	145
História de fundo	145
Definições de entrada	146
Definição do modelo de detector	148
BatchPutMessage exemplos	166
BatchUpdateDetector exemplo	172
AWS IoT Core mecanismo de regras	174
Guindastes	177
Enviar comandos	178
Modelos de detector	180
Entradas	186
Mensagens	187

Exemplo: Detecção de eventos com sensores	189
Dispositivo HeartBeat	191
Alarmes ISA	193
Alarme simples	203
Monitorar com alarme do	208
Trabalhando com AWS IoT SiteWise	208
Fluxo de reconhecimento	208
Como criar um modelo de alarme	209
Requisitos	210
Como criar um modelo de alarme (console)	210
Responder a alarmes	213
Como gerenciar notificações	215
Criação de uma função do Lambda	215
Usando a função Lambda	224
Gerenciar destinatários de alarmes	225
Segurança	227
Gerenciamento de identidade e acesso	227
Público	228
Autenticação com identidades	228
Gerenciar o acesso usando políticas	229
Mais sobre gerenciamento de identidade e acesso	231
Como AWS IoT Events funciona com o IAM	231
Exemplos de políticas baseadas em identidade	235
Prevenção confusa de delegados entre serviços para AWS IoT Events	242
Solução de problemas	246
Monitoramento	248
Ferramentas disponíveis para monitorar AWS IoT Events	249
Monitoramento AWS IoT Events com a Amazon CloudWatch	251
Registrando chamadas de AWS IoT Events API com AWS CloudTrail	252
Validação de conformidade	272
Resiliência	272
Segurança da infraestrutura	273
Cotas	274
Tags	275
Conceitos Básicos de Tags	275
Restrições e limitações de tags	276

Utilização de tags com políticas do IAM	276
Solução de problemas	280
AWS IoT Events Problemas e soluções comuns	280
Erros na criação do modelo do detector	281
Atualizações de um modelo de detector excluído	281
Falha no gatilho da ação (ao atender a uma condição)	281
Falha no gatilho de ação (ao ultrapassar um limite)	282
Uso incorreto de estados	282
Mensagem de conexão	282
InvalidRequestException mensagem	283
action.setTimerErros do Amazon CloudWatch Logs	283
Erros de CloudWatch carga útil da Amazon	284
Tipos de dados incompatíveis	286
Falha ao enviar mensagem para AWS IoT Events	287
Solução de problemas para um modelo de detector	288
Informações de diagnóstico	289
Análise um modelo de detector (console)	302
Análise um modelo de detector (AWS CLI)	304
Comandos	309
AWS IoT Events ações	309
AWS IoT Events dados	309
Histórico do documento	310
Atualizações anteriores	311

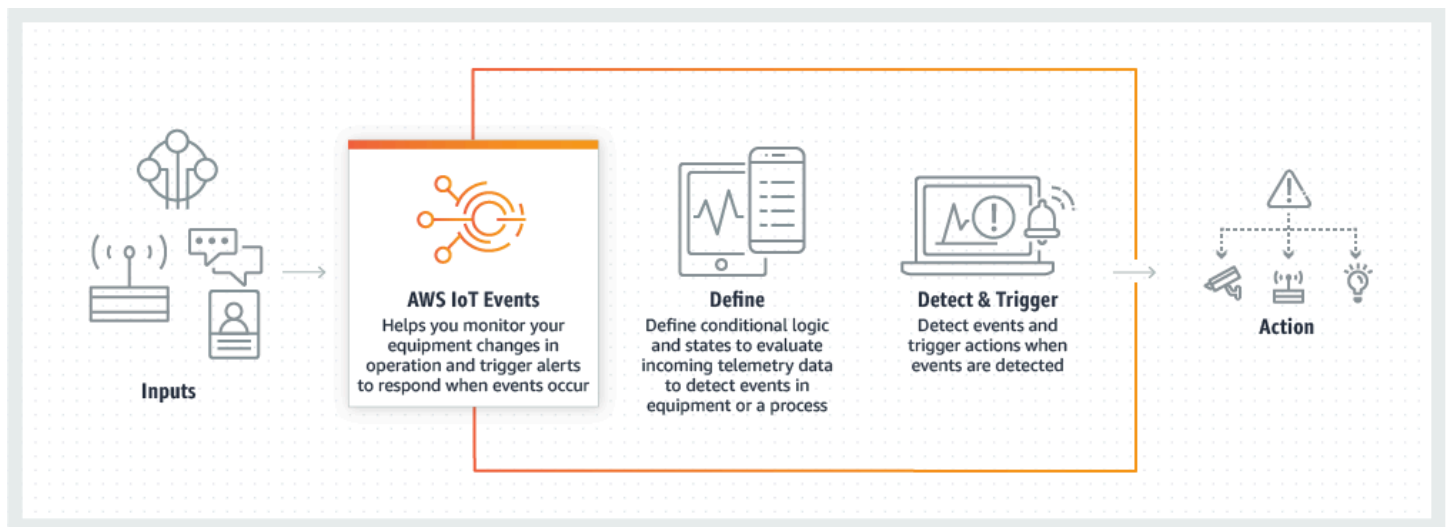
Aviso de fim do suporte: em 20 de maio de 2026, AWS encerrará o suporte para AWS IoT Events. Depois de 20 de maio de 2026, você não poderá mais acessar o AWS IoT Events console ou os AWS IoT Events recursos. Para obter mais informações, consulte [AWS IoT Events Fim do suporte](#).

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.

O que AWS IoT Events é

AWS IoT Events permite monitorar suas frotas de equipamentos ou dispositivos em busca de falhas ou mudanças na operação e acionar ações quando tais eventos ocorrerem. AWS IoT Events monitora continuamente os dados dos sensores de IoT de dispositivos, processos, aplicativos e outros AWS serviços para identificar eventos significativos para que você possa agir.

Use AWS IoT Events para criar aplicativos complexos de monitoramento de eventos na AWS nuvem que você pode acessar por meio do AWS IoT Events console ou APIs.



Tópicos

- [Benefícios e recursos](#)
- [Casos de uso](#)

Benefícios e recursos

Aceite entradas de várias fontes

AWS IoT Events aceita entradas de várias fontes de dados de telemetria de IoT. Isso inclui dispositivos de sensores, aplicativos de gerenciamento e outros AWS IoT serviços, como AWS IoT Core AWS IoT Analytics e. Você pode enviar qualquer entrada de dados de telemetria AWS IoT Events usando uma interface de API (BatchPutMessageAPI) padrão ou o AWS IoT Events console.

Para obter mais informações sobre como começar a usar AWS IoT Events, consulte [Começando a usar o AWS IoT Events console](#).

Use expressões lógicas simples para reconhecer padrões complexos de eventos

AWS IoT Events pode reconhecer padrões de eventos que envolvem várias entradas de um único dispositivo ou aplicativo de IoT ou de diversos equipamentos e muitos sensores independentes. Isso é especialmente útil porque cada sensor e aplicativo fornecem informações importantes. Mas somente combinando diversos dados de sensores e aplicações você pode obter uma visão completa do desempenho e da qualidade das operações. Você pode configurar AWS IoT Events detectores para reconhecer esses eventos usando expressões lógicas simples em vez de código complexo.

Para obter mais informações sobre expressões lógicas, consulte [Expressões para filtrar, transformar e processar dados de eventos](#).

Acione ações com base em eventos

AWS IoT Events permite que você acione ações diretamente no Amazon Simple Notification Service (Amazon SNS), no Lambda AWS IoT Core, no Amazon SQS e no Amazon Kinesis Firehose. Você também pode acionar uma AWS Lambda função usando o mecanismo de AWS IoT regras que possibilita a realização de ações usando outros serviços, como o Amazon Connect, ou seus próprios aplicativos de planejamento de recursos corporativos (ERP).

AWS IoT Events inclui uma biblioteca pré-construída de ações que você pode realizar e também permite que você defina as suas próprias.

Para saber mais sobre como acionar ações com base em eventos, consulte [Ações suportadas para receber dados e acionar ações em AWS IoT Events](#)

Dimensione automaticamente para atender às demandas de sua frota

AWS IoT Events é dimensionado automaticamente quando você está conectando dispositivos homogêneos. Você pode definir um detector uma vez para um tipo específico de dispositivo, e o serviço escalará e gerenciará automaticamente todas as instâncias desse dispositivo às quais se conectam AWS IoT Events.

Para explorar exemplos de modelos de detectores, consulte [AWS IoT Events exemplos de modelos de detectores](#).

Casos de uso

AWS IoT Events tem muitos usos. Aqui estão alguns exemplos de casos de uso.

Monitore e mantenha dispositivos remotos

Monitorar uma frota de máquinas implantadas remotamente pode ser um desafio, especialmente quando ocorre uma avaria sem um contexto claro. Se uma máquina parar de funcionar, isso pode significar a substituição de toda a unidade de processamento ou máquina. Mas isso não é sustentável. Com AWS IoT Events ele, você pode receber mensagens de vários sensores em cada máquina para ajudá-lo a diagnosticar problemas específicos ao longo do tempo. Em vez de substituir a unidade inteira, agora você tem as informações necessárias para enviar um técnico com a peça exata que precisa ser substituída. Com milhões de máquinas, a economia pode chegar a milhões de dólares, reduzindo o custo total de propriedade ou manutenção de cada máquina.

Gerencie robôs industriais

A implantação de robôs em suas instalações para automatizar a movimentação de pacotes pode aumentar muito a eficiência. Para minimizar os custos, os robôs podem ser equipados com sensores simples e de baixo custo que reportam dados para a nuvem. No entanto, com dezenas de sensores e centenas de modos de operação, detectar problemas em tempo real pode ser um desafio. Usando AWS IoT Events, você pode criar um sistema especializado que processa esses dados do sensor na nuvem, criando alertas para notificar automaticamente a equipe técnica se uma falha for iminente.

Rastreie sistemas de automação predial

Nos data centers, o monitoramento de altas temperaturas e baixa umidade ajuda a evitar falhas no equipamento. Os sensores geralmente são adquiridos de vários fabricantes e cada tipo vem com seu próprio software de gerenciamento. No entanto, o software de gerenciamento de diferentes fornecedores às vezes não é compatível, dificultando a detecção de problemas. Usando AWS IoT Events, você pode configurar alertas para notificar seus analistas de operações sobre problemas com seus sistemas de aquecimento e resfriamento bem antes das falhas. Dessa forma, é possível evitar um desligamento não programado do datacenter que custaria milhares de dólares em substituição de equipamentos e em potencial perda de receita.

AWS IoT Events fim do suporte

Após uma análise cuidadosa, decidimos encerrar o suporte para o AWS IoT Events serviço, a partir de 20 de maio de 2026. AWS IoT Events não aceitará mais novos clientes a partir de 20 de maio de 2025. Como cliente existente com uma conta cadastrada no serviço antes de 20 de maio de 2025, você pode continuar usando os AWS IoT Events recursos. Depois de 20 de maio de 2026, você não poderá mais usar AWS IoT Events.

Esta página fornece instruções e considerações para que AWS IoT Events os clientes façam a transição para uma solução alternativa que atenda às suas necessidades comerciais.

Note

As soluções apresentadas nesses guias devem servir como exemplos ilustrativos, não como substitutos de funcionalidade prontos para produção. AWS IoT Events Personalize o código, o fluxo de trabalho e AWS os recursos relacionados às suas necessidades comerciais.

Tópicos

- [Considerações ao migrar para fora do AWS IoT Events](#)
- [Procedimento de migração para modelos de detectores em AWS IoT Events](#)
- [Procedimento de migração para AWS IoT SiteWise alarmes em AWS IoT Events](#)

Considerações ao migrar para fora do AWS IoT Events

- Implemente as melhores práticas de segurança, incluindo o uso de funções do IAM com menos privilégios para cada componente e a criptografia de dados em repouso e em trânsito. Para obter mais informações, consulte [Práticas recomendadas de segurança no IAM](#) no Guia do usuário do IAM.
- Considere o número de fragmentos do stream do Kinesis com base em seus requisitos de ingestão de dados. Para obter mais informações sobre os fragmentos do Kinesis, consulte a [terminologia e os conceitos do Amazon Kinesis Data Streams](#) no Guia do desenvolvedor do Amazon Kinesis Data Streams.

- Configure monitoramento e depuração abrangentes usando CloudWatch métricas e registros. Para obter mais informações, consulte [O que é CloudWatch?](#) no Guia do CloudWatch usuário da Amazon.
- Considere a estrutura do tratamento de erros, incluindo como gerenciar mensagens que falham no processamento repetido, implementar políticas de repetição e configurar um processo para isolar e analisar mensagens problemáticas.
- Use a [Calculadora de AWS preços](#) para estimar os custos para seu caso de uso específico.

Procedimento de migração para modelos de detectores em AWS IoT Events

Esta seção descreve soluções alternativas que oferecem funcionalidade semelhante ao modelo de detector à medida que você migra para fora do AWS IoT Events.

Você pode migrar a ingestão de dados por meio de AWS IoT Core regras para uma combinação de outros AWS serviços. Em vez da ingestão de dados por meio da [BatchPutMessageAPI](#), os dados podem ser roteados para o tópico do AWS IoT Core MQTT.

Essa abordagem de migração utiliza os tópicos do AWS IoT Core MQTT como ponto de entrada para seus dados de IoT, substituindo a entrada direta para AWS IoT Events. Os tópicos do MQTT são escolhidos por vários motivos principais. Eles oferecem ampla compatibilidade com dispositivos de IoT devido ao amplo uso do MQTT no setor. Esses tópicos podem lidar com grandes volumes de mensagens de vários dispositivos, garantindo escalabilidade. Eles também oferecem flexibilidade no roteamento e na filtragem de mensagens com base no conteúdo ou no tipo de dispositivo. Além disso, os tópicos do AWS IoT Core MQTT se integram perfeitamente a outros AWS serviços, facilitando o processo de migração.

Os dados fluem dos tópicos do MQTT para uma arquitetura que combina o Amazon Kinesis Data Streams, AWS Lambda uma função, uma tabela do Amazon DynamoDB e agendas da Amazon EventBridge. Essa combinação de serviços replica e aprimora a funcionalidade fornecida anteriormente AWS IoT Events, oferecendo mais flexibilidade e controle sobre seu pipeline de processamento de dados de IoT.

Comparando arquiteturas

A AWS IoT Events arquitetura atual ingere dados por meio de uma AWS IoT Core regra e da BatchPutMessage API. Essa arquitetura é usada AWS IoT Core para ingestão de dados e

publicação de eventos, com mensagens roteadas por meio de AWS IoT Events entradas para modelos de detectores que definem a lógica do estado. Uma função do IAM gerencia as permissões necessárias.

A nova solução é mantida AWS IoT Core para ingestão de dados (agora com tópicos dedicados de entrada e saída do MQTT). Ele apresenta o Kinesis Data Streams para particionamento de dados e uma função Lambda avaliadora para lógica de estado. Os estados do dispositivo agora são armazenados em uma tabela do DynamoDB, e uma função aprimorada do IAM gerencia as permissões nesses serviços.

Finalidade	Solução	Diferenças
Ingestão de dados — recebe dados de dispositivos de IoT	AWS IoT Core	Agora requer dois tópicos distintos do MQTT: um para ingerir dados do dispositivo e outro para publicar eventos de saída
Direção da mensagem — encaminha as mensagens recebidas para os serviços apropriados	AWS IoT Core regra de roteamento de mensagens	Mantém a mesma funcionalidade de roteamento, mas agora direciona as mensagens para o Kinesis Data Streams em vez de AWS IoT Events
Processamento de dados — manipula e organiza fluxos de dados recebidos	Kinesis Data Streams	Substitui a funcionalidade AWS IoT Events de entrada, fornecendo ingestão de dados com particionamento de ID de dispositivo para processamento de mensagens
Avaliação lógica — Processa mudanças de estado e aciona ações	Avaliador Lambda	Substitui o modelo do AWS IoT Events detector, fornecendo avaliação de lógica de estado personalizável por meio de código em vez de fluxo de trabalho visual
Gerenciamento de estado — mantém os estados do dispositivo	Tabela DynamoDB	Novo componente que fornece armazenamento persistente dos estados do dispositivo, substituindo o gerenciamento interno AWS IoT Events do estado

Finalidade	Solução	Diferenças
Segurança — gerencia as permissões do serviço	perfil do IAM	As permissões atualizadas agora incluem acesso ao Kinesis Data Streams, ao DynamoDB e às permissões EventBridge existentes AWS IoT Core

Etapa 1: (opcional) exportar configurações do modelo do AWS IoT Events detector

Antes de criar novos recursos, exporte as definições do modelo do AWS IoT Events detector. Eles contêm sua lógica de processamento de eventos e podem servir como referência histórica para implementar sua nova solução.

Console

Usando o AWS IoT Events Console de gerenciamento da AWS, execute as seguintes etapas para exportar as configurações do modelo do detector:

Para exportar modelos de detectores usando o Console de gerenciamento da AWS

1. Faça login no [console do AWS IoT Events](#).
2. No painel de navegação à esquerda, escolha Detector models (Modelos de detector).
3. Selecione o modelo do detector a ser exportado.
4. Escolha Exportar. Leia a mensagem informativa sobre a saída e escolha Exportar novamente.
5. Repita o processo para cada modelo de detector que você deseja exportar.

Um arquivo contendo uma saída JSON do seu modelo de detector é adicionado à pasta de download do seu navegador. Opcionalmente, você pode salvar cada configuração do modelo de detector para preservar os dados históricos.

AWS CLI

Usando o AWS CLI, execute os seguintes comandos para exportar as configurações do modelo do detector:

Para exportar modelos de detectores usando AWS CLI

1. Liste todos os modelos de detectores em sua conta:

```
aws iotevents list-detector-models
```

2. Para cada modelo de detector, exporte sua configuração executando:

```
aws iotevents describe-detector-model \  
  --detector-model-name your-detector-model-name
```

3. Salve a saída para cada modelo de detector.

Etapa 2: Criar um perfil do IAM

Crie uma função do IAM para fornecer permissões para replicar a funcionalidade do AWS IoT Events. A função neste exemplo concede acesso ao DynamoDB para gerenciamento de estado, agendamento EventBridge, Kinesis Data Streams para AWS IoT Core ingestão de dados, publicação de mensagens e registro em log. CloudWatch Juntos, esses serviços funcionarão como substitutos AWS IoT Events.

1. Criar uma função do IAM com as permissões a seguir. Para obter instruções mais detalhadas sobre como criar uma função do IAM, consulte [Criar uma função para delegar permissões a um AWS serviço](#) no Guia do usuário do IAM.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "DynamoDBAccess",  
      "Effect": "Allow",  
      "Action": [  
        "dynamodb:GetItem",  
        "dynamodb:PutItem",  
        "dynamodb:UpdateItem",  
        "dynamodb>DeleteItem",  
        "dynamodb:Query",  
        "dynamodb:Scan"  
      ]  
    }  
  ]  
}
```

```

    ],
    "Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/
EventsStateTable"
  },
  {
    "Sid": "SchedulerAccess",
    "Effect": "Allow",
    "Action": [
      "scheduler:CreateSchedule",
      "scheduler>DeleteSchedule"
    ],
    "Resource": "arn:aws:scheduler:us-east-1:123456789012:schedule/*"
  },
  {
    "Sid": "KinesisAccess",
    "Effect": "Allow",
    "Action": [
      "kinesis:GetRecords",
      "kinesis:GetShardIterator",
      "kinesis:DescribeStream",
      "kinesis:ListStreams"
    ],
    "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/*"
  },
  {
    "Sid": "IoTPublishAccess",
    "Effect": "Allow",
    "Action": "iot:Publish",
    "Resource": "arn:aws:iot:us-east-1:123456789012:topic/*"
  },
  {
    "Effect": "Allow",
    "Action": "logs:CreateLogGroup",
    "Resource": "arn:aws:logs:us-east-1:123456789012:*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "logs:CreateLogStream",
      "logs:PutLogEvents"
    ],
    "Resource": [
      "arn:aws:logs:us-east-1:123456789012:log-group:/aws/lambda/your-
Lambda:*"

```

```

    ]
  }
]
}

```

2. Adicione a seguinte política de confiança da função do IAM. Uma política de confiança permite que os AWS serviços especificados assumam a função do IAM para que possam realizar as ações necessárias. Para obter instruções mais detalhadas sobre como criar uma política de confiança do IAM, consulte [Criar uma função usando políticas de confiança personalizadas](#) no Guia do usuário do IAM.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "scheduler.amazonaws.com",
          "lambda.amazonaws.com",
          "iot.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

Etapa 3: Criar Amazon Kinesis Data Streams

Crie Amazon Kinesis Data Streams Console de gerenciamento da AWS usando o ou. AWS CLI

Console

Para criar um stream de dados do Kinesis usando o Console de gerenciamento da AWS, siga o procedimento encontrado na página [Criar um stream de dados](#) no Guia do desenvolvedor do Amazon Kinesis Data Streams.

Ajuste a contagem de fragmentos com base na contagem de dispositivos e no tamanho da carga útil da mensagem.

AWS CLI

Usando AWS CLI, crie o Amazon Kinesis Data Streams para ingerir e particionar os dados de seus dispositivos.

Os Kinesis Data Streams são usados nessa migração para substituir a funcionalidade de ingestão de dados do AWS IoT Events. Ele fornece uma maneira escalável e eficiente de coletar, processar e analisar dados de streaming em tempo real de seus dispositivos de IoT, ao mesmo tempo em que fornece tratamento flexível de dados e integração com outros serviços AWS.

```
aws kinesis create-stream --stream-name your-kinesis-stream-name --shard-count 4 --  
region your-region
```

Ajuste a contagem de fragmentos com base na contagem de dispositivos e no tamanho da carga útil da mensagem.

Etapa 4: criar ou atualizar a regra de roteamento de mensagens do MQTT

Você pode criar uma nova regra de roteamento de mensagens MQTT ou atualizar uma regra existente.

Console

1. Determine se você precisa de uma nova regra de roteamento de mensagens do MQTT ou se você pode atualizar uma regra existente.
2. Abra o [console do AWS IoT Core](#).
3. No painel de navegação, escolha Encaminhamento de mensagens e, em seguida, escolha Regras.
4. Na seção Gerenciar, escolha Roteamento de mensagens e, em seguida, Regras.
5. Escolha Criar regra.
6. Na página Especificar propriedades da regra, insira o nome da AWS IoT Core regra em Nome da regra. Em Descrição da regra, opcional, insira uma descrição para identificar que você está processando eventos e encaminhando-os para o Kinesis Data Streams.

7. Na página Configurar instrução SQL, insira o seguinte para a instrução SQL: **SELECT * FROM 'your-database'** e escolha Avançar.
8. Na página Anexar ações de regras e em Ações de regras, escolha kinesis.
9. Escolha seu stream do Kinesis para o stream. Para a chave de partição, insira **your-instance-id**. Selecione a função apropriada para a função do IAM e, em seguida, escolha Adicionar ação de regra.

Para obter mais informações, consulte [Criação de regras de AWS IoT para rotear dados do dispositivo para outros serviços](#).

AWS CLI

1. Crie um arquivo JSON com o conteúdo apresentado a seguir. Esse arquivo de configuração JSON define uma AWS IoT Core regra que seleciona todas as mensagens de um tópico e as encaminha para o stream especificado do Kinesis, usando o ID da instância como chave de partição.

```
{
  "sql": "SELECT * FROM 'your-config-file'",
  "description": "Rule to process events and forward to Kinesis Data Streams",
  "actions": [
    {
      "kinesis": {
        "streamName": "your-kinesis-stream-name",
        "roleArn": "arn:aws:iam::your-account-id:role/service-role/your-iam-role",
        "partitionKey": "${your-instance-id}"
      }
    }
  ],
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23"
}
```

2. Crie a regra de tópico do MQTT usando o AWS CLI. Essa etapa usa o AWS CLI para criar uma regra de AWS IoT Core tópico usando a configuração definida no `events_rule.json` arquivo.

```
aws iot create-topic-rule \
  --rule-name "your-iot-core-rule" \
```

```
--topic-rule-payload file://your-file-name.json
```

Etapa 5: Obtenha o endpoint para o tópico MQTT de destino

Use o tópico MQTT de destino para configurar onde seus tópicos publicam mensagens de saída, substituindo a funcionalidade anteriormente tratada pelo. AWS IoT Events O endpoint é exclusivo para sua AWS conta e região.

Console

1. Abra o [console do AWS IoT Core](#).
2. Na seção Connect no painel de navegação esquerdo, escolha Configuração do domínio.
3. Escolha a configuração do domínio IoT:Data-ATS para abrir a página de detalhes da configuração.
4. Copie o valor do nome de domínio. Esse valor é o endpoint. Salve o valor do endpoint porque você precisará dele em etapas posteriores.

AWS CLI

Execute o comando a seguir para obter o AWS IoT Core endpoint para publicar mensagens de saída para sua conta.

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

Etapa 6: criar uma tabela do Amazon DynamoDB

Uma tabela do Amazon DynamoDB substitui a funcionalidade de gerenciamento de estado AWS IoT Events do, fornecendo uma forma escalável e flexível de persistir e gerenciar o estado de seus dispositivos e a lógica do modelo de detector em sua nova arquitetura de solução.

Console

Crie uma tabela do Amazon DynamoDB para manter o estado dos modelos de detectores. Para obter mais informações, consulte [Criar uma tabela no DynamoDB no Amazon DynamoDB Developer Guide](#).

Use o seguinte para obter os detalhes da tabela:

- Em Nome da tabela, insira um nome de tabela de sua escolha.
- Em Partition Key, insira seu próprio ID de instância.
- Você pode usar as configurações padrão para as configurações da tabela

AWS CLI

Execute o comando a seguir para criar uma tabela do DynamoDB.

```
aws dynamodb create-table \  
    --table-name your-table-name \  
    --attribute-definitions AttributeName=your-instance-  
id,AttributeType=S \  
    --key-schema AttributeName=your-instance-id,KeyType=HASH \  
    --
```

Etapa 7: criar uma AWS Lambda função (console)

A função Lambda serve como o principal mecanismo de processamento, substituindo a lógica de avaliação do modelo de detector do. AWS IoT Events No exemplo, nos integramos com outros AWS serviços para lidar com dados recebidos, gerenciar o estado e acionar ações com base nas regras definidas por você.

Crie uma função Lambda com NodeJS tempo de execução. Use o seguinte trecho de código, substituindo as constantes codificadas:

1. Abra a [AWS Lambda console](#).
2. Escolha a opção Criar função.
3. Insira um nome para o nome da função.
4. Selecione NodeJS 22.x como Runtime.
5. No menu suspenso Alterar função de execução padrão, escolha Usar função existente e selecione a função do IAM que você criou nas etapas anteriores.
6. Escolha a opção Criar função.
7. Cole o seguinte trecho de código depois de substituir as constantes codificadas.
8. Depois que sua função for criada, na guia Código, cole o exemplo de código a seguir, substituindo o **your-destination-endpoint** endpoint pelo seu.

```
import { DynamoDBClient, GetItemCommand } from '@aws-sdk/client-dynamodb';
import { PutItemCommand } from '@aws-sdk/client-dynamodb';
import { IoTDataPlaneClient, PublishCommand } from "@aws-sdk/client-iot-data-plane";
import { SchedulerClient, CreateScheduleCommand, DeleteScheduleCommand } from "@aws-
sdk/client-scheduler"; // ES Modules import

//// External Clients and Constants
const scheduler = new SchedulerClient({});
const iot = new IoTDataPlaneClient({
  endpoint: 'https://your-destination-endpoint-ats.iot.your-region.amazonaws.com/'
});
const ddb = new DynamoDBClient({});

//// Lambda Handler function
export const handler = async (event) => {
  console.log('Incoming event:', JSON.stringify(event, null, 2));

  if (!event.Records) {
    throw new Error('No records found in event');
  }

  const processedRecords = [];

  for (const record of event.Records) {
    try {
      if (record.eventSource !== 'aws:kinesis') {
        console.log(`Skipping non-Kinesis record from ${record.eventSource}`);
        continue;
      }

      // Assumes that we are processing records from Kinesis
      const payload = record.kinesis.data;
      const decodedData = Buffer.from(payload, 'base64').toString();
      console.log("decoded payload is ", decodedData);

      const output = await handleDecodedData(decodedData);

      // Add additional processing logic here
      const processedData = {
        output,
```

```
        sequenceNumber: record.kinesis.sequenceNumber,
        partitionKey: record.kinesis.partitionKey,
        timestamp: record.kinesis.approximateArrivalTimestamp
    };

    processedRecords.push(processedData);

} catch (error) {
    console.error('Error processing record:', error);
    console.error('Failed record:', record);
    // Decide whether to throw error or continue processing other records
    // throw error; // Uncomment to stop processing on first error
}
}

return {
    statusCode: 200,
    body: JSON.stringify({
        message: 'Processing complete',
        processedCount: processedRecords.length,
        records: processedRecords
    })
};
};

// Helper function to handle decoded data
async function handleDecodedData(payload) {
    try {
        // Parse the decoded data
        const parsedData = JSON.parse(payload);

        // Extract instanceId
        const instanceId = parsedData.instanceId;
        // Parse the input field
        const inputData = JSON.parse(parsedData.payload);
        const temperature = inputData.temperature;
        console.log('For InstanceId: ', instanceId, ' the temperature is:',
temperature);

        await iotEvents.process(instanceId, inputData)

        return {
            instanceId,
            temperature,

```

```

        // Add any other fields you want to return
        rawInput: inputData
    };
} catch (error) {
    console.error('Error handling decoded data:', error);
    throw error;
}
}

///// Classes for declaring/defining the state machine
class CurrentState {
    constructor(instanceId, stateName, variables, inputs) {
        this.stateName = stateName;
        this.variables = variables;
        this.inputs = inputs;
        this.instanceId = instanceId
    }

    static async load(instanceId) {
        console.log(`Loading state for id ${instanceId}`);
        try {
            const { Item: { state: { S: stateContent } } } = await ddb.send(new
GetItemCommand({
                TableName: 'EventsStateTable',
                Key: {
                    'InstanceId': { S: `${instanceId}` }
                }
            }));

            const { stateName, variables, inputs } = JSON.parse(stateContent);

            return new CurrentState(instanceId, stateName, variables, inputs);
        } catch (e) {
            console.log(`No state for id ${instanceId}: ${e}`);
            return undefined;
        }
    }

    static async save(instanceId, state) {
        console.log(`Saving state for id ${instanceId}`);
        await ddb.send(new PutItemCommand({
            TableName: 'your-events-state-table-name',
            Item: {

```

```

        'InstanceId': { S: `${instanceId}` },
        'state': { S: state }
    }
    }));
}

setVariable(name, value) {
    this.variables[name] = value;
}

changeState(stateName) {
    console.log(`Changing state from ${this.stateName} to ${stateName}`);
    this.stateName = stateName;
}

async setTimer(instanceId, frequencyInMinutes, payload) {
    console.log(`Setting timer ${instanceId} for frequency of ${frequencyInMinutes}
minutes`);

    const base64Payload = Buffer.from(JSON.stringify(payload)).toString();
    console.log(base64Payload);

    const scheduleName = `your-schedule-name-${instanceId}-schedule`;
    const scheduleParams = {
        Name: scheduleName,
        FlexibleTimeWindow: {
            Mode: 'OFF'
        },
        ScheduleExpression: `rate(${frequencyInMinutes} minutes)`,
        Target: {
            Arn: "arn:aws::kinesis:your-region:your-account-id:stream/your-kinesis-
stream-name",
            RoleArn: "arn:aws::iam:your-account-id:role/service-role/your-iam-
role",
            Input: base64Payload,
            KinesisParameters: {
                PartitionKey: instanceId,
            },
            RetryPolicy: {
                MaximumRetryAttempts: 3
            }
        },
    };
};

```

```
    const command = new CreateScheduleCommand(scheduleParams);
    console.log(`Sending command to set timer ${JSON.stringify(command)}`);
    await scheduler.send(command);
  }

  async clearTimer(instanceId) {
    console.log(`Cleaning timer ${instanceId}`);

    const scheduleName = `your-schedule-name-${instanceId}-schedule`;
    const command = new DeleteScheduleCommand({
      Name: scheduleName
    });
    await scheduler.send(command);
  }

  async executeAction(actionType, actionPayload) {
    console.log(`Will execute the ${actionType} with payload ${actionPayload}`);
    await iot.send(new PublishCommand({
      topic: `${this.instanceId}`,
      payload: actionPayload,
      qos: 0
    })));
  }

  setInput(value) {
    this.inputs = { ...this.inputs, ...value };
  }

  input(name) {
    return this.inputs[name];
  }
}

class IoTEvents {

  constructor(initialState) {
    this.initialState = initialState;
    this.states = {};
  }

  state(name) {
    const state = new IoTEventsState();
  }
}
```

```
    this.states[name] = state;
    return state;
  }

  async process(instanceId, input) {
    let currentState = await CurrentState.load(instanceId) || new
    CurrentState(instanceId, this.initialState, {}, {});
    currentState.setInput(input);

    console.log(`With inputs as: ${JSON.stringify(currentState)}`);
    const state = this.states[currentState.stateName];

    currentState = await state.evaluate(currentState);
    console.log(`With output as: ${JSON.stringify(currentState)}`);

    await CurrentState.save(instanceId, JSON.stringify(currentState));
  }
}

class Event {
  constructor(condition, action) {
    this.condition = condition;
    this.action = action;
  }
}

class IoTEventsState {
  constructor() {
    this.eventsList = []
  }

  events(eventListArg) {
    this.eventsList.push(...eventListArg);
    return this;
  }

  async evaluate(currentState) {
    for (const e of this.eventsList) {
      console.log(`Evaluating event ${e.condition}`);
      if (e.condition(currentState)) {
        console.log(`Event condition met`);
        // Execute any action as defined in iotEvents DM Definition
        await e.action(currentState);
      }
    }
  }
}
```

```

    }

    return currentState;
  }
}

///// DetectorModel Definitions - replace with your own defintions
let processAlarmStateEvent = new Event(
  (currentState) => {
    const source = currentState.input('source');
    return (
      currentState.input('temperature') < 70
    );
  },
  async (currentState) => {
    currentState.changeState('normal');
    await currentState.clearTimer(currentState.instanceId)
    await currentState.executeAction('MQTT', `{"state": "alarm cleared, timer
deleted" }`);
  }
);

let processTimerEvent = new Event(
  (currentState) => {
    const source = currentState.input('source');
    console.log(`Evaluating timer event with source ${source}`);
    const booleanOutput = (source !== undefined && source !== null &&
      typeof source === 'string' &&
      source.toLowerCase() === 'timer' &&
      // check if the currentState == state from the timer payload
      currentState.input('currentState') !== undefined &&
      currentState.input('currentState') !== null &&
      currentState.input('currentState').toLowerCase !== 'normal');
    console.log(`Timer event evaluated as ${booleanOutput}`);
    return booleanOutput;
  },
  async (currentState) => {
    await currentState.executeAction('MQTT', `{"state": "timer timed out in
Alarming state" }`);
  }
);

let processNormalEvent = new Event(
  (currentState) => currentState.input('temperature') > 70,

```

```

    async (currentState) => {
        currentState.changeState('alarm');
        await currentState.executeAction('MQTT', `{"state": "alarm detected, timer
started" }`);
        await currentState.setTimer(currentState.instanceId, 5, {
            "instanceId": currentState.instanceId,
            "payload": "{\"currentState\": \"alarm\", \"source\": \"timer\"}"
        });
    }
};
const iotEvents = new IoTEvents('normal');
iotEvents
    .state('normal')
    .events(
        [
            processNormalEvent
        ]
    );
iotEvents
    .state('alarm')
    .events([
        processAlarmStateEvent,
        processTimerEvent
    ]
);

```

Etapa 8: Adicionar um gatilho do Amazon Kinesis Data Streams

Adicione um gatilho do Kinesis Data Streams à função Lambda usando o ou. Console de gerenciamento da AWS AWS CLI

Adicionar um gatilho do Kinesis Data Streams à sua função Lambda estabelece a conexão entre seu pipeline de ingestão de dados e sua lógica de processamento, permitindo que ele avalie automaticamente os fluxos de dados de IoT recebidos e reaja aos eventos em tempo real, da mesma forma que os processos entram. AWS IoT Events

Console

Para obter mais informações, consulte [Criar um mapeamento de origem de eventos para invocar uma função Lambda](#) no Guia AWS Lambda do desenvolvedor.

Use o seguinte para obter os detalhes do mapeamento da origem do evento:

- Em Nome da função, insira o nome lambda usado em [Etapa 7: criar uma AWS Lambda função \(console\)](#).
- Para Consumidor, opcional, insira o ARN do seu stream do Kinesis.
- Em Tamanho do lote, insira **10**.

AWS CLI

Execute o comando a seguir para criar o gatilho da função Lambda.

```
aws lambda create-event-source-mapping \  
  --function-name your-lambda-name \  
  --event-source arn:aws:kinesis:your-region:your-account-id:stream/your-kinesis-  
stream-name \  
  --batch-size 10 \  
  --starting-position LATEST \  
  --region your-region
```

Etapa 9: testar a funcionalidade de ingestão e saída de dados ()AWS CLI

Publique uma carga no tópico do MQTT com base no que você definiu em seu modelo de detector. Veja a seguir um exemplo de carga útil para o tópico MQTT `your-topic-name` para testar uma implementação.

```
{  
  "instanceId": "your-instance-id",  
  "payload": "{\"temperature\":78}"  
}
```

Você deve ver uma mensagem MQTT publicada em um tópico com o seguinte conteúdo (ou similar):

```
{  
  "state": "alarm detected, timer started"  
}
```

Procedimento de migração para AWS IoT SiteWise alarmes em AWS IoT Events

Esta seção descreve soluções alternativas que oferecem funcionalidade de alarme semelhante à da AWS IoT Events qual você migra.

Para AWS IoT SiteWise propriedades que usam AWS IoT Events alarmes, você pode migrar para uma solução usando CloudWatch alarmes. Essa abordagem fornece recursos robustos de monitoramento com recursos estabelecidos SLAs e adicionais, como detecção de anomalias e alarmes agrupados.

Comparando arquiteturas

A configuração atual de AWS IoT Events alarmes para AWS IoT SiteWise propriedades requer a criação `AssetModelCompositeModels` no modelo de ativos, conforme descrito em [Definir alarmes externos AWS IoT SiteWise no Guia do AWS IoT SiteWise usuário](#). As modificações na nova solução geralmente são gerenciadas por meio do AWS IoT Events console.

A nova solução fornece gerenciamento de alarmes aproveitando os CloudWatch alarmes. Essa abordagem usa AWS IoT SiteWise notificações para publicar pontos de dados de propriedades em tópicos do AWS IoT Core MQTT, que são então processados por uma função Lambda. A função transforma essas notificações em CloudWatch métricas, permitindo o monitoramento de alarmes por meio CloudWatch da estrutura de alarmes.

Finalidade	Solução	Diferenças
Fonte de dados — dados de propriedades de AWS IoT SiteWise	AWS IoT SiteWise Notificações MQTT	Substitui a integração direta do IoT Events por notificações MQTT de propriedades AWS IoT SiteWise
Processamento de dados — transforma dados de propriedades	Função do Lambda	AWS IoT SiteWise Processa notificações de propriedades e as converte em CloudWatch métricas

Finalidade	Solução	Diferenças
Avaliação de alarmes — monitora métricas e aciona alarmes	CloudWatch Alarmes da Amazon	Substitui AWS IoT Events alarmes por alarmes, oferecendo recursos adicionais, como detecção de CloudWatch anomalias
Integração — Conexão com AWS IoT SiteWise	AWS IoT SiteWise alarmes externos	Capacidade opcional de importar CloudWatch alarmes de volta AWS IoT SiteWise como alarmes externos

Etapa 1: ativar as notificações do MQTT na propriedade do ativo

Se você estiver usando AWS IoT Events integrações para AWS IoT SiteWise alarmes, poderá ativar as notificações do MQTT para cada propriedade a ser monitorada.

1. Siga o AWS IoT SiteWise procedimento [Configurar alarmes em ativos](#) até chegar à etapa de edição das propriedades do modelo de ativo.
2. Para cada propriedade a ser migrada, altere o status da Notificação MQTT para ATIVO.

The screenshot shows the 'Properties' configuration interface. On the left, a sidebar lists 'Property Type' with options: 'Attributes (1)' (selected), 'Measurements (2)', 'Transforms (0)', and 'Metrics (2)'. The main area is titled 'Attributes' and shows a field for 'Alarm-Recipient' with a note 'Must be less than 2048 characters.' To the right, a red-bordered box highlights the 'MQTT Notification status' dropdown menu, which is currently set to 'ACTIVE'. Below the dropdown, the notification topic path is displayed: 'Notification will be published to topic \$aws/sitewise/asset-models/{asset-id}/assets/{property-id}/properties/{property-name}'.

3. Observe o caminho do tópico no qual o alarme é publicado para cada atributo de alarme modificado.

Para obter mais informações, consulte os seguintes recursos de documentação:

- [Entenda as propriedades dos ativos nos tópicos do MQTT](#) no Guia AWS IoT SiteWise do usuário.
- [Tópicos do MQTT](#) no Guia do AWS IoT Desenvolvedor.

Etapa 2: criar uma AWS Lambda função

Crie uma função Lambda para ler a matriz TQV publicada pelo tópico MQTT e publique valores individuais em. CloudWatch Usaremos essa função Lambda como uma ação de destino a ser acionada nas Regras de AWS IoT Core Mensagens.

1. Abra a [AWS Lambda console](#).
2. Escolha a opção Criar função.
3. Insira um nome para o nome da função.
4. Selecione NodeJS 22.x como Runtime.
5. No menu suspenso Alterar função de execução padrão, escolha Usar função existente e selecione a função do IAM que você criou nas etapas anteriores.

Note

Esse procedimento pressupõe que você já migrou seu modelo de detector. Se você não tiver uma função do IAM, consulte [???](#).

6. Escolha a opção Criar função.
7. Cole o seguinte trecho de código depois de substituir as constantes codificadas.

```
import json
import boto3
from datetime import datetime

# Initialize CloudWatch client
cloudwatch = boto3.client('cloudwatch')

def lambda_handler(message, context):
    try:
        # Parse the incoming IoT message
        # Extract relevant information
        asset_id = message['payload']['assetId']
        property_id = message['payload']['propertyId']

        # Process each value in the values array
        for value in message['payload']['values']:
            # Extract timestamp and value
            timestamp = datetime.fromtimestamp(value['timestamp']['timeInSeconds'])
            metric_value = value['value']['doubleValue']
```

```
quality = value.get('quality', 'UNKNOWN')

# Publish to CloudWatch
response = cloudwatch.put_metric_data(
    Namespace='IoTSiteWise/AssetMetrics',
    MetricData=[
        {
            'MetricName': f'Property_your-property-id',
            'Value': metric_value,
            'Timestamp': timestamp,
            'Dimensions': [
                {
                    'Name': 'AssetId',
                    'Value': 'your-asset-id'
                },
                {
                    'Name': 'Quality',
                    'Value': quality
                }
            ]
        }
    ]
)

return {
    'statusCode': 200,
    'body': json.dumps('Successfully published metrics to CloudWatch')
}

except Exception as e:
    print(f'Error processing message: {str(e)}')
    return {
        'statusCode': 500,
        'body': json.dumps(f'Error: {str(e)}')
    }
```

Etapa 3: criar uma regra de roteamento de AWS IoT Core mensagens

- Siga o [tutorial: Procedimento de republicação de uma mensagem MQTT](#) inserindo as seguintes informações quando solicitado:
 - a. Nomeie a regra SiteWiseToCloudwatchAlarms de roteamento de mensagens.

- b. Para a consulta, você pode usar o seguinte:

```
SELECT * FROM '$aws/sitewise/asset-models/your-asset-model-id/assets/your-asset-id/properties/your-property-id'
```

- c. Em Ações de regra, selecione a ação Lambda para enviar os dados gerados de AWS IoT SiteWise para. CloudWatch Por exemplo:

Rule actions Info
Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. You can add up to 10 actions.

Action 1

▼ **Lambda**
Send a message to a Lambda function Remove

Lambda function Info
ListenForSiteWiseUpdates View Create a Lambda function

Lambda function version
\$LATEST Refresh

Add rule action

Etapa 4: visualizar CloudWatch métricas

À medida que você ingere dados para AWS IoT SiteWise a propriedade selecionada anteriormente [Etapa 1: ativar as notificações do MQTT na propriedade do ativo](#), os dados são encaminhados para a função Lambda que criamos em [Etapa 2: criar uma AWS Lambda função](#). Nesta etapa, você pode conferir se o Lambda está enviando suas métricas para. CloudWatch

1. Abra o [CloudWatch do Console de gerenciamento da AWS](#).
2. No painel de navegação à esquerda, escolha Métricas e, em seguida, Todas as métricas.
3. Escolha o URL de um alarme para abri-lo.
4. Na guia Fonte, a CloudWatch saída é semelhante a este exemplo. Essas informações de origem confirmam que os dados métricos estão sendo CloudWatch inseridos.

```
{
  "view": "timeSeries",
  "stacked": false,
  "metrics": [
    [ "IoTSiteWise/AssetMetrics", "Property_your-property-id-hash", "Quality",
    "GOOD", "AssetId", "your-asset-id-hash", { "id": "m1" } ]
  ],
  "region": "your-region"
}
```

Etapa 5: criar CloudWatch alarmes

Siga o procedimento [Criar um CloudWatch alarme com base em um limite estático](#) no Guia do CloudWatch usuário da Amazon para criar alarmes para cada métrica relevante.

Note

Há muitas opções para configuração de alarmes na Amazon. CloudWatch Para obter mais informações sobre CloudWatch alarmes, consulte [Usando CloudWatch alarmes da Amazon no Guia CloudWatch](#) do usuário da Amazon.

Etapa 6: (opcional) importar o CloudWatch alarme para AWS IoT SiteWise

Você pode configurar CloudWatch alarmes para enviar dados de volta AWS IoT SiteWise usando ações de CloudWatch alarme e Lambda. Essa integração permite que você visualize os estados e propriedades do alarme no portal do SiteWise Monitor.

1. Configure o alarme externo como uma propriedade em um modelo de ativo. Para obter mais informações, consulte [Definir alarmes externos AWS IoT SiteWise no](#) Guia do AWS IoT SiteWise usuário.
2. Crie uma função Lambda que use a [BatchPutAssetPropertyValueAPI](#) encontrada no Guia do AWS IoT SiteWise usuário para enviar dados de alarme para o. AWS IoT SiteWise
3. Configure ações de CloudWatch alarme para invocar sua função Lambda quando os estados do alarme mudarem. Para obter mais informações, consulte a seção [Ações de alarme](#) no Guia CloudWatch do usuário da Amazon.

Conf AWS IoT Events configuração

Esta seção fornece um guia de configuração AWS IoT Events, incluindo a criação de uma AWS conta, a configuração das permissões necessárias e o estabelecimento de funções para gerenciar o acesso aos recursos.

Tópicos

- [Configurando um Conta da AWS](#)
- [Configurando permissões para AWS IoT Events](#)

Configurando um Conta da AWS

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 AWS é 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 AWS](#) como 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 .

Configurando permissões para AWS IoT Events

A implementação de permissões adequadas é importante para o uso seguro e eficaz do AWS IoT Events. Esta seção descreve as permissões necessárias para usar alguns recursos do AWS IoT Events. Você pode usar AWS CLI comandos ou o console AWS Identity and Access Management (IAM) para criar funções e políticas de permissão associadas para acessar recursos ou realizar determinadas funções no AWS IoT Events.

O [Guia do usuário do IAM](#) tem informações mais detalhadas sobre como controlar com segurança as permissões de acesso AWS aos recursos. Para obter informações específicas sobre AWS IoT Events, consulte [Ações, recursos e chaves de condição para AWS IoT Events](#).

Para usar o console do IAM para criar e gerenciar funções e permissões, consulte o [tutorial do IAM: Delegar acesso entre AWS contas usando funções do IAM](#).

Note

As chaves podem ter de 1 a 128 caracteres e podem incluir:

- letras maiúsculas ou minúsculas de a-z
- números 0-9
- caracteres especiais -, _ ou .:

Permissões de ação para AWS IoT Events

AWS IoT Events permite que você acione ações que usam outros AWS serviços. Para fazer isso, você deve conceder AWS IoT Events permissão para realizar essas ações em seu nome. Esta seção contém uma lista das ações e um exemplo de política que concede permissão para realizar todas essas ações em seus recursos. *region* Altere as *account-id* referências e conforme necessário. Quando possível, você também deve alterar os curingas (*) para se referir a recursos específicos que serão acessados. Você pode usar o console do IAM para conceder permissão AWS IoT Events para enviar um alerta do Amazon SNS que você definiu.

AWS IoT Events suporta as seguintes ações que permitem usar um cronômetro ou definir uma variável:

- [setTimer](#) para criar um temporizador.
- [resetTimer](#) para redefinir o temporizador.
- [clearTimer](#) para excluir o temporizador.
- [setVariable](#) para criar uma variável.

AWS IoT Events suporta as seguintes ações que permitem trabalhar com AWS serviços:

- [iotTopicPublish](#) para publicar uma mensagem em um tópico MQTT.
- [iotEvents](#) para enviar dados para AWS IoT Events como um valor de entrada.
- [iotSiteWise](#) para enviar dados para uma propriedade de ativo no AWS IoT SiteWise.
- [dynamoDB](#) para enviar dados para uma tabela do Amazon DynamoDB.
- [dynamoDBv2](#) para enviar dados para uma tabela do Amazon DynamoDB.
- [firehose](#) para enviar dados para um stream do Amazon Data Firehose.
- [lambda](#) para invocar uma função do AWS Lambda .
- [sns](#) para enviar dados como uma notificação por push.
- [sqs](#) para enviar dados para uma fila do Amazon SQS.

Example Política

JSON

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": "iot:Publish",
    "Resource": "arn:aws:iot:us-east-1:123456789012:topic/*"
  },
  {
    "Effect": "Allow",
    "Action": "iotevents:BatchPutMessage",
    "Resource": "arn:aws:iotevents:us-east-1:123456789012:input/*"
  },
  {
    "Effect": "Allow",
    "Action": "iotsitewise:BatchPutAssetPropertyValue",
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": "dynamodb:PutItem",
    "Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "firehose:PutRecord",
      "firehose:PutRecordBatch"
    ],
    "Resource": "arn:aws:firehose:us-east-1:123456789012:deliverystream/
**
  },
  {
    "Effect": "Allow",
    "Action": "lambda:InvokeFunction",
    "Resource": "arn:aws:lambda:us-east-1:123456789012:function:*"
  },
  {
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:us-east-1:123456789012:*"
  },
  {
    "Effect": "Allow",
    "Action": "sqs:SendMessage",

```

```

    "Resource": "arn:aws:sqs:us-east-1:123456789012:*"
  }
]
}

```

Protegendo os dados de entrada em AWS IoT Events

É importante considerar quem pode conceder acesso aos dados de entrada para uso em um modelo de detector. Caso tenha um usuário ou entidade cujas permissões gerais deseja restringir, mas que tem permissão para criar ou atualizar um modelo de detector, você também deve conceder permissão para que esse usuário ou entidade atualize o roteamento de entrada. Isso significa que, além de conceder permissão para `iotevents:CreateDetectorModel` e `iotevents:UpdateDetectorModel`, você também deve conceder permissão para `iotevents:UpdateInputRouting`.

Example

A política a seguir adiciona permissões para `iotevents:UpdateInputRouting`.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "updateRoutingPolicy",
      "Effect": "Allow",
      "Action": [
        "iotevents:UpdateInputRouting"
      ],
      "Resource": "*"
    }
  ]
}

```

Você pode especificar uma lista de nomes de recursos da Amazon de entrada (ARNs) em vez do caractere curinga `* *` para o `"Resource"` para limitar essa permissão a entradas específicas. Isso permite restringir o acesso aos dados de entrada que são consumidos pelos modelos de detectores criados ou atualizados pelo usuário ou pela entidade.

Política de função de CloudWatch registro da Amazon para AWS IoT Events

Os documentos de política a seguir fornecem a política de funções e AWS IoT Events a política de confiança que permitem enviar registros CloudWatch em seu nome.

Política da função:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:PutMetricFilter",
        "logs:PutRetentionPolicy",
        "logs:GetLogEvents",
        "logs>DeleteLogStream"
      ],
      "Resource": [
        "arn:aws:logs:*:*:*"
      ]
    }
  ]
}
```

Política de confiança:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```

    "Effect": "Allow",
    "Principal": {
      "Service": [

        "iotevents.amazonaws.com"
      ]
    },
    "Action": "sts:AssumeRole"
  }
]
}

```

Você também precisa de uma política de permissões anexada ao usuário do IAM que permita ao usuário transmitir funções, como segue. Para obter mais informações, consulte [Conceder permissões a um usuário para passar uma função para um AWS serviço](#) no Guia do usuário do IAM.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/Role_To_Pass"
    }
  ]
}

```

Você pode usar o comando a seguir para colocar a política de recursos para CloudWatch registros. Isso permite AWS IoT Events colocar eventos de log em CloudWatch fluxos.

```

aws logs put-resource-policy --policy-name ioteventsLoggingPolicy --policy-
document "{ \"Version\": \"2012-10-17\",          \"Statement\": [ { \"Sid\":
  \"IoTEventsToCloudWatchLogs\", \"Effect\": \"Allow\", \"Principal\": { \"Service\":

```

```
[ \"iotevents.amazonaws.com\" ] }, \"Action\": \"logs:PutLogEvents\", \"Resource\": \"*\n\" } ] ]\"
```

Use o comando a seguir para colocar opções de registro em log. Substitua o `roleArn` pela função de registro em log criada.

```
aws iotevents put-logging-options --cli-input-json \"{ \"loggingOptions\": {\"roleArn\":\n\"arn:aws:iam::123456789012:role/testLoggingRole\", \"level\": \"INFO\", \"enabled\":\ntrue } }\"
```

Política de função de mensagens do Amazon SNS para AWS IoT Events

A integração AWS IoT Events com o Amazon SNS exige um gerenciamento cuidadoso de permissões para a entrega segura e eficiente de notificações. Este guia mostra o processo de configuração de funções e políticas do IAM para permitir AWS IoT Events a publicação de mensagens em tópicos do Amazon SNS.

Os documentos de políticas a seguir fornecem as políticas de função e de confiança que permitem que o AWS IoT Events envie entradas de log ao SNS em seu nome.

Política da função:

JSON

```
{\n  \"Version\": \"2012-10-17\", \n  \"Statement\": [\n    {\n      \"Action\": [\n        \"sns:*\"\n      ],\n      \"Effect\": \"Allow\", \n      \"Resource\": \"arn:aws:sns:us-east-1:123456789012:testAction\"\n    }\n  ]\n}
```

Política de confiança:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Começando a usar o AWS IoT Events console

Esta seção mostra como criar uma entrada e um modelo de detector usando o [console do AWS IoT Events](#). Você modela dois estados de um motor: um estado normal e uma condição de sobrepressão. Quando a pressão medida no motor excede um determinado limite, o modelo passa do estado normal para o estado de sobrepressão. Em seguida, ele envia uma mensagem do Amazon SNS para alertar um técnico sobre a condição. Quando a pressão cai novamente abaixo do limite por três leituras de pressão consecutivas, o modelo retorna ao estado normal e envia outra mensagem do Amazon SNS como confirmação.

Verificamos três leituras consecutivas abaixo do limite de pressão para eliminar possíveis interrupções de sobrepressão ou mensagens normais, no caso de uma fase de recuperação não linear ou de uma leitura de pressão anômala.

No console, você também pode encontrar vários modelos de detectores pré-fabricados que podem ser personalizados. Você também pode usar o console para importar modelos de detectores que outras pessoas escreveram ou exportar seus modelos de detectores e usá-los em diferentes AWS regiões. Se você importar um modelo de detector, certifique-se de criar as entradas necessárias ou recriá-las para a nova região e atualizar qualquer função ARNs usada.

Use o AWS IoT Events console para saber mais sobre o seguinte.

Definir entradas

Para monitorar seus dispositivos e processos, eles devem ter uma maneira de inserir dados de telemetria no AWS IoT Events. Isso é feito enviando mensagens como entradas para AWS IoT Events. Você pode fazer isso de várias maneiras:

- Use a operação [BatchPutMessage](#).
- Em AWS IoT Core, escreva uma regra de [AWS IoT Events ação](#) para o mecanismo de AWS IoT regras que encaminha os dados da mensagem para o AWS IoT Events. Você deve identificar a entrada pelo nome.
- Em AWS IoT Analytics, use a [CreateDataset](#) operação para criar um conjunto de dados com `contentDeliveryRules`. Essas regras especificam a AWS IoT Events entrada para a qual o conteúdo do conjunto de dados é enviado automaticamente.

Antes que seus dispositivos possam enviar dados dessa forma, é preciso definir uma ou mais entradas. Para fazer isso, dê um nome a cada entrada e especifique quais campos nos dados da mensagem recebida a entrada monitora.

Como criar um modelo de detector

Defina um modelo de detector (um modelo de seu equipamento ou processo) usando estados. Para cada estado, você define a lógica condicional (booliana) que avalia as entradas recebidas para detectar eventos significativos. Quando o modelo do detector detecta um evento, ele pode alterar o estado ou iniciar ações personalizadas ou predefinidas usando outros serviços. AWS É possível definir eventos adicionais que acionam ações ao entrar ou sair de um estado e, opcionalmente, quando uma condição é atendida.

Neste tutorial, você envia uma mensagem do Amazon SNS como a ação quando o modelo entra ou sai de um determinado estado.

Como monitorar um dispositivo ou processo

Se você monitorar vários dispositivos ou processos, especifique um campo em cada entrada que identifique o dispositivo ou processo específico do qual a entrada vem. Veja o campo `key` em `CreateDetectorModel`. Quando o campo de entrada identificado pelo `key` reconhece um novo valor, um novo dispositivo é identificado e um detector é criado. Cada detector é uma instância de um modelo de detector. O novo detector continua respondendo às entradas provenientes desse dispositivo até que seu modelo seja atualizado ou excluído.

Se você monitora um único processo (mesmo que vários dispositivos ou subprocessos estejam enviando entradas), você não especifica um campo de identificação `key` exclusivo. Nesse caso, o modelo cria um único detector (instância) quando a primeira entrada chega.

Como enviar mensagens como entradas para seu modelo de detector

Há várias maneiras de enviar uma mensagem de um dispositivo ou processo como entrada em um detector do AWS IoT Events que não exigem que você execute formatação adicional na mensagem. Neste tutorial, você usa o AWS IoT console para escrever uma regra de [AWS IoT Events ação](#) para o mecanismo de AWS IoT regras que encaminha os dados da mensagem para o. AWS IoT Events

Para fazer isso, identifique a entrada pelo nome e continue usando o AWS IoT console para gerar mensagens que são encaminhadas como entradas para. AWS IoT Events

Note

Este tutorial usa o console para criar o mesmo input e detector model mostrado no exemplo em [Tutoriais para casos de uso AWS IoT Events](#). É possível usar este exemplo de JSON para ajudar a seguir o tutorial.

Tópicos

- [Pré-requisitos para começar AWS IoT Events](#)
- [Crie uma entrada para modelos em AWS IoT Events](#)
- [Crie um modelo de detector em AWS IoT Events](#)
- [Envie entradas para testar o modelo do detector em AWS IoT Events](#)

Pré-requisitos para começar AWS IoT Events

Se você não tiver uma AWS conta, crie uma.

1. Siga as etapas [Conf AWS IoT Events iguração](#) para garantir a configuração e as permissões adequadas da conta.
2. Criar dois tópicos Amazon Simple Notification Service (Amazon SNS).

Este tutorial (e o exemplo correspondente) supõe que você criou dois tópicos do Amazon SNS. Esses tópicos são mostrados como: `arn:aws:sns:us-east-1:123456789012:underPressureAction` e `arn:aws:sns:us-east-1:123456789012:pressureClearedAction`. Substitua esses valores pelos tópicos ARNs do Amazon SNS que você criar. Para obter mais informações, consulte o [Manual do desenvolvedor do Amazon Simple Notification Service](#).

Como alternativa à publicação de alertas para tópicos do Amazon SNS, é possível fazer com que os detectores enviem mensagens MQTT com um tópico especificado por você. Com essa opção, você pode verificar se seu modelo de detector está criando instâncias e se essas instâncias estão enviando alertas usando o console AWS IoT principal para assinar e monitorar as mensagens enviadas para esses tópicos do MQTT. Também é possível definir o nome do tópico MQTT dinamicamente em runtime usando uma entrada ou variável criada no modelo do detector.

3. Escolha um Região da AWS que suporte AWS IoT Events. Para ter mais informações, consulte [AWS IoT Events](#) no Referência geral da AWS. Para [obter ajuda, consulte Introdução a um serviço Console de gerenciamento da AWS em](#) Introdução ao Console de gerenciamento da AWS.

Crie uma entrada para modelos em AWS IoT Events

Ao criar as entradas para seus modelos, recomendamos reunir arquivos que contenham amostras de cargas úteis de mensagens que seus dispositivos ou processos enviam para relatar seu estado de saúde. Ter esses arquivos ajuda a definir as entradas necessárias.

Você pode criar uma entrada por meio de vários métodos descritos nesta seção.

Crie um arquivo de entrada JSON

1. Para começar, crie um arquivo chamado `input.json` em seu sistema de arquivos local com o seguinte conteúdo:

```
{
  "motorid": "Fulton-A32",
  "sensorData": {
    "pressure": 23,
    "temperature": 47
  }
}
```

2. Agora que tem esse arquivo inicial `input.json`, você pode criar uma entrada. Há duas maneiras de criar uma entrada. Você pode criar uma entrada usando o painel de navegação no [AWS IoT Events console](#). Ou você pode criar uma entrada dentro do modelo do detector depois que ele for criado.

Criar e configurar uma entrada

Aprenda a criar uma entrada para um modelo de alarme ou um modelo de detector.

1. Faça login no [AWS IoT Events console](#) ou selecione a opção Criar uma nova AWS IoT Events conta.

2. No AWS IoT Events console, no canto superior esquerdo, selecione e expanda o painel de navegação.
3. No painel de navegação à esquerda, escolha Entradas.
4. No canto direito do console, escolha Criar entrada.
5. Forneça um exclusivo InputName.
6. Opcional — insira uma Descrição para sua entrada.
7. Para fazer upload de um arquivo JSON, selecione o `input.json` arquivo que você criou na visão geral para [Crie um arquivo de entrada JSON](#). A opção Escolher atributos de entrada aparece com uma lista dos atributos inseridos.
8. Em Selecionar atributo de entrada, escolha os atributos a serem usados e selecione Criar. Neste exemplo, selecionamos `motorid` e `sensorData.pressure`.
9. Opcional — Adicione tags relevantes à entrada.

Note

Você também pode criar entradas adicionais dentro do modelo do detector no [AWS IoT Events console](#). Para obter mais informações, consulte [Crie uma entrada dentro do modelo do detector em AWS IoT Events](#).

Crie uma entrada dentro do modelo do detector em AWS IoT Events

As entradas do detector AWS IoT Events servem como ponte entre suas fontes de dados e os modelos de detectores. As entradas do detector fornecem os dados brutos que potencializam os recursos de detecção e automação de eventos do AWS IoT Events. Aprenda a configurar as entradas do detector para ajudar seus modelos a responder com precisão aos eventos e condições do mundo real em seu ecossistema de IoT.

Esta seção mostra como definir uma entrada para um modelo de detector receber dados ou mensagens de telemetria.

Para definir uma entrada para um modelo de detector

1. Abra o [console do AWS IoT Events](#).
2. No AWS IoT Events console, escolha Criar modelo de detector.

3. Selecione Create new (Criar novo).
4. Escolha Create input (Criar entrada).
5. Para a entrada, insira uma InputNameDescrição opcional e escolha Carregar arquivo. Na caixa de diálogo exibida, selecione o `input.json` arquivo que você criou na visão geral para [Crie um arquivo de entrada JSON](#).
6. Em Selecionar atributo de entrada, escolha os atributos a serem usados e selecione Criar. Neste exemplo, selecionamos MotorID e SensorData.pressure.

Crie um modelo de detector em AWS IoT Events

Neste tópico, você define um modelo de detector (um modelo de seu equipamento ou processo) usando estados.

Para cada estado, você define a lógica condicional (booliana) que avalia as entradas recebidas para detectar um evento significativo. Quando um evento é detectado, ele muda o estado e pode iniciar ações adicionais. Esses eventos são conhecidos como eventos de transição.

Em seus estados, você também define eventos que podem executar ações sempre que o detector entra ou sai desse estado ou quando uma entrada é recebida (esses são conhecidos como eventos `OnEnter`, `OnExit`, e `OnInput`). As ações são executadas somente se a lógica condicional do evento for avaliada como `true`.

Para criar um modelo de detector

1. O primeiro estado do detector foi criado para você. Para modificá-lo, selecione o círculo com o rótulo `State_1` no espaço de edição principal.
2. No painel Estado, insira o nome do estado e `OnEnter` escolha Adicionar evento.
3. Na página Adicionar `OnEnter` evento, insira o nome do evento e a condição do evento. Neste exemplo, insira `true` para indicar que o evento sempre é iniciado quando o estado é inserido.
4. Em Ações do evento, escolha Adicionar ação.
5. Em Ações do evento, faça o seguinte:
 - a. Selecione Definir variável
 - b. Para Operação variável, escolha Atribuir valor.
 - c. Para Nome da variável, insira o nome da variável a ser definida.
 - d. Para Valor da variável, insira o valor `0` (zero).

6. Escolha Salvar.

Uma variável, como a que você definiu pode ser definida (com um valor) em qualquer evento no modelo do detector. O valor da variável só pode ser referenciado (por exemplo, na lógica condicional de um evento) depois que o detector atingir um estado e executar uma ação em que esteja definido ou definido.

7. No painel Estado, escolha o X ao lado de Estado para retornar à paleta do Modelo de detector.
8. Para criar um segundo estado do detector, na paleta Modelo do detector, escolha Estado e arraste-o para o espaço de edição principal. Isso cria um estado intitulado `untitled_state_1`.
9. Pausa no primeiro estado (Normal). Uma seta aparece na circunferência do estado.
10. Clique e arraste a seta do primeiro estado para o segundo estado. Uma linha direcionada do primeiro estado para o segundo estado (chamada Sem título) é exibida.
11. Selecione a linha Sem título. No painel Eventos de transição, insira o Nome do evento e a Lógica do gatilho do evento.
12. No painel Evento de transição, escolha Adicionar ação.
13. No painel Adicionar ações do evento de transição, escolha Adicionar ação.
14. Em Escolher uma ação, escolha Definir variável.
 - a. Para Operação variável, escolha Atribuir valor.
 - b. Em Nome da variável, insira o nome da variável.
 - c. Em Atribuir valor, insira o valor, como: `$variable.pressureThresholdBreached + 3`
 - d. Escolha Salvar.
15. Selecione o segundo estado `untitled_state_1`.
16. No painel Estado, insira o Nome do estado e para em Entrar, escolha Adicionar evento.
17. Na página Adicionar OnEnter evento, insira o nome do evento e a condição do evento. Selecione Adicionar ação.
18. Em Escolher uma ação, escolha Enviar mensagem SNS.
 - a. Para Tópico de SNS, insira o ARN alvo do tópico do seu Amazon SNS.
 - b. Escolha Salvar.
19. Continue adicionando os eventos no exemplo.
 - a. Para OnInput, escolha Adicionar evento e insira e salve as seguintes informações do evento.

```

Event name: Overpressurized
Event condition: $input.PressureInput.sensorData.pressure > 70
Event actions:
  Set variable:
    Variable operation: Assign value
    Variable name: pressureThresholdBreached
    Assign value: 3

```

- b. Para OnInput, escolha Adicionar evento e insira e salve as seguintes informações do evento.

```

Event name: Pressure Okay
Event condition: $input.PressureInput.sensorData.pressure <= 70
Event actions:
  Set variable:
    Variable operation: Decrement
    Variable name: pressureThresholdBreached

```

- c. Para OnExit, escolha Adicionar evento e insira e salve as seguintes informações do evento usando o ARN do tópico do Amazon SNS que você criou.

```

Event name: Normal Pressure Restored
Event condition: true
Event actions:
  Send SNS message:
    Target arn: arn:aws:sns:us-east-1:123456789012:pressureClearedAction

```

20. Pausa no segundo estado (Perigoso). Uma seta aparece na circunferência do estado
21. Clique e arraste a seta do segundo estado para o primeiro estado. Uma linha direcionada com o rótulo Sem título é exibida.
22. Escolha a linha Sem título e, no painel Evento de transição, insira o Nome do evento e a Lógica do gatilho do evento usando as informações a seguir.

```

{
  Event name: BackToNormal
  Event trigger logic: $input.PressureInput.sensorData.pressure <= 70 &&
  $variable.pressureThresholdBreached <= 0
}

```

Para obter mais informações sobre por que testamos o valor `$input` e o valor `$variable` na lógica de gatilho, consulte a entrada para disponibilidade de valores de variáveis em [AWS IoT Events restrições e limitações do modelo de detector](#).

23. Selecione o estado Iniciar. Por padrão, esse estado foi criado quando você criou um modelo de detector. No painel Iniciar, escolha o Estado destino (por exemplo, Normal).
24. Em seguida, configure o seu modelo de detector para ouvir as entradas. Escolha Publicar no canto superior direito.
25. Na caixa de diálogo Publicar modelo de detector, faça o seguinte:
 - a. Insira o Nome do modelo de detector, uma Descrição e o nome de uma Função. Esta função foi criada para você.
 - b. Escolha Criar um detector para cada valor de chave exclusivo. Para criar e usar sua própria Função, siga as etapas [Configurando permissões para AWS IoT Events](#) e insira-a como a Função aqui.
26. Em Chave de criação do detector, escolha o nome de um dos atributos da entrada que você definiu anteriormente. O atributo que você escolhe como chave de criação do detector deve estar presente em cada entrada de mensagem e deve ser exclusivo para cada dispositivo que envia mensagens. Este exemplo usa o atributo `motorid`.
27. Escolha Save and publish (Salvar e publicar).

Note

O número de detectores exclusivos criados para um determinado modelo de detector é baseado nas mensagens de entrada enviadas. Quando um modelo de detector é criado, uma chave é selecionada a partir dos atributos de entrada. Essa chave determina qual instância do detector usar. Se a chave não tiver sido vista antes (para este modelo de detector), uma nova instância de detector será criada. Se a chave já foi vista antes, usamos a instância existente do detector correspondente a esse valor de chave.

É possível fazer uma cópia de backup da definição do modelo do detector (em JSON), recriar ou atualizar o modelo do detector ou usá-lo como modelo para criar outro modelo de detector.

Você pode fazer isso a partir do console ou usando o seguinte comando CLI. Se necessário, altere o nome do modelo do detector para corresponder ao que você usou ao publicá-lo na etapa anterior.

```
aws iotevents describe-detector-model --detector-model-name motorDetectorModel >
motorDetectorModel.json
```

Isso cria um arquivo (`motorDetectorModel.json`) com conteúdo semelhante ao seguinte.

```
{
  "detectorModel": {
    "detectorModelConfiguration": {
      "status": "ACTIVE",
      "lastUpdateTime": 1552072424.212,
      "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
      "creationTime": 1552072424.212,
      "detectorModelArn": "arn:aws:iotevents:us-
west-2:123456789012:detectorModel/motorDetectorModel",
      "key": "motorid",
      "detectorModelName": "motorDetectorModel",
      "detectorModelVersion": "1"
    },
    "detectorModelDefinition": {
      "states": [
        {
          "onInput": {
            "transitionEvents": [
              {
                "eventName": "Overpressurized",
                "actions": [
                  {
                    "setVariable": {
                      "variableName":
"pressureThresholdBreached",
                      "value":
"$variable.pressureThresholdBreached + 3"
                    }
                  }
                ],
                "condition": "$input.PressureInput.sensorData.pressure
> 70",
                "nextState": "Dangerous"
              }
            ],
            "events": []
          },
          "stateName": "Normal",

```

```

        "onEnter": {
            "events": [
                {
                    "eventName": "init",
                    "actions": [
                        {
                            "setVariable": {
                                "variableName":
"pressureThresholdBreach",
                                "value": "0"
                            }
                        }
                    ],
                    "condition": "true"
                }
            ],
            "onExit": {
                "events": []
            }
        },
        {
            "onInput": {
                "transitionEvents": [
                    {
                        "eventName": "Back to Normal",
                        "actions": [],
                        "condition": "$variable.pressureThresholdBreach <= 1
&& $input.PressureInput.sensorData.pressure <= 70",
                        "nextState": "Normal"
                    }
                ],
                "events": [
                    {
                        "eventName": "Overpressurized",
                        "actions": [
                            {
                                "setVariable": {
                                    "variableName":
"pressureThresholdBreach",
                                    "value": "3"
                                }
                            }
                        ]
                    }
                ],
            }
        }
    ]
}

```

```

        "condition": "$input.PressureInput.sensorData.pressure
> 70"
        },
        {
            "eventName": "Pressure Okay",
            "actions": [
                {
                    "setVariable": {
                        "variableName":
"pressureThresholdBreached",
                        "value":
"$variable.pressureThresholdBreached - 1"
                    }
                }
            ],
            "condition": "$input.PressureInput.sensorData.pressure
<= 70"
        }
    ]
},
"stateName": "Dangerous",
"onEnter": {
    "events": [
        {
            "eventName": "Pressure Threshold Breached",
            "actions": [
                {
                    "sns": {
                        "targetArn": "arn:aws:sns:us-
west-2:123456789012:MyIoTButtonSNSTopic"
                    }
                }
            ],
            "condition": "$variable.pressureThresholdBreached > 1"
        }
    ]
},
"onExit": {
    "events": [
        {
            "eventName": "Normal Pressure Restored",
            "actions": [
                {
                    "sns": {

```


```
        "targetArn": "arn:aws:sns:us-  
west-2:123456789012:IoTVirtualButtonTopic"  
    }  
    }  
  ],  
  "condition": "true"  
}  
]  
}  
  ],  
  "initialStateName": "Normal"  
}  
}
```

Envie entradas para testar o modelo do detector em AWS IoT Events

Existem várias maneiras de receber dados de telemetria em AWS IoT Events (consulte [Ações suportadas para receber dados e acionar ações em AWS IoT Events](#)). Este tópico mostra como criar uma AWS IoT regra no AWS IoT console que encaminha mensagens como entradas para o AWS IoT Events detector. Você pode usar o cliente MQTT do AWS IoT console para enviar mensagens de teste. Você pode usar esse método para obter dados de telemetria AWS IoT Events quando seus dispositivos são capazes de enviar mensagens MQTT usando o AWS IoT agente de mensagens.

Para enviar entradas para testar o modelo do detector

1. Abra o [console do AWS IoT Core](#). No painel de navegação esquerdo, em Gerenciar, escolha Roteamento de mensagens e, em seguida, escolha Regras.
2. Escolha Criar regra no canto superior direito.
3. Na página Criar regra, conclua as seguintes etapas:
 1. Etapa 1. Especifique as propriedades da regra. Preencha os seguintes campos:
 - Nome da regra. Insira um nome para a regra, como MyIoTEventsRule.

 Note

Não use espaços.

- Descrição da regra. Isso é opcional.
- Escolha Próximo.

2. Etapa 2. Configure a declaração SQL. Preencha os seguintes campos:

- Versão do SQL. Selecione a opção apropriada na lista.
- Declaração do SQL. Insira **SELECT *, topic(2) as motorid FROM 'motors/+/
status'**.

Escolha Próximo.

3. Etapa 3. Anexe as ações de regras. Na seção Ações, preencha o seguinte:

- Ação 1. Selecione IoT Events. Os campos a seguir são mostrados:
 - a. Nome da entrada. Selecione a opção apropriada na lista. Se a entrada não aparecer, escolha Atualizar.

Para criar uma nova entrada, escolha Criar entrada IoT Events. Preencha os seguintes campos:

- Nome da entrada. Insira `PressureInput`.
- Descrição. Isso é opcional.
- Faça upload de um arquivo JSON. Faça upload de uma cópia do seu arquivo JSON. Há um link para um arquivo de amostra nessa tela, caso você não tenha um arquivo. O código inclui:

```
{
  "motorid": "Fulton-A32",
  "sensorData": {
    "pressure": 23,
    "temperature": 47
  }
}
```

- Selecionar atributo de entrada. Selecione a(s) opção(ões) apropriada(s).
- Tags. Isso é opcional.

Escolha Criar.

Retorne à tela Criar regra e atualize o campo Nome de entrada. Selecione a entrada que você acabou de criar.

- b. Modo Batch. Isso é opcional. Se a carga for uma matriz de mensagens, selecione essa opção.
- c. ID da mensagem. Isso é opcional, mas recomendado.
- d. IAM role (Perfil do IAM. Selecione o perfil apropriado na lista. Se a função não estiver listada, escolha Criar nova função.

Digite um nome de função e escolha Criar.

Para adicionar outra regra, escolha Adicionar regra de ação

- Ação de erro. Esta seção é opcional. Para adicionar uma ação, escolha Adicionar ação de erro e selecione a ação apropriada na lista.

Preencha os campos que aparecem.

- Escolha Próximo.


4. Etapa 4. Revisar e criar. Revise as informações e escolha Criar função.

4. No painel de navegação esquerdo, em Teste, escolha o Cliente de teste do MQTT.

5. Selecione Publish to a topic (Publicar em um tópico). Preencha os seguintes campos:

- Nome do tópico . Insira um nome para identificar a mensagem, como `motors/Fulton-A32/status`.
- Carga útil de mensagem. Insira o seguinte:

```
{
  "messageId": 100,
  "sensorData": {
    "pressure": 39
  }
}
```

 Note

Altere a `messageId` cada vez que você publicar uma nova mensagem.

6. Para Publicar, mantenha o tópico o mesmo, mas altere o "pressure" na carga para um valor maior que o valor limite que você especificou no modelo do detector (como **85**).
7. Selecione Publish.

A instância do detector que você criou gera e envia uma mensagem do Amazon SNS para você. Continue enviando mensagens com leituras de pressão acima ou abaixo do limite de pressão (70 neste exemplo) para ver o detector em operação.

Neste exemplo, você deve enviar três mensagens com leituras de pressão abaixo do limite para voltar ao estado Normal e receber uma mensagem do Amazon SNS indicando que a condição de sobrepressão foi eliminada. Depois da volta ao estado Normal, uma mensagem com uma leitura de pressão acima do limite faz com que o detector entre no estado Perigoso e envie uma mensagem do Amazon SNS indicando essa condição.

Agora que você criou um modelo simples de entrada e de detector, tente o seguinte.

- Veja mais exemplos de modelos de detectores (modelos) no console.
- Siga as etapas [Crie um AWS IoT Events detector para dois estados usando CLI](#) para criar um modelo de entrada e detector usando o AWS CLI
- Conheça os detalhes do [Expressões para filtrar, transformar e processar dados de eventos](#) usado em eventos.
- Saiba mais sobre o [Ações suportadas para receber dados e acionar ações em AWS IoT Events](#).
- Se algo não estiver funcionando, consulte [Solução de problemas AWS IoT Events](#).

Melhores práticas para AWS IoT Events

Siga estas melhores práticas para obter o benefício máximo do AWS IoT Events.

Tópicos

- [Ative o CloudWatch registro na Amazon ao desenvolver modelos AWS IoT Events de detectores](#)
- [Publique regularmente para salvar seu modelo de detector ao trabalhar no AWS IoT Events console](#)

Ative o CloudWatch registro na Amazon ao desenvolver modelos AWS IoT Events de detectores

A Amazon CloudWatch monitora seus AWS recursos e os aplicativos nos quais você executa AWS em tempo real. Com isso CloudWatch, você obtém visibilidade de todo o sistema sobre o uso de recursos, desempenho de aplicativos e integridade operacional. Quando você desenvolve ou depura um modelo de AWS IoT Events detector, CloudWatch ajuda você a saber o que AWS IoT Events está fazendo e os erros que ele encontra.

Para habilitar CloudWatch

1. Se ainda não o fez, siga as etapas [Configurando permissões para AWS IoT Events](#) para criar uma função com uma política anexada que conceda permissão para criar e gerenciar CloudWatch registros para AWS IoT Events.
2. Acesse o console do [AWS IoT Events](#).
3. No painel de navegação, selecione Configurações.
4. Na página Configurações, selecione Editar.
5. Na página Editar opções de registro, na seção Opções de registro, faça o seguinte:
 - a. Em Nível de verbosidade, selecione uma opção.
 - b. Em Selecionar função, selecione uma função com permissões suficientes para realizar as ações de registro em log escolhida.
 - c. (Opcional) Se você escolher Depurar para o nível de detalhamento, poderá adicionar destinos de depuração fazendo o seguinte:
 - i. Em Depurar destinos, escolha Adicionar opção de modelo.

- ii. Insira o nome do modelo do detector e (opcional) especifique KeyValues modelos de detectores e detectores específicos (instâncias) a serem registrados.

6. Selecione Atualizar.

Suas opções de registro em log foram atualizadas com sucesso.

Publique regularmente para salvar seu modelo de detector ao trabalhar no AWS IoT Events console

Quando você usa o AWS IoT Events console, seu trabalho em andamento é salvo localmente no seu navegador. No entanto, você deve escolher Publicar para salvar seu modelo de detector do AWS IoT Events. Depois de publicar um modelo de detector, seu trabalho publicado ficará disponível em qualquer navegador usado para acessar sua conta.

Note

Se você não publicar seu trabalho, ele não será salvo. Depois de publicar um modelo de detector, não é possível alterar o nome. No entanto, você pode continuar modificando a sua definição.

Tutoriais para casos de uso AWS IoT Events

AWS IoT Events os tutoriais fornecem uma coleção de procedimentos que abrangem vários aspectos AWS IoT Events, desde a configuração básica até casos de uso mais específicos. Cada tutorial mostra exemplos de cenários práticos, ajudando você a desenvolver habilidades reais na criação de modelos de detectores, configuração de entradas, configuração de ações e integração com outros serviços AWS para criar soluções de IoT poderosas.

Este capítulo destaca como:

- Obter ajuda para decidir quais estados incluir em seu modelo de detector e determinar se você precisa de uma instância de detector ou de várias instâncias.
- Siga um exemplo que usa a AWS CLI.
- Crie uma entrada para receber dados de telemetria de um dispositivo e um modelo de detector para monitorar e relatar o estado do dispositivo que envia esses dados.
- Analise as restrições e limitações nas entradas, nos modelos de detectores e no serviço do AWS IoT Events .
- Veja um exemplo mais complexo de um modelo de detector, com comentários incluídos.

Tópicos

- [Usando AWS IoT Events para monitorar seus dispositivos de IoT](#)
- [Crie um AWS IoT Events detector para dois estados usando CLI](#)
- [AWS IoT Events restrições e limitações do modelo de detector](#)
- [Um exemplo comentado: controle de temperatura HVAC com AWS IoT Events](#)

Usando AWS IoT Events para monitorar seus dispositivos de IoT

Você pode usar AWS IoT Events para monitorar seus dispositivos ou processos e agir com base em eventos significativos. Para fazer isso, siga estas etapas básicas:

Criar entradas

Para monitorar seus dispositivos e processos para inserir dados de telemetria no AWS IoT Events. Isso é feito enviando mensagens como entradas para o AWS IoT Events. Você pode enviar mensagens como entradas de várias maneiras:

- Use a [BatchPutMessage](#) operação.
- Defina uma [ação de regra do IoTEvents](#) para o [mecanismo de regras do AWS IoT Core](#). A ação de regra encaminha os dados da mensagem de sua entrada para o AWS IoT Events.
- Em AWS IoT Analytics, use a [CreateDataset](#) operação para criar um conjunto de dados com `contentDeliveryRules`. Essas regras especificam a AWS IoT Events entrada para a qual o conteúdo do conjunto de dados é enviado automaticamente.
- Defina uma [ação do IoTEvents](#) em um modelo ou evento de AWS IoT Events `onInput` detector. `onExit transitionEvents` As informações sobre a instância do modelo de detector e o evento que iniciou a ação são devolvidas ao sistema como uma entrada com o nome que você especificar.

Antes que seus dispositivos comecem a enviar dados dessa forma, é preciso definir uma ou mais entradas. Para fazer isso, dê um nome a cada entrada e especifique quais campos nos dados da mensagem recebida a entrada monitora. AWS IoT Events recebe sua entrada, na forma de carga útil JSON, de várias fontes. Cada entrada pode ser acionada sozinha ou em combinação com outras entradas para detectar eventos mais complexos.

Como criar um modelo de detector

Defina um modelo de detector (um modelo de seu equipamento ou processo) usando estados. Para cada estado, você define a lógica condicional (booleana) que avalia as entradas recebidas para detectar eventos significativos. Quando um evento é detectado, ele pode alterar o estado ou iniciar ações personalizadas ou predefinidas usando outros serviços. AWS É possível definir eventos adicionais que acionam ações ao entrar ou sair de um estado e, opcionalmente, quando uma condição é atendida.

Neste tutorial, você envia uma mensagem do Amazon SNS como a ação quando o modelo entra ou sai de um determinado estado.

Como monitorar um dispositivo ou processo

Se você estiver monitorando vários dispositivos ou processos, especifique um campo em cada entrada que identifica o dispositivo ou processo específico do qual a entrada vem. (Veja o campo `key` no `CreateDetectorModel`.) Quando um novo dispositivo é identificado (um novo valor é visto no campo de entrada identificado pelo `key`), um detector é criado. (Cada detector é uma instância de um modelo de detector). Em seguida, o novo detector continua respondendo às entradas provenientes desse dispositivo até que o modelo do detector seja atualizado ou excluído.

Se você estiver monitorando um único processo (mesmo que vários dispositivos ou subprocessos estejam enviando entradas), você não especifica um campo de identificação key exclusivo. Nesse caso, um único detector (instância) é criado quando a primeira entrada chega.

Como enviar mensagens como entradas para seu modelo de detector

Há várias maneiras de enviar uma mensagem de um dispositivo ou processo como entrada em um AWS IoT Events detector que não exigem que você execute formatação adicional na mensagem. Neste tutorial, você usa o AWS IoT console para escrever uma regra de [AWS IoT Events ação](#) para o mecanismo de AWS IoT Core regras que encaminha os dados da sua mensagem para o. AWS IoT Events Para fazer isso, você identifica a entrada pelo nome. Em seguida, você continua usando o AWS IoT console para gerar algumas mensagens que são encaminhadas como entradas para. AWS IoT Events

Como descobrir quais estados são necessários em um modelo de detector?

Para determinar quais estados seu modelo de detector deve ter, primeiro decida quais ações devem ser tomadas. Por exemplo, se seu automóvel funciona com gasolina, você olha o medidor de combustível ao iniciar uma viagem para ver se precisa reabastecer. Aqui você tem uma ação: diga ao motorista que “vá buscar gasolina”. Seu modelo de detector precisa de dois estados: “o carro não precisa de combustível” e “o carro precisa de combustível”. Em geral, a intenção é definir um estado para cada ação possível, além de um extra para quando nenhuma ação for necessária. Isso funciona mesmo que a ação em si seja mais complicada. Por exemplo, talvez você queira pesquisar e incluir informações sobre onde encontrar o posto de gasolina mais próximo ou o preço mais barato, mas faça isso ao enviar a mensagem “vá buscar gasolina”.

Para decidir em qual estado entrar em seguida, as entradas são analisadas. As entradas contêm as informações necessárias para decidir em que estado você deve estar. Para criar uma entrada, selecione um ou mais campos em uma mensagem enviada pelo seu dispositivo ou processo para ajudar na decisão. Neste exemplo, é preciso uma entrada que informe o nível atual de combustível (“porcentagem cheia”). Talvez seu carro esteja enviando várias mensagens diferentes, cada uma com vários campos diferentes. Para criar essa entrada, é preciso selecionar a mensagem e o campo que informa o nível atual do medidor de gasolina. A duração da viagem que está prestes a fazer (“distância até o destino”) pode ser codificada para simplificar a situação; a duração média da viagem pode ser usada. Você fará alguns cálculos com base na entrada (em quantos litros essa porcentagem total se traduz? A duração média da viagem é maior do que os quilômetros que você pode viajar, considerando os litros disponíveis e sua média de “quilômetros por litro”?). Realize esses cálculos e envie mensagens em eventos.

Até agora, você tem dois estados e uma entrada. É preciso um evento no primeiro estado que realize os cálculos com base na entrada e decida se deseja ir para o segundo estado. Esse é um evento de transição. (`transitionEvents` estão na lista de eventos `onInput` de um estado. Ao receber uma entrada nesse primeiro estado, o evento executa uma transição para o segundo estado, se a `condition` do evento for atendida.) Ao chegar ao segundo estado, você envia a mensagem assim que entra no estado. (Um evento `onEnter` é usado. Ao entrar no segundo estado, esse evento envia a mensagem. Não é necessário esperar a chegada de outra entrada). Existem outros tipos de eventos, mas isso é tudo o que precisa para um exemplo simples.

Os outros tipos de eventos são `onExit` e `onInput`. Assim que uma entrada é recebida e a condição é atendida, um evento `onInput` executa as ações especificadas. Quando uma operação sai de seu estado atual e a condição é atendida, o evento `onExit` executa as ações especificadas.

Perdeu alguma coisa? Sim, como voltar ao primeiro estado de “carro não precisa de combustível”? Depois de encher o tanque de gasolina, a entrada mostra um tanque cheio. Em seu segundo estado, é preciso um evento de transição de volta ao primeiro estado que acontece quando a entrada é recebida (nos eventos `onInput`: do segundo estado). Ele deve voltar ao primeiro estado se seus cálculos mostrarem que agora você tem gasolina suficiente para chegar aonde deseja.

Isso é o básico. Alguns modelos de detectores ficam mais complexos ao adicionar estados que refletem entradas importantes, não apenas ações possíveis. Por exemplo, é possível ter três estados em um modelo de detector que monitora a temperatura: um estado “normal”, um estado “muito quente” e um estado de “problema potencial”. Você faz a transição para o estado de problema potencial quando a temperatura sobe acima de um determinado nível, mas ainda não está muito quente. Não se deve enviar um alarme a menos que essa temperatura continue por mais de 15 minutos. Se a temperatura voltar ao normal antes disso, o detector voltará ao estado normal. Se o temporizador expirar, o detector passará para o estado muito quente e enviará um alarme, só por precaução. É possível fazer a mesma coisa usando variáveis e um conjunto mais complexo de condições de eventos. Mas, muitas vezes, é mais fácil usar outro estado para armazenar os resultados de seus cálculos.

Como saber se você precisa de uma ou várias instâncias de um detector?

Para decidir quantas instâncias são necessárias, pergunte “O que tenho interesse em saber?”. Digamos que quer saber como está o clima hoje. Está chovendo (estado)? Precisa pegar um guarda-chuva (ação)? É possível ter um sensor que informa a temperatura, outro que informa a umidade e outros que relatam a pressão barométrica, a velocidade e direção do vento e a precipitação. Mas é necessário monitorar todos esses sensores juntos para determinar o estado do clima (chuva, neve,

céu nublado, sol) e a ação apropriada a ser tomada (pegar um guarda-chuva ou aplicar protetor solar). Apesar do número de sensores, você quer que uma instância de detector monitore o estado do clima e informe qual ação tomar.

Mas caso seja o meteorologista da sua região, poderá ter várias instâncias dessas matrizes de sensores, localizadas em locais diferentes da região. As pessoas em cada local precisam saber como está o clima em seu próprio local. Nesse caso, várias instâncias do detector serão necessárias. Os dados relatados por cada sensor em cada local devem incluir um campo designado como campo key. Esse campo permite ao AWS IoT Events criar uma instância do detector para a área e, em seguida, continuar a rotear tais informações para essa instância do detector à medida que elas continuam chegando. Chega de ventos destruindo penteados ou narizes queimados pelo sol!

Basicamente, você precisa de uma instância de detector se tiver uma situação (um processo ou um local) para monitorar. Caso tenha muitas situações (locais, processos) para monitorar, precisará de várias instâncias de detector.

Crie um AWS IoT Events detector para dois estados usando CLI

Neste exemplo, chamamos os AWS CLI comandos de AWS IoT Events APIs uso para criar um detector que modela dois estados de um motor: um estado normal e uma condição de sobrepressão.

Quando a pressão medida no motor excede um determinado limite, o modelo passa para o estado de sobrepressão e envia uma mensagem do Amazon Simple Notification Service (Amazon SNS) para alertar um técnico sobre a condição. Quando a pressão cai abaixo do limite por três leituras de pressão consecutivas, o modelo retorna ao estado normal e envia outra mensagem do Amazon SNS como confirmação de que a condição foi eliminada. Exigimos três leituras consecutivas abaixo do limite de pressão para eliminar possíveis interrupções de sobrepressão/mensagens normais no caso de uma fase de recuperação não linear ou de uma leitura de recuperação anômala única.

Confira abaixo uma visão geral das etapas para criar o detector.

Criar entradas.

Para monitorar seus dispositivos e processos, eles devem ter uma maneira de inserir dados de telemetria no AWS IoT Events. Isso é feito enviando mensagens como entradas para AWS IoT Events. Você pode fazer isso de várias maneiras:

- Use a [BatchPutMessage](#) operação. Esse método é fácil, mas exige que seus dispositivos ou processos possam acessar a AWS IoT Events API por meio de um SDK ou do AWS CLI.

- Em AWS IoT Core, escreva uma regra de [AWS IoT Events ação](#) para o mecanismo de AWS IoT Core regras que encaminha os dados da sua mensagem para o. AWS IoT Events Isso identifica a entrada pelo nome. Use esse método se seus dispositivos ou processos puderem, ou já estiverem, enviando mensagens por meio de AWS IoT Core. Geralmente, esse método requer menos poder de computação de um dispositivo.
- Em AWS IoT Analytics, use a [CreateDataset](#) operação para criar um conjunto de dados com `contentDeliveryRules` essa especificação da AWS IoT Events entrada, para a qual o conteúdo do conjunto de dados é enviado automaticamente. Use este método se quiser controlar seus dispositivos ou processos com base nos dados agregados ou analisados no AWS IoT Analytics.

Antes que seus dispositivos possam enviar dados dessa forma, é preciso definir uma ou mais entradas. Para fazer isso, dê um nome a cada entrada e especifique quais campos nos dados da mensagem recebida a entrada monitora.

Como criar um modelo de detector

Crie um modelo de detector (um modelo de seu equipamento ou processo) usando estados. Para cada estado, você define a lógica condicional (booliana) que avalia as entradas recebidas para detectar eventos significativos. Quando um evento é detectado, ele pode alterar o estado ou iniciar ações personalizadas ou predefinidas usando outros serviços. AWS É possível definir eventos adicionais que acionam ações ao entrar ou sair de um estado e, opcionalmente, quando uma condição é atendida.

Monitorar vários dispositivos ou processos

Caso esteja monitorando vários dispositivos ou processos e queira acompanhar cada um deles separadamente, especifique um campo em cada entrada que identifique o dispositivo ou processo específico do qual a entrada vem. Veja o campo `key` em `CreateDetectorModel`. Quando um novo dispositivo é identificado (um novo valor é visto no campo de entrada identificado pelo `key`), uma instância do detector é criada. A nova instância do detector continua respondendo às entradas provenientes desse dispositivo específico até que seu modelo de detector seja atualizado ou excluído. Você tem tantos detectores (instâncias) exclusivos quanto valores exclusivos nos campos de entrada `key`.

Monitorar um único dispositivo ou processo

Se você estiver monitorando um único processo (mesmo que vários dispositivos ou subprocessos estejam enviando entradas), você não especifica um campo de identificação `key` exclusivo. Nesse caso, um único detector (instância) é criado quando a primeira entrada chega. Por

exemplo, é possível ter sensores de temperatura em cada cômodo de uma casa, mas apenas uma unidade HVAC para aquecer ou resfriar a casa inteira. Portanto, só é possível controlar isso como um único processo, mesmo que cada ocupante da sala queira que seu voto (entrada) prevaleça.

Enviar mensagens de seus dispositivos ou processos como entradas para seu modelo de detector

Descrevemos as várias maneiras de enviar uma mensagem de um dispositivo ou processo como entrada para um AWS IoT Events detector em entradas. Depois de criar as entradas e criar o modelo de detector, tudo estará pronto para começar a enviar os dados.

Note

Quando cria um modelo de detector ou atualiza um existente, leva vários minutos até que o modelo de detector novo ou atualizado comece a receber mensagens e a criar detectores (instâncias). Se o modelo do detector for atualizado, durante esse período é possível continuar observando o comportamento com base na versão anterior.

Tópicos

- [Crie uma AWS IoT Events entrada para capturar dados do dispositivo](#)
- [Crie um modelo de detector para representar os estados do dispositivo no AWS IoT Events](#)
- [Envie mensagens como entradas para um detector em AWS IoT Events](#)

Crie uma AWS IoT Events entrada para capturar dados do dispositivo

Ao configurar as entradas para AWS IoT Events, você pode aproveitar o AWS CLI para definir como seus dispositivos comunicam os dados do sensor. Por exemplo, se seus dispositivos enviarem mensagens no formato JSON com identificadores de motor e leituras de sensores, você poderá capturar esses dados criando uma entrada que mapeie atributos específicos das mensagens, como a pressão e a ID do motor. O processo começa definindo uma entrada em um arquivo JSON, especificando os pontos de dados relevantes e usando o AWS CLI para registrar a entrada. AWS IoT Events Isso permite AWS IoT monitorar e responder às condições críticas com base nos dados do sensor em tempo real.

Por exemplo, suponha que seus dispositivos enviem mensagens com o formato a seguir.

```
{
```

```
"motorid": "Fulton-A32",
"sensorData": {
  "pressure": 23,
  "temperature": 47
}
}
```

Você pode criar uma entrada para capturar os dados de pressão e o `motorid` (que identifica o dispositivo específico que enviou a mensagem) usando o AWS CLI comando a seguir.

```
aws iotevents create-input --cli-input-json file://pressureInput.json
```

O arquivo `pressureInput.json` contém o seguinte.

```
{
  "inputName": "PressureInput",
  "inputDescription": "Pressure readings from a motor",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "sensorData.pressure" },
      { "jsonPath": "motorid" }
    ]
  }
}
```

Ao criar suas próprias entradas, lembre-se primeiro de coletar mensagens de exemplo como arquivos JSON de seus dispositivos ou processos. Você pode usá-los para criar uma entrada do console ou da CLI.

Crie um modelo de detector para representar os estados do dispositivo no AWS IoT Events

Em [Crie uma AWS IoT Events entrada para capturar dados do dispositivo](#), você criou um `input` com base em uma mensagem que relata dados de pressão de um motor. Para continuar com o exemplo, confira aqui um modelo de detector que responde a um evento de sobrepressão em um motor.

Você cria dois estados: “Normal” e “Dangerous”. Cada detector (instância) entra no estado “Normal” quando é criado. A instância é criada quando uma entrada chega com um valor exclusivo para o “`motorid`” do `key`.

Se a instância do detector receber uma leitura de pressão de 70 ou mais, ela entrará no estado “Dangerous” e enviará uma mensagem do Amazon SNS como aviso. Se as leituras de pressão voltarem ao normal (menos de 70) por três entradas consecutivas, o detector retornará ao estado “Normal” e enviará outra mensagem do Amazon SNS indicando que tudo está correto.

Este exemplo de modelo de detector pressupõe que você tenha criado dois tópicos do Amazon SNS cujos nomes de recursos da Amazon ARNs () são mostrados na definição "targetArn": "arn:aws:sns:us-east-1:123456789012:underPressureAction" como e "targetArn": "arn:aws:sns:us-east-1:123456789012:pressureClearedAction"

Para obter mais informações, consulte o [Guia do desenvolvedor do Amazon Simple Notification Service](#) e, mais especificamente, a documentação da [CreateTopic](#) operação na Referência da API do Amazon Simple Notification Service.

Este exemplo também pressupõe que você tenha criado uma função AWS Identity and Access Management (IAM) com as permissões apropriadas. O ARN dessa função é exibido na definição do modelo do detector como "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole". Siga as etapas em [Configurando permissões para AWS IoT Events](#) para criar essa função e copiar o ARN da função no local apropriado na definição do modelo do detector.

Você pode criar o modelo do detector usando o AWS CLI comando a seguir.

```
aws iotevents create-detector-model --cli-input-json file://motorDetectorModel.json
```

Veja a seguir o conteúdo do arquivo "motorDetectorModel.json".

```
{
  "detectorModelName": "motorDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "Normal",
        "onEnter": {
          "events": [
            {
              "eventName": "init",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
```

```

        "variableName": "pressureThresholdBreached",
        "value": "0"
      }
    ]
  },
  "onInput": {
    "transitionEvents": [
      {
        "eventName": "Overpressurized",
        "condition": "$input.PressureInput.sensorData.pressure > 70",
        "actions": [
          {
            "setVariable": {
              "variableName": "pressureThresholdBreached",
              "value": "$variable.pressureThresholdBreached + 3"
            }
          }
        ],
        "nextState": "Dangerous"
      }
    ]
  }
},
{
  "stateName": "Dangerous",
  "onEnter": {
    "events": [
      {
        "eventName": "Pressure Threshold Breached",
        "condition": "$variable.pressureThresholdBreached > 1",
        "actions": [
          {
            "sns": {
              "targetArn": "arn:aws:sns:us-
east-1:123456789012:underPressureAction"
            }
          }
        ]
      }
    ]
  }
},

```

```
"onInput": {
  "events": [
    {
      "eventName": "Overpressurized",
      "condition": "$input.PressureInput.sensorData.pressure > 70",
      "actions": [
        {
          "setVariable": {
            "variableName": "pressureThresholdBreached",
            "value": "3"
          }
        }
      ]
    },
    {
      "eventName": "Pressure Okay",
      "condition": "$input.PressureInput.sensorData.pressure <= 70",
      "actions": [
        {
          "setVariable": {
            "variableName": "pressureThresholdBreached",
            "value": "$variable.pressureThresholdBreached - 1"
          }
        }
      ]
    }
  ],
  "transitionEvents": [
    {
      "eventName": "BackToNormal",
      "condition": "$input.PressureInput.sensorData.pressure <= 70 &&
$variable.pressureThresholdBreached <= 1",
      "nextState": "Normal"
    }
  ]
},
"onExit": {
  "events": [
    {
      "eventName": "Normal Pressure Restored",
      "condition": "true",
      "actions": [
        {
          "sns": {
```

```
        "targetArn": "arn:aws:sns:us-
east-1:123456789012:pressureClearedAction"
      }
    }
  ]
}
],
"initialStateName": "Normal"
},
"key" : "motorid",
"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole"
}
```

Envie mensagens como entradas para um detector em AWS IoT Events

Agora você definiu uma entrada que identifica os campos importantes nas mensagens enviadas de um dispositivo (consulte [Crie uma AWS IoT Events entrada para capturar dados do dispositivo](#)). Na seção anterior, você criou um detector `model` que responde a um evento de sobrepessão em um motor (consulte [Crie um modelo de detector para representar os estados do dispositivo no AWS IoT Events](#)).

Para concluir o exemplo, envie mensagens de um dispositivo (neste caso, um computador com a AWS CLI instalada) como entradas para o detector.

Note

Quando cria um modelo de detector ou atualiza um existente, leva vários minutos até que o modelo de detector novo ou atualizado comece a receber mensagens e criar detectores (instâncias). Se atualizar o modelo do detector, durante esse período poderá continuar vendo o comportamento com base na versão anterior.

Use o AWS CLI comando a seguir para enviar uma mensagem com dados que ultrapassam o limite.

```
aws iotevents-data batch-put-message --cli-input-json file://highPressureMessage.json
--cli-binary-format raw-in-base64-out
```

O arquivo "highPressureMessage.json" contém o código a seguir.

```
{
  "messages": [
    {
      "messageId": "00001",
      "inputName": "PressureInput",
      "payload": "{\"motorid\": \"Fulton-A32\", \"sensorData\": {\"pressure\": 80,
        \"temperature\": 39} }"
    }
  ]
}
```

Você deve alterar o messageId em cada mensagem enviada. Se você não alterar, o AWS IoT Events sistema desduplica as mensagens. AWS IoT Events ignora uma mensagem se ela tiver a messageID mesma mensagem que foi enviada nos últimos cinco minutos.

Nesse ponto, um detector (instância) é criado para monitorar os eventos do motor "Fulton-A32". Esse detector entra no estado "Normal" quando é criado. Mas como enviamos um valor de pressão acima do limite, ele imediatamente passa para o estado "Dangerous". Ao fazer isso, o detector envia uma mensagem para o endpoint do Amazon SNS cujo ARN é arn:aws:sns:us-east-1:123456789012:underPressureAction.

Execute o AWS CLI comando a seguir para enviar uma mensagem com dados abaixo do limite de pressão.

```
aws iotevents-data batch-put-message --cli-input-json file://normalPressureMessage.json
--cli-binary-format raw-in-base64-out
```

O arquivo normalPressureMessage.json contém o seguinte.

```
{
  "messages": [
    {
      "messageId": "00002",
      "inputName": "PressureInput",
      "payload": "{\"motorid\": \"Fulton-A32\", \"sensorData\": {\"pressure\": 60,
        \"temperature\": 29} }"
    }
  ]
}
```

```
]
}
```

Você deve alterar o `messageId` no arquivo sempre que invocar o comando `BatchPutMessage` em um período de cinco minutos. Envie a mensagem mais duas vezes. Depois que a mensagem é enviada três vezes, o detector (instância) do motor "Fulton-A32" envia uma mensagem para o endpoint do Amazon SNS "`arn:aws:sns:us-east-1:123456789012:pressureClearedAction`" e entra novamente no estado "Normal".

Note

É possível enviar várias mensagens ao mesmo tempo com o `BatchPutMessage`. No entanto, a ordem em que essas mensagens são processadas não é garantida. Para garantir que as mensagens (entradas) sejam processadas em ordem, envie-as uma de cada vez e aguarde uma resposta bem-sucedida sempre que a API for chamada.

Confira a seguir exemplos de cargas úteis de mensagens do SNS criadas pelo exemplo do modelo de detector descrito nesta seção.

no evento "Limite de pressão violado"

```
IoT> {
  "eventTime":1558129816420,
  "payload":{
    "actionExecutionId":"5d7444df-a655-3587-a609-dbd7a0f55267",
    "detector":{
      "detectorModelName":"motorDetectorModel",
      "keyValue":"Fulton-A32",
      "detectorModelVersion":"1"
    },
    "eventTriggerDetails":{
      "inputName":"PressureInput",
      "messageId":"00001",
      "triggerType":"Message"
    },
    "state":{
      "stateName":"Dangerous",
      "variables":{
        "pressureThresholdBreach":3
      }
    }
  }
}
```

```

    "timers":{}
  }
},
"eventName":"Pressure Threshold Breached"
}

```

no evento “Pressão normal restaurada”

```

IoT> {
  "eventTime":1558129925568,
  "payload":{
    "actionExecutionId":"7e25fd38-2533-303d-899f-c979792a12cb",
    "detector":{
      "detectorModelName":"motorDetectorModel",
      "keyValue":"Fulton-A32",
      "detectorModelVersion":"1"
    },
    "eventTriggerDetails":{
      "inputName":"PressureInput",
      "messageId":"00004",
      "triggerType":"Message"
    },
    "state":{
      "stateName":"Dangerous",
      "variables":{
        "pressureThresholdBreached":0
      },
      "timers":{}
    }
  },
  "eventName":"Normal Pressure Restored"
}

```

Se você definiu um temporizador, o estado atual dele também será mostrado nas cargas úteis de mensagens do SNS.

As cargas úteis de mensagens contêm informações sobre o estado do detector (instância) no momento em que a mensagem foi enviada (ou seja, no momento em que a ação do SNS foi executada). É possível usar a operação https://docs.aws.amazon.com/iotevents/latest/apireference/API_iotevents-data_DescribeDetector.html para obter informações semelhantes sobre o estado do detector.

AWS IoT Events restrições e limitações do modelo de detector

É importante considerar o seguinte ao criar um modelo de detector.

Como usar o campo **actions**

O campo `actions` é uma lista de objetos. Você pode ter mais de um objeto, mas somente uma ação é permitida em cada objeto.

Example

```
"actions": [  
  {  
    "setVariable": {  
      "variableName": "pressureThresholdBreached",  
      "value": "$variable.pressureThresholdBreached - 1"  
    }  
  }  
  {  
    "setVariable": {  
      "variableName": "temperatureIsTooHigh",  
      "value": "$variable.temperatureIsTooHigh - 1"  
    }  
  }  
]
```

Como usar o campo **condition**

O `condition` é obrigatório para o `transitionEvents` e opcional em outros casos.

Se o campo `condition` não estiver presente, ele é equivalente ao `"condition": true`.

O resultado da avaliação de uma expressão de condição deverá ser um valor booleano. Se o resultado não for um valor booleano, ele é equivalente ao `false` e não iniciará o `actions` ou fará a transição para o `nextState` especificado no evento.

Disponibilidade de valores variáveis

Por padrão, se o valor de uma variável for definido em um evento, seu novo valor não estará disponível nem será usado para avaliar condições em outros eventos no mesmo grupo. O novo

valor não está disponível nem é usado em uma condição de evento no mesmo campo `onInput`, `onEnter` ou `onExit`.

Defina o parâmetro `evaluationMethod` na definição do modelo do detector para alterar esse comportamento. Quando o `evaluationMethod` é definido como `SERIAL`, as variáveis são atualizadas e as condições do evento são avaliadas na ordem em que os eventos são definidos. Caso contrário, quando o `evaluationMethod` é definido como `BATCH` ou padronizado como ele, as variáveis dentro de um estado serão atualizadas e os eventos dentro de um estado serão executados somente depois que todas as condições do evento forem avaliadas.

No estado "Dangerous", no campo `onInput`, `"$variable.pressureThresholdBreach"` é diminuído em um no evento "Pressure Okay" quando a condição é atendida (quando a entrada de corrente tem pressão menor ou igual a 70).

```
{
  "eventName": "Pressure Okay",
  "condition": "$input.PressureInput.sensorData.pressure <= 70",
  "actions": [
    {
      "setVariable": {
        "variableName": "pressureThresholdBreach",
        "value": "$variable.pressureThresholdBreach - 1"
      }
    }
  ]
}
```

O detector deve voltar ao estado "Normal" quando `"$variable.pressureThresholdBreach"` atingir 0 (ou seja, quando o detector tiver recebido 3 leituras de pressão contíguas menores ou iguais a 70). O evento "BackToNormal" em `transitionEvents` deve testar se `"$variable.pressureThresholdBreach"` é menor ou igual a 1 (não 0) e também verificar novamente se o valor atual fornecido por `"$input.PressureInput.sensorData.pressure"` é menor ou igual a 70.

```
"transitionEvents": [
  {
    "eventName": "BackToNormal",
```

```
        "condition": "$input.PressureInput.sensorData.pressure <= 70 &&
$variable.pressureThresholdBreach <= 1",
        "nextState": "Normal"
    }
]
```

Caso contrário, se a condição testar apenas o valor da variável, duas leituras normais seguidas por uma leitura de sobrepressão atenderiam à condição e retornariam ao estado "Normal". A condição é analisar o valor fornecido na última vez em que "\$variable.pressureThresholdBreach" foi dado durante o tempo anterior que uma entrada foi processada. O valor da variável é redefinido para três no evento "Overpressurized", mas não esqueça que esse novo valor ainda não está disponível para nenhuma `condition`.

Por padrão, toda vez que um controle entra no campo `onInput`, uma `condition` só pode ver o valor de uma variável como ela estava no início do processamento da entrada, antes de ser alterada por qualquer ação especificada no `onInput`. O mesmo se aplica para `onEnter` e `onExit`. Qualquer alteração feita em uma variável quando entramos ou saímos do estado não está disponível para outras condições especificadas no mesmo nos campos `onEnter` ou `onExit`.

Latência ao atualizar um modelo de detector

Se você atualizar, excluir e recriar um modelo de detector (consulte [UpdateDetectorModel](#)), haverá algum atraso até que todos os detectores gerados (instâncias) sejam excluídos e o novo modelo seja usado para recriar os detectores. Eles são recriados depois que o novo modelo de detector entra em vigor e novas entradas chegam. Durante esse período, as entradas podem continuar sendo processadas pelos detectores gerados pela versão anterior do modelo do detector. Durante esse período, é possível continuar recebendo alertas definidos pelo modelo de detector anterior.

Espaços nas teclas de entrada

Espaços são permitidos nas teclas de entrada, mas as referências à chave devem estar entre crases, tanto na definição do atributo de entrada, quanto quando o valor da chave é referenciado em uma expressão. Por exemplo, dada uma carga útil de mensagem como a seguinte:

```
{
  "motor id": "A32",
  "sensorData" {
    "motor pressure": 56,
```

```
"motor temperature": 39
}
}
```

Use o seguinte para definir a entrada.

```
{
  "inputName": "PressureInput",
  "inputDescription": "Pressure readings from a motor",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "sensorData.`motor pressure`" },
      { "jsonPath": "`motor id`" }
    ]
  }
}
```

Em uma expressão condicional, você também deve se referir ao valor de qualquer chave usando crases.

```
$input.PressureInput.sensorData.`motor pressure`
```

Um exemplo comentado: controle de temperatura HVAC com AWS IoT Events

Alguns dos exemplos de arquivos JSON a seguir têm comentários embutidos, sendo assim JSON inválidos. Versões completas desses exemplos, sem comentários, estão disponíveis em [Exemplo: Usando o controle de temperatura HVAC com AWS IoT Events](#).

Este exemplo implementa um modelo de controle de termostato que permite fazer o seguinte.

- Definir apenas um modelo de detector que possa ser usado para monitorar e controlar várias áreas. Uma instância de detector é criada para cada área.
- Ingerir dados de temperatura de vários sensores em cada área de controle.
- Alterar o ponto de ajuste da temperatura para uma área.
- Definir parâmetros operacionais para cada área e redefinir esses parâmetros enquanto a instância estiver em uso.

- Adicionar ou excluir dinamicamente sensores de uma área.
- Especificar um runtime mínimo para proteger as unidades de aquecimento e resfriamento.
- Rejeitar leituras anômalas do sensor.
- Definir pontos de ajuste de emergência que ativem imediatamente o aquecimento ou o resfriamento se algum sensor relatar uma temperatura acima ou abaixo de um determinado limite.
- Relatar leituras anômalas e picos de temperatura.

Tópicos

- [Definições de entrada para modelos de detectores em AWS IoT Events](#)
- [Crie uma definição AWS IoT Events de modelo de detector](#)
- [Use BatchUpdateDetector para atualizar um modelo AWS IoT Events de detector](#)
- [Use BatchPutMessage para entradas em AWS IoT Events](#)
- [Ingerir mensagens MQTT em AWS IoT Events](#)
- [Gere mensagens do Amazon SNS em AWS IoT Events](#)
- [Configure a DescribeDetector API em AWS IoT Events](#)
- [Use o mecanismo de AWS IoT Core regras para AWS IoT Events](#)

Definições de entrada para modelos de detectores em AWS IoT Events

Queremos criar um modelo de detector que possa ser usado para monitorar e controlar a temperatura em várias áreas diferentes. Cada área pode ter vários sensores que informam a temperatura. Presumimos que cada área é servida por uma unidade de aquecimento e uma unidade de resfriamento que podem ser ligadas ou desligadas para controlar a temperatura na área. Cada área é controlada por uma instância de detector.

Como as diferentes áreas que monitoramos e controlamos podem ter características diferentes que exigem parâmetros de controle diferentes, definimos o 'seedTemperatureInput' para fornecer esses parâmetros para cada área. Quando enviamos uma dessas mensagens de entrada para o AWS IoT Events, é criada uma nova instância do modelo de detector com os parâmetros que queremos usar nessa área. Confira aqui a definição dessa entrada.

Comando da CLI:

```
aws iotevents create-input --cli-input-json file://seedInput.json
```

Arquivo: seedInput.json

```
{
  "inputName": "seedTemperatureInput",
  "inputDescription": "Temperature seed values.",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "areaId" },
      { "jsonPath": "desiredTemperature" },
      { "jsonPath": "allowedError" },
      { "jsonPath": "rangeHigh" },
      { "jsonPath": "rangeLow" },
      { "jsonPath": "anomalousHigh" },
      { "jsonPath": "anomalousLow" },
      { "jsonPath": "sensorCount" },
      { "jsonPath": "noDelay" }
    ]
  }
}
```

Resposta:

```
{
  "inputConfiguration": {
    "status": "ACTIVE",
    "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/seedTemperatureInput",
    "lastUpdateTime": 1557519620.736,
    "creationTime": 1557519620.736,
    "inputName": "seedTemperatureInput",
    "inputDescription": "Temperature seed values."
  }
}
```

Observações

- Uma nova instância de detector é criada para cada 'areaId' exclusivo recebido em qualquer mensagem. Veja o campo 'key' na definição 'areaDetectorModel'.
- A temperatura média pode variar de 'desiredTemperature' até 'allowedError' antes que as unidades de aquecimento ou resfriamento sejam ativadas para a área.

- Se algum sensor relatar uma temperatura acima de 'rangeHigh', o detector relatará um pico e iniciará imediatamente a unidade de resfriamento.
- Se algum sensor relatar uma temperatura abaixo de 'rangeLow', o detector relatará um pico e iniciará imediatamente a unidade de aquecimento.
- Se algum sensor relatar uma temperatura acima de 'anomalousHigh' ou abaixo de 'anomalousLow', o detector relatará uma leitura anômala do sensor, mas ignorará a leitura da temperatura relatada.
- O 'sensorCount' informa ao detector quantos sensores estão reportando para a área. O detector calcula a temperatura média na área fornecendo o fator de peso apropriado para cada leitura de temperatura que recebe. Por causa disso, o detector não precisará acompanhar o que cada sensor relata, e o número de sensores pode ser alterado dinamicamente, conforme necessário. No entanto, se um sensor individual ficar off-line, o detector não saberá sobre ou aceitará essa situação. Recomendamos que você crie outro modelo de detector específico para monitorar o status da conexão de cada sensor. Ter dois modelos de detectores complementares simplifica o projeto de ambos.
- O valor de 'noDelay' pode ser true ou false. Depois que uma unidade de aquecimento ou resfriamento é ligada, ela deve permanecer ligada por um certo tempo mínimo para proteger a integridade da unidade e prolongar sua vida útil. Se 'noDelay' estiver configurado como false, a instância do detector impõe um atraso antes de desligar as unidades de resfriamento e aquecimento, para garantir que elas funcionem pelo tempo mínimo. O número de segundos de atraso foi codificado na definição do modelo do detector porque não podemos usar um valor variável para definir um temporizador.

O 'temperatureInput' é usado para transmitir dados do sensor para uma instância do detector.

Comando da CLI:

```
aws iotevents create-input --cli-input-json file://temperatureInput.json
```

Arquivo: temperatureInput.json

```
{
  "inputName": "temperatureInput",
  "inputDescription": "Temperature sensor unit data.",
  "inputDefinition": {
```

```
"attributes": [
  { "jsonPath": "sensorId" },
  { "jsonPath": "areaId" },
  { "jsonPath": "sensorData.temperature" }
]
}
```

Resposta:

```
{
  "inputConfiguration": {
    "status": "ACTIVE",
    "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/temperatureInput",
    "lastUpdateTime": 1557519707.399,
    "creationTime": 1557519707.399,
    "inputName": "temperatureInput",
    "inputDescription": "Temperature sensor unit data."
  }
}
```

Observações

- O 'sensorId' não é usado por uma instância de detector de exemplo para controlar ou monitorar um sensor diretamente. Ele é automaticamente passado para as notificações enviadas pela instância do detector. A partir daí, ele pode ser usado para identificar os sensores que estão falhando (por exemplo, um sensor que envia regularmente leituras anômalas pode estar prestes a falhar) ou que estão off-line (quando usado como entrada para um modelo de detector adicional que monitora o batimento cardíaco do dispositivo). O 'sensorId' também pode ajudar a identificar zonas quentes ou frias em uma área se suas leituras regularmente diferirem da média.
- O 'areaId' é usado para rotear os dados do sensor para a instância apropriada do detector. Uma instância de detector é criada para cada 'areaId' exclusivo recebido em qualquer mensagem. Veja o campo 'key' na definição 'areaDetectorModel'.

Crie uma definição AWS IoT Events de modelo de detector

O exemplo 'areaDetectorModel' tem comentários embutidos.

Comando da CLI:

```
aws iotevents create-detector-model --cli-input-json file://areaDetectorModel.json
```

Arquivo: areaDetectorModel.json

```
{
  "detectorModelName": "areaDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "start",
        // In the 'start' state we set up the operation parameters of the new detector
        // instance.
        // We get here when the first input message arrives. If that is a
        // 'seedTemperatureInput'
        // message, we save the operation parameters, then transition to the 'idle'
        // state. If
        // the first message is a 'temperatureInput', we wait here until we get a
        // 'seedTemperatureInput' input to ensure our operation parameters are set.
        // We can
        // also reenter this state using the 'BatchUpdateDetector' API. This enables
        // us to
        // reset the operation parameters without needing to delete the detector
        // instance.
        "onEnter": {
          "events": [
            {
              "eventName": "prepare",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    // initialize 'sensorId' to an invalid value (0) until an actual
                    // sensor reading
                    // arrives
                    "variableName": "sensorId",
                    "value": "0"
                  }
                },
                {
                  "setVariable": {
                    // initialize 'reportedTemperature' to an invalid value (0.1) until
                    // an actual
```



```
    }
  },
  {
    "setVariable": {
      // Assume we're at the desired temperature when we start.
      "variableName": "averageTemperature",
      "value": "$input.seedTemperatureInput.desiredTemperature"
    }
  },
  {
    "setVariable": {
      "variableName": "allowedError",
      "value": "$input.seedTemperatureInput.allowedError"
    }
  },
  {
    "setVariable": {
      "variableName": "anomalousHigh",
      "value": "$input.seedTemperatureInput.anomalousHigh"
    }
  },
  {
    "setVariable": {
      "variableName": "anomalousLow",
      "value": "$input.seedTemperatureInput.anomalousLow"
    }
  },
  {
    "setVariable": {
      "variableName": "sensorCount",
      "value": "$input.seedTemperatureInput.sensorCount"
    }
  },
  {
    "setVariable": {
      "variableName": "noDelay",
      "value": "$input.seedTemperatureInput.noDelay == true"
    }
  }
],
"nextState": "idle"
},
{
  "eventName": "reset",
```

```

        "condition": "($variable.resetMe == true) &&
($input.temperatureInput.sensorData.temperature < $variable.anomalousHigh &&
$input.temperatureInput.sensorData.temperature > $variable.anomalousLow)",
        // This event is triggered if we have reentered the 'start' state using
the
        // 'BatchUpdateDetector' API with 'resetMe' set to true. When we
reenter using
        // 'BatchUpdateDetector' we do not automatically continue to the 'idle'
state, but
        // wait in 'start' until the next input message arrives. This event
enables us to
        // transition to 'idle' on the next valid 'temperatureInput' message
that arrives.
        "actions": [
            {
                "setVariable": {
                    "variableName": "averageTemperature",
                    "value": "(((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
                }
            }
        ],
        "nextState": "idle"
    }
]
},
"onExit": {
    "events": [
        {
            "eventName": "resetHeatCool",
            "condition": "true",
            // Make sure the heating and cooling units are off before entering
'idle'.
            "actions": [
                {
                    "sns": {
                        "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
                    }
                },
                {
                    "sns": {
                        "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOff"
                    }
                }
            ],
        }
    ]
}

```

```

        {
            "iotTopicPublish": {
                "mqttTopic": "hvac/Heating/Off"
            }
        },
        {
            "iotTopicPublish": {
                "mqttTopic": "hvac/Cooling/Off"
            }
        }
    ]
}
],
},
},

{
    "stateName": "idle",
    "onInput": {
        "events": [
            {
                "eventName": "whatWasInput",
                "condition": "true",
                // By storing the 'sensorId' and the 'temperature' in variables, we make
them
                // available in any messages we send out to report anomalies, spikes,
or just
                // if needed for debugging.
                "actions": [
                    {
                        "setVariable": {
                            "variableName": "sensorId",
                            "value": "$input.temperatureInput.sensorId"
                        }
                    },
                    {
                        "setVariable": {
                            "variableName": "reportedTemperature",
                            "value": "$input.temperatureInput.sensorData.temperature"
                        }
                    }
                ]
            }
        ]
    },
},

```

```

    {
      "eventName": "changeDesired",
      "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
      // This event enables us to change the desired temperature at any time by
      sending a
      // 'seedTemperatureInput' message. But note that other operational
      parameters are not
      // read or changed.
      "actions": [
        {
          "setVariable": {
            "variableName": "desiredTemperature",
            "value": "$input.seedTemperatureInput.desiredTemperature"
          }
        }
      ]
    },
    {
      "eventName": "calculateAverage",
      "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
      // If a valid temperature reading arrives, we use it to update the
      average temperature.
      // For simplicity, we assume our sensors will be sending updates at
      about the same rate,
      // so we can calculate an approximate average by giving equal weight to
      each reading we receive.
      "actions": [
        {
          "setVariable": {
            "variableName": "averageTemperature",
            "value": "((( $variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
          }
        }
      ]
    }
  ],
  "transitionEvents": [
    {
      "eventName": "anomalousInputArrived",

```

```

        "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
        // When an anomalous reading arrives, send an MQTT message, but stay in
the current state.
        "actions": [
            {
                "iotTopicPublish": {
                    "mqttTopic": "temperatureSensor/anomaly"
                }
            }
        ],
        "nextState": "idle"
    },

    {
        "eventName": "highTemperatureSpike",
        "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
        // When even a single temperature reading arrives that is above the
'rangeHigh', take
        // emergency action to begin cooling, and report a high temperature
spike.
        "actions": [
            {
                "iotTopicPublish": {
                    "mqttTopic": "temperatureSensor/spike"
                }
            },
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:cool0n"
                }
            },
            {
                "iotTopicPublish": {
                    "mqttTopic": "hvac/Cooling/On"
                }
            },
            {
                "setVariable": {
                    // This is necessary because we want to set a timer to delay the
shutoff

```

```

        // of a cooling/heating unit, but we only want to set the timer
when we
        // enter that new state initially.
        "variableName": "enteringNewState",
        "value": "true"
    }
}
],
"nextState": "cooling"
},

{
    "eventName": "lowTemperatureSpike",
    "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
    // When even a single temperature reading arrives that is below the
'rangeLow', take
    // emergency action to begin heating, and report a low-temperature
spike.
    "actions": [
        {
            "iotTopicPublish": {
                "mqttTopic": "temperatureSensor/spike"
            }
        },
        {
            "sns": {
                "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOn"
            }
        },
        {
            "iotTopicPublish": {
                "mqttTopic": "hvac/Heating/On"
            }
        },
        {
            "setVariable": {
                "variableName": "enteringNewState",
                "value": "true"
            }
        }
    ],
    "nextState": "heating"
},

```

```

    {
      "eventName": "highTemperatureThreshold",
      "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) >
($variable.desiredTemperature + $variable.allowedError))",
      // When the average temperature is above the desired temperature plus the
allowed error factor,
      // it is time to start cooling. Note that we calculate the average
temperature here again
      // because the value stored in the 'averageTemperature' variable is not
yet available for use
      // in our condition.
      "actions": [
        {
          "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:cool0n"
          }
        },
        {
          "iotTopicPublish": {
            "mqttTopic": "hvac/Cooling/0n"
          }
        },
        {
          "setVariable": {
            "variableName": "enteringNewState",
            "value": "true"
          }
        }
      ],
      "nextState": "cooling"
    },

    {
      "eventName": "lowTemperatureThreshold",
      "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) <
($variable.desiredTemperature - $variable.allowedError))",
      // When the average temperature is below the desired temperature minus
the allowed error factor,
      // it is time to start heating. Note that we calculate the average
temperature here again

```

```
        // because the value stored in the 'averageTemperature' variable is not
yet available for use
        // in our condition.
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOn"
                }
            },
            {
                "iotTopicPublish": {
                    "mqttTopic": "hvac/Heating/On"
                }
            },
            {
                "setVariable": {
                    "variableName": "enteringNewState",
                    "value": "true"
                }
            }
        ],
        "nextState": "heating"
    }
]
}
},

{
    "stateName": "cooling",
    "onEnter": {
        "events": [
            {
                "eventName": "delay",
                "condition": "!$variable.noDelay && $variable.enteringNewState",
                // If the operational parameters specify that there should be a minimum
time that the
                // heating and cooling units should be run before being shut off again,
we set
                // a timer to ensure the proper operation here.
                "actions": [
                    {
                        "setTimer": {
                            "timerName": "coolingTimer",
```

```
        "seconds": 180
      }
    },
    {
      "setVariable": {
        // We use this 'goodToGo' variable to store the status of the timer
expiration
        // for use in conditions that also use input variable values. If
lost.
        // 'timeout()' is used in such mixed conditionals, its value is

        "variableName": "goodToGo",
        "value": "false"
      }
    }
  ]
},
{
  "eventName": "dontDelay",
  "condition": "$variable.noDelay == true",
  // If the heating/cooling unit shutoff delay is not used, no need to
wait.
  "actions": [
    {
      "setVariable": {
        "variableName": "goodToGo",
        "value": "true"
      }
    }
  ]
},
{
  "eventName": "beenHere",
  "condition": "true",
  "actions": [
    {
      "setVariable": {
        "variableName": "enteringNewState",
        "value": "false"
      }
    }
  ]
}
],
},
],
},
```

```
"onInput": {
  "events": [
    // These are events that occur when an input is received (if the condition
is
    // satisfied), but don't cause a transition to another state.
    {
      "eventName": "whatWasInput",
      "condition": "true",
      "actions": [
        {
          "setVariable": {
            "variableName": "sensorId",
            "value": "$input.temperatureInput.sensorId"
          }
        },
        {
          "setVariable": {
            "variableName": "reportedTemperature",
            "value": "$input.temperatureInput.sensorData.temperature"
          }
        }
      ]
    },
    {
      "eventName": "changeDesired",
      "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
      "actions": [
        {
          "setVariable": {
            "variableName": "desiredTemperature",
            "value": "$input.seedTemperatureInput.desiredTemperature"
          }
        }
      ]
    },
    {
      "eventName": "calculateAverage",
      "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
      "actions": [
        {
```

```

        "setVariable": {
            "variableName": "averageTemperature",
            "value": "((( $variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
        }
    }
]
},
{
    "eventName": "areWeThereYet",
    "condition": "(timeout(\"coolingTimer\"))",
    "actions": [
        {
            "setVariable": {
                "variableName": "goodToGo",
                "value": "true"
            }
        }
    ]
}
],
"transitionEvents": [
    // Note that some tests of temperature values (for example, the test for an
anomalous value)
    // must be placed here in the 'transitionEvents' because they work
together with the tests
    // in the other conditions to ensure that we implement the proper
    "if..elseif..else" logic.
    // But each transition event must have a destination state ('nextState'),
and even if that
    // is actually the current state, the "onEnter" events for this state
will be executed again.
    // This is the reason for the 'enteringNewState' variable and related.
    {
        "eventName": "anomalousInputArrived",
        "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
        "actions": [
            {
                "iotTopicPublish": {
                    "mqttTopic": "temperatureSensor/anomaly"
                }
            }
        ]
    }
]
}

```

```
    ],
    "nextState": "cooling"
  },
  {
    "eventName": "highTemperatureSpike",
    "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
    "actions": [
      {
        "iotTopicPublish": {
          "mqttTopic": "temperatureSensor/spike"
        }
      }
    ],
    "nextState": "cooling"
  },
  {
    "eventName": "lowTemperatureSpike",
    "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
    "actions": [
      {
        "iotTopicPublish": {
          "mqttTopic": "temperatureSensor/spike"
        }
      },
      {
        "sns": {
          "targetArn": "arn:aws:sns:us-west-2:123456789012:cool0ff"
        }
      },
      {
        "sns": {
          "targetArn": "arn:aws:sns:us-west-2:123456789012:heat0n"
        }
      }
    ],
    {
      "iotTopicPublish": {
        "mqttTopic": "hvac/Cooling/Off"
      }
    }
  },
  {
```

```

        "iotTopicPublish": {
            "mqttTopic": "hvac/Heating/On"
        }
    },
    {
        "setVariable": {
            "variableName": "enteringNewState",
            "value": "true"
        }
    }
],
"nextState": "heating"
},
{
    "eventName": "desiredTemperature",
    "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) <=
($variable.desiredTemperature - $variable.allowedError)) && $variable.goodToGo ==
true",
    "actions": [
        {
            "sns": {
                "targetArn": "arn:aws:sns:us-west-2:123456789012:cool0ff"
            }
        },
        {
            "iotTopicPublish": {
                "mqttTopic": "hvac/Cooling/Off"
            }
        }
    ],
    "nextState": "idle"
}
]
}
},
{
    "stateName": "heating",
    "onEnter": {
        "events": [
            {

```

```
    "eventName": "delay",
    "condition": "!$variable.noDelay && $variable.enteringNewState",
    "actions": [
      {
        "setTimer": {
          "timerName": "heatingTimer",
          "seconds": 120
        }
      },
      {
        "setVariable": {
          "variableName": "goodToGo",
          "value": "false"
        }
      }
    ]
  },
  {
    "eventName": "dontDelay",
    "condition": "$variable.noDelay == true",
    "actions": [
      {
        "setVariable": {
          "variableName": "goodToGo",
          "value": "true"
        }
      }
    ]
  },
  {
    "eventName": "beenHere",
    "condition": "true",
    "actions": [
      {
        "setVariable": {
          "variableName": "enteringNewState",
          "value": "false"
        }
      }
    ]
  }
]
},
```

```

"onInput": {
  "events": [
    {
      "eventName": "whatWasInput",
      "condition": "true",
      "actions": [
        {
          "setVariable": {
            "variableName": "sensorId",
            "value": "$input.temperatureInput.sensorId"
          }
        },
        {
          "setVariable": {
            "variableName": "reportedTemperature",
            "value": "$input.temperatureInput.sensorData.temperature"
          }
        }
      ]
    },
    {
      "eventName": "changeDesired",
      "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
      "actions": [
        {
          "setVariable": {
            "variableName": "desiredTemperature",
            "value": "$input.seedTemperatureInput.desiredTemperature"
          }
        }
      ]
    },
    {
      "eventName": "calculateAverage",
      "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
      "actions": [
        {
          "setVariable": {
            "variableName": "averageTemperature",
            "value": "((( $variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
          }
        }
      ]
    }
  ]
}

```

```

    }
  }
]
},
{
  "eventName": "areWeThereYet",
  "condition": "(timeout(\"heatingTimer\"))",
  "actions": [
    {
      "setVariable": {
        "variableName": "goodToGo",
        "value": "true"
      }
    }
  ]
}
],
"transitionEvents": [
  {
    "eventName": "anomalousInputArrived",
    "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
    "actions": [
      {
        "iotTopicPublish": {
          "mqttTopic": "temperatureSensor/anomaly"
        }
      }
    ],
    "nextState": "heating"
  },
  {
    "eventName": "highTemperatureSpike",
    "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
    "actions": [
      {
        "iotTopicPublish": {
          "mqttTopic": "temperatureSensor/spike"
        }
      }
    ],
  }
]

```

```

        "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
        }
    },
    {
        "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOn"
        }
    },
    {
        "iotTopicPublish": {
            "mqttTopic": "hvac/Heating/Off"
        }
    },
    {
        "iotTopicPublish": {
            "mqttTopic": "hvac/Cooling/On"
        }
    },
    {
        "setVariable": {
            "variableName": "enteringNewState",
            "value": "true"
        }
    }
],
"nextState": "cooling"
},

{
    "eventName": "lowTemperatureSpike",
    "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
    "actions": [
        {
            "iotTopicPublish": {
                "mqttTopic": "temperatureSensor/spike"
            }
        }
    ]
},
"nextState": "heating"
},

{

```

```

        "eventName": "desiredTemperature",
        "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) >=
($variable.desiredTemperature + $variable.allowedError)) && $variable.goodToGo ==
true",
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
                }
            },
            {
                "iotTopicPublish": {
                    "mqttTopic": "hvac/Heating/Off"
                }
            }
        ],
        "nextState": "idle"
    }
]
}
},
"initialStateName": "start"
},
"key": "areaId",
"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole"
}

```

Resposta:

```

{
  "detectorModelConfiguration": {
    "status": "ACTIVATING",
    "lastUpdateTime": 1557523491.168,
    "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
    "creationTime": 1557523491.168,
    "detectorModelArn": "arn:aws:iotevents:us-west-2:123456789012:detectorModel/
areaDetectorModel",
    "key": "areaId",
    "detectorModelName": "areaDetectorModel",

```

```
    "detectorModelVersion": "1"
  }
}
```

Use BatchUpdateDetector para atualizar um modelo AWS IoT Events de detector

É possível usar a operação `BatchUpdateDetector` para colocar uma instância do detector em um estado conhecido, incluindo valores de temporizador e variáveis. No exemplo a seguir, a operação `BatchUpdateDetector` redefine os parâmetros operacionais de uma área que está sob monitoramento e controle de temperatura. Esta operação permite que você faça isso sem precisar excluir, recriar ou atualizar o modelo do detector.

Comando da CLI:

```
aws iotevents-data batch-update-detector --cli-input-json file://areaDM.BUD.json
```

Arquivo: `areaDM.BUD.json`

```
{
  "detectors": [
    {
      "messageId": "0001",
      "detectorModelName": "areaDetectorModel",
      "keyValue": "Area51",
      "state": {
        "stateName": "start",
        "variables": [
          {
            "name": "desiredTemperature",
            "value": "22"
          },
          {
            "name": "averageTemperature",
            "value": "22"
          },
          {
            "name": "allowedError",
            "value": "1.0"
          }
        ]
      }
    }
  ]
}
```

```
{
  "name": "rangeHigh",
  "value": "30.0"
},
{
  "name": "rangeLow",
  "value": "15.0"
},
{
  "name": "anomalousHigh",
  "value": "60.0"
},
{
  "name": "anomalousLow",
  "value": "0.0"
},
{
  "name": "sensorCount",
  "value": "12"
},
{
  "name": "noDelay",
  "value": "true"
},
{
  "name": "goodToGo",
  "value": "true"
},
{
  "name": "sensorId",
  "value": "0"
},
{
  "name": "reportedTemperature",
  "value": "0.1"
},
{
  "name": "resetMe",
  // When 'resetMe' is true, our detector model knows that we have reentered
the 'start' state
  // to reset operational parameters, and will allow the next valid
temperature sensor
  // reading to cause the transition to the 'idle' state.
  "value": "true"
}
```

```
    }
  ],
  "timers": [
  ]
}
]
```

Resposta:

```
{
  "batchUpdateDetectorErrorEntries": []
}
```

Use BatchPutMessage para entradas em AWS IoT Events

Example 1

Use a operação `BatchPutMessage` para enviar uma mensagem `"seedTemperatureInput"` que define os parâmetros operacionais para uma determinada área sob controle e monitoramento de temperatura. Qualquer mensagem recebida por AWS IoT Events isso tem uma nova `"areaId"` causa a criação de uma nova instância de detector. Mas a nova instância do detector não mudará de estado para `"idle"` e começará a monitorar a temperatura e controlar as unidades de aquecimento ou resfriamento até que uma mensagem `"seedTemperatureInput"` seja recebida para a nova área.

Comando da CLI:

```
aws iotevents-data batch-put-message --cli-input-json file://seedExample.json --cli-binary-format raw-in-base64-out
```

Arquivo: `seedExample.json`

```
{
  "messages": [
    {
      "messageId": "00001",
      "inputName": "seedTemperatureInput",
```

```
    "payload": "{\\"areaId\\": \\"Area51\\", \\"desiredTemperature\\": 20.0, \\"allowedError\\": 0.7, \\"rangeHigh\\": 30.0, \\"rangeLow\\": 15.0, \\"anomalousHigh\\": 60.0, \\"anomalousLow\\": 0.0, \\"sensorCount\\": 10, \\"noDelay\\": false}"
  }
]
```

Resposta:

```
{
  "BatchPutMessageErrorEntries": []
}
```

Example

2

Use a operação `BatchPutMessage` para enviar uma mensagem "temperatureInput" para relatar os dados do sensor de temperatura de um sensor em uma determinada área de controle e monitoramento.

Comando da CLI:

```
aws iotevents-data batch-put-message --cli-input-json file://temperatureExample.json --cli-binary-format raw-in-base64-out
```

Arquivo: `temperatureExample.json`

```
{
  "messages": [
    {
      "messageId": "00005",
      "inputName": "temperatureInput",
      "payload": "{\\"sensorId\\": \\"05\\", \\"areaId\\": \\"Area51\\", \\"sensorData\\": {\\"temperature\\": 23.12} }"
    }
  ]
}
```

Resposta:

```
{
  "BatchPutMessageErrorEntries": []
}
```

Example 3

Use a operação `BatchPutMessage` para enviar uma mensagem `"seedTemperatureInput"` para alterar o valor da temperatura desejada para uma determinada área.

Comando da CLI:

```
aws iotevents-data batch-put-message --cli-input-json file://seedSetDesiredTemp.json --cli-binary-format raw-in-base64-out
```

Arquivo: `seedSetDesiredTemp.json`

```
{
  "messages": [
    {
      "messageId": "00001",
      "inputName": "seedTemperatureInput",
      "payload": "{\"areaId\": \"Area51\", \"desiredTemperature\": 23.0}"
    }
  ]
}
```

Resposta:

```
{
  "BatchPutMessageErrorEntries": []
}
```

Ingerir mensagens MQTT em AWS IoT Events

Se seus recursos de computação de sensores não puderem usar a `"BatchPutMessage"` API, mas puderem enviar seus dados para o agente de AWS IoT Core mensagens usando um cliente MQTT leve, você poderá criar uma regra de AWS IoT Core tópico para redirecionar os dados da mensagem para uma AWS IoT Events entrada. A seguir está uma definição de uma regra de AWS

IoT Events tópico que usa os campos "areaId" e de "sensorId" entrada do tópico MQTT e o "sensorData.temperature" campo do campo de carga útil "temp" da mensagem e ingere esses dados em nosso. AWS IoT Events "temperatureInput"

Comando da CLI:

```
aws iot create-topic-rule --cli-input-json file://temperatureTopicRule.json
```

Arquivo: seedSetDesiredTemp.json

```
{
  "ruleName": "temperatureTopicRule",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as areaId, topic(4) as sensorId, temp as
sensorData.temperature FROM 'update/temperature/#'",
    "description": "Ingest temperature sensor messages into IoT Events",
    "actions": [
      {
        "iotEvents": {
          "inputName": "temperatureInput",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/anotherRole"
        }
      }
    ],
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23"
  }
}
```

Resposta: [nenhuma]

Se o sensor enviar uma mensagem sobre o assunto "update/temperature/Area51/03" com a seguinte carga útil.

```
{ "temp": 24.5 }
```

Isso resulta na ingestão de dados AWS IoT Events como se a seguinte chamada de "BatchPutMessage" API tivesse sido feita.

```
aws iotevents-data batch-put-message --cli-input-json file://spoofoExample.json --cli-binary-format raw-in-base64-out
```

Arquivo: spoofoExample.json

```
{
  "messages": [
    {
      "messageId": "54321",
      "inputName": "temperatureInput",
      "payload": "{\"sensorId\": \"03\", \"areaId\": \"Area51\", \"sensorData\": {\"temperature\": 24.5} }"
    }
  ]
}
```

Gere mensagens do Amazon SNS em AWS IoT Events

Veja a seguir exemplos de mensagens de SNS geradas pela instância "Area51" do detector.

AWS IoT Events pode se integrar ao Amazon SNS para gerar e publicar notificações com base nos eventos detectados. Esta seção demonstra como uma instância do AWS IoT Events detector, especificamente o detector "Area51", gera mensagens do Amazon SNS. Esses exemplos mostram a estrutura e o conteúdo das notificações do Amazon SNS acionadas por vários estados e eventos dentro AWS IoT Events do detector, ilustrando o poder da combinação com o AWS IoT Events Amazon SNS para alertas e comunicação em tempo real.

```
Heating system off command> {
  "eventTime":1557520274729,
  "payload":{
    "actionExecutionId":"f3159081-bac3-38a4-96f7-74af0940d0a4",
    "detector":{

      "detectorModelName":"areaDetectorModel","keyValue":"Area51","detectorModelVersion":"1"}, "event
{"inputName":"seedTemperatureInput","messageId":"00001","triggerType":"Message"},"state":
{"stateName":"start","variables":
{"sensorCount":10,"rangeHigh":30.0,"resetMe":false,"enteringNewState":true,"averageTemperature"
{}}},"eventName":"resetHeatCool"}
```

```
Cooling system off command> {"eventTime":1557520274729,"payload":
{"actionExecutionId":"98f6a1b5-8f40-3cdb-9256-93afd4d66192","detector":
{"detectorModelName":"areaDetectorModel","keyValue":"Area51","detectorModelVersion":"1"},"event
{"inputName":"seedTemperatureInput","messageId":"00001","triggerType":"Message"},"state":
{"stateName":"start","variables":
{"sensorCount":10,"rangeHigh":30.0,"resetMe":false,"enteringNewState":true,"averageTemperature"
{}}},"eventName":"resetHeatCool"}
```

Configure a DescribeDetector API em AWS IoT Events

A `DescribeDetector` API AWS IoT Events permite que você recupere informações detalhadas sobre uma instância específica do detector. Essa operação fornece informações sobre o estado atual, valores variáveis e temporizadores ativos de um detector. Ao usar essa API, você pode monitorar o status em tempo real de seus AWS IoT Events detectores, facilitando a depuração, a análise e o gerenciamento de seus fluxos de trabalho de processamento de eventos de IoT.

Comando da CLI:

```
aws iotevents-data describe-detector --detector-model-name areaDetectorModel --key-
value Area51
```

Resposta:

```
{
  "detector": {
    "lastUpdateTime": 1557521572.216,
    "creationTime": 1557520274.405,
    "state": {
      "variables": [
        {
          "name": "resetMe",
          "value": "false"
        },
        {
          "name": "rangeLow",
          "value": "15.0"
        },
        {
          "name": "noDelay",
          "value": "false"
        }
      ]
    }
  }
}
```

```
{
  "name": "desiredTemperature",
  "value": "20.0"
},
{
  "name": "anomalousLow",
  "value": "0.0"
},
{
  "name": "sensorId",
  "value": "\"01\""
},
{
  "name": "sensorCount",
  "value": "10"
},
{
  "name": "rangeHigh",
  "value": "30.0"
},
{
  "name": "enteringNewState",
  "value": "false"
},
{
  "name": "averageTemperature",
  "value": "19.572"
},
{
  "name": "allowedError",
  "value": "0.7"
},
{
  "name": "anomalousHigh",
  "value": "60.0"
},
{
  "name": "reportedTemperature",
  "value": "15.72"
},
{
  "name": "goodToGo",
  "value": "false"
}
}
```

```

    ],
    "stateName": "idle",
    "timers": [
      {
        "timestamp": 1557520454.0,
        "name": "idleTimer"
      }
    ]
  },
  "keyValue": "Area51",
  "detectorModelName": "areaDetectorModel",
  "detectorModelVersion": "1"
}
}

```

Use o mecanismo de AWS IoT Core regras para AWS IoT Events

As regras a seguir republicam mensagens AWS IoT Core MQTT como mensagens de solicitação de atualização paralela. Assumimos que AWS IoT Core as coisas são definidas para uma unidade de aquecimento e uma unidade de resfriamento para cada área controlada pelo modelo do detector. Neste exemplo, definimos coisas chamadas "Area51HeatingUnit" e "Area51CoolingUnit".

Comando da CLI:

```
aws iot create-topic-rule --cli-input-json file://ADMShadowCoolOffRule.json
```

Arquivo: ADMShadowCoolOffRule.json

```

{
  "ruleName": "ADMShadowCoolOff",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Cooling/Off'",
    "description": "areaDetectorModel mqtt topic publish to cooling unit shadow request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republish": {
          "topic": "$$aws/things/${payload.detector.keyValue}CoolingUnit/shadow/update",

```

```

        "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMSHadowRole"
    }
}
]
}
}

```

Resposta: [vazio]

Comando da CLI:

```
aws iot create-topic-rule --cli-input-json file://ADMSHadowCoolOnRule.json
```

Arquivo: ADMSHadowCoolOnRule.json

```

{
  "ruleName": "ADMSHadowCoolOn",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Cooling/On'",
    "description": "areaDetectorModel mqtt topic publish to cooling unit shadow
request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republish": {
          "topic": "$$aws/things/${payload.detector.keyValue}CoolingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMSHadowRole"
        }
      }
    ]
  }
}

```

Resposta: [vazio]

Comando da CLI:

```
aws iot create-topic-rule --cli-input-json file://ADMSHadowHeatOffRule.json
```

Arquivo: ADMSHadowHeatOffRule.json

```
{
  "ruleName": "ADMSHadowHeatOff",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Heating/Off'",
    "description": "areaDetectorModel mqtt topic publish to heating unit shadow
request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republsh": {
          "topic": "$$aws/things/${payload.detector.keyValue}HeatingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMSHadowRole"
        }
      }
    ]
  }
}
```

Resposta: [vazio]

Comando da CLI:

```
aws iot create-topic-rule --cli-input-json file://ADMSHadowHeatOnRule.json
```

Arquivo: ADMSHadowHeatOnRule.json

```
{
  "ruleName": "ADMSHadowHeatOn",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Heating/On'",
    "description": "areaDetectorModel mqtt topic publish to heating unit shadow
request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republsh": {
```

```
        "topic": "$$aws/things/${payload.detector.keyValue}HeatingUnit/shadow/  
update",  
        "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMShadowRole"  
    }  
  }  
]  
}  
}
```

Resposta: [vazio]

Ações suportadas para receber dados e acionar ações em AWS IoT Events

AWS IoT Events pode acionar ações ao detectar um evento específico ou evento de transição. É possível definir ações integradas para usar um temporizador, definir uma variável ou enviar dados para outros recursos da AWS. Saiba como configurar e personalizar essas ações para criar respostas automatizadas para seus vários eventos de IoT.

Note

Ao definir uma ação em um modelo de detector, é possível usar expressões para parâmetros que são do tipo de dados de string. Para obter mais informações, consulte [Expressões](#).

AWS IoT Events suporta as seguintes ações que permitem usar um cronômetro ou definir uma variável:

- [setTimer](#) para criar um temporizador.
- [resetTimer](#) para redefinir o temporizador.
- [clearTimer](#) para excluir o temporizador.
- [setVariable](#) para criar uma variável.

AWS IoT Events suporta as seguintes ações que permitem trabalhar com AWS serviços:

- [iotTopicPublish](#) para publicar uma mensagem em um tópico MQTT.
- [iotEvents](#) para enviar dados para AWS IoT Events como um valor de entrada.
- [iotSiteWise](#) para enviar dados para uma propriedade de ativo no AWS IoT SiteWise.
- [dynamoDB](#) para enviar dados para uma tabela do Amazon DynamoDB.
- [dynamoDBv2](#) para enviar dados para uma tabela do Amazon DynamoDB.
- [firehose](#) para enviar dados para um stream do Amazon Data Firehose.
- [lambda](#) para invocar uma função do AWS Lambda.
- [sns](#) para enviar dados como uma notificação por push.
- [sqs](#) para enviar dados para uma fila do Amazon SQS.

Use o cronômetro AWS IoT Events embutido e as ações variáveis

AWS IoT Events suporta as seguintes ações que permitem usar um cronômetro ou definir uma variável:

- [setTimer](#) para criar um temporizador.
- [resetTimer](#) para redefinir o temporizador.
- [clearTimer](#) para excluir o temporizador.
- [setVariable](#) para criar uma variável.

Definir ação do temporizador

Set timer action

A ação `setTimer` permite criar um temporizador com duração em segundos.

More information (2)

Ao criar um temporizador, você deve especificar os seguintes parâmetros obrigatórios.

timerName

O nome do temporizador.

durationExpression

(Opcional) A duração do temporizador, em segundos.

O resultado avaliado da de uma expressão de duração é arredondado para baixo para o número inteiro mais próximo. Por exemplo, se você definir o temporizador para 60,99 segundos, o resultado avaliado da expressão de duração será 60 segundos.

Para obter mais informações, consulte [SetTimerAction](#) na Referência de APIs do AWS IoT Events

Redefinir ação do temporizador

Reset timer action

A ação `resetTimer` permite definir o temporizador para o resultado previamente avaliado da expressão de duração.

More information (1)

Ao criar um temporizador, você deve especificar o seguinte parâmetro.

timerName

O nome do temporizador.

AWS IoT Events não reavalia a expressão de duração quando você redefine o cronômetro.

Para obter mais informações, consulte [ResetTimerAction](#) na Referência de APIs do AWS IoT Events .

Apagar a ação do temporizador

Clear timer action

A ação `clearTimer` permite que você exclua um temporizador existente.

More information (1)

Ao excluir um temporizador, você deve especificar o seguinte parâmetro.

timerName

O nome do temporizador.

Para obter mais informações, consulte [ClearTimerAction](#) na Referência de APIs do AWS IoT Events .

Definir ação variável

Set variable action

A ação `setVariable` permite criar uma variável com um valor especificado.

More information (2)

Ao criar uma regra de retenção, você deve especificar os seguintes parâmetros.

variableName

O nome da variável.

value

O novo valor da variável.

Para obter mais informações, consulte [SetVariableAction](#) na Referência de APIs do AWS IoT Events .

AWS IoT Events trabalhando com outros AWS serviços

AWS IoT Events suporta as seguintes ações que permitem trabalhar com AWS serviços:

- [iotTopicPublish](#) para publicar uma mensagem em um tópico MQTT.
- [iotEvents](#) para enviar dados para AWS IoT Events como um valor de entrada.
- [iotSiteWise](#) para enviar dados para uma propriedade de ativo no AWS IoT SiteWise.
- [dynamoDB](#) para enviar dados para uma tabela do Amazon DynamoDB.
- [dynamoDBv2](#) para enviar dados para uma tabela do Amazon DynamoDB.
- [firehose](#) para enviar dados para um stream do Amazon Data Firehose.
- [lambda](#) para invocar uma função do AWS Lambda .
- [sns](#) para enviar dados como uma notificação por push.
- [sqs](#) para enviar dados para uma fila do Amazon SQS.

Important

- Você deve escolher a mesma AWS região para ambas AWS IoT Events e os AWS serviços com os quais trabalhar. Para obter a lista de regiões compatíveis, consulte [Endpoints e cotas do AWS IoT Events](#) no Referência geral da Amazon Web Services.
- Você deve usar a mesma AWS região ao criar outros AWS recursos para as AWS IoT Events ações. Se você mudar de AWS região, poderá ter problemas para acessar os AWS recursos.

Por padrão, AWS IoT Events gera uma carga padrão em JSON para qualquer ação. Essa carga útil da ação contém todos os pares de atributo-valor que têm as informações sobre a instância do modelo de detector e o evento que acionou a ação. Para configurar a carga útil da ação, você pode usar uma expressão do conteúdo. Para obter mais informações, consulte [Expressões para filtrar, transformar e processar dados de eventos](#) e o tipo de dados [Carga útil](#) na Referência API do AWS IoT Events .

AWS IoT Core

IoT topic publish action

A AWS IoT Core ação permite que você publique uma mensagem MQTT por meio do agente de AWS IoT mensagens. Para obter a lista de regiões compatíveis, consulte [Endpoints e cotas do AWS IoT Core](#) no Referência geral da Amazon Web Services.

O agente de AWS IoT mensagens conecta AWS IoT clientes enviando mensagens de clientes de publicação para clientes assinantes. Para obter mais informações, consulte [Protocolos de comunicação do dispositivo](#) no Guia do AWS IoT desenvolvedor.

More information (2)

Ao publicar uma mensagem MQTT, você deve especificar os seguintes parâmetros.

mqttTopic

O tópico MQTT que recebe a mensagem.

É possível definir um nome de tópico do MQTT dinamicamente em runtime usando variáveis ou valores de entrada criados no modelo do detector.

payload

(Opcional) A carga útil da ação padrão contém todos os pares de atributo-valor que têm as informações sobre a instância do modelo de detector e o evento que acionou a ação. Também é possível personalizar a carga útil. Para obter mais informações, consulte [Carga útil](#) na Referência da API do AWS IoT Events .

Note

Certifique-se de que a política anexada à sua função AWS IoT Events de serviço conceda a `iot:Publish` permissão. Para obter mais informações, consulte [Gerenciamento de identidade e acesso para AWS IoT Events](#).

Para obter mais informações, consulte [lotTopicPublishAction](#) na Referência de APIs do AWS IoT Events .

AWS IoT Events

IoT Events action

A AWS IoT Events ação permite que você envie dados AWS IoT Events como entrada. Para obter a lista de regiões compatíveis, consulte [Endpoints e cotas do AWS IoT Events](#) no Referência geral da Amazon Web Services.

AWS IoT Events permite monitorar suas frotas de equipamentos ou dispositivos em busca de falhas ou alterações na operação e acionar ações quando esses eventos ocorrerem. Para obter mais informações, consulte [O que é AWS IoT Events?](#) no Guia do AWS IoT Events desenvolvedor.

More information (2)

Ao enviar dados para AWS IoT Events, você deve especificar os seguintes parâmetros.

inputName

O nome da AWS IoT Events entrada que recebe os dados.

payload

(Opcional) A carga útil da ação padrão contém todos os pares de atributo-valor que têm as informações sobre a instância do modelo de detector e o evento que acionou a ação. Também é possível personalizar a carga útil. Para obter mais informações, consulte [Carga útil](#) na Referência da API do AWS IoT Events .

Note

Certifique-se de que a política anexada à sua função AWS IoT Events de serviço conceda a `iotevents:BatchPutMessage` permissão. Para obter mais informações, consulte [Gerenciamento de identidade e acesso para AWS IoT Events](#).

Para obter mais informações, consulte [lotEventsAction](#) na Referência de APIs do AWS IoT Events .

AWS IoT SiteWise

IoT SiteWise action

A AWS IoT SiteWise ação permite que você envie dados para uma propriedade do ativo em AWS IoT SiteWise. Para obter a lista de regiões compatíveis, consulte [Endpoints e cotas do AWS IoT SiteWise](#) no Referência geral da Amazon Web Services.

AWS IoT SiteWise é um serviço gerenciado que permite coletar, organizar e analisar dados de equipamentos industriais em grande escala. Para obter mais informações, consulte [O que é o AWS IoT SiteWise?](#) no Guia do usuário do AWS IoT SiteWise .

More information (11)

Ao enviar dados para uma propriedade do ativo em AWS IoT SiteWise, você deve especificar os seguintes parâmetros.

Important

Para receber os dados, você deve usar uma propriedade de ativo existente em AWS IoT SiteWise.

- Se você usar o AWS IoT Events console, deverá especificar `propertyAlias` para identificar a propriedade do ativo de destino.
- Se você usar o AWS CLI, deverá especificar um `propertyAlias` ou ambos `assetId` e identificar `propertyId` a propriedade do ativo de destino.

Para ter mais informações, consulte [Mapping industrial data streams to asset properties](#) no Guia do usuário do AWS IoT SiteWise .

propertyAlias

(Opcional) O alias da propriedade do ativo. Também é possível especificar uma expressão.

assetId

(Opcional) A ID do ativo que tem a propriedade especificada. Também é possível especificar uma expressão.

propertyId

(Opcional) A ID de uma propriedade de ativo. Também é possível especificar uma expressão.

entryId

(Opcional) Um identificador exclusivo para essa entrada. É possível usar o ID de entrada para rastrear qual entrada de dados causa um erro em caso de falha. O padrão é um novo identificador exclusivo. Também é possível especificar uma expressão.

propertyValue

Uma estrutura que contém detalhes sobre o valor da propriedade.

quality

(Opcional) A qualidade do valor da propriedade do ativo. O valor deve ser GOOD, BAD ou UNCERTAIN. Também é possível especificar uma expressão.

timestamp

(Opcional) Uma estrutura que contém informações do timestamp. Se esse valor não for especificado, o valor padrão será o tempo do evento.

timeInSeconds

O time stamp, em segundos, no formato Unix epoch. O intervalo válido é entre 1-31556889864403199. Também é possível especificar uma expressão.

offsetInNanos

(Opcional) O deslocamento em nanossegundos convertido de `timeInSeconds`. O intervalo válido é entre 0-999999999. Também é possível especificar uma expressão.

value

Uma estrutura que contém um valor de propriedade de ativo.

Important

É necessário especificar um dos seguintes tipos de valor, dependendo do `dataType` da propriedade de ativo especificada. Para obter mais informações, consulte [AssetProperty](#) na Referência de APIs do AWS IoT SiteWise .

booleanValue

(Opcional) O valor da propriedade do ativo é um valor booleano que deve ser `TRUE` ou `FALSE`. Também é possível especificar uma expressão. Se você usar uma expressão, o resultado avaliado deverá ser um valor booleano.

doubleValue

(Opcional) O valor da propriedade do ativo é um dobro. Também é possível especificar uma expressão. Se você usar uma expressão, o resultado avaliado deverá ser um dobro.

integerValue

(Opcional) O valor da propriedade do ativo é um inteiro. Também é possível especificar uma expressão. Se você usar uma expressão, o resultado avaliado deverá ser um inteiro.

stringValue

(Opcional) O valor da propriedade do ativo é uma string. Também é possível especificar uma expressão. Se você usar uma expressão, o resultado avaliado deverá ser uma string.

Note

Certifique-se de que a política anexada à sua função AWS IoT Events de serviço conceda a `iotsitewise:BatchPutAssetPropertyValue` permissão. Para obter mais informações, consulte [Gerenciamento de identidade e acesso para AWS IoT Events](#).

Para obter mais informações, consulte [lotSiteWiseAction](#) na Referência de APIs do AWS IoT Events .

Amazon DynamoDB

DynamoDB action

A ação do Amazon DynamoDB permite enviar dados para uma tabela do DynamoDB. Uma coluna da tabela do DynamoDB recebe todos os pares de atributo-valor na carga útil que você especifica. Para obter a lista de regiões compatíveis, consulte [Amazon DynamoDB endpoints e cotas do](#) no Referência geral da Amazon Web Services.

O Amazon DynamoDB é um serviço de banco de dados NoSQL totalmente gerenciado que fornece uma performance rápida e previsível com escalabilidade integrada. Para obter mais informações, consulte [O que é DynamoDB?](#) no Guia do desenvolvedor Amazon DynamoDB.

More information (10)

Ao enviar dados para uma coluna de uma tabela do DynamoDB, você deve especificar os seguintes parâmetros.

tableName

O nome da tabela do DynamoDB que recebe os dados. O valor `tableName` deve corresponder ao nome da tabela DynamoDB na tabela. Também é possível especificar uma expressão.

hashKeyField

O nome da chave de hash (também chamada de chave de partição). O valor `hashKeyField` deve corresponder à chave de partição da tabela DynamoDB. Também é possível especificar uma expressão.

hashKeyType

(Opcional) O tipo de dados da chave de hash. O valor do tipo de chave de hash deve ser `STRING` ou `NUMBER`. O padrão é `STRING`. Também é possível especificar uma expressão.

hashKeyValue

O valor da chave de hash. O `hashKeyValue` usa modelos de substituição. Esses modelos fornecem dados em runtime. Também é possível especificar uma expressão.

rangeKeyField

(Opcional) O nome da chave de intervalo (também chamada de chave de classificação). O valor `rangeKeyField` deve corresponder à chave de classificação da tabela do DynamoDB. Também é possível especificar uma expressão.

rangeKeyType

(Opcional) O tipo de dados da chave de intervalo. O valor do tipo de chave de hash deve ser `STRING` ou `NUMBER`. O padrão é `STRING`. Também é possível especificar uma expressão.

rangeKeyValue

(Opcional) O valor da chave de intervalo. O `rangeKeyValue` usa modelos de substituição. Esses modelos fornecem dados em runtime. Também é possível especificar uma expressão.

operação

(Opcional) O tipo de operação a executar. Também é possível especificar uma expressão. O valor da operação deve ser um dos seguintes valores:

- `INSERT` – insira dados como um novo item na tabela do DynamoDB. Este é o valor padrão.
- `UPDATE` – atualize um item existente da tabela do DynamoDB com novos dados.
- `DELETE`: exclua um item existente da tabela do DynamoDB.

payloadField

(Opcional) O nome da coluna do DynamoDB que recebe a carga útil da ação. O nome padrão é `payload`. Também é possível especificar uma expressão.

payload

(Opcional) A carga útil da ação padrão contém todos os pares de atributo-valor que têm as informações sobre a instância do modelo de detector e o evento que acionou a ação. Também

é possível personalizar a carga útil. Para obter mais informações, consulte [Carga útil](#) na Referência da API do AWS IoT Events .

Se o tipo de carga útil definido for uma string, `DynamoDBAction` grava dados que não estão no formato JSON na tabela do DynamoDB como dados binários. O console do DynamoDB exibe os dados como texto codificado em Base64. O valor de `payloadField` é `payload-field_raw`. Também é possível especificar uma expressão.

Note

Certifique-se de que a política anexada à sua função AWS IoT Events de serviço conceda a `dynamodb:PutItem` permissão. Para obter mais informações, consulte [Gerenciamento de identidade e acesso para AWS IoT Events](#).

Para obter mais informações, consulte [Dynamo DBAction](#) na Referência da AWS IoT Events API.

Amazon DynamoDB(v2)

DynamoDBv2 action

A ação do Amazon DynamoDB(v2) permite gravar dados em uma tabela do DynamoDB. Uma coluna separada da tabela do DynamoDB recebe um par de atributo-valor na carga útil que você especifica. Para obter a lista de regiões compatíveis, consulte [Amazon DynamoDB endpoints e cotas do](#) no Referência geral da Amazon Web Services.

O Amazon DynamoDB é um serviço de banco de dados NoSQL totalmente gerenciado que fornece uma performance rápida e previsível com escalabilidade integrada. Para obter mais informações, consulte [O que é DynamoDB?](#) no Guia do desenvolvedor Amazon DynamoDB.

More information (2)

Ao enviar dados para várias colunas de uma tabela do DynamoDB, você deve especificar os parâmetros a seguir.

tableName

O nome da tabela do DynamoDB que recebe os dados. Também é possível especificar uma expressão.

payload

(Opcional) A carga útil da ação padrão contém todos os pares de atributo-valor que têm as informações sobre a instância do modelo de detector e o evento que acionou a ação. Também é possível personalizar a carga útil. Para obter mais informações, consulte [Carga útil](#) na Referência da API do AWS IoT Events .

Important

O tipo de carga útil deve ser JSON. Também é possível especificar uma expressão.

Note

Certifique-se de que a política anexada à sua função AWS IoT Events de serviço conceda a `dynamodb:PutItem` permissão. Para obter mais informações, consulte [Gerenciamento de identidade e acesso para AWS IoT Events](#).

Para obter mais informações, consulte [Dynamo DBv2 Action](#) na Referência da AWS IoT Events API.

Amazon Data Firehose

Firehose action

A ação Amazon Data Firehose permite que você envie dados para um stream de distribuição do Firehose. Para ver a lista de regiões suportadas, consulte os [endpoints e cotas do Amazon Data Firehose](#) no. Referência geral da Amazon Web Services

O Amazon Data Firehose é um serviço totalmente gerenciado para fornecer dados de streaming em tempo real para destinos como Amazon Simple Storage Service (Amazon Simple Storage Service), Amazon Redshift, OpenSearch Amazon OpenSearch Service (Service) e Splunk. Para obter mais informações, consulte [O que é o Amazon Data Firehose?](#) no Guia do desenvolvedor do Amazon Data Firehose.

More information (3)

Ao enviar dados para um stream de distribuição do Firehose, você deve especificar os seguintes parâmetros.

deliveryStreamName

O nome do stream de entrega do Firehose que recebe os dados.

separator

(Opcional) Você pode usar um separador de caracteres para separar dados contínuos enviados para o stream de distribuição do Firehose. O valor do separador deve ser '\n' (nova linha), '\t' (guia), '\r\n' (nova linha do Windows) ou ',' (vírgula).

payload

(Opcional) A carga útil da ação padrão contém todos os pares de atributo-valor que têm as informações sobre a instância do modelo de detector e o evento que acionou a ação. Também é possível personalizar a carga útil. Para obter mais informações, consulte [Carga útil](#) na Referência da API do AWS IoT Events .

Note

Certifique-se de que a política anexada à sua função AWS IoT Events de serviço conceda a `firehose:PutRecord` permissão. Para obter mais informações, consulte [Gerenciamento de identidade e acesso para AWS IoT Events](#).

Para obter mais informações, consulte [FirehoseAction](#) na Referência de APIs do AWS IoT Events .

AWS Lambda

Lambda action

A AWS Lambda ação permite que você chame uma função Lambda. Para obter a lista de regiões compatíveis, consulte [Endpoints e cotas do AWS Lambda](#) no Referência geral da Amazon Web Services.

AWS Lambda é um serviço de computação que permite executar código sem provisionar ou gerenciar servidores. Para obter mais informações, consulte [O que é AWS Lambda?](#) no Guia do AWS Lambda desenvolvedor.

More information (2)

Ao chamar uma função do Lambda, você deve especificar os seguintes parâmetros.

functionArn

O ARN da função do Lambda a ser chamada.

payload

(Opcional) A carga útil da ação padrão contém todos os pares de atributo-valor que têm as informações sobre a instância do modelo de detector e o evento que acionou a ação. Também é possível personalizar a carga útil. Para obter mais informações, consulte [Carga útil](#) na Referência da API do AWS IoT Events .

Note

Certifique-se de que a política anexada à sua função AWS IoT Events de serviço conceda a `lambda:InvokeFunction` permissão. Para obter mais informações, consulte [Gerenciamento de identidade e acesso para AWS IoT Events](#).

Para obter mais informações, consulte [LambdaAction](#) na Referência de APIs do AWS IoT Events .


Amazon Simple Notification Service

SNS action

A ação de publicação de tópicos do Amazon SNS permite que você publique uma mensagem do Amazon SNS. Para obter a lista de regiões compatíveis, consulte [Amazon Simple Notification Service endpoints e cotas](#) no Referência geral da Amazon Web Services.

O Amazon Simple Notification Service (Amazon Simple Notification Service) é um serviço da Web que coordena e gerencia a entrega ou o envio de mensagens para endpoints ou clientes inscritos.

Para obter mais informações, consulte [O que é o Amazon SNS?](#) no Guia do desenvolvedor do Amazon Simple Notification Service.

 Note

A ação de publicação de tópicos do Amazon SNS não é compatível com tópicos Amazon SNS FIFO (primeiro a entrar, primeiro a sair). Como o mecanismo de regras é um serviço totalmente distribuído, as mensagens podem não ser exibidas em uma ordem especificada quando a ação do Amazon SNS é iniciada.

More information (2)


Ao publicar uma mensagem do Amazon SNS, você deve especificar os seguintes parâmetros.

targetArn

O ARN do destino do Amazon SNS que recebe a mensagem.

payload

(Opcional) A carga útil da ação padrão contém todos os pares de atributo-valor que têm as informações sobre a instância do modelo de detector e o evento que acionou a ação. Também é possível personalizar a carga útil. Para obter mais informações, consulte [Carga útil na Referência da API do AWS IoT Events](#) .

 Note

Certifique-se de que a política anexada à sua função AWS IoT Events de serviço conceda a `sns:Publish` permissão. Para obter mais informações, consulte [Gerenciamento de identidade e acesso para AWS IoT Events](#).

Para obter mais informações, consulte [SNSTopicPublishAction](#) na Referência de APIs do AWS IoT Events .

Amazon Simple Queue Service

SQS action

A ação do Amazon SQS permite enviar dados para uma fila do Amazon SQS. Para obter a lista de regiões compatíveis, consulte [Amazon Simple Queue Service endpoints e cotas](#) no Referência geral da Amazon Web Services.

O Amazon Simple Queue Service (Amazon SQS) oferece uma fila hospedada segura, durável e disponível que permite integrar e desacoplar sistemas de software e componentes distribuídos. Para obter mais informações, consulte [O que é o Simple Queue Service?](#) no Guia do desenvolvedor do Amazon Simple Queue Service.

Note

A ação do Amazon SQS não oferece suporte a tópicos >Amazon SQS FIFO (primeiro a entrar, primeiro a sair). Como o mecanismo de regras é um serviço totalmente distribuído, as mensagens podem não ser exibidas em uma ordem especificada quando a ação do Amazon SQS é iniciada.

More information (3)

Ao enviar dados para uma fila do Amazon SQS, você deve especificar os seguintes parâmetros.

queueUrl

A URL da fila do Amazon SQS que recebe os dados.

useBase64

(Opcional) AWS IoT Events codifica os dados em texto Base64, se você especificar. TRUE O padrão é FALSE.

payload

(Opcional) A carga útil da ação padrão contém todos os pares de atributo-valor que têm as informações sobre a instância do modelo de detector e o evento que acionou a ação. Também é possível personalizar a carga útil. Para obter mais informações, consulte [Carga útil](#) na Referência da API do AWS IoT Events .

Note

Certifique-se de que a política anexada à sua função AWS IoT Events de serviço conceda a `sqs:SendMessage` permissão. Para obter mais informações, consulte [Gerenciamento de identidade e acesso para AWS IoT Events](#).

Para obter mais informações, consulte [SNSTopicPublishAction](#) na Referência de APIs do AWS IoT Events .

Você também pode usar o Amazon SNS e o mecanismo de AWS IoT Core regras para acionar uma AWS Lambda função. Isso possibilita realizar ações usando outros serviços, como o Amazon Connect, ou até mesmo um aplicativo de planejamento de recursos empresariais (ERP) da empresa.

Note

Para coletar e processar grandes fluxos de registros de dados em tempo real, você pode usar outros AWS serviços, como o [Amazon Kinesis](#). A partir daí, você pode concluir uma análise inicial e, em seguida, enviar os resultados AWS IoT Events como entrada para um detector.

Expressões para filtrar, transformar e processar dados de eventos

As expressões são usadas para avaliar dados recebidos, realizar cálculos e determinar as condições sob as quais ações específicas ou transições de estado devem ocorrer. AWS IoT Events fornece várias maneiras de especificar valores ao criar e atualizar modelos de detectores. Você pode usar expressões para especificar valores literais ou AWS IoT Events avaliar as expressões antes de especificar valores específicos.

Tópicos

- [Sintaxe para filtrar dados do dispositivo e definir ações no AWS IoT Events](#)
- [Exemplos de expressões e uso para AWS IoT Events](#)

Sintaxe para filtrar dados do dispositivo e definir ações no AWS IoT Events

As expressões oferecem sintaxe para filtrar dados do dispositivo e definir ações. É possível usar literais, operadores, funções, referências e modelos de substituição nas expressões do AWS IoT Events . Ao combinar esses componentes, você pode criar expressões poderosas e flexíveis para processar dados de IoT, realizar cálculos, manipular cadeias de caracteres e tomar decisões lógicas em seus modelos de detectores.

Literais

- Inteiro
- Decimal
- String
- Booleano

Operadores

Unário

- Não (booleano): !

- Não (bitwise): \sim
- Menos (aritmética): $-$

String

- Concatenação: $+$

Ambos os operandos devem ser strings. Literais de string devem estar entre aspas simples (').

Por exemplo: `'my' + 'string' -> 'mystring'`

Aritmética

- Adição: $+$

Ambos os operandos devem ser numéricos.

- Subtração: $-$
- Divisão: $/$

O resultado da divisão é um valor inteiro arredondado, a menos que pelo menos um dos operandos (divisor ou dividendo) seja um valor decimal.

- Multiplicação: $*$

Bit a bit (inteiro)

- OU: $|$

Por exemplo: `13 | 5 -> 13`

- E: $\&$

Por exemplo: `13 & 5 -> 5`

- XOR: \wedge

Por exemplo: `13 ^ 5 -> 8`


- NÃO: \sim

Por exemplo: `\sim 13 -> -14`

Booleano

- Menor que: $<$
- Menor ou igual a: $<=$
- Igual a: $==$

- Não igual a: `!=`
- Maior ou igual a: `>=`
- Maior que: `>`
- E: `&&`
- OU: `||`

 Note

Quando uma subexpressão de `||` contém dados indefinidos, essa subexpressão é tratada como `false`.

Parênteses

É possível usar parênteses para agrupar termos em uma expressão.

Funções para usar em AWS IoT Events expressões

AWS IoT Events fornece um conjunto de funções integradas para aprimorar os recursos das expressões do seu modelo de detector. Essas funções permitem gerenciamento de cronômetro, conversão de tipo, verificação de nulos, identificação do tipo de gatilho, verificação de entrada, manipulação de seqüências de caracteres e operações bit a bit. Ao aproveitar essas funções, você pode criar uma lógica de AWS IoT Events processamento responsiva, melhorando a eficácia geral de seus aplicativos de IoT.

Funções incorporadas


`timeout("timer-name")`

Será avaliado como `true` se o temporizador especificado expirou. Substitua `timer-name` pelo nome de um cronômetro que você definiu, entre aspas. Em uma ação de evento, é possível definir um temporizador e, em seguida, iniciá-lo, reiniciá-lo ou limpar um que você definiu anteriormente. Consulte o `campodetectorModelDefinition.states.onInput|onEnter|onExit.events.actions.setTimer.timerName`.

Um temporizador definido em um estado pode ser referenciado em um estado diferente. Você deve visitar o estado em que criou o temporizador antes de entrar no estado em que o temporizador é referenciado.

Por exemplo, um modelo de detector tem dois estados, `TemperatureChecked` e `RecordUpdated`. Você criou um cronômetro no `TemperatureChecked` estado. Você deve visitar o `TemperatureChecked` estado primeiro antes de poder usar o cronômetro no `RecordUpdated` estado.

Para garantir a precisão, o tempo mínimo que um temporizador deve ser configurado é de 60 segundos.

 Note

`timeout()` retorna `true` somente na primeira vez em que é verificado após a expiração real do temporizador e retorna `false` depois disso.

convert(*type*, *expression*)

Será avaliado como o valor da expressão convertida para o tipo especificado. O *type* valor deve ser `StringBoolean`, ou `Decimal`. Use uma dessas palavras-chave ou uma expressão que seja avaliada como uma string contendo a palavra-chave. Somente as seguintes conversões são bem-sucedidas e retornam um valor válido:

- Booleano -> string

Retorna uma string "true" ou "false".

- Decimal -> string
- String -> booleano
- String -> decimal

A string especificada deve ser uma representação válida de um número decimal ou `convert()` falhará.

Se `convert()` não retornar um valor válido, a expressão da qual ele faz parte também é inválida. Esse resultado é equivalente a `false` e não acionará `actions` ou a transição para o `nextState` especificado como parte do evento no qual a expressão ocorre.

isNull(*expression*)

Será avaliado como `true` se a expressão retornar para `NULL`. Por exemplo, se a entrada `MyInput` receber a mensagem `{ "a": null }`, o seguinte será avaliado como `true`, mas `isUndefined($input.MyInput.a)` será avaliado como `false`.

```
isNull($input.MyInput.a)
```

isUndefined(*expression*)

Será avaliado como `true` se a expressão for indefinida. Por exemplo, se a entrada `MyInput` receber a mensagem `{ "a": null }`, o seguinte será avaliado como `true`, mas `isNull($input.MyInput.a)` será avaliado como `false`.

```
isUndefined($input.MyInput.a)
```

triggerType("type")

O valor de *type* pode ser `"Message"` ou `"Timer"`. Será avaliado como `true` se a condição do evento em que ele aparece está sendo avaliada porque um temporizador expirou, como no exemplo a seguir.

```
triggerType("Timer")
```

Ou uma mensagem de entrada foi recebida.

```
triggerType("Message")
```

currentInput("input")

Será avaliado como `true` se a condição do evento em que ele aparece está sendo avaliada porque a mensagem de entrada especificada foi recebida. Por exemplo, se a entrada `Command` receber a mensagem `{ "value": "Abort" }`, o seguinte será avaliado como `true`.

```
currentInput("Command")
```

Use essa função para verificar se a condição está sendo avaliada porque uma entrada específica foi recebida e um temporizador não expirou, como na expressão a seguir.

```
currentInput("Command") && $input.Command.value == "Abort"
```

Funções de correspondência de strings

startsWith(*expression1*, *expression2*)

Será avaliado como `true` se a primeira expressão de string começa com a segunda expressão de string. Por exemplo, se a entrada `MyInput` receber a mensagem `{ "status": "offline" }`, a seguinte será avaliada como `true`.

```
startsWith($input.MyInput.status, "off")
```

Ambas as expressões devem ser avaliadas para um valor de string. Se uma das expressões não for avaliada como um valor de string, o resultado da função será indefinido. Nenhuma conversão é realizada.

endsWith(*expression1*, *expression2*)

Será avaliado como `true` se a primeira expressão de string termina com a segunda expressão de string. Por exemplo, se a entrada `MyInput` receber a mensagem `{ "status": "offline" }`, a seguinte será avaliada como `true`.

```
endsWith($input.MyInput.status, "line")
```

Ambas as expressões devem ser avaliadas para um valor de string. Se uma das expressões não for avaliada como um valor de string, o resultado da função será indefinido. Nenhuma conversão é realizada.

contains(*expression1*, *expression2*)

Será avaliado como `true` se a primeira expressão de string contém a segunda expressão de string. Por exemplo, se a entrada `MyInput` receber a mensagem `{ "status": "offline" }`, a seguinte será avaliada como `true`.

```
contains($input.MyInput.value, "fli")
```

Ambas as expressões devem ser avaliadas para um valor de string. Se uma das expressões não for avaliada como um valor de string, o resultado da função será indefinido. Nenhuma conversão é realizada.

Funções de manipulação de números inteiros bit a bit

bitor(*expression1*, *expression2*)

Avalia o OR bit a bit das expressões inteiras (a operação binária OR é executada nos bits correspondentes dos números inteiros). Por exemplo, se a entrada MyInput receber a mensagem { "value1": 13, "value2": 5 }, a seguinte será avaliada como 13.

```
bitor($input.MyInput.value1, $input.MyInput.value2)
```

Ambas as expressões devem ser avaliadas como um valor inteiro. Se uma das expressões não for avaliada como um valor inteiro, o resultado da função será indefinido. Nenhuma conversão é realizada.

bitand(*expression1*, *expression2*)

Avalia o AND bit a bit das expressões inteiras (a operação binária AND é executada nos bits correspondentes dos números inteiros). Por exemplo, se a entrada MyInput receber a mensagem { "value1": 13, "value2": 5 }, a seguinte será avaliada como 5.

```
bitand($input.MyInput.value1, $input.MyInput.value2)
```

Ambas as expressões devem ser avaliadas como um valor inteiro. Se uma das expressões não for avaliada como um valor inteiro, o resultado da função será indefinido. Nenhuma conversão é realizada.

bitxor(*expression1*, *expression2*)

Avalia o XOR bit a bit das expressões inteiras (a operação binária XOR é executada nos bits correspondentes dos números inteiros). Por exemplo, se a entrada MyInput receber a mensagem { "value1": 13, "value2": 5 }, a seguinte será avaliada como 8.

```
bitxor($input.MyInput.value1, $input.MyInput.value2)
```

Ambas as expressões devem ser avaliadas como um valor inteiro. Se uma das expressões não for avaliada como um valor inteiro, o resultado da função será indefinido. Nenhuma conversão é realizada.

bitnot(*expression*)

Avalia o NOT bit a bit da expressão inteira (a operação binária NOT é executada nos bits do inteiro). Por exemplo, se a entrada MyInput receber a mensagem { "value": 13 }, a seguinte será avaliada como -14.

```
bitnot($input.MyInput.value)
```

Ambas as expressões devem ser avaliadas como um valor inteiro. Se uma das expressões não for avaliada como um valor inteiro, o resultado da função será indefinido. Nenhuma conversão é realizada.

AWS IoT Events referência para entradas e variáveis em expressões

Entradas

`$input.input-name.path-to-data`

`input-name` é uma entrada que você cria usando a [CreateIn](#)putação.

Por exemplo, se você tiver uma entrada nomeada TemperatureInput para a qual definiu entradas `inputDefinition.attributes.jsonPath`, os valores podem aparecer nos seguintes campos disponíveis.

```
{
  "temperature": 78.5,
  "date": "2018-10-03T16:09:09Z"
}
```

Para referenciar o valor do campo `temperature`, use o comando a seguir.

```
$input.TemperatureInput.temperature
```

Para campos cujos valores são matrizes, é possível referenciar membros da matriz usando `[n]`. Por exemplo, considerando os seguintes dados:

```
{
  "temperatures": [
    78.4,
```

```
    77.9,  
    78.8  
  ],  
  "date": "2018-10-03T16:09:09Z"  
}
```

O valor 78.8 pode ser referenciado com o comando a seguir.

```
$input.TemperatureInput.temperatures[2]
```

Variáveis

`$variable.variable-name`

A *variable-name* é uma variável que você definiu usando a [CreateDetectorModelação](#).

Por exemplo, caso tenha uma variável chamada TechnicianID definida usando `detectorDefinition.states.onInputEvents.actions.setVariable.variableName`, é possível referenciar o valor (string) dado mais recentemente à variável com o comando a seguir.

```
$variable.TechnicianID
```

Você pode definir os valores das variáveis somente usando a ação `setVariable`. Não é possível atribuir valores para variáveis em uma expressão. Uma variável não pode ser desativada. Por exemplo, não é possível atribuir o valor `null` a ela.

Note

Nas referências que usam identificadores que não seguem o padrão (expressão regular) `[a-zA-Z][a-zA-Z0-9_]*`, você deve colocar esses identificadores em acentos graves (```). Por exemplo, uma referência a uma entrada nomeada MyInput com um campo chamado `_value` deve especificar esse campo como `$input.MyInput.`_value``.

Ao usar referências em expressões, verifique o seguinte:

- Ao usar uma referência como operando com um ou mais operadores, verifique se todos os tipos de dados referenciados são compatíveis.

Por exemplo, na expressão a seguir, o inteiro 2 é um operando dos operadores == e &&. Para garantir que os operandos sejam compatíveis, `$variable.testVariable + 1` e `$variable.testVariable` devem referenciar um número inteiro ou decimal.

Além disso, o inteiro 1 é um operando do operador +. Portanto, `$variable.testVariable` deve fazer referência a um número inteiro ou decimal.

```
'$variable.testVariable + 1 == 2 && $variable.testVariable'
```

- Ao usar uma referência como argumento passado para uma função, verifique se a função é compatível com os tipos de dados aos quais você faz referência.

Por exemplo, a função a seguir `timeout("time-name")` requer uma string com aspas duplas como argumento. Se você usar uma referência para o `timer-name` valor, deverá referenciar uma string com aspas duplas.

```
timeout("timer-name")
```

Note

Para a `convert(type, expression)` função, se você usar uma referência para o `type` valor, o resultado avaliado da sua referência deverá ser `String`, `Decimal`, ou `Boolean`.

AWS IoT Events as expressões oferecem suporte aos tipos de dados inteiros, decimais, strings e booleanos. A tabela a seguir fornece uma lista de pares de tipos incompatíveis.

Pares de tipos incompatíveis

Inteiro, string

Inteiro, booleano

Decimal, sequência

Decimal, booleano

String, booleano

Modelos de substituição para expressões AWS IoT Events

'\${*expression*}'

O `${}` identifica a string como uma string interpolada. *expression* pode ser qualquer AWS IoT Events expressão. Isso inclui Operadores, Funções e Referências.

Por exemplo, você usou a [SetVariableAction](#) para definir uma variável. O `variableName` é `SensorID`, e o `value` é `10`. É possível criar os seguintes modelos de substituição.

Modelo de substituição	String de resultados
'\${'Sensor ' + \$variable.SensorID}'	"Sensor 10"
'Sensor ' + '\${\$variable.SensorID + 1}'	"Sensor 11"
'Sensor 10: \${\$variable.SensorID == 10}'	"Sensor 10: true"
'{"sensor\":"\${\$variable.SensorID + 1}\}'	"{"sensor\":"11\"}"
'{"sensor\":"\${\$variable.SensorID + 1}}'	"{"sensor\":"11}"

Exemplos de expressões e uso para AWS IoT Events

É possível especificar valores em um modelo de detector das seguintes maneiras:

- Insira as expressões suportadas no AWS IoT Events console.
- Passe as expressões para o AWS IoT Events APIs como parâmetros.

As expressões são compatíveis com modelos literais, operadores, funções, referências e de substituição.

⚠ Important

Suas expressões devem fazer referência a um valor inteiro, decimal, string ou booleano.

Escrevendo AWS IoT Events expressões

Veja os exemplos a seguir para ajudá-lo a escrever suas AWS IoT Events expressões:

Literal

Para valores literais, as expressões devem conter aspas simples. Um valor booleano deve ser `true` ou `false`.

```
'123'      # Integer
'123.12'   # Decimal
'hello'    # String
'true'     # Boolean
```

Referência

Para referências, é necessário especificar variáveis ou valores de entrada.

- A entrada a seguir faz referência a um número decimal, `10.01`.

```
$input.GreenhouseInput.temperature
```

- A variável a seguir faz referência a uma string, `Greenhouse Temperature Table`.

```
$variable.TableName
```

Modelo de substituição

Para um modelo de substituição, é necessário usar `${}`, e o modelo deve estar entre aspas simples. Um modelo de substituição também pode conter uma combinação de literais, operadores, funções, referências e modelos de substituição.

- O resultado avaliado da expressão a seguir é uma string, `50.018 in Fahrenheit`.

```
'${$input.GreenhouseInput.temperature * 9 / 5 + 32} in Fahrenheit'
```

- O resultado avaliado da expressão a seguir é uma string, `{"sensor_id\":\"Sensor_1\", \"temperature\":\"50.018\"}`.

```
'{"sensor_id\":\"${$input.GreenhouseInput.sensors[0].sensor1}\", \"temperature\": \"${$input.GreenhouseInput.temperature*9/5+32}\"}'
```

Concatenação de strings

Para uma concatenação de string, use `+`. Uma concatenação de string também pode conter uma combinação de literais, operadores, funções, referências e modelos de substituição.

- O resultado avaliado da expressão a seguir é uma string, `Greenhouse Temperature Table 2000-01-01`.

```
'Greenhouse Temperature Table ' + $input.GreenhouseInput.date
```

AWS IoT Events exemplos de modelos de detectores

Esta página fornece uma lista de exemplos de casos de uso que demonstram como configurar vários AWS IoT Events recursos. Os exemplos variam de detecções básicas, como limites de temperatura, até cenários mais avançados de detecção de anomalias e aprendizado de máquina. Cada exemplo inclui procedimentos e trechos de código para ajudá-lo a configurar AWS IoT Events detecções, ações e integrações. Esses exemplos mostram a flexibilidade do AWS IoT Events serviço e como ele pode ser personalizado para diversos aplicativos e casos de uso de IoT. Consulte esta página ao explorar os AWS IoT Events recursos ou se precisar de orientação para implementar um fluxo de trabalho específico de detecção ou automação.

Tópicos

- [Exemplo: Usando o controle de temperatura HVAC com AWS IoT Events](#)
- [Exemplo: Um guindaste detectando condições usando AWS IoT Events](#)
- [Envie comandos em resposta às condições detectadas no AWS IoT Events](#)
- [Um modelo AWS IoT Events de detector para monitoramento de guindastes](#)
- [AWS IoT Events entradas para monitoramento de guindastes](#)
- [Envie mensagens operacionais e de alarme com AWS IoT Events](#)
- [Exemplo: detecção de AWS IoT Events eventos com sensores e aplicativos](#)
- [Exemplo: dispositivo HeartBeat para monitorar as conexões do dispositivo com AWS IoT Events](#)
- [Exemplo: Um alarme ISA em AWS IoT Events](#)
- [Exemplo: Crie um alarme simples com AWS IoT Events](#)

Exemplo: Usando o controle de temperatura HVAC com AWS IoT Events

História de fundo

Este exemplo implementa um modelo de controle de temperatura (um termostato) com os seguintes atributos:

- Um modelo de detector definido por você que pode monitorar e controlar várias áreas. (Uma instância de detector será criada para cada área).

- Cada instância do detector recebe dados de temperatura de vários sensores colocados em cada área de controle.
- Você pode alterar a temperatura desejada (o ponto de ajuste) para cada área a qualquer momento.
- Você pode definir os parâmetros operacionais para cada área e alterá-los a qualquer momento.
- Você pode adicionar ou excluir sensores de uma área a qualquer momento.
- Você pode ativar um tempo mínimo de funcionamento das unidades de aquecimento e resfriamento para protegê-las contra danos.
- Os detectores rejeitarão e reportarão leituras anômalas do sensor.
- Você pode definir pontos de ajuste de temperatura de emergência. Se algum sensor relatar uma temperatura acima ou abaixo dos pontos de ajuste que você definiu, as unidades de aquecimento ou resfriamento serão acionadas imediatamente e o detector relatará esse pico de temperatura.

Este exemplo demonstra os seguintes recursos funcionais:

- Crie modelos de detectores de eventos.
- Cria entradas.
- Ingira entradas em um modelo de detector.
- Avalie as condições do gatilho.
- Consulte as variáveis de estado em condições e defina os valores das variáveis dependendo das condições.
- Consulte os temporizadores em condições e defina-os de acordo com as condições.
- Execute ações que enviem mensagens do Amazon SNS e MQTT.

Definições de entrada para um sistema HVAC em AWS IoT Events

Uma `seedTemperatureInput` é usada para criar uma instância de detector para uma área e definir seus parâmetros operacionais.

Configurar entradas para sistemas HVAC em AWS IoT Events é importante para um controle climático eficaz. Este exemplo mostra como configurar entradas que capturam parâmetros como temperatura, umidade, ocupação e dados de consumo de energia. Aprenda a definir atributos de entrada, configurar fontes de dados e configurar regras de pré-processamento para ajudar seus

modelos de detectores a receber informações precisas e oportunas para gerenciamento e eficiência ideais.

Comando CLI usado:

```
aws iotevents create-input --cli-input-json file://seedInput.json
```

Arquivo: seedInput.json

```
{
  "inputName": "seedTemperatureInput",
  "inputDescription": "Temperature seed values.",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "areaId" },
      { "jsonPath": "desiredTemperature" },
      { "jsonPath": "allowedError" },
      { "jsonPath": "rangeHigh" },
      { "jsonPath": "rangeLow" },
      { "jsonPath": "anomalousHigh" },
      { "jsonPath": "anomalousLow" },
      { "jsonPath": "sensorCount" },
      { "jsonPath": "noDelay" }
    ]
  }
}
```

Resposta:

```
{
  "inputConfiguration": {
    "status": "ACTIVE",
    "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/seedTemperatureInput",
    "lastUpdateTime": 1557519620.736,
    "creationTime": 1557519620.736,
    "inputName": "seedTemperatureInput",
    "inputDescription": "Temperature seed values."
  }
}
```

Uma `temperatureInput` deve ser enviada por cada sensor em cada área, conforme necessário.

Comando CLI usado:

```
aws iotevents create-input --cli-input-json file://temperatureInput.json
```

Arquivo: `temperatureInput.json`

```
{
  "inputName": "temperatureInput",
  "inputDescription": "Temperature sensor unit data.",
  "inputDefinition": {
    "attributes": [
      { "jsonPath": "sensorId" },
      { "jsonPath": "areaId" },
      { "jsonPath": "sensorData.temperature" }
    ]
  }
}
```

Resposta:

```
{
  "inputConfiguration": {
    "status": "ACTIVE",
    "inputArn": "arn:aws:iotevents:us-west-2:123456789012:input/temperatureInput",
    "lastUpdateTime": 1557519707.399,
    "creationTime": 1557519707.399,
    "inputName": "temperatureInput",
    "inputDescription": "Temperature sensor unit data."
  }
}
```

Definição do modelo de detector para um sistema HVAC usando AWS IoT Events

O `areaDetectorModel` define como cada instância do detector funciona. Cada instância `state machine` ingerirá as leituras do sensor de temperatura, depois mudará de estado e enviará mensagens de controle, dependendo dessas leituras.

Comando CLI usado:

```
aws iotevents create-detector-model --cli-input-json file://areaDetectorModel.json
```

Arquivo: areaDetectorModel.json

```
{
  "detectorModelName": "areaDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "start",
        "onEnter": {
          "events": [
            {
              "eventName": "prepare",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "sensorId",
                    "value": "0"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "reportedTemperature",
                    "value": "0.1"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "resetMe",
                    "value": "false"
                  }
                }
              ]
            }
          ]
        }
      }
    ],
    "onInput": {
      "transitionEvents": [
```

```
{
  "eventName": "initialize",
  "condition": "$input.seedTemperatureInput.sensorCount > 0",
  "actions": [
    {
      "setVariable": {
        "variableName": "rangeHigh",
        "value": "$input.seedTemperatureInput.rangeHigh"
      }
    },
    {
      "setVariable": {
        "variableName": "rangeLow",
        "value": "$input.seedTemperatureInput.rangeLow"
      }
    },
    {
      "setVariable": {
        "variableName": "desiredTemperature",
        "value": "$input.seedTemperatureInput.desiredTemperature"
      }
    },
    {
      "setVariable": {
        "variableName": "averageTemperature",
        "value": "$input.seedTemperatureInput.desiredTemperature"
      }
    },
    {
      "setVariable": {
        "variableName": "allowedError",
        "value": "$input.seedTemperatureInput.allowedError"
      }
    },
    {
      "setVariable": {
        "variableName": "anomalousHigh",
        "value": "$input.seedTemperatureInput.anomalousHigh"
      }
    },
    {
      "setVariable": {
        "variableName": "anomalousLow",
        "value": "$input.seedTemperatureInput.anomalousLow"
      }
    }
  ]
}
```

```

    }
  },
  {
    "setVariable": {
      "variableName": "sensorCount",
      "value": "$input.seedTemperatureInput.sensorCount"
    }
  },
  {
    "setVariable": {
      "variableName": "noDelay",
      "value": "$input.seedTemperatureInput.noDelay == true"
    }
  }
],
"nextState": "idle"
},
{
  "eventName": "reset",
  "condition": "($variable.resetMe == true) &&
($input.temperatureInput.sensorData.temperature < $variable.anomalousHigh &&
$input.temperatureInput.sensorData.temperature > $variable.anomalousLow)",
  "actions": [
    {
      "setVariable": {
        "variableName": "averageTemperature",
        "value": "((( $variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
      }
    }
  ],
  "nextState": "idle"
}
]
},
"onExit": {
  "events": [
    {
      "eventName": "resetHeatCool",
      "condition": "true",
      "actions": [
        {
          "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
          }
        }
      ]
    }
  ]
}
}

```

```

    }
  },
  {
    "sns": {
      "targetArn": "arn:aws:sns:us-west-2:123456789012:cool0ff"
    }
  },
  {
    "iotTopicPublish": {
      "mqttTopic": "hvac/Heating/Off"
    }
  },
  {
    "iotTopicPublish": {
      "mqttTopic": "hvac/Cooling/Off"
    }
  }
]
}
]
}
},

{
  "stateName": "idle",
  "onInput": {
    "events": [
      {
        "eventName": "whatWasInput",
        "condition": "true",
        "actions": [
          {
            "setVariable": {
              "variableName": "sensorId",
              "value": "$input.temperatureInput.sensorId"
            }
          },
          {
            "setVariable": {
              "variableName": "reportedTemperature",
              "value": "$input.temperatureInput.sensorData.temperature"
            }
          }
        ]
      }
    ]
  }
}

```

```
    ]
  },
  {
    "eventName": "changeDesired",
    "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
    "actions": [
      {
        "setVariable": {
          "variableName": "desiredTemperature",
          "value": "$input.seedTemperatureInput.desiredTemperature"
        }
      }
    ]
  },
  {
    "eventName": "calculateAverage",
    "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
    "actions": [
      {
        "setVariable": {
          "variableName": "averageTemperature",
          "value": "((( $variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
        }
      }
    ]
  }
],
"transitionEvents": [
  {
    "eventName": "anomalousInputArrived",
    "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
    "actions": [
      {
        "iotTopicPublish": {
          "mqttTopic": "temperatureSensor/anomaly"
        }
      }
    ]
  }
],
```

```
    "nextState": "idle"
  },

  {
    "eventName": "highTemperatureSpike",
    "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
    "actions": [
      {
        "iotTopicPublish": {
          "mqttTopic": "temperatureSensor/spike"
        }
      },
      {
        "sns": {
          "targetArn": "arn:aws:sns:us-west-2:123456789012:cool0n"
        }
      },
      {
        "iotTopicPublish": {
          "mqttTopic": "hvac/Cooling/On"
        }
      },
      {
        "setVariable": {
          "variableName": "enteringNewState",
          "value": "true"
        }
      }
    ],
    "nextState": "cooling"
  },

  {
    "eventName": "lowTemperatureSpike",
    "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
    "actions": [
      {
        "iotTopicPublish": {
          "mqttTopic": "temperatureSensor/spike"
        }
      },
      {

```

```

        "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOn"
        }
    },
    {
        "iotTopicPublish": {
            "mqttTopic": "hvac/Heating/On"
        }
    },
    {
        "setVariable": {
            "variableName": "enteringNewState",
            "value": "true"
        }
    }
],
"nextState": "heating"
},
{
    "eventName": "highTemperatureThreshold",
    "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) >
($variable.desiredTemperature + $variable.allowedError))",
    "actions": [
        {
            "sns": {
                "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOn"
            }
        },
        {
            "iotTopicPublish": {
                "mqttTopic": "hvac/Cooling/On"
            }
        }
    ],
    {
        "setVariable": {
            "variableName": "enteringNewState",
            "value": "true"
        }
    }
],
"nextState": "cooling"
},

```

```
    {
      "eventName": "lowTemperatureThreshold",
      "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) <
($variable.desiredTemperature - $variable.allowedError))",
      "actions": [
        {
          "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOn"
          }
        },
        {
          "iotTopicPublish": {
            "mqttTopic": "hvac/Heating/On"
          }
        },
        {
          "setVariable": {
            "variableName": "enteringNewState",
            "value": "true"
          }
        }
      ],
      "nextState": "heating"
    }
  ]
},

{
  "stateName": "cooling",
  "onEnter": {
    "events": [
      {
        "eventName": "delay",
        "condition": "!$variable.noDelay && $variable.enteringNewState",
        "actions": [
          {
            "setTimer": {
              "timerName": "coolingTimer",
              "seconds": 180
            }
          }
        ]
      }
    ]
  }
}
```

```
    },
    {
      "setVariable": {
        "variableName": "goodToGo",
        "value": "false"
      }
    }
  ]
},
{
  "eventName": "dontDelay",
  "condition": "$variable.noDelay == true",
  "actions": [
    {
      "setVariable": {
        "variableName": "goodToGo",
        "value": "true"
      }
    }
  ]
},
{
  "eventName": "beenHere",
  "condition": "true",
  "actions": [
    {
      "setVariable": {
        "variableName": "enteringNewState",
        "value": "false"
      }
    }
  ]
}
],
},

"onInput": {
  "events": [
    {
      "eventName": "whatWasInput",
      "condition": "true",
      "actions": [
        {
          "setVariable": {
```

```

        "variableName": "sensorId",
        "value": "$input.temperatureInput.sensorId"
    }
  },
  {
    "setVariable": {
      "variableName": "reportedTemperature",
      "value": "$input.temperatureInput.sensorData.temperature"
    }
  }
]
},
{
  "eventName": "changeDesired",
  "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
  "actions": [
    {
      "setVariable": {
        "variableName": "desiredTemperature",
        "value": "$input.seedTemperatureInput.desiredTemperature"
      }
    }
  ]
},
{
  "eventName": "calculateAverage",
  "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
  "actions": [
    {
      "setVariable": {
        "variableName": "averageTemperature",
        "value": "((( $variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
      }
    }
  ]
},
{
  "eventName": "areWeThereYet",
  "condition": "(timeout(\"coolingTimer\"))",
  "actions": [

```

```

        {
            "setVariable": {
                "variableName": "goodToGo",
                "value": "true"
            }
        }
    ],
    "transitionEvents": [
        {
            "eventName": "anomalousInputArrived",
            "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
            "actions": [
                {
                    "iotTopicPublish": {
                        "mqttTopic": "temperatureSensor/anomaly"
                    }
                }
            ],
            "nextState": "cooling"
        },
        {
            "eventName": "highTemperatureSpike",
            "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
            "actions": [
                {
                    "iotTopicPublish": {
                        "mqttTopic": "temperatureSensor/spike"
                    }
                }
            ],
            "nextState": "cooling"
        },
        {
            "eventName": "lowTemperatureSpike",
            "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
            "actions": [

```

```

    {
      "iotTopicPublish": {
        "mqttTopic": "temperatureSensor/spike"
      }
    },
    {
      "sns": {
        "targetArn": "arn:aws:sns:us-west-2:123456789012:cool0ff"
      }
    },
    {
      "sns": {
        "targetArn": "arn:aws:sns:us-west-2:123456789012:heat0n"
      }
    },
    {
      "iotTopicPublish": {
        "mqttTopic": "hvac/Cooling/Off"
      }
    },
    {
      "iotTopicPublish": {
        "mqttTopic": "hvac/Heating/On"
      }
    },
    {
      "setVariable": {
        "variableName": "enteringNewState",
        "value": "true"
      }
    }
  ],
  "nextState": "heating"
},

{
  "eventName": "desiredTemperature",
  "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) <=
($variable.desiredTemperature - $variable.allowedError)) && $variable.goodToGo ==
true",
  "actions": [
    {
      "sns": {

```

```

        "targetArn": "arn:aws:sns:us-west-2:123456789012:cool10ff"
      }
    },
    {
      "iotTopicPublish": {
        "mqttTopic": "hvac/Cooling/Off"
      }
    }
  ],
  "nextState": "idle"
}
]
}
},

{
  "stateName": "heating",
  "onEnter": {
    "events": [
      {
        "eventName": "delay",
        "condition": "!$variable.noDelay && $variable.enteringNewState",
        "actions": [
          {
            "setTimer": {
              "timerName": "heatingTimer",
              "seconds": 120
            }
          },
          {
            "setVariable": {
              "variableName": "goodToGo",
              "value": "false"
            }
          }
        ]
      },
      {
        "eventName": "dontDelay",
        "condition": "$variable.noDelay == true",
        "actions": [
          {
            "setVariable": {

```

```
        "variableName": "goodToGo",
        "value": "true"
      }
    ]
  },
  {
    "eventName": "beenHere",
    "condition": "true",
    "actions": [
      {
        "setVariable": {
          "variableName": "enteringNewState",
          "value": "false"
        }
      }
    ]
  }
]
},
"onInput": {
  "events": [
    {
      "eventName": "whatWasInput",
      "condition": "true",
      "actions": [
        {
          "setVariable": {
            "variableName": "sensorId",
            "value": "$input.temperatureInput.sensorId"
          }
        },
        {
          "setVariable": {
            "variableName": "reportedTemperature",
            "value": "$input.temperatureInput.sensorData.temperature"
          }
        }
      ]
    },
    {
      "eventName": "changeDesired",
```

```

        "condition": "$input.seedTemperatureInput.desiredTemperature !=
$variable.desiredTemperature",
        "actions": [
            {
                "setVariable": {
                    "variableName": "desiredTemperature",
                    "value": "$input.seedTemperatureInput.desiredTemperature"
                }
            }
        ],
    },
    {
        "eventName": "calculateAverage",
        "condition": "$input.temperatureInput.sensorData.temperature <
$variable.anomalousHigh && $input.temperatureInput.sensorData.temperature >
$variable.anomalousLow",
        "actions": [
            {
                "setVariable": {
                    "variableName": "averageTemperature",
                    "value": "((( $variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount)"
                }
            }
        ],
    },
    {
        "eventName": "areWeThereYet",
        "condition": "(timeout(\"heatingTimer\"))",
        "actions": [
            {
                "setVariable": {
                    "variableName": "goodToGo",
                    "value": "true"
                }
            }
        ],
    }
],
"transitionEvents": [
    {
        "eventName": "anomalousInputArrived",

```

```
    "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
    "actions": [
      {
        "iotTopicPublish": {
          "mqttTopic": "temperatureSensor/anomaly"
        }
      }
    ],
    "nextState": "heating"
  },
  {
    "eventName": "highTemperatureSpike",
    "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
    "actions": [
      {
        "iotTopicPublish": {
          "mqttTopic": "temperatureSensor/spike"
        }
      },
      {
        "sns": {
          "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
        }
      },
      {
        "sns": {
          "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOn"
        }
      },
      {
        "iotTopicPublish": {
          "mqttTopic": "hvac/Heating/Off"
        }
      },
      {
        "iotTopicPublish": {
          "mqttTopic": "hvac/Cooling/On"
        }
      }
    ]
  }
}
```

```
        "setVariable": {
          "variableName": "enteringNewState",
          "value": "true"
        }
      ],
      "nextState": "cooling"
    },
    {
      "eventName": "lowTemperatureSpike",
      "condition": "$input.temperatureInput.sensorData.temperature <
$variable.rangeLow",
      "actions": [
        {
          "iotTopicPublish": {
            "mqttTopic": "temperatureSensor/spike"
          }
        }
      ],
      "nextState": "heating"
    },
    {
      "eventName": "desiredTemperature",
      "condition": "((((($variable.averageTemperature * ($variable.sensorCount
- 1)) + $input.temperatureInput.sensorData.temperature) / $variable.sensorCount) >=
($variable.desiredTemperature + $variable.allowedError)) && $variable.goodToGo ==
true",
      "actions": [
        {
          "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:heatOff"
          }
        },
        {
          "iotTopicPublish": {
            "mqttTopic": "hvac/Heating/Off"
          }
        }
      ],
      "nextState": "idle"
    }
  ]
}
```

```

    }
  }

],

  "initialStateName": "start"
},
"key": "areaId",
"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole"
}

```

Resposta:

```

{
  "detectorModelConfiguration": {
    "status": "ACTIVATING",
    "lastUpdateTime": 1557523491.168,
    "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
    "creationTime": 1557523491.168,
    "detectorModelArn": "arn:aws:iotevents:us-west-2:123456789012:detectorModel/
areaDetectorModel",
    "key": "areaId",
    "detectorModelName": "areaDetectorModel",
    "detectorModelVersion": "1"
  }
}

```

BatchPutMessage exemplos de um sistema HVAC em AWS IoT Events

Neste exemplo, BatchPutMessage é usado para criar uma instância de detector para uma área e definir os parâmetros operacionais iniciais.

Comando CLI usado:

```
aws iotevents-data batch-put-message --cli-input-json file://seedExample.json --cli-binary-format raw-in-base64-out
```

Arquivo: seedExample.json

```

{
  "messages": [

```

```
{
  "messageId": "00001",
  "inputName": "seedTemperatureInput",
  "payload": "{\"areaId\": \"Area51\", \"desiredTemperature\": 20.0, \"allowedError\": 0.7, \"rangeHigh\": 30.0, \"rangeLow\": 15.0, \"anomalousHigh\": 60.0, \"anomalousLow\": 0.0, \"sensorCount\": 10, \"noDelay\": false}"
}
```

Resposta:

```
{
  "BatchPutMessageErrorEntries": []
}
```

Neste exemplo, BatchPutMessage é usado para relatar as leituras do sensor de temperatura para um único sensor em uma área.

Comando CLI usado:

```
aws iotevents-data batch-put-message --cli-input-json file://temperatureExample.json --cli-binary-format raw-in-base64-out
```

Arquivo: temperatureExample.json

```
{
  "messages": [
    {
      "messageId": "00005",
      "inputName": "temperatureInput",
      "payload": "{\"sensorId\": \"05\", \"areaId\": \"Area51\", \"sensorData\": {\"temperature\": 23.12} }"
    }
  ]
}
```

Resposta:

```
{
  "BatchPutMessageErrorEntries": []
}
```

```
}
```

Neste exemplo, BatchPutMessage é usado para alterar a temperatura desejada para uma área.

Comando CLI usado:

```
aws iotevents-data batch-put-message --cli-input-json file:///seedSetDesiredTemp.json --cli-binary-format raw-in-base64-out
```

Arquivo: seedSetDesiredTemp.json

```
{
  "messages": [
    {
      "messageId": "00001",
      "inputName": "seedTemperatureInput",
      "payload": "{\"areaId\": \"Area51\", \"desiredTemperature\": 23.0}"
    }
  ]
}
```

Resposta:

```
{
  "BatchPutMessageErrorEntries": []
}
```

Exemplos de mensagens do Amazon SNS geradas pela instância do Area51 detector:

```
Heating system off command> {
  "eventTime":1557520274729,
  "payload":{
    "actionExecutionId":"f3159081-bac3-38a4-96f7-74af0940d0a4",
    "detector":{
      "detectorModelName":"areaDetectorModel",
      "keyValue":"Area51",
      "detectorModelVersion":"1"
    },
    "eventTriggerDetails":{
```

```

    "inputName": "seedTemperatureInput",
    "messageId": "00001",
    "triggerType": "Message"
  },
  "state": {
    "stateName": "start",
    "variables": {
      "sensorCount": 10,
      "rangeHigh": 30.0,
      "resetMe": false,
      "enteringNewState": true,
      "averageTemperature": 20.0,
      "rangeLow": 15.0,
      "noDelay": false,
      "allowedError": 0.7,
      "desiredTemperature": 20.0,
      "anomalousHigh": 60.0,
      "reportedTemperature": 0.1,
      "anomalousLow": 0.0,
      "sensorId": 0
    },
    "timers": {}
  }
},
"eventName": "resetHeatCool"
}

```

```

Cooling system off command> {
  "eventTime": 1557520274729,
  "payload": {
    "actionExecutionId": "98f6a1b5-8f40-3cdb-9256-93afd4d66192",
    "detector": {
      "detectorModelName": "areaDetectorModel",
      "keyValue": "Area51",
      "detectorModelVersion": "1"
    },
    "eventTriggerDetails": {
      "inputName": "seedTemperatureInput",
      "messageId": "00001",
      "triggerType": "Message"
    },
    "state": {

```

```

    "stateName": "start",
    "variables": {
      "sensorCount": 10,
      "rangeHigh": 30.0,
      "resetMe": false,
      "enteringNewState": true,
      "averageTemperature": 20.0,
      "rangeLow": 15.0,
      "noDelay": false,
      "allowedError": 0.7,
      "desiredTemperature": 20.0,
      "anomalousHigh": 60.0,
      "reportedTemperature": 0.1,
      "anomalousLow": 0.0,
      "sensorId": 0
    },
    "timers": {}
  }
},
"eventName": "resetHeatCool"
}

```

Neste exemplo, usamos a `DescribeDetector` API para obter informações sobre o estado atual de uma instância do detector.

```
aws iotevents-data describe-detector --detector-model-name areaDetectorModel --key-value Area51
```

Resposta:

```

{
  "detector": {
    "lastUpdateTime": 1557521572.216,
    "creationTime": 1557520274.405,
    "state": {
      "variables": [
        {
          "name": "resetMe",
          "value": "false"
        },
        {
          "name": "rangeLow",
          "value": "15.0"
        }
      ]
    }
  }
}

```

```
    },
    {
      "name": "noDelay",
      "value": "false"
    },
    {
      "name": "desiredTemperature",
      "value": "20.0"
    },
    {
      "name": "anomalousLow",
      "value": "0.0"
    },
    {
      "name": "sensorId",
      "value": "\"01\""
    },
    {
      "name": "sensorCount",
      "value": "10"
    },
    {
      "name": "rangeHigh",
      "value": "30.0"
    },
    {
      "name": "enteringNewState",
      "value": "false"
    },
    {
      "name": "averageTemperature",
      "value": "19.572"
    },
    {
      "name": "allowedError",
      "value": "0.7"
    },
    {
      "name": "anomalousHigh",
      "value": "60.0"
    },
    {
      "name": "reportedTemperature",
      "value": "15.72"
    }
  ],
  "value": "15.72"
}
```

```
        },
        {
            "name": "goodToGo",
            "value": "false"
        }
    ],
    "stateName": "idle",
    "timers": [
        {
            "timestamp": 1557520454.0,
            "name": "idleTimer"
        }
    ]
},
"keyValue": "Area51",
"detectorModelName": "areaDetectorModel",
"detectorModelVersion": "1"
}
}
```

BatchUpdateDetector exemplo de um sistema HVAC em AWS IoT Events

Neste exemplo, `BatchUpdateDetector` é usado para alterar os parâmetros operacionais de uma instância de detector em funcionamento.

O gerenciamento eficiente do sistema HVAC geralmente requer atualizações em lote para vários detectores. Esta seção demonstra como usar o recurso AWS IoT Events de atualização em lote para detectores. Aprenda a modificar simultaneamente vários parâmetros de controle e atualizar os valores limite para que você possa ajustar as ações de resposta em uma frota de dispositivos, melhorando sua capacidade de gerenciar sistemas de grande escala com eficiência.

Comando CLI usado:

```
aws iotevents-data batch-update-detector --cli-input-json file://areaDM.BUD.json
```

Arquivo: `areaDM.BUD.json`

```
{
  "detectors": [
    {
      "messageId": "0001",
```

```
"detectorModelName": "areaDetectorModel",
"keyValue": "Area51",
"state": {
  "stateName": "start",
  "variables": [
    {
      "name": "desiredTemperature",
      "value": "22"
    },
    {
      "name": "averageTemperature",
      "value": "22"
    },
    {
      "name": "allowedError",
      "value": "1.0"
    },
    {
      "name": "rangeHigh",
      "value": "30.0"
    },
    {
      "name": "rangeLow",
      "value": "15.0"
    },
    {
      "name": "anomalousHigh",
      "value": "60.0"
    },
    {
      "name": "anomalousLow",
      "value": "0.0"
    },
    {
      "name": "sensorCount",
      "value": "12"
    },
    {
      "name": "noDelay",
      "value": "true"
    },
    {
      "name": "goodToGo",
      "value": "true"
    }
  ]
}
```

```
    },
    {
      "name": "sensorId",
      "value": "0"
    },
    {
      "name": "reportedTemperature",
      "value": "0.1"
    },
    {
      "name": "resetMe",
      "value": "true"
    }
  ],
  "timers": [
  ]
}
]
```

Resposta:

```
{
  "message": "An error occurred (InvalidRequestException) when calling the BatchUpdateDetector operation: Number of variables in the detector exceeds the limit 10"
}
```

O mecanismo de AWS IoT Core regras e AWS IoT Events

As regras a seguir republicam mensagens AWS IoT Events MQTT como mensagens de solicitação de atualização paralela. Assumimos que AWS IoT Core as coisas são definidas para uma unidade de aquecimento e uma unidade de resfriamento para cada área controlada pelo modelo do detector.

Neste exemplo, definimos coisas chamadas `Area51HeatingUnit` e `Area51CoolingUnit`.

Comando CLI usado:

```
aws iot create-topic-rule --cli-input-json file://ADMSHadowCool0ffRule.json
```

Arquivo: `ADMSHadowCool0ffRule.json`

```
{
  "ruleName": "ADMSHadowCool0ff",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Cooling/Off'",
    "description": "areaDetectorModel mqtt topic publish to cooling unit shadow
request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republish": {
          "topic": "$aws/things/${payload.detector.keyValue}CoolingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMSHadowRole"
        }
      }
    ]
  }
}
```

Resposta: [vazio]

Comando CLI usado:

```
aws iot create-topic-rule --cli-input-json file://ADMSHadowCool0nRule.json
```

Arquivo: ADMSHadowCool0nRule.json

```
{
  "ruleName": "ADMSHadowCool0n",
  "topicRulePayload": {
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Cooling/On'",
    "description": "areaDetectorModel mqtt topic publish to cooling unit shadow
request",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "republish": {
          "topic": "$aws/things/${payload.detector.keyValue}CoolingUnit/shadow/
update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMSHadowRole"
        }
      }
    ]
  }
}
```

```
    }  
  ]  
}  
}
```

Resposta: [vazio]

Comando CLI usado:

```
aws iot create-topic-rule --cli-input-json file://ADMSHadowHeatOffRule.json
```

Arquivo: ADMSHadowHeatOffRule.json

```
{  
  "ruleName": "ADMSHadowHeatOff",  
  "topicRulePayload": {  
    "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Heating/Off'",  
    "description": "areaDetectorModel mqtt topic publish to heating unit shadow  
request",  
    "ruleDisabled": false,  
    "awsIotSqlVersion": "2016-03-23",  
    "actions": [  
      {  
        "republish": {  
          "topic": "$$aws/things/${payload.detector.keyValue}HeatingUnit/shadow/  
update",  
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMSHadowRole"  
        }  
      }  
    ]  
  }  
}
```

Resposta: [vazio]

Comando CLI usado:

```
aws iot create-topic-rule --cli-input-json file://ADMSHadowHeatOnRule.json
```

Arquivo: ADMSHadowHeatOnRule.json

```
{
```

```
"ruleName": "ADMSHadowHeatOn",
"topicRulePayload": {
  "sql": "SELECT topic(3) as state.desired.command FROM 'hvac/Heating/On'",
  "description": "areaDetectorModel mqtt topic publish to heating unit shadow
request",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
    {
      "republish": {
        "topic": "$aws/things/${payload.detector.keyValue}HeatingUnit/shadow/
update",
        "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMSHadowRole"
      }
    }
  ]
}
```

Resposta: [vazio]

Exemplo: Um guindaste detectando condições usando AWS IoT Events

Um operador de muitos guindastes deseja detectar quando as máquinas precisam de manutenção ou substituição e acionar as notificações apropriadas. Cada guindaste tem um motor. Um motor emite mensagens (entradas) com informações sobre pressão e temperatura. O operador quer dois níveis de detectores de eventos:

- Um detector de eventos em nível de guindaste
- Um detector de eventos em nível de motor

Usando mensagens dos motores (que contêm metadados com o `craneId` e o `motorid`), o operador pode executar os dois níveis de detectores de eventos usando o roteamento apropriado. Quando as condições do evento forem atendidas, as notificações devem ser enviadas para os tópicos apropriados do Amazon SNS. O operador pode configurar os modelos do detector para que notificações duplicadas não sejam geradas.

Este exemplo demonstra os seguintes recursos funcionais:

- Criar, ler, atualizar, excluir (CRUD) de entradas.
- Criar, ler, atualizar, excluir (CRUD) de modelos de detector de eventos e versões diferentes de detectores de eventos.
- Roteamento de uma entrada para vários detectores de eventos.
- Ingestão de entradas em um modelo de detector.
- Avaliação das condições de gatilho e eventos do ciclo de vida.
- Capacidade de se referir às variáveis de estado em condições e definir seus valores dependendo das condições.
- Orquestração de runtime com definição, estado, avaliador de gatilho e executor de ações.
- Execução de ações em `ActionsExecutor` com um alvo do SNS.

Envie comandos em resposta às condições detectadas no AWS IoT Events

Esta página fornece um exemplo de uso de AWS IoT Events comandos para configurar entradas, criar modelos de detectores e enviar dados simulados de sensores. Os exemplos demonstram como aproveitar o monitoramento AWS IoT Events de equipamentos industriais, como motores e guindastes.

```
#Create Pressure Input
aws iotevents create-input --cli-input-json file://pressureInput.json
aws iotevents describe-input --input-name PressureInput
aws iotevents update-input --cli-input-json file://pressureInput.json
aws iotevents list-inputs
aws iotevents delete-input --input-name PressureInput

#Create Temperature Input
aws iotevents create-input --cli-input-json file://temperatureInput.json
aws iotevents describe-input --input-name TemperatureInput
aws iotevents update-input --cli-input-json file://temperatureInput.json
aws iotevents list-inputs
aws iotevents delete-input --input-name TemperatureInput

#Create Motor Event Detector using pressure and temperature input
aws iotevents create-detector-model --cli-input-json file://motorDetectorModel.json
aws iotevents describe-detector-model --detector-model-name motorDetectorModel
```

```
aws iotevents update-detector-model --cli-input-json file://
updateMotorDetectorModel.json
aws iotevents list-detector-models
aws iotevents list-detector-model-versions --detector-model-name motorDetectorModel
aws iotevents delete-detector-model --detector-model-name motorDetectorModel

#Create Crane Event Detector using temperature input
aws iotevents create-detector-model --cli-input-json file://craneDetectorModel.json
aws iotevents describe-detector-model --detector-model-name craneDetectorModel
aws iotevents update-detector-model --cli-input-json file://
updateCraneDetectorModel.json
aws iotevents list-detector-models
aws iotevents list-detector-model-versions --detector-model-name craneDetectorModel
aws iotevents delete-detector-model --detector-model-name craneDetectorModel

#Replace craneIds
sed -i ' "s/100008/100009/g" messages/*

#Replace motorIds
sed -i ' "s/200008/200009/g" messages/*

#Send HighPressure message
aws iotevents-data batch-put-message --cli-input-json file://messages/
highPressureMessage.json --cli-binary-format raw-in-base64-out

#Send HighTemperature message
aws iotevents-data batch-put-message --cli-input-json file://messages/
highTemperatureMessage.json --cli-binary-format raw-in-base64-out

#Send LowPressure message
aws iotevents-data batch-put-message --cli-input-json file://messages/
lowPressureMessage.json --cli-binary-format raw-in-base64-out

#Send LowTemperature message
aws iotevents-data batch-put-message --cli-input-json file://messages/
lowTemperatureMessage.json --cli-binary-format raw-in-base64-out
```

Um modelo AWS IoT Events de detector para monitoramento de guindastes

Monitore suas frotas de equipamentos ou dispositivos em busca de falhas ou mudanças na operação e acione ações quando esses eventos ocorrerem. Você define modelos de detectores em JSON que especificam estados, regras e ações. Isso permite monitorar entradas como temperatura e pressão, rastrear violações de limites e enviar alertas. Os exemplos mostram modelos de detectores para guindaste e motor, detectando problemas de superaquecimento e notificando o Amazon SNS quando um limite é excedido. Você pode atualizar os modelos para refinar o comportamento sem interromper o monitoramento.

Arquivo: craneDetectorModel.json

```
{
  "detectorModelName": "craneDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "Running",
        "onEnter": {
          "events": [
            {
              "eventName": "init",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "craneThresholdBreach",
                    "value": "0"
                  }
                }
              ]
            }
          ]
        },
        "onInput": {
          "events": [
            {
              "eventName": "Overheated",
              "condition": "$input.TemperatureInput.temperature > 35",
              "actions": [
```

```

        {
            "setVariable": {
                "variableName": "craneThresholdBreached",
                "value": "$variable.craneThresholdBreached + 1"
            }
        }
    ],
},
{
    "eventName": "Crane Threshold Breached",
    "condition": "$variable.craneThresholdBreached > 5",
    "actions": [
        {
            "sns": {
                "targetArn": "arn:aws:sns:us-
east-1:123456789012:CraneSNSTopic"
            }
        }
    ],
},
{
    "eventName": "Underheated",
    "condition": "$input.TemperatureInput.temperature < 25",
    "actions": [
        {
            "setVariable": {
                "variableName": "craneThresholdBreached",
                "value": "0"
            }
        }
    ]
}
]
}
},
    ],
    "initialStateName": "Running"
},
    "key": "craneid",
    "roleArn": "arn:aws:iam::123456789012:role/columboSNSRole"
}

```

Para atualizar um modelo de detector existente. Arquivo: `updateCraneDetectorModel.json`

```
{
  "detectorModelName": "craneDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "Running",
        "onEnter": {
          "events": [
            {
              "eventName": "init",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "craneThresholdBreach",
                    "value": "0"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "alarmRaised",
                    "value": "'false'"
                  }
                }
              ]
            }
          ]
        },
        "onInput": {
          "events": [
            {
              "eventName": "Overheated",
              "condition": "$input.TemperatureInput.temperature > 30",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "craneThresholdBreach",
                    "value": "$variable.craneThresholdBreach + 1"
                  }
                }
              ]
            }
          ]
        }
      }
    ]
  }
}
```

```

        "eventName": "Crane Threshold Breached",
        "condition": "$variable.craneThresholdBreached > 5 &&
$variable.alarmRaised == 'false'",
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-
east-1:123456789012:CraneSNSTopic"
                }
            },
            {
                "setVariable": {
                    "variableName": "alarmRaised",
                    "value": "'true'"
                }
            }
        ]
    },
    {
        "eventName": "Underheated",
        "condition": "$input.TemperatureInput.temperature < 10",
        "actions": [
            {
                "setVariable": {
                    "variableName": "craneThresholdBreached",
                    "value": "0"
                }
            }
        ]
    }
]
},
"initialStateName": "Running"
},
"roleArn": "arn:aws:iam::123456789012:role/columboSNSRole"
}

```

Arquivo: motorDetectorModel.json

```

{
    "detectorModelName": "motorDetectorModel",

```

```

"detectorModelDefinition": {
  "states": [
    {
      "stateName": "Running",
      "onEnter": {
        "events": [
          {
            "eventName": "init",
            "condition": "true",
            "actions": [
              {
                "setVariable": {
                  "variableName": "motorThresholdBreach",
                  "value": "0"
                }
              }
            ]
          }
        ]
      },
      "onInput": {
        "events": [
          {
            "eventName": "Overheated And Overpressurized",
            "condition": "$input.PressureInput.pressure > 70 &&
$input.TemperatureInput.temperature > 30",
            "actions": [
              {
                "setVariable": {
                  "variableName": "motorThresholdBreach",
                  "value": "$variable.motorThresholdBreach + 1"
                }
              }
            ]
          },
          {
            "eventName": "Motor Threshold Breached",
            "condition": "$variable.motorThresholdBreach > 5",
            "actions": [
              {
                "sns": {
                  "targetArn": "arn:aws:sns:us-
east-1:123456789012:MotorSNSTopic"
                }
              }
            ]
          }
        ]
      }
    }
  ]
}

```

```

    }
  ]
}
},
"initialStateName": "Running"
},
"key": "motorId",
"roleArn": "arn:aws:iam::123456789012:role/columboSNSRole"
}

```

Para atualizar um modelo de detector existente. Arquivo: `updateMotorDetectorModel.json`

```

{
  "detectorModelName": "motorDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "Running",
        "onEnter": {
          "events": [
            {
              "eventName": "init",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "motorThresholdBreach",
                    "value": "0"
                  }
                }
              ]
            }
          ]
        },
        "onInput": {
          "events": [
            {
              "eventName": "Overheated And Overpressurized",
              "condition": "$input.PressureInput.pressure > 70 &&
$input.TemperatureInput.temperature > 30",

```

```

        "actions": [
            {
                "setVariable": {
                    "variableName": "motorThresholdBreach",
                    "value": "$variable.motorThresholdBreach + 1"
                }
            }
        ],
    },
    {
        "eventName": "Motor Threshold Breach",
        "condition": "$variable.motorThresholdBreach > 5",
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-
east-1:123456789012:MotorSNSTopic"
                }
            }
        ]
    }
]
}
},
"initialStateName": "Running"
},
"roleArn": "arn:aws:iam::123456789012:role/columboSNSRole"
}

```

AWS IoT Events entradas para monitoramento de guindastes

Neste exemplo, demonstramos como configurar entradas para um sistema de monitoramento de guindaste usando AWS IoT Events. Ele captura entradas de pressão e temperatura para ilustrar como estruturar entradas para monitoramento complexo de equipamentos industriais.

Arquivo: `pressureInput.json`

```

{
  "inputName": "PressureInput",
  "inputDescription": "this is a pressure input description",
  "inputDefinition": {

```

```
    "attributes": [  
      {"jsonPath": "pressure"}  
    ]  
  }  
}
```

Arquivo: temperatureInput.json

```
{  
  "inputName": "TemperatureInput",  
  "inputDescription": "this is temperature input description",  
  "inputDefinition": {  
    "attributes": [  
      {"jsonPath": "temperature"}  
    ]  
  }  
}
```

Envie mensagens operacionais e de alarme com AWS IoT Events

O tratamento eficaz de mensagens é importante nos sistemas de monitoramento de guindastes. Esta seção mostra como configurar para processar e responder AWS IoT Events a vários tipos de mensagens dos sensores do guindaste. Configurar alarmes com base em uma mensagem específica pode ajudá-lo a analisar, filtrar e rotear atualizações de status para acionar as ações apropriadas.

Arquivo: highPressureMessage.json

```
{  
  "messages": [  
    {  
      "messageId": "1",  
      "inputName": "PressureInput",  
      "payload": "{\"craneid\": \"100009\", \"pressure\": 80, \"motorid\":  
\"200009\"}"  
    }  
  ]  
}
```

Arquivo: highTemperatureMessage.json

```
{
  "messages": [
    {
      "messageId": "2",
      "inputName": "TemperatureInput",
      "payload": "{\"craneid\": \"100009\", \"temperature\": 40, \"motorid\": \"200009\"}"
    }
  ]
}
```

Arquivo: lowPressureMessage.json

```
{
  "messages": [
    {
      "messageId": "1",
      "inputName": "PressureInput",
      "payload": "{\"craneid\": \"100009\", \"pressure\": 20, \"motorid\": \"200009\"}"
    }
  ]
}
```

Arquivo: lowTemperatureMessage.json

```
{
  "messages": [
    {
      "messageId": "2",
      "inputName": "TemperatureInput",
      "payload": "{\"craneid\": \"100009\", \"temperature\": 20, \"motorid\": \"200009\"}"
    }
  ]
}
```

Exemplo: detecção de AWS IoT Events eventos com sensores e aplicativos

Esse modelo de detector é um dos modelos disponíveis no AWS IoT Events console. Está incluído aqui para a sua conveniência.

Este exemplo demonstra a detecção de eventos AWS IoT Events do aplicativo usando dados do sensor. Ele mostra como você pode criar um modelo de detector que monitora eventos específicos para que você possa acionar as ações apropriadas. Você pode criar várias entradas de sensor, definir condições complexas de eventos e configurar mecanismos de resposta graduados.

```
{
  "detectorModelName": "EventDetectionSensorsAndApplications",
  "detectorModelDefinition": {
    "states": [
      {
        "onInput": {
          "transitionEvents": [],
          "events": []
        },
        "stateName": "Device_exception",
        "onEnter": {
          "events": [
            {
              "eventName": "Send_mqtt",
              "actions": [
                {
                  "iotTopicPublish": {
                    "mqttTopic": "Device_stolen"
                  }
                }
              ],
              "condition": "true"
            }
          ]
        },
        "onExit": {
          "events": []
        }
      },
      {
        "onInput": {
```

```

        "transitionEvents": [
            {
                "eventName": "To_in_use",
                "actions": [],
                "condition": "$variable.position !=
$input.AWS_IoTEvents_Blueprints_Tracking_DeviceInput.gps_position",
                "nextState": "Device_in_use"
            }
        ],
        "events": []
    },
    "stateName": "Device_idle",
    "onEnter": {
        "events": [
            {
                "eventName": "Set_position",
                "actions": [
                    {
                        "setVariable": {
                            "variableName": "position",
                            "value":
"$input.AWS_IoTEvents_Blueprints_Tracking_DeviceInput.gps_position"
                        }
                    }
                ],
                "condition": "true"
            }
        ]
    },
    "onExit": {
        "events": []
    }
},
{
    "onInput": {
        "transitionEvents": [
            {
                "eventName": "To_exception",
                "actions": [],
                "condition":
"$input.AWS_IoTEvents_Blueprints_Tracking_UserInput.device_id !=
$input.AWS_IoTEvents_Blueprints_Tracking_DeviceInput.device_id",
                "nextState": "Device_exception"
            }
        ]
    }
}

```

```

        ],
        "events": []
    },
    "stateName": "Device_in_use",
    "onEnter": {
        "events": []
    },
    "onExit": {
        "events": []
    }
}
],
"initialStateName": "Device_idle"
}
}

```

Exemplo: dispositivo HeartBeat para monitorar as conexões do dispositivo com AWS IoT Events

Esse modelo de detector é um dos modelos disponíveis no AWS IoT Events console. Está incluído aqui para a sua conveniência.

O exemplo de batimento cardíaco defeituoso (DHB) ilustra como AWS IoT Events pode ser usado no monitoramento de serviços de saúde. Este exemplo mostra como você pode criar um modelo de detector que analisa dados de frequência cardíaca, detecta padrões irregulares e aciona respostas apropriadas. Aprenda a configurar entradas, definir limites e configurar alertas para possíveis problemas cardíacos, mostrando a versatilidade AWS IoT Events da empresa em aplicações relacionadas à saúde.

```

{
  "detectorModelDefinition": {
    "states": [
      {
        "onInput": {
          "transitionEvents": [
            {
              "eventName": "To_normal",
              "actions": [],
              "condition":
"currentInput(\"AWS_IoTEvents_Blueprints_Heartbeat_Input\")",
              "nextState": "Normal"
            }
          ]
        }
      }
    ]
  }
}

```

```
        }
      ],
      "events": []
    },
    "stateName": "Offline",
    "onEnter": {
      "events": [
        {
          "eventName": "Send_notification",
          "actions": [
            {
              "sns": {
                "targetArn": "sns-topic-arn"
              }
            }
          ],
          "condition": "true"
        }
      ]
    },
    "onExit": {
      "events": []
    }
  },
  {
    "onInput": {
      "transitionEvents": [
        {
          "eventName": "Go_offline",
          "actions": [],
          "condition": "timeout(\"awake\")",
          "nextState": "Offline"
        }
      ],
      "events": [
        {
          "eventName": "Reset_timer",
          "actions": [
            {
              "resetTimer": {
                "timerName": "awake"
              }
            }
          ]
        }
      ]
    }
  }
],
```

```

        "condition":
"currentInput(\"AWS_IoTEvents_Blueprints_Heartbeat_Input\")"
        }
    ]
},
"stateName": "Normal",
"onEnter": {
    "events": [
        {
            "eventName": "Create_timer",
            "actions": [
                {
                    "setTimer": {
                        "seconds": 300,
                        "timerName": "awake"
                    }
                }
            ],
            "condition":
"$input.AWS_IoTEvents_Blueprints_Heartbeat_Input.value > 0"
        }
    ]
},
"onExit": {
    "events": []
}
}
],
"initialStateName": "Normal"
}
}

```

Exemplo: Um alarme ISA em AWS IoT Events

Esse modelo de detector é um dos modelos disponíveis no AWS IoT Events console. Está incluído aqui para a sua conveniência.

```

{
  "detectorModelName": "AWS_IoTEvents_Blueprints_ISA_Alarm",
  "detectorModelDefinition": {
    "states": [
      {

```

```

    "onInput": {
      "transitionEvents": [
        {
          "eventName": "unshelve",
          "actions": [],
          "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unshelve\" &&
$variable.state == \"rtnunack\"",
          "nextState": "RTN_Unacknowledged"
        },
        {
          "eventName": "unshelve",
          "actions": [],
          "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unshelve\" &&
$variable.state == \"ack\"",
          "nextState": "Acknowledged"
        },
        {
          "eventName": "unshelve",
          "actions": [],
          "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unshelve\" &&
$variable.state == \"unack\"",
          "nextState": "Unacknowledged"
        },
        {
          "eventName": "unshelve",
          "actions": [],
          "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unshelve\" &&
$variable.state == \"normal\"",
          "nextState": "Normal"
        }
      ],
      "events": []
    },
    "stateName": "Shelved",
    "onEnter": {
      "events": []
    },
    "onExit": {
      "events": []
    }
  }

```

```

    },
    {
      "onInput": {
        "transitionEvents": [
          {
            "eventName": "abnormal_condition",
            "actions": [],
            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value > $variable.higher_threshold ||
$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value < $variable.lower_threshold",
            "nextState": "Unacknowledged"
          },
          {
            "eventName": "acknowledge",
            "actions": [],
            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"acknowledge\"",
            "nextState": "Normal"
          },
          {
            "eventName": "shelve",
            "actions": [],
            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"shelve\"",
            "nextState": "Shelved"
          },
          {
            "eventName": "remove_from_service",
            "actions": [],
            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"remove\"",
            "nextState": "Out_of_service"
          },
          {
            "eventName": "suppression",
            "actions": [],
            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"suppressed\"",
            "nextState": "Suppressed_by_design"
          }
        ],
        "events": []
      },
      "stateName": "RTN_Unacknowledged",

```

```

    "onEnter": {
      "events": [
        {
          "eventName": "State Save",
          "actions": [
            {
              "setVariable": {
                "variableName": "state",
                "value": "\\rtnunack\\"
              }
            }
          ],
          "condition": "true"
        }
      ]
    },
    "onExit": {
      "events": []
    }
  },
  {
    "onInput": {
      "transitionEvents": [
        {
          "eventName": "abnormal_condition",
          "actions": [],
          "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value > $variable.higher_threshold ||
$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value < $variable.lower_threshold",
          "nextState": "Unacknowledged"
        },
        {
          "eventName": "shelve",
          "actions": [],
          "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \\shelve\\",
          "nextState": "Shelved"
        },
        {
          "eventName": "remove_from_service",
          "actions": [],
          "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \\remove\\",
          "nextState": "Out_of_service"
        }
      ]
    }
  }
}

```

```

        },
        {
            "eventName": "suppression",
            "actions": [],
            "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"suppressed\"",
            "nextState": "Suppressed_by_design"
        }
    ],
    "events": [
        {
            "eventName": "Create Config variables",
            "actions": [
                {
                    "setVariable": {
                        "variableName": "lower_threshold",
                        "value":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.lower_threshold"
                    }
                },
                {
                    "setVariable": {
                        "variableName": "higher_threshold",
                        "value":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.higher_threshold"
                    }
                }
            ],
            "condition": "$variable.lower_threshold !=
$variable.lower_threshold"
        }
    ]
},
"stateName": "Normal",
"onEnter": {
    "events": [
        {
            "eventName": "State Save",
            "actions": [
                {
                    "setVariable": {
                        "variableName": "state",
                        "value": "\"normal\""
                    }
                }
            ]
        }
    ]
}

```

```

        }
    ],
    "condition": "true"
}
]
},
"onExit": {
    "events": []
}
},
{
    "onInput": {
        "transitionEvents": [
            {
                "eventName": "acknowledge",
                "actions": [],
                "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"acknowledge\"",
                "nextState": "Acknowledged"
            },
            {
                "eventName": "return_to_normal",
                "actions": [],
                "condition":
"($input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value <= $variable.higher_threshold
&& $input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.value >=
$variable.lower_threshold)",
                "nextState": "RTN_Unacknowledged"
            },
            {
                "eventName": "shelve",
                "actions": [],
                "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"shelve\"",
                "nextState": "Shelved"
            },
            {
                "eventName": "remove_from_service",
                "actions": [],
                "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"remove\"",
                "nextState": "Out_of_service"
            }
        ]
    }
}

```

```

        "eventName": "suppression",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"suppressed\"",
        "nextState": "Suppressed_by_design"
    }
  ],
  "events": []
},
"stateName": "Unacknowledged",
"onEnter": {
  "events": [
    {
      "eventName": "State Save",
      "actions": [
        {
          "setVariable": {
            "variableName": "state",
            "value": "\"unack\""
          }
        }
      ],
      "condition": "true"
    }
  ]
},
"onExit": {
  "events": []
}
},
{
  "onInput": {
    "transitionEvents": [
      {
        "eventName": "unsuppression",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unsuppressed\" &&
$variable.state == \"normal\"",
        "nextState": "Normal"
      },
      {
        "eventName": "unsuppression",
        "actions": [],

```

```

        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unsuppressed\" &&
$variable.state == \"unack\"",
        "nextState": "Unacknowledged"
    },
    {
        "eventName": "unsuppression",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unsuppressed\" &&
$variable.state == \"ack\"",
        "nextState": "Acknowledged"
    },
    {
        "eventName": "unsuppression",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"unsuppressed\" &&
$variable.state == \"rtnunack\"",
        "nextState": "RTN_Unacknowledged"
    }
],
"events": []
},
"stateName": "Suppressed_by_design",
"onEnter": {
    "events": []
},
"onExit": {
    "events": []
}
},
{
    "onInput": {
        "transitionEvents": [
            {
                "eventName": "return_to_service",
                "actions": [],
                "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"add\" && $variable.state
== \"rtnunack\"",
                "nextState": "RTN_Unacknowledged"
            }
        ]
    }
}

```

```

        "eventName": "return_to_service",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"add\" && $variable.state
== \"unack\"",
        "nextState": "Unacknowledged"
    },
    {
        "eventName": "return_to_service",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"add\" && $variable.state
== \"ack\"",
        "nextState": "Acknowledged"
    },
    {
        "eventName": "return_to_service",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"add\" && $variable.state
== \"normal\"",
        "nextState": "Normal"
    }
],
"events": []
},
"stateName": "Out_of_service",
"onEnter": {
    "events": []
},
"onExit": {
    "events": []
}
},
{
    "onInput": {
        "transitionEvents": [
            {
                "eventName": "re-alarm",
                "actions": [],
                "condition": "timeout(\"snooze\")",
                "nextState": "Unacknowledged"
            },
            {

```

```

        "eventName": "return_to_normal",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"reset\"",
        "nextState": "Normal"
    },
    {
        "eventName": "shelve",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"shelve\"",
        "nextState": "Shelved"
    },
    {
        "eventName": "remove_from_service",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"remove\"",
        "nextState": "Out_of_service"
    },
    {
        "eventName": "suppression",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_ISA_Alarm_Input.command == \"suppressed\"",
        "nextState": "Suppressed_by_design"
    }
],
"events": []
},
"stateName": "Acknowledged",
"onEnter": {
    "events": [
        {
            "eventName": "Create Timer",
            "actions": [
                {
                    "setTimer": {
                        "seconds": 60,
                        "timerName": "snooze"
                    }
                }
            ]
        }
    ],
    "condition": "true"

```

```

        },
        {
            "eventName": "State Save",
            "actions": [
                {
                    "setVariable": {
                        "variableName": "state",
                        "value": "\"ack\""
                    }
                }
            ],
            "condition": "true"
        }
    ],
    "onExit": {
        "events": []
    }
},
],
"initialStateName": "Normal"
},
"detectorModelDescription": "This detector model is used to detect if a monitored
device is in an Alarming State in accordance to the ISA 18.2.",
"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
"key": "alarmId"
}

```

Exemplo: Crie um alarme simples com AWS IoT Events

Esse modelo de detector é um dos modelos disponíveis no AWS IoT Events console. Está incluído aqui para a sua conveniência.

```

{
  "detectorModelDefinition": {
    "states": [
      {
        "onInput": {
          "transitionEvents": [
            {
              "eventName": "not_fixed",
              "actions": [],
            }
          ]
        }
      }
    ]
  }
}

```

```

        "condition": "timeout(\"snoozeTime\")",
        "nextState": "Alarming"
    },
    {
        "eventName": "reset",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.command == \"reset\"",
        "nextState": "Normal"
    }
],
"events": [
    {
        "eventName": "DND",
        "actions": [
            {
                "setVariable": {
                    "variableName": "dnd_active",
                    "value": "1"
                }
            }
        ],
        "condition":
"$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.command == \"dnd\""
    }
],
"stateName": "Snooze",
"onEnter": {
    "events": [
        {
            "eventName": "Create Timer",
            "actions": [
                {
                    "setTimer": {
                        "seconds": 120,
                        "timerName": "snoozeTime"
                    }
                }
            ],
            "condition": "true"
        }
    ]
}
},

```

```

        "onExit": {
            "events": []
        }
    },
    {
        "onInput": {
            "transitionEvents": [
                {
                    "eventName": "out_of_range",
                    "actions": [],
                    "condition":
"$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.value > $variable.threshold",
                    "nextState": "Alarming"
                }
            ],
            "events": [
                {
                    "eventName": "Create Config variables",
                    "actions": [
                        {
                            "setVariable": {
                                "variableName": "threshold",
                                "value":
"$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.threshold"
                            }
                        }
                    ],
                    "condition": "$variable.threshold != $variable.threshold"
                }
            ]
        },
        "stateName": "Normal",
        "onEnter": {
            "events": [
                {
                    "eventName": "Init",
                    "actions": [
                        {
                            "setVariable": {
                                "variableName": "dnd_active",
                                "value": "0"
                            }
                        }
                    ]
                }
            ],

```

```

        "condition": "true"
      }
    ]
  },
  "onExit": {
    "events": []
  }
},
{
  "onInput": {
    "transitionEvents": [
      {
        "eventName": "reset",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.command == \"reset\"",
        "nextState": "Normal"
      },
      {
        "eventName": "acknowledge",
        "actions": [],
        "condition":
"$input.AWS_IoTEvents_Blueprints_Simple_Alarm_Input.command == \"acknowledge\"",
        "nextState": "Snooze"
      }
    ],
    "events": [
      {
        "eventName": "Escalated Alarm Notification",
        "actions": [
          {
            "sns": {
              "targetArn": "arn:aws:sns:us-
west-2:123456789012:escalatedAlarmNotification"
            }
          }
        ],
        "condition": "timeout(\"unacknowledgeTime\")"
      }
    ]
  },
  "stateName": "Alarming",
  "onEnter": {
    "events": [

```

```
        {
            "eventName": "Alarm Notification",
            "actions": [
                {
                    "sns": {
                        "targetArn": "arn:aws:sns:us-
west-2:123456789012:alarmNotification"
                    }
                },
                {
                    "setTimer": {
                        "seconds": 300,
                        "timerName": "unacknowledgeTime"
                    }
                }
            ],
            "condition": "$variable.dnd_active != 1"
        }
    ],
    "onExit": {
        "events": []
    }
},
"initialStateName": "Normal"
},
"detectorModelDescription": "This detector model is used to detect if a monitored
device is in an Alarming State.",
"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
"key": "alarmId"
}
```

Monitoramento com alarmes em AWS IoT Events

AWS IoT Events os alarmes ajudam você a monitorar seus dados em busca de alterações. Os dados podem ser indicadores que você mede para seus equipamentos e processos. É possível criar alarmes que enviam notificações quando um limite é violado. Os alarmes ajudam você a detectar problemas, racionalizar a manutenção e otimizar o desempenho de seus equipamentos e processos.

Alarmes são exemplos de modelos de alarme. O modelo de alarme especifica o que detectar, quando enviar notificações, quem é notificado e muito mais. Você também pode especificar uma ou mais [ações suportadas](#) que ocorrem quando o estado do alarme muda. AWS IoT Events direciona [os atributos de entrada](#) derivados de seus dados para os alarmes apropriados. Se os dados que você está monitorando estiverem fora do intervalo especificado, o alarme será invocado. Você também pode reconhecer os alarmes ou configurá-los para o modo de soneca.

Trabalhando com AWS IoT SiteWise

Você pode usar AWS IoT Events alarmes para monitorar as propriedades do ativo em AWS IoT SiteWise. AWS IoT SiteWise envia valores de propriedades de ativos para AWS IoT Events alarmes. AWS IoT Events envia o estado do alarme para AWS IoT SiteWise.

AWS IoT SiteWise também suporta alarmes externos. Você pode escolher alarmes externos se usar alarmes externos AWS IoT SiteWise e tiver uma solução que retorne os dados do estado do alarme. O alarme externo contém uma propriedade de medição que ingere os dados do estado do alarme.

AWS IoT SiteWise não avalia o estado dos alarmes externos. Além disso, não é possível reconhecer ou adiar um alarme externo quando o estado do alarme muda.

Você pode usar o recurso SiteWise Monitor para visualizar o estado dos alarmes externos nos portais do SiteWise Monitor.

Para obter mais informações, consulte [Monitoramento de dados com alarmes](#) no Guia do AWS IoT SiteWise usuário e [Monitoramento com alarmes](#) no Guia de aplicação do SiteWise Monitor.

Fluxo de reconhecimento

Ao criar um modelo de alarme, você escolhe se deseja ativar o fluxo de reconhecimento. Se você ativar o fluxo de reconhecimento, a sua equipe será notificada quando o estado do alarme mudar. A

sua equipe pode reconhecer o alarme e deixar uma observação. Por exemplo, é possível incluir as informações do alarme e as ações que vai tomar para resolver o problema. Se os dados que você está monitorando estiverem fora do intervalo especificado, o alarme será invocado.

Os alarmes têm um dos seguintes estados:

DISABLED

Quando o alarme está no estado DISABLED, ele não está pronto para avaliar os dados. Para ativar o alarme, você deve alterar o alarme para o estado NORMAL.

NORMAL

Quando o alarme está no estado NORMAL, ele está pronto para avaliar os dados.

ACTIVE

Se o alarme estiver no estado ACTIVE, o alarme será invocado. Os dados que você está monitorando estão fora do intervalo especificado.

ACKNOWLEDGED

Quando o alarme está no estado ACKNOWLEDGED, o alarme foi invocado e você reconheceu o alarme.

LATCHED

O alarme foi invocado, mas você não o reconheceu após um período de tempo. O alarme muda automaticamente para o estado NORMAL.

SNOOZE_DISABLED

Quando o alarme está no estado SNOOZE_DISABLED, o alarme é desativado por um período de tempo especificado. Após o tempo de soneca, o alarme muda automaticamente para o estado NORMAL.

Criando um modelo de alarme em AWS IoT Events

Você pode usar AWS IoT Events alarmes para monitorar seus dados e ser notificado quando um limite for violado. Os alarmes fornecem parâmetros que você usa para criar ou configurar um modelo de alarme. Você pode usar o AWS IoT Events console ou a AWS IoT Events API para criar ou configurar o modelo de alarme. Quando você configura o modelo de alarme, as alterações entram em vigor à medida em que novos dados chegam.

Requisitos

Os seguintes requisitos se aplicam ao criar um modelo de alarme.

- Você pode criar um modelo de alarme para monitorar um atributo de entrada AWS IoT Events ou uma propriedade de ativo em AWS IoT SiteWise.
 - Se você optar por monitorar um atributo de entrada em AWS IoT Events, [Crie uma entrada para modelos em AWS IoT Events](#) antes de criar o modelo de alarme.
 - Se você optar por monitorar uma propriedade de ativo, deverá [criar um modelo de ativo](#) AWS IoT SiteWise antes de criar o modelo de alarme.
- Você deve ter uma função do IAM que permita que seu alarme execute ações e acesse AWS recursos. Para obter mais informações, consulte [Configuração de permissões do IAM para o AWS IoT Events](#).
- Todos os AWS recursos que este tutorial usa devem estar na mesma AWS região.

Como criar um modelo de alarme (console)

Veja a seguir como criar um modelo de alarme para monitorar um AWS IoT Events atributo no AWS IoT Events console.

1. Faça login no [console do AWS IoT Events](#).
2. No painel de navegação, selecione Modelos de alarme.
3. Na página Modelos de alarme, selecione Criar modelo de alarme.
4. Na seção Detalhes de modelos de alarme, faça o seguinte:
 - a. Insira um nome exclusivo.
 - b. (Opcional) Insira uma descrição.
5. Na seção Alvo do alarme, faça o seguinte:

Important

Se escolher Propriedade do ativo do AWS IoT SiteWise , você deverá ter criado um modelo de ativo no AWS IoT SiteWise.

- a. Escolha o atributo de entrada do AWS IoT Events .

- b. Escolha a entrada.
- c. Escolha a chave do atributo de entrada. Esse atributo de entrada é usado como uma chave para criar o alarme. AWS IoT Events direciona as entradas associadas a essa chave para o alarme.

 Important

Se a carga útil de mensagem de entrada não contiver essa chave de atributo de entrada ou se a chave não estiver no mesmo caminho JSON especificado na chave, a mensagem falhará na ingestão no AWS IoT Events.

6. Na seção Definições de limite, você define o atributo de entrada, o valor limite e o operador de comparação AWS IoT Events usados para alterar o estado do alarme.

- a. Em Atributo de entrada, escolha o atributo que você deseja monitorar.

Cada vez que esse atributo de entrada recebe novos dados, ele é avaliado para determinar o estado do alarme.

- b. Em Operador, escolha o operador de comparação. O operador compara seu atributo de entrada com o valor limite do seu atributo.

É possível escolher entre as seguintes opções:

- > maior que
- >= maior ou igual a
- < menor que
- <= menor ou igual a
- = igual a
- != não igual a

- c. Para Valor limite, insira um número ou escolha um atributo nas AWS IoT Events entradas. AWS IoT Events compara esse valor com o valor do atributo de entrada que você escolher.
 - d. (Opcional) Para Gravidade, use um número que sua equipe entenda para refletir a gravidade desse alarme.
7. (Opcional) Na seção Configurações de notificação, defina as configurações de notificação para o alarme.

É possível adicionar até 10 notificações. Em Notificação 1, faça o seguinte:

- a. Em Protocolo, escolha uma das seguintes opções:
 - E-mail e texto: o alarme envia uma notificação por SMS e uma notificação por e-mail.
 - E-mail: o alarme envia uma notificação por e-mail.
 - Texto: o alarme envia uma notificação por SMS.
- b. Para Remetente, especifique o endereço de e-mail que pode enviar notificações sobre esse alarme.

Para adicionar mais endereços de e-mail à sua lista de remetentes, escolha Adicionar remetente.

- c. (Opcional) Em Destinatário, escolha o destinatário.

Para adicionar mais usuários à sua lista de destinatários, escolha Adicionar novo usuário. Você deve adicionar novos usuários à sua loja do IAM Identity Center antes de poder adicioná-los ao seu modelo de alarme. Para obter mais informações, consulte [Gerencie o acesso dos destinatários do alarme ao IAM Identity Center em AWS IoT Events](#).

- d. (Opcional) Em Mensagem personalizada adicional, insira uma mensagem que descreva o que o alarme detecta e quais ações os destinatários devem tomar.
8. Na seção Instância, é possível ativar ou desativar todas as instâncias de alarme criadas com base nesse modelo de alarme.
 9. Na seção Configurações avançadas, faça o seguinte:

- a. Para o Fluxo de reconhecimento, você pode ativar ou desativar as notificações.
 - Se for Ativado, você receberá uma notificação quando o estado do alarme mudar. Você precisa escolher confirmar a notificação antes que o estado de alarme possa retornar para o normal.
 - Se você escolher Desativado, nenhuma ação será necessária. O alarme muda automaticamente para o estado normal quando de medição retorna ao intervalo especificado.

Para obter mais informações, consulte [Fluxo de reconhecimento](#).

- b. Para Permissões, selecione uma das seguintes opções:

- Você pode criar uma nova função a partir de modelos de AWS política e criar AWS IoT Events automaticamente uma função do IAM para você.
- Você pode usar uma função existente do IAM que permite que esse modelo de alarme execute ações e acesse outros AWS recursos.

Para obter mais informações, consulte [Gerenciamento de identidade e acesso do AWS IoT Events](#).

- c. Para Configurações adicionais de notificação, você pode editar sua AWS Lambda função para gerenciar as notificações de alarme. Escolha uma das seguintes opções para sua AWS Lambda função:
 - Crie uma nova AWS Lambda função - AWS IoT Events cria uma nova AWS Lambda função para você.
 - Use uma AWS Lambda função existente - Use uma AWS Lambda função existente escolhendo um nome de AWS Lambda função.

Para obter mais informações sobre as ações possíveis, consulte [AWS IoT Events trabalhando com outros AWS serviços](#).

- d. (Opcional) Em Definir ação de estado, você pode adicionar uma ou mais AWS IoT Events ações a serem tomadas quando o estado do alarme mudar.
10. (Opcional) Você pode adicionar Tags para gerenciar os seus alarmes. Para obter mais informações, consulte [Marcar seus recursos do AWS IoT Events](#).
 11. Escolha Criar.

Respondendo aos alarmes em AWS IoT Events

Responder aos alarmes de forma eficaz é um aspecto importante do gerenciamento de sistemas de IoT com. AWS IoT Events Explore várias maneiras de configurar e lidar com alarmes, incluindo: configurar canais de notificação, definir procedimentos de escalonamento e implementar ações de resposta automatizadas. Aprenda a criar condições de alarme diferenciadas, priorizar alertas e integrar-se a outros AWS serviços para criar um sistema de gerenciamento de alarmes responsivo para seus aplicativos de IoT.

Se você ativou o [fluxo de reconhecimento](#), você receberá uma notificação quando o estado do alarme mudar. Para responder ao alarme, você pode reconhecer, desativar, ativar, redefinir ou suspender o alarme.

Console

Veja a seguir como responder a um alarme no console do AWS IoT Events .

1. Faça login no [console do AWS IoT Events](#).
2. No painel de navegação, selecione Modelos de alarme.
3. Escolha o modelo de alarme alvo.
4. Na seção Lista de alarmes, escolha o alarme alvo.
5. É possível escolher uma das seguintes Ações:
 - Reconhecer: o alarme muda para o estado ACKNOWLEDGED.
 - Desativar: o alarme muda para o estado DISABLED.
 - Ativar: o alarme muda para o estado NORMAL.
 - Redefinir: o alarme muda para o estado NORMAL.
 - Suspender, e em seguida, faça o seguinte:
 1. Escolha a Duração da suspensão ou insira uma Duração de suspensão personalizada.
 2. Escolha Salvar.

O alarme muda para o estado SNOOZE_DISABLED

Para mais informações sobre esses estados, consulte [Fluxo de reconhecimento](#).

API

Para responder a um ou mais alarmes, você pode usar as seguintes operações de AWS IoT Events API:

- [BatchAcknowledgeAlarm](#)
- [BatchDisableAlarm](#)
- [BatchEnableAlarm](#)
- [BatchResetAlarm](#)
- [BatchSnoozeAlarm](#)

Gerenciando notificações de alarme em AWS IoT Events

AWS IoT Events integra-se ao Lambda, oferecendo recursos personalizados de processamento de eventos. Esta seção explora como usar as funções do Lambda em AWS IoT Events seus modelos de detectores, permitindo que você execute uma lógica complexa, interaja com serviços externos e implemente um tratamento sofisticado de eventos.

AWS IoT Events usa uma função Lambda para gerenciar notificações de alarme. Você pode usar a função Lambda fornecida por AWS IoT Events ou criar uma nova.

Tópicos

- [Criação de uma função Lambda em AWS IoT Events](#)
- [Usando a função Lambda fornecida pelo AWS IoT Events](#)
- [Gerencie o acesso dos destinatários do alarme ao IAM Identity Center em AWS IoT Events](#)

Criação de uma função Lambda em AWS IoT Events

AWS IoT Events fornece uma função Lambda que permite que os alarmes enviem e recebam notificações por e-mail e SMS.

Requisitos

Os seguintes requisitos se aplicam ao criar uma função do Lambda para alarmes:

- Se seu alarme enviar notificações por SMS, certifique-se de que o Amazon SNS esteja configurado para entregar mensagens SMS.
 - Para saber mais, consulte a documentação a seguir:
 - [Mensagens de texto móveis com Amazon SNS](#) e [identidades de origem para mensagens SMS do Amazon SNS no Guia do desenvolvedor do Amazon Simple Notification Service](#).
 - [O que é o AWS End User Messaging SMS?](#) no Guia do AWS SMS usuário.
- Se seu alarme enviar notificações por e-mail ou SMS, você deverá ter uma função do IAM que permita AWS Lambda trabalhar com o Amazon SES e o Amazon SNS.

Exemplo de política:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ses:GetIdentityVerificationAttributes",
        "ses:SendEmail",
        "ses:VerifyEmailIdentity"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish",
        "sns:OptInPhoneNumber",
        "sns:CheckIfPhoneNumberIsOptedOut",
        "sms-voice:DescribeOptedOutNumbers"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:*:*:*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource" : "*"
    }
  ]
}
```

- Você deve escolher a mesma AWS região para AWS IoT Events AWS Lambda e. Para obter a lista de regiões compatíveis, consulte [Endpoints e cotas do AWS IoT Events](#) e [Endpoints e cotas do AWS Lambda](#) no Referência geral da Amazon Web Services.

Implemente uma função Lambda para usar AWS IoT Events CloudFormation

Este tutorial usa um CloudFormation modelo para implantar uma função Lambda. Esse modelo cria automaticamente um perfil do IAM que permite que a função do Lambda funcione com o Amazon SES e o Amazon SNS.

Veja a seguir como usar o AWS Command Line Interface (AWS CLI) para criar uma CloudFormation pilha.

1. No terminal do seu dispositivo, execute `aws --version` para verificar se você instalou AWS CLI o. Para obter mais informações, consulte [Instalar ou atualizar para a versão mais recente da AWS CLI](#) no Guia do usuário do AWS Command Line Interface .
2. Execute `aws configure list` para verificar se você configurou o AWS CLI na AWS região que tem todos os seus AWS recursos para este tutorial. Para obter mais informações, consulte [Definir e visualizar as configurações usando comandos](#) no Guia AWS Command Line Interface do usuário
3. Faça o download do CloudFormation modelo, [NotificationLambda.template.yaml.zip](#).

Note

Se você tiver dificuldade em baixar o arquivo, o modelo também está disponível no [CloudFormation modelo](#).

4. Descompacte o conteúdo e salve-o localmente como `notificationLambda.template.yaml`.
5. Abra um terminal em seu dispositivo e navegue até o diretório em que você fez o download do arquivo `notificationLambda.template.yaml`.
6. Para criar uma CloudFormation pilha, execute o seguinte comando:

```
aws cloudformation create-stack --stack-name notificationLambda-stack --template-body file://notificationLambda.template.yaml --capabilities CAPABILITY_IAM
```

Você pode modificar esse CloudFormation modelo para personalizar a função Lambda e seu comportamento.

Note

AWS Lambda repete os erros de função duas vezes. Se a função não tiver capacidade suficiente para lidar com todas as solicitações em andamento, os eventos poderão ter de aguardar na fila por horas ou dias até serem enviados para a função. É possível configurar uma fila de mensagens não entregues (DLQ) na função para capturar eventos que não foram processados com êxito. Para obter mais informações, consulte [Invocação assíncrona](#) no Guia do desenvolvedor do AWS Lambda .

Você também pode criar ou configurar a pilha no CloudFormation console. Para mais informações, consulte [Como trabalhar com pilhas](#) no Guia do usuário do AWS CloudFormation .

Criação de uma função Lambda personalizada para AWS IoT Events

Você pode criar uma função do Lambda ou modificar a fornecida pelo AWS IoT Events.

Os seguintes requisitos se aplicam ao criar uma função do Lambda personalizada.

- Adicione permissões que permitam que sua função Lambda execute ações específicas e acesse AWS recursos.
- Se você usar a função Lambda fornecida por AWS IoT Events, certifique-se de escolher o tempo de execução do Python 3.7.

Exemplo da função do Lambda:

```
import boto3
import json
import logging
import datetime
logger = logging.getLogger()
logger.setLevel(logging.INFO)
ses = boto3.client('ses')
sns = boto3.client('sns')
def check_value(target):
    if target:
        return True
```

```
    return False

# Check whether email is verified. Only verified emails are allowed to send emails to
# or from.
def check_email(email):
    if not check_value(email):
        return False
    result = ses.get_identity_verification_attributes(Identities=[email])
    attr = result['VerificationAttributes']
    if (email not in attr or attr[email]['VerificationStatus'] != 'Success'):
        logging.info('Verification email for {} sent. You must have all the emails
verified before sending email.'.format(email))
        ses.verify_email_identity(EmailAddress=email)
        return False
    return True

# Check whether the phone holder has opted out of receiving SMS messages from your
# account
def check_phone_number(phone_number):
    try:
        result = sns.check_if_phone_number_is_opted_out(phoneNumber=phone_number)
        if (result['isOptedOut']):
            logger.info('phoneNumber {} is not opt in of receiving SMS messages. Phone
number must be opt in first.'.format(phone_number))
            return False
        return True
    except Exception as e:
        logging.error('Your phone number {} must be in E.164 format in SS0. Exception
thrown: {}'.format(phone_number, e))
        return False

def check_emails(emails):
    result = True
    for email in emails:
        if not check_email(email):
            result = False
    return result

def lambda_handler(event, context):
    logging.info('Received event: ' + json.dumps(event))
    nep = json.loads(event.get('notificationEventPayload'))
    alarm_state = nep['alarmState']
    default_msg = 'Alarm ' + alarm_state['stateName'] + '\n'
```

```

timestamp =
datetime.datetime.utcnow().timestamp(float(nep['stateUpdateTime']/1000)).strftime('%Y-
%m-%d %H:%M:%S')
alarm_msg = "{} {} {} at {} UTC ".format(nep['alarmModelName'], nep.get('keyValue',
'Singleton'), alarm_state['stateName'], timestamp)
default_msg += 'Sev: ' + str(nep['severity']) + '\n'
if (alarm_state['ruleEvaluation']):
    property = alarm_state['ruleEvaluation']['simpleRule']['inputProperty']
    default_msg += 'Current Value: ' + str(property) + '\n'
    operator = alarm_state['ruleEvaluation']['simpleRule']['operator']
    threshold = alarm_state['ruleEvaluation']['simpleRule']['threshold']
    alarm_msg += '({} {} {})' .format(str(property), operator, str(threshold))
default_msg += alarm_msg + '\n'

emails = event.get('emailConfigurations', [])
logger.info('Start Sending Emails')
for email in emails:
    from_adr = email.get('from')
    to_adrs = email.get('to', [])
    cc_adrs = email.get('cc', [])
    bcc_adrs = email.get('bcc', [])
    msg = default_msg + '\n' + email.get('additionalMessage', '')
    subject = email.get('subject', alarm_msg)
    fa_ver = check_email(from_adr)
    tas_ver = check_emails(to_adrs)
    ccas_ver = check_emails(cc_adrs)
    bccas_ver = check_emails(bcc_adrs)
    if (fa_ver and tas_ver and ccas_ver and bccas_ver):
        ses.send_email(Source=from_adr,
                        Destination={'ToAddresses': to_adrs, 'CcAddresses': cc_adrs,
'BccAddresses': bcc_adrs},
                        Message={'Subject': {'Data': subject}, 'Body': {'Text': {'Data':
msg}}})
    logger.info('Emails have been sent')

logger.info('Start Sending SNS message to SMS')
sns_configs = event.get('smsConfigurations', [])
for sns_config in sns_configs:
    sns_msg = default_msg + '\n' + sns_config.get('additionalMessage', '')
    phone_numbers = sns_config.get('phoneNumbers', [])
    sender_id = sns_config.get('senderId')
    for phone_number in phone_numbers:
        if check_phone_number(phone_number):
            if check_value(sender_id):

```

```
sns.publish(PhoneNumber=phone_number, Message=sns_msg,
MessageAttributes={'AWS.SNS.SMS.SenderID':{'DataType': 'String','StringValue':
sender_id}})
else:
sns.publish(PhoneNumber=phone_number, Message=sns_msg)
logger.info('SNS messages have been sent')
```

Para obter mais informações, consulte [O que é o AWS Lambda?](#) no Guia do desenvolvedor do AWS Lambda .

CloudFormation modelo

Use o CloudFormation modelo a seguir para criar sua função Lambda.

```
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Notification Lambda for Alarm Model'
Resources:
  NotificationLambdaRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
            Action: sts:AssumeRole
      Path: "/"
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/AWSLambdaExecute'
    Policies:
      - PolicyName: "NotificationLambda"
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
            - Effect: "Allow"
              Action:
                - "ses:GetIdentityVerificationAttributes"
                - "ses:SendEmail"
                - "ses:VerifyEmailIdentity"
              Resource: "*"
            - Effect: "Allow"
              Action:
                - "sns:Publish"
```

```

    - "sns:OptInPhoneNumber"
    - "sns:CheckIfPhoneNumberIsOptedOut"
    - "sms-voice:DescribeOptedOutNumbers"
  Resource: "*"
- Effect: "Deny"
  Action:
    - "sns:Publish"
  Resource: "arn:aws:sns:*:*:*"

```

NotificationLambdaFunction:

Type: AWS::Lambda::Function

Properties:

Role: !GetAtt NotificationLambdaRole.Arn

Runtime: python3.7

Handler: index.lambda_handler

Timeout: 300

MemorySize: 3008

Code:

```

ZipFile: |
    import boto3
    import json
    import logging
    import datetime
    logger = logging.getLogger()
    logger.setLevel(logging.INFO)
    ses = boto3.client('ses')
    sns = boto3.client('sns')
    def check_value(target):
        if target:
            return True
        return False

```

Check whether email is verified. Only verified emails are allowed to send emails to or from.

```

def check_email(email):
    if not check_value(email):
        return False
    result = ses.get_identity_verification_attributes(Identities=[email])
    attr = result['VerificationAttributes']
    if (email not in attr or attr[email]['VerificationStatus'] != 'Success'):
        logging.info('Verification email for {} sent. You must have all the
emails verified before sending email.'.format(email))
        ses.verify_email_identity(EmailAddress=email)
        return False
    return True

```

```

    # Check whether the phone holder has opted out of receiving SMS messages from
    your account
    def check_phone_number(phone_number):
        try:
            result = sns.check_if_phone_number_is_opted_out(phoneNumber=phone_number)
            if (result['isOptedOut']):
                logger.info('phoneNumber {} is not opt in of receiving SMS messages.
                Phone number must be opt in first.'.format(phone_number))
                return False
            return True
        except Exception as e:
            logging.error('Your phone number {} must be in E.164 format in SS0.
            Exception thrown: {}'.format(phone_number, e))
            return False

    def check_emails(emails):
        result = True
        for email in emails:
            if not check_email(email):
                result = False
        return result

    def lambda_handler(event, context):
        logging.info('Received event: ' + json.dumps(event))
        nep = json.loads(event.get('notificationEventPayload'))
        alarm_state = nep['alarmState']
        default_msg = 'Alarm ' + alarm_state['stateName'] + '\n'
        timestamp =
        datetime.datetime.utcnow().timestamp(float(nep['stateUpdateTime']/1000)).strftime('%Y-
        %m-%d %H:%M:%S')
        alarm_msg = "{} {} {} at {} UTC ".format(nep['alarmModelName'],
        nep.get('keyValue', 'Singleton'), alarm_state['stateName'], timestamp)
        default_msg += 'Sev: ' + str(nep['severity']) + '\n'
        if (alarm_state['ruleEvaluation']):
            property = alarm_state['ruleEvaluation']['simpleRule']['inputProperty']
            default_msg += 'Current Value: ' + str(property) + '\n'
            operator = alarm_state['ruleEvaluation']['simpleRule']['operator']
            threshold = alarm_state['ruleEvaluation']['simpleRule']['threshold']
            alarm_msg += '({} {} {})'.format(str(property), operator, str(threshold))
            default_msg += alarm_msg + '\n'

        emails = event.get('emailConfigurations', [])
        logger.info('Start Sending Emails')

```

```
for email in emails:
    from_adr = email.get('from')
    to_adrs = email.get('to', [])
    cc_adrs = email.get('cc', [])
    bcc_adrs = email.get('bcc', [])
    msg = default_msg + '\n' + email.get('additionalMessage', '')
    subject = email.get('subject', alarm_msg)
    fa_ver = check_email(from_adr)
    tas_ver = check_emails(to_adrs)
    ccas_ver = check_emails(cc_adrs)
    bccas_ver = check_emails(bcc_adrs)
    if (fa_ver and tas_ver and ccas_ver and bccas_ver):
        ses.send_email(Source=from_adr,
                       Destination={'ToAddresses': to_adrs, 'CcAddresses':
cc_adrs, 'BccAddresses': bcc_adrs},
                       Message={'Subject': {'Data': subject}, 'Body': {'Text':
{'Data': msg}}})
        logger.info('Emails have been sent')

logger.info('Start Sending SNS message to SMS')
sns_configs = event.get('smsConfigurations', [])
for sns_config in sns_configs:
    sns_msg = default_msg + '\n' + sns_config.get('additionalMessage', '')
    phone_numbers = sns_config.get('phoneNumbers', [])
    sender_id = sns_config.get('senderId')
    for phone_number in phone_numbers:
        if check_phone_number(phone_number):
            if check_value(sender_id):
                sns.publish(PhoneNumber=phone_number, Message=sns_msg,
MessageAttributes={'AWS.SNS.SMS.SenderID':{'DataType': 'String', 'StringValue':
sender_id}})
            else:
                sns.publish(PhoneNumber=phone_number, Message=sns_msg)
        logger.info('SNS messages have been sent')
```

Usando a função Lambda fornecida pelo AWS IoT Events

Com as notificações de alarme, você pode usar a função Lambda fornecida por AWS IoT Events para gerenciar notificações de alarme.

Os requisitos a seguir se aplicam quando você usa a função do Lambda fornecida pelo AWS IoT Events para gerenciar suas notificações de alarme:

- Você deve verificar o endereço de e-mail que envia as notificações por e-mail no Amazon Simple Email Service (Amazon SES). Para ter mais informações, consulte [Verifying an email address identity](#) no Guia do desenvolvedor do Amazon Simple Email Service.

Se receber um link de verificação, clique no link para verificar seu endereço de e-mail. Você também pode verificar se há um e-mail de verificação na pasta de spam.

- Se o alarme enviar notificações por SMS, você deverá usar a formatação de número de telefone internacional E.164 para números de telefone. Esse formato contém `+<country-calling-code><area-code><phone-number>`.

Exemplos de números de telefone:


País	Número de telefone local	Número formatado E.164
Estados Unidos	206-555-0100	+12065550100
Reino Unido	020-1234-1234	+442012341234
Lituânia	8+601+12345	+370 60112345

Para encontrar o código de chamada de um país, acesse [countrycode.org](#).

A função Lambda fornecida por AWS IoT Events verifica se você usa números de telefone no formato E.164. No entanto, ele não verifica os números de telefone. Se você garantir que inseriu números de telefone corretos, mas não recebeu notificações por SMS, entre em contato com as operadoras telefônicas. As operadoras podem bloquear as mensagens.

Gerencie o acesso dos destinatários do alarme ao IAM Identity Center em AWS IoT Events

AWS IoT Events usa Centro de Identidade do AWS IAM para gerenciar o acesso SSO dos destinatários dos alarmes. A implementação do IAM Identity Center para destinatários de AWS IoT Events notificações pode aprimorar a segurança e a experiência do usuário. Para ativar o alarme para enviar notificações aos destinatários, você deve ativar o IAM Identity Center e adicionar destinatários à sua loja do IAM Identity Center. Para obter mais informações, consulte [Adicionar usuários](#) no Guia do usuário do Centro de Identidade do AWS IAM .

 Important

- Você deve escolher a mesma AWS região para AWS IoT Events, AWS Lambda, e o IAM Identity Center.
- AWS Organizations só oferece suporte a uma região do IAM Identity Center por vez. Se você quiser disponibilizar o IAM Identity Center em uma região diferente, primeiro exclua a configuração atual do IAM Identity Center. Para obter mais informações, consulte [Dados da região do IAM Identity Center](#) no Guia do usuário do Centro de Identidade do AWS IAM

Segurança em AWS IoT Events

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. A eficácia da nossa segurança é regularmente testada e verificada por auditores de terceiros como parte dos [Programas de conformidade da AWS](#). Para saber mais sobre os programas de conformidade que se aplicam AWS IoT Events, 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, inclusive a confidencialidade dos dados, os requisitos da organização, as leis e as regulamentações vigentes.

Esta documentação ajudará você a entender como aplicar o modelo de responsabilidade compartilhada ao usar AWS IoT Events. Os tópicos a seguir mostram como configurar para atender AWS IoT Events aos seus objetivos de segurança e conformidade. Você também aprenderá a usar outros AWS serviços que podem ajudá-lo a monitorar e proteger seus AWS IoT Events recursos.

Tópicos

- [Gerenciamento de identidade e acesso para AWS IoT Events](#)
- [Monitoramento AWS IoT Events para manter a confiabilidade, a disponibilidade e o desempenho](#)
- [Validação de conformidade para AWS IoT Events](#)
- [Resiliência em AWS IoT Events](#)
- [Segurança da infraestrutura em AWS IoT Events](#)

Gerenciamento de identidade e acesso para AWS IoT Events

AWS Identity and Access Management (IAM) é um AWS serviço 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 Events os recursos. O IAM é um AWS serviço que você pode usar sem custo adicional.

Tópicos

- [Público](#)
- [Autenticação com identidades](#)
- [Gerenciar o acesso usando políticas](#)
- [Mais sobre gerenciamento de identidade e acesso](#)
- [Como AWS IoT Events funciona com o IAM](#)
- [AWS IoT Events exemplos de políticas baseadas em identidade](#)
- [Prevenção confusa de delegados entre serviços para AWS IoT Events](#)
- [Solucionar problemas de AWS IoT Events 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 [Solucionar problemas de AWS IoT Events identidade e acesso](#)).
- Administrador do serviço: determine o acesso do usuário e envie solicitações de permissão (consulte [Como AWS IoT Events funciona com o IAM](#))
- Administrador do IAM: escreva políticas para gerenciar o acesso (consulte [AWS IoT Events exemplos de políticas baseadas em identidade](#))

Autenticação com identidades

A autenticação é a forma 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.

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.

Mais sobre gerenciamento de identidade e acesso

Para obter mais informações sobre gerenciamento de identidade e acesso para AWS IoT Events, continue nas seguintes páginas:

- [Como AWS IoT Events funciona com o IAM](#)
- [Solucionar problemas de AWS IoT Events identidade e acesso](#)

Como AWS IoT Events funciona com o IAM

Antes de usar o IAM para gerenciar o acesso AWS IoT Events, você deve entender quais recursos do IAM estão disponíveis para uso AWS IoT Events. Para ter uma visão de alto nível de como AWS IoT Events e outros AWS serviços funcionam com o IAM, consulte [AWS os serviços que funcionam com o IAM](#) no Guia do usuário do IAM.

Tópicos

- [AWS IoT Events políticas baseadas em identidade](#)
- [AWS IoT Events políticas baseadas em recursos](#)
- [Autorização baseada em AWS IoT Events tags](#)
- [AWS IoT Events Funções do IAM](#)

AWS IoT Events 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 Events oferece suporte a ações, recursos e chaves de condição específicos. Para saber mais sobre todos os elementos usados em uma política JSON, consulte [Referência de elementos de política JSON do IAM](#) no Guia do usuário do IAM.

Ações

O elemento `Action` de uma política baseada em identidade do IAM descreve a ação ou ações específicas que serão permitidas ou negadas pela política. As ações de política geralmente têm o mesmo nome da operação de AWS API associada. A ação é usada em uma política para conceder permissões para executar a operação associada.

As ações políticas AWS IoT Events usam o seguinte prefixo antes da ação: `iotevents:`. Por exemplo, para conceder permissão a alguém para criar uma AWS IoT Events entrada com a operação da AWS IoT Events `CreateInput` API, você inclui a `iotevents:CreateInput` ação na política dessa pessoa. Para conceder permissão a alguém para enviar uma entrada com a operação da AWS IoT Events `BatchPutMessage` API, você inclui a `iotevents-data:BatchPutMessage` ação na política dessa pessoa. As declarações de política devem incluir um `NotAction` elemento `Action` ou. AWS IoT Events 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": [  
    "iotevents:action1",  
    "iotevents: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": "iotevents:Describe*"
```

Para ver uma lista de AWS IoT Events ações, consulte [Ações definidas por AWS IoT Events](#) no Guia do usuário do IAM.

Recursos

O elemento `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou um elemento `NotResource`. Você especifica um recurso

usando um ARN ou usando o caractere curinga (*) para indicar que a instrução se aplica a todos os recursos.

O recurso do modelo do AWS IoT Events detector tem o seguinte ARN:

```
arn:${Partition}:iotevents:${Region}:${Account}:detectorModel/${detectorModelName}
```

Para obter mais informações sobre o formato de ARNs, consulte [Identificar AWS recursos com nomes de recursos da Amazon \(ARNs\)](#).

Por exemplo, para especificar modelo de detector Foobar na sua declaração, use o seguinte ARN:

```
"Resource": "arn:aws:iotevents:us-east-1:123456789012:detectorModel/Foobar"
```

Para especificar todas as instâncias que pertencem a uma conta específica, use o caractere curinga (*):

```
"Resource": "arn:aws:iotevents:us-east-1:123456789012:detectorModel/*"
```

Algumas AWS IoT Events ações, como as de criação de recursos, não podem ser executadas em um recurso específico. Nesses casos, é necessário utilizar o caractere curinga (*).

```
"Resource": "*"
```

Algumas ações AWS IoT Events da API envolvem vários recursos. Por exemplo, `CreateDetectorModel` faz referência às entradas em suas declarações de condição, portanto, um usuário deve ter permissões para usar a entrada e o modelo do detector. Para especificar vários recursos em uma única instrução, separe-os ARNs com vírgulas.

```
"Resource": [  
  "resource1",  
  "resource2"
```

Para ver uma lista dos tipos de AWS IoT Events recursos e seus ARNs, consulte [Resources Defined by AWS IoT Events](#) no Guia do usuário do IAM. Para saber com quais ações é possível especificar o ARN de cada recurso, consulte [Ações definidas pelo AWS IoT Events](#).

Chaves de condição

O elemento `Condition` (ou bloco `Condition`) permite que você especifique condições nas quais uma instrução estiver em vigor. O elemento `Condition` é opcional. É possível criar expressões condicionais que usam [operadores de condição](#), como "igual a" ou "menor que", para fazer a condição da política corresponder aos valores na solicitação.

Se você especificar vários elementos de `Condition` em uma declaração ou várias chaves em um único elemento de `Condition`, a AWS os avaliará usando uma operação lógica AND. Se você especificar vários valores para uma única chave de condição, AWS avalia a condição usando uma OR operação lógica. Todas as condições devem ser atendidas antes que as permissões da instrução sejam concedidas.

Você também pode usar variáveis de espaço reservado ao especificar condições. Por exemplo, você pode conceder uma permissão de usuário para acessar um recurso somente se ela estiver marcado com seu nome de usuário. Para obter mais informações, consulte [Elementos de política do IAM: variáveis e tags](#) no Guia do usuário do IAM.

AWS IoT Events não fornece nenhuma chave de condição específica do serviço, mas oferece suporte ao uso de algumas chaves de condição globais. Para ver todas as chaves de condição AWS globais, consulte as [chaves de contexto de condição AWS global](#) no Guia do usuário do IAM.”

Exemplos

Para ver exemplos de políticas AWS IoT Events baseadas em identidade, consulte. [AWS IoT Events exemplos de políticas baseadas em identidade](#)

AWS IoT Events políticas baseadas em recursos

AWS IoT Events não oferece suporte a políticas baseadas em recursos.” Para visualizar um exemplo de uma política baseada em recurso detalhada, consulte <https://docs.aws.amazon.com/lambda/latest/dg/access-control-resource-based.html>.

Autorização baseada em AWS IoT Events tags

Você pode anexar tags a AWS IoT Events recursos ou passar tags em uma solicitação para AWS IoT Events. Para controlar o acesso baseado em tags, forneça informações sobre as tags no [elemento de condição](#) de uma política usando as `iotevents:ResourceTag/key-name`, `aws:RequestTag/key-name` ou chaves de condição `aws:TagKeys`. Para obter mais informações

sobre recursos de marcação do AWS IoT Events , consulte [Marcando seus recursos AWS IoT Events](#).

Para visualizar um exemplo de política baseada em identidade que visa limitar o acesso a um recurso baseado nas tags desse recurso, consulte [Exibir AWS IoT Events entradas com base em tags](#).

AWS IoT Events 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 Events

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 AWS Security Token Service (AWS STS) operações de API, como [AssumeRole](#) ou [GetFederationToken](#).

AWS IoT Events não 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 Events 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 Events suporta funções de serviço.

AWS IoT Events exemplos de políticas baseadas em identidade

Por padrão, usuários e funções não têm permissão para criar ou modificar AWS IoT Events recursos. 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 utilizando 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 Events console](#)
- [Permita que os usuários visualizem suas próprias permissões no AWS IoT Events](#)
- [Acesse uma AWS IoT Events entrada](#)
- [Exibir AWS IoT Events entradas com base em tags](#)

Práticas recomendadas de política

As políticas baseadas em identidade são muito poderosas. Eles determinam se alguém pode criar, acessar ou excluir AWS IoT Events 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 a usar políticas AWS gerenciadas — Para começar a usar AWS IoT Events rapidamente, use políticas AWS gerenciadas para dar a seus funcionários as permissões de que precisam. Essas políticas já estão disponíveis em sua conta e são mantidas e atualizadas pela AWS. Para obter mais informações, consulte [Comece a usar permissões com políticas AWS gerenciadas](#) no Guia do usuário do IAM.
- Conceder privilégio mínimo: ao criar políticas personalizadas, conceda apenas as permissões necessárias para executar uma tarefa. Comece com um conjunto mínimo de permissões e conceda permissões adicionais conforme necessário. Fazer isso é mais seguro do que começar com permissões que são muito lenientes e tentar restringi-las superiormente. Para obter mais informações, consulte [Conceder privilégio mínimo](#) no Guia do usuário do IAM.
- Habilitar o MFA para operações confidenciais: para segurança adicional, exija que os usuários do IAM usem a autenticação multifator (MFA) para acessar recursos ou operações de API confidenciais. Para obter mais informações, consulte [Usar autenticação multifator \(MFA\) AWS](#) no Guia do usuário do IAM.

- Usar condições de política para segurança adicional: na medida do possível, defina as condições sob as quais suas políticas baseadas em identidade permitem o acesso a um recurso. Por exemplo, você pode gravar condições para especificar um intervalo de endereços IP permitidos do qual a solicitação deve partir. Você também pode escrever condições para permitir somente solicitações em uma data especificada ou período ou para exigir o uso de SSL ou MFA. Para obter mais informações, consulte [Elementos de política JSON do IAM: condição](#) no Guia do usuário do IAM.

Usando o AWS IoT Events console

Para acessar o AWS IoT Events console, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os AWS IoT Events 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 Events console, anexe também a seguinte política AWS gerenciada às entidades. Para obter mais informações, consulte [Adicionar permissões a um usuário](#) no Manual do usuário do IAM:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotevents:BatchPutMessage",
        "iotevents:BatchUpdateDetector",
        "iotevents:CreateDetectorModel",
        "iotevents:CreateInput",
        "iotevents>DeleteDetectorModel",
        "iotevents>DeleteInput",
        "iotevents:DescribeDetector",
        "iotevents:DescribeDetectorModel",
        "iotevents:DescribeInput",
        "iotevents:DescribeLoggingOptions",
        "iotevents:ListDetectorModelVersions",
        "iotevents:ListDetectorModels",

```

```

        "iotevents:ListDetectors",
        "iotevents:ListInputs",
        "iotevents:ListTagsForResource",
        "iotevents:PutLoggingOptions",
        "iotevents:TagResource",
        "iotevents:UntagResource",
        "iotevents:UpdateDetectorModel",
        "iotevents:UpdateInput",
        "iotevents:UpdateInputRouting"
    ],
    "Resource": "arn:aws:iotevents:us-
east-1:123456789012:detectorModel/your-detector-model-name",
    "Resource": "arn:aws:iotevents:us-east-1:123456789012:input/your-input-name"
  }
]
}

```

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 corresponderem a operação da API que você estiver tentando executar.

Permita que os usuários visualizem suas próprias permissões no AWS IoT Events

Este exemplo mostra como você pode criar uma política que permite que os usuários visualizem as políticas gerenciadas e embutidas anexadas a sua identidade de usuário. Permitir que os usuários visualizem suas próprias permissões do IAM é útil para conscientização sobre segurança e recursos de autoatendimento. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou AWS .

JSON

```

{
  "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": "*"
}
]
}

```

Acesse uma AWS IoT Events entrada

O controle de acesso granular às AWS IoT Events entradas é importante para manter a segurança em ambientes com vários usuários ou várias equipes. Esta seção mostra como criar políticas do IAM que concedem acesso a AWS IoT Events entradas específicas e restringem o acesso a outras.

Neste exemplo, você pode conceder a um usuário Conta da AWS acesso a uma de suas AWS IoT Events entradas, `exampleInput`. Você também pode permitir que o usuário adicione, atualize e exclua entradas.

A política concede as permissões `iotevents:ListInputs`, `iotevents:DescribeInput`, `iotevents>CreateInput`, `iotevents>DeleteInput` e `iotevents:UpdateInput` ao usuário. Para ver um exemplo de apresentação do Amazon Simple Storage Service (Amazon S3), que concede permissões aos usuários e os testa usando o console, [consulte Controle do acesso a um bucket com políticas de usuário](#).

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListInputsInConsole",
      "Effect": "Allow",
      "Action": [
        "iotevents:ListInputs"
      ],
      "Resource": "arn:aws:iotevents:us-east-2:123456789012:input/*"
    },
    {
      "Sid": "ViewSpecificInputInfo",
      "Effect": "Allow",
      "Action": [
        "iotevents:DescribeInput"
      ],
      "Resource": "arn:aws:iotevents:us-east-1:123456789012:input/inputName"
    },
    {
      "Sid": "ManageInputs",
      "Effect": "Allow",
      "Action": [
        "iotevents:CreateInput",
        "iotevents>DeleteInput",
        "iotevents:DescribeInput",
        "iotevents:ListInputs",
        "iotevents:UpdateInput"
      ],
      "Resource": "arn:aws:iotevents:us-east-1:123456789012:input/*"
    }
  ]
}
```

Exibir AWS IoT Events entradas com base em tags

As tags ajudam você a organizar AWS IoT Events os recursos. Você pode usar condições em sua política baseada em identidade para controlar o acesso aos AWS IoT Events recursos com base em tags. Este exemplo mostra como você pode criar uma política que permita visualizar um *input*. No

entanto, a permissão é concedida somente se a tag `Owner` de `input` tiver o valor do nome desse usuário. Essa política também concede as permissões necessárias concluir essa ação no console.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListInputsInConsole",
      "Effect": "Allow",
      "Action": "iotevents:ListInputs",
      "Resource": "*"
    },
    {
      "Sid": "ViewInputsIfOwner",
      "Effect": "Allow",
      "Action": "iotevents:ListInputs",
      "Resource": "arn:aws:iotevents:*:*:input/*",
      "Condition": {
        "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
      }
    }
  ]
}
```

Você pode anexar essa política aos usuários na sua conta. Se um usuário chamado `richard-roe` tentar visualizar um AWS IoT Events `input`, ele `input` deverá ser marcado como `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 [Elementos de política JSON do IAM: condição](#) no Guia do usuário do IAM.

Prevenção confusa de delegados entre serviços para AWS IoT Events

Note

- O AWS IoT Events serviço só permite que você use funções para iniciar ações na mesma conta na qual um recurso foi criado. Isso ajuda a evitar um confuso ataque policial em AWS IoT Events.
- Esta página serve como referência para você ver como o problema confuso de deputados funciona e pode ser evitado caso recursos cruzados de contas sejam permitidos no AWS IoT Events serviço.

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 em um problema confuso de delegado.

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 usar as chaves de contexto de condição [aws:SourceAccount](#) global [aws:SourceArn](#) as chaves de contexto nas políticas de recursos para limitar as permissões que AWS IoT Events concedem outro serviço ao recurso. Se o valor de `aws:SourceArn` não contém ID da conta, como um ARN do bucket do Amazon S3, você deve usar ambas as chaves de contexto de condição global para limitar as permissões. Se você usa ambas as chaves de contexto de condição global, e o valor `aws:SourceArn` contém o ID da conta, o valor `aws:SourceAccount` e a conta no valor `aws:SourceArn` deverão utilizar a mesma ID de conta quando na mesma declaração de política.

Use `aws:SourceArn` se quiser apenas um recurso associado a acessibilidade de serviço. Use `aws:SourceAccount` se quiser permitir que qualquer recurso nessa conta seja associado ao uso entre serviços. O valor de `aws:SourceArn` deve ser o modelo do detector ou o modelo de alarme associado à solicitação `sts:AssumeRole`.

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 especificar vários recursos, use a chave de condição de contexto global `aws:SourceArn` com curingas (*) para as partes desconhecidas do ARN. Por exemplo, `.arn:aws:iotevents*:123456789012:*`

Os exemplos a seguir mostram como você pode usar as chaves de contexto de condição `aws:SourceAccount` global `aws:SourceArn` e as chaves de contexto AWS IoT Events para evitar o confuso problema substituto.

Tópicos

- [Exemplo: acesso seguro a um modelo AWS IoT Events de detector](#)
- [Exemplo: acesso seguro a um modelo AWS IoT Events de alarme](#)
- [Exemplo: acessar um AWS IoT Events recurso em uma região especificada](#)
- [Exemplo: Configurar opções de registro para AWS IoT Events](#)

Exemplo: acesso seguro a um modelo AWS IoT Events de detector

Este exemplo demonstra como criar uma política de IAM que concede acesso seguro a um modelo de detector específico em AWS IoT Events. A política usa condições para garantir que somente a AWS conta e o AWS IoT Events serviço especificados possam assumir a função, adicionando uma camada extra de segurança. Neste exemplo, a função só pode acessar o modelo de detector chamado `WindTurbine01`.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
```

```

        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:iotevents:us-
east-1:123456789012:detectorModel/WindTurbine01"
        }
      }
    ]
  }
}

```

Exemplo: acesso seguro a um modelo AWS IoT Events de alarme

Este exemplo demonstra como criar uma política do IAM que permite AWS IoT Events acessar modelos de alarme com segurança. A política usa condições para garantir que somente a AWS conta e o AWS IoT Events serviço especificados possam assumir a função.

Neste exemplo, a função pode acessar qualquer modelo de alarme dentro da AWS conta especificada, conforme indicado pelo caractere * curinga no ARN do modelo de alarme. As `aws:SourceArn` condições `aws:SourceAccount` e funcionam juntas para evitar o confuso problema do deputado.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {

```

```

        "aws:SourceArn": "arn:aws:iotevents:us-
east-1:123456789012:alarmModel/*"
    }
}
]
}

```

Exemplo: acessar um AWS IoT Events recurso em uma região especificada

Este exemplo demonstra como configurar uma função do IAM para acessar AWS IoT Events recursos em uma AWS região específica. Ao usar regiões específicas ARNs em suas políticas do IAM, você pode restringir o acesso a AWS IoT Events recursos em diferentes áreas geográficas. Essa abordagem pode ajudar a manter a segurança e a conformidade em implantações em várias regiões. A região neste exemplo é *us-east-1*.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:iotevents:us-east-1:123456789012:*"
        }
      }
    }
  ]
}

```

Exemplo: Configurar opções de registro para AWS IoT Events

O registro adequado é importante para monitorar, depurar e auditar seus aplicativos. AWS IoT Events Esta seção fornece uma visão geral das opções de registro disponíveis em AWS IoT Events.

Este exemplo demonstra como configurar uma função do IAM que permite AWS IoT Events registrar dados no CloudWatch Logs. O uso de curingas (*) no ARN do recurso permite um registro abrangente em AWS IoT Events sua infraestrutura.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "iotevents.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:iotevents:us-east-1:123456789012:*"
        }
      }
    }
  ]
}
```

Solucionar problemas de AWS IoT Events 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 Events um IAM.

Tópicos

- [Não estou autorizado a realizar uma ação em AWS IoT Events](#)
- [Não estou autorizado a executar iam:PassRole](#)
- [Quero permitir que pessoas fora da minha Conta da AWS acessem meus AWS IoT Events recursos](#)

Não estou autorizado a realizar uma ação em AWS IoT Events

Se isso Console de gerenciamento da AWS indicar que você não está autorizado a realizar uma ação, entre em contato com o administrador para obter ajuda. O administrador é a pessoa que forneceu o seu nome de usuário e senha.

O erro de exemplo a seguir ocorre quando o usuário mateojackson do IAM tenta usar o console para visualizar detalhes sobre um *input*, mas não tem as permissões `iotevents:ListInputs`.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
iotevents:ListInputs on resource: my-example-input
```

Neste caso, Mateo pede ao administrador para atualizar suas políticas para permitir a ele o acesso ao recurso *my-example-input* usando a ação `iotevents:ListInput`.

Não estou autorizado a executar 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 Events.

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 Events. 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 Events 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.

Consulte os tópicos a seguir para determinar suas melhores opções:

- Para saber se é AWS IoT Events compatível com esses recursos, consulte [Como AWS IoT Events 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 conhecer 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.

Monitoramento AWS IoT Events para manter a confiabilidade, a disponibilidade e o desempenho

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho de AWS IoT Events 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. Antes de iniciar o monitoramento AWS IoT Events, você deve criar um plano de monitoramento que inclua respostas às seguintes perguntas:

- 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?

A próxima etapa é estabelecer uma linha de base para o AWS IoT Events desempenho normal em seu ambiente, medindo o desempenho em vários momentos e sob diferentes condições de carga. À medida que você monitora o AWS IoT Events, armazene dados de monitoramento históricos para compará-los com os dados de performance atuais, identificar padrões de performance normais e anomalias de performance e elaborar métodos para resolver problemas.

Por exemplo, se você estiver usando o Amazon EC2, poderá monitorar a utilização da CPU, do disco ou reduzir o I/O, and network utilization for your instances. When performance falls outside your established baseline, you might need to reconfigure or optimize the instance to reduce CPU utilization, improve disk I/O tráfego de rede.

Tópicos

- [Ferramentas disponíveis para monitorar AWS IoT Events](#)
- [Monitoramento AWS IoT Events com a Amazon CloudWatch](#)
- [Registrando chamadas de AWS IoT Events API com AWS CloudTrail](#)

Ferramentas disponíveis para monitorar AWS IoT Events

AWS fornece várias ferramentas que você pode usar para monitorar AWS IoT Events. É possível configurar algumas dessas ferramentas para fazer o monitoramento em seu lugar, e, ao mesmo tempo, 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 Events e relatar quando algo está errado:

- Amazon CloudWatch Logs — Monitore, armazene e acesse seus arquivos de log de AWS CloudTrail ou de outras fontes. Para obter mais informações, consulte [Usando CloudWatch painéis da Amazon](#) no Guia do CloudWatch 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 registros em Java e valide se seus arquivos de log não foram alterados após a entrega. CloudTrail Para obter mais informações, consulte Como [trabalhar com arquivos de CloudTrail log](#) no Guia AWS CloudTrail do usuário.

Ferramentas de monitoramento manual

Outra parte importante do monitoramento AWS IoT Events envolve o monitoramento manual dos itens que os CloudWatch alarmes não cobrem. O AWS IoT Events, CloudWatch, e outros painéis AWS do console fornecem uma at-a-glance visão do estado do seu AWS ambiente. Recomendamos que você também verifique os arquivos de log AWS IoT Events.

- O AWS IoT Events console mostra:
 - Modelos de detector
 - Detectores
 - Entradas
 - Configurações
- A página CloudWatch inicial mostra:
 - Alertas e status atual
 - Gráficos de alertas e recursos
 - Estado de integridade do serviço

Além disso, você pode usar CloudWatch para fazer o seguinte:

- [Criar Criação de um CloudWatch painel](#) para monitorar os serviços que lhe interessam
- Colocar em gráfico dados de métrica para solucionar problemas e descobrir tendências
- Pesquise e navegue em todas as suas métricas AWS de recursos
- Criar e editar alertas para ser notificado sobre problemas

Monitoramento AWS IoT Events com a Amazon CloudWatch

Ao desenvolver ou depurar um modelo de AWS IoT Events detector, você precisa saber o que AWS IoT Events está fazendo e os erros encontrados. A Amazon CloudWatch monitora seus AWS recursos e os aplicativos em que você executa AWS em tempo real. Com isso CloudWatch, você obtém visibilidade de todo o sistema sobre o uso de recursos, desempenho de aplicativos e integridade operacional. [Ative o CloudWatch registro na Amazon ao desenvolver modelos AWS IoT Events de detectores](#) tem informações sobre como habilitar o CloudWatch registro para AWS IoT Events. Para gerar registros como o mostrado abaixo, você deve definir o nível de detalhamento como 'Depurar' e fornecer um ou mais destinos de depuração que seja um nome de modelo de detector e um opcional. KeyValue

O exemplo a seguir mostra uma entrada de registro no nível CloudWatch DEBUG gerada por AWS IoT Events.

```
{
  "timestamp": "2019-03-15T15:56:29.412Z",
  "level": "DEBUG",
  "logMessage": "Summary of message evaluation",
  "context": "MessageEvaluation",
  "status": "Success",
  "messageId": "SensorAggregate_2th846h",
  "keyValue": "boiler_1",
  "detectorModelName": "BoilerAlarmDetector",
  "initialState": "high_temp_alarm",
  "initialVariables": {
    "high_temp_count": 1,
    "high_pressure_count": 1
  },
  "finalState": "no_alarm",
  "finalVariables": {
    "high_temp_count": 0,
    "high_pressure_count": 0
  },
  "message": "{\"temp\": 34.9, \"pressure\": 84.5}",
  "messageType": "CUSTOMER_MESSAGE",
  "conditionEvaluationResults": [
    {
      "result": "True",
      "eventName": "alarm_cleared",
      "state": "high_temp_alarm",
      "lifeCycle": "OnInput",
    }
  ]
}
```

```
    "hasTransition": true
  },
  {
    "result": "Skipped",
    "eventName": "alarm_escalated",
    "state": "high_temp_alarm",
    "lifeCycle": "OnInput",
    "hasTransition": true,
    "resultDetails": "Skipped due to transition from alarm_cleared event"
  },
  {
    "result": "True",
    "eventName": "should_recall_technician",
    "state": "no_alarm",
    "lifeCycle": "OnEnter",
    "hasTransition": true
  }
]
}
```

Registrando chamadas de AWS IoT Events API com AWS CloudTrail

AWS IoT Events é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou AWS serviço em AWS IoT Events. CloudTrail captura todas as chamadas de API para eventos AWS IoT Events as, incluindo chamadas do AWS IoT Events console e de chamadas de código para o. AWS IoT Events 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 Events 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 Events, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para saber mais sobre isso CloudTrail, consulte o [Guia AWS CloudTrail do usuário](#).

AWS IoT Events informações em CloudTrail

CloudTrail é ativado em sua AWS conta quando você cria a conta. Quando a atividade ocorre em AWS IoT Events, essa atividade é registrada em um CloudTrail evento com outros eventos AWS de serviço no histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes em

sua AWS conta. Para obter mais informações, consulte [Trabalhando com o histórico de CloudTrail eventos](#).

Para um registro contínuo dos eventos em sua AWS conta, incluindo eventos para AWS IoT Events, 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 todas as AWS regiões. A trilha registra eventos de todas as regiões na AWS partição e entrega os arquivos de log ao bucket do Amazon S3 que você especificar. Além disso, 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:

- [Criando uma trilha para sua AWS conta](#)
- [Serviços e integrações compatíveis com o CloudTrail](#)
- [Configurando notificações do Amazon SNS para CloudTrail](#)
- [Recebendo arquivos de CloudTrail log de várias regiões](#) e [Recebendo arquivos de CloudTrail log de várias contas](#)

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 o elemento [CloudTrailuserIdentity](#). AWS IoT Events as ações são documentadas na [referência AWS IoT Events da API](#).

Entendendo as entradas do arquivo de AWS IoT Events 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. AWS 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.

Quando o CloudTrail registro está ativado em sua AWS conta, a maioria das chamadas de API feitas para AWS IoT Events ações são rastreadas em arquivos de CloudTrail log, onde são gravadas com outros registros AWS de serviço. CloudTrail determina quando criar e gravar em um novo arquivo com base no período e no tamanho do arquivo.

Cada entrada de log contém informações sobre quem gerou a solicitação. As informações de identidade do usuário na entrada de log ajudam você a determinar o seguinte:

- 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.

É possível armazenar os arquivos de log no seu bucket do Amazon S3 pelo tempo que desejar, mas também pode definir regras do ciclo de vida do Amazon S3 para arquivar ou excluir os arquivos de log automaticamente. Por padrão, os arquivos de log são criptografados com criptografia do lado do servidor (SSE) do Amazon S3.

Para ser notificado sobre a entrega do arquivo de log, você pode configurar CloudTrail para publicar notificações do Amazon SNS quando novos arquivos de log forem entregues. Para obter mais informações, consulte [Configuração de notificações do Amazon SNS para o CloudTrail](#).

Você também pode agregar arquivos de AWS IoT Events log de várias AWS regiões e várias AWS contas em um único bucket do Amazon S3.

Para obter mais informações, consulte [Recebimento de arquivos de CloudTrail log de várias regiões](#) e [Recebimento de arquivos de CloudTrail log de várias contas](#).

Exemplo: DescribeDetector ação para CloudTrail

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a DescribeDetector ação.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE",
    "arn": "arn:aws:sts::123456789012:assumed-role/Admin/bertholt-brecht",
```

```
"accountId": "123456789012",
"accessKeyId": "access-key-id",
"sessionContext": {
  "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2019-02-08T18:53:58Z"
  },
  "sessionIssuer": {
    "type": "Role",
    "principalId": "AKIAI44QH8DHBEXAMPLE",
    "arn": "arn:aws:iam::123456789012:role/Admin",
    "accountId": "123456789012",
    "userName": "Admin"
  }
},
"eventTime": "2019-02-08T19:02:44Z",
"eventSource": "iotevents.amazonaws.com",
"eventName": "DescribeDetector",
"awsRegion": "us-east-1",
"sourceIPAddress": "192.168.0.1",
"userAgent": "aws-cli/1.15.65 Python/3.7.1 Darwin/16.7.0 boto3/1.10.65",
"requestParameters": {
  "detectorModelName": "pressureThresholdEventDetector-brecht",
  "keyValue": "1"
},
"responseElements": null,
"requestID": "00f41283-ea0f-4e85-959f-bee37454627a",
"eventID": "5eb0180d-052b-49d9-a289-0eb8d08d4c27",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}
```

Exemplo: CreateDetectorModel ação para CloudTrail

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a CreateDetectorModel ação.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-Lambda",
```

```

    "arn": "arn:aws:sts::123456789012:assumed-role/IotEvents-RoleForIotEvents-
ABC123DEF456/IotEvents-Lambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABC123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABC123DEF456"
      }
    },
    "eventTime": "2019-02-07T23:54:43Z",
    "eventSource": "iotevents.amazonaws.com",
    "eventName": "CreateDetectorModel",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.168.0.1",
    "userAgent": "aws-internal/3",
    "requestParameters": {
      "detectorModelName": "myDetectorModel",
      "key": "HIDDEN_DUE_TO_SECURITY_REASONS",
      "roleArn": "arn:aws:iam::123456789012:role/events_action_execution_role"
    },
    "responseElements": null,
    "requestID": "cecfbfa1-e452-4fa6-b86b-89a89f392b66",
    "eventID": "8138d46b-50a3-4af0-9c5e-5af5ef75ea55",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
  }
}

```

Exemplo: CreateInput ação para CloudTrail

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a CreateInput ação.

```
{
```

```
"eventVersion": "1.05",
"userIdentity": {
  "type": "AssumedRole",
  "principalId": "AKIAI44QH8DHBEXAMPLE",
  "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABC123DEF456/IotEvents-Lambda",
  "accountId": "123456789012",
  "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
  "sessionContext": {
    "attributes": {
      "mfaAuthenticated": "false",
      "creationDate": "2019-02-07T22:22:30Z"
    },
    "sessionIssuer": {
      "type": "Role",
      "principalId": "AKIAI44QH8DHBEXAMPLE",
      "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABC123DEF456",
      "accountId": "123456789012",
      "userName": "IotEventsLambda-RoleForIotEvents-ABC123DEF456"
    }
  }
},
"eventTime": "2019-02-07T23:54:43Z",
"eventSource": "iotevents.amazonaws.com",
"eventName": "CreateInput",
"awsRegion": "us-east-1",
"sourceIPAddress": "192.168.0.1",
"userAgent": "aws-internal/3",
"requestParameters": {
  "inputName": "batchputmessagedetectorupdated",
  "inputDescription": "batchputmessagedetectorupdated"
},
"responseElements": null,
"requestID": "fb315af4-39e9-4114-94d1-89c9183394c1",
"eventID": "6d8cf67b-2a03-46e6-bbff-e113a7bded1e",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}
```

Exemplo: DeleteDetectorModel ação para CloudTrail

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a DeleteDetectorModel ação.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:54:11Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "DeleteDetectorModel",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "detectorModelName": "myDetectorModel"
  },
  "responseElements": null,
  "requestID": "149064c1-4e24-4160-a5b2-1065e63ee2e4",
  "eventID": "7669db89-dcc0-4c42-904b-f24b764dd808",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

```
}
```

Exemplo: DeleteInput ação para CloudTrail

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a DeleteInput ação.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:54:38Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "DeleteInput",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "errorCode": "ResourceNotFoundException",
  "errorMessage": "Input of name: NoSuchInput not found",
  "requestParameters": {
    "inputName": "NoSuchInput"
  },
  "responseElements": null,
```

```

"requestID": "ce6d28ac-5baf-423d-a5c3-afd009c967e3",
"eventID": "be0ef01d-1c28-48cd-895e-c3ff3172c08e",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Exemplo: DescribeDetectorModel ação para CloudTrail

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a DescribeDetectorModel ação.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AAKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:54:20Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "DescribeDetectorModel",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",
  "userAgent": "aws-internal/3",
  "requestParameters": {
    "detectorModelName": "myDetectorModel"
  }
}

```

```

},
"responseElements": null,
"requestID": "18a11622-8193-49a9-85cb-1fa6d3929394",
"eventID": "1ad80ff8-3e2b-4073-ac38-9cb3385beb04",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Exemplo: DescribeInput ação para CloudTrail

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a DescribeInput ação.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AAKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:56:09Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "DescribeInput",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.168.0.1",

```

```

"userAgent": "aws-internal/3",
"requestParameters": {
  "inputName": "input_createinput"
},
"responseElements": null,
"requestID": "3af641fa-d8af-41c9-ba77-ac9c6260f8b8",
"eventID": "bc4e6cc0-55f7-45c1-b597-ec99aa14c81a",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Exemplo: DescribeLoggingOptions ação para CloudTrail

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a DescribeLoggingOptions ação.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:53:23Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "DescribeLoggingOptions",

```

```

"awsRegion": "us-east-1",
"sourceIPAddress": "192.168.0.1",
"userAgent": "aws-internal/3",
"requestParameters": null,
"responseElements": null,
"requestID": "b624b6c5-aa33-41d8-867b-025ec747ee8f",
"eventID": "9c7ce626-25c8-413a-96e7-92b823d6c850",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Exemplo: ListDetectorModels ação para CloudTrail

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a ListDetectorModels ação.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
  "eventTime": "2019-02-07T23:53:23Z",
  "eventSource": "iotevents.amazonaws.com",
  "eventName": "ListDetectorModels",

```

```

"awsRegion": "us-east-1",
"sourceIPAddress": "192.168.0.1",
"userAgent": "aws-internal/3",
"requestParameters": {
  "nextToken": "CkZEZXRlY3Rvck1vZGVsMl9saXN0ZGV0ZWN0b3Jtb2R1bHN0ZXN0X2VlOWJkZTk1YT",
  "maxResults": 3
},
"responseElements": null,
"requestID": "6d70f262-da95-4bb5-94b4-c08369df75bb",
"eventID": "2d01a25c-d5c7-4233-99fe-ce1b8ec05516",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Exemplo: ListDetectorModelVersions ação para CloudTrail

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a ListDetectorModelVersions ação.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456",
        "accountId": "123456789012",
        "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
      }
    }
  },
}

```

```

"eventTime": "2019-02-07T23:53:33Z",
"eventSource": "iotevents.amazonaws.com",
"eventName": "ListDetectorModelVersions",
"awsRegion": "us-east-1",
"sourceIPAddress": "192.168.0.1",
"userAgent": "aws-internal/3",
"requestParameters": {
  "detectorModelName": "myDetectorModel",
  "maxResults": 2
},
"responseElements": null,
"requestID": "ebecb277-6bd8-44ea-8abd-fbf40ac044ee",
"eventID": "fc6281a2-3fac-4e1e-98e0-ca6560b8b8be",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Exemplo: ListDetectors ação para CloudTrail

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a ListDetectors ação.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      }
    },
    "sessionIssuer": {
      "type": "Role",
      "principalId": "AKIAI44QH8DHBEXAMPLE",
      "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456",
      "accountId": "123456789012",
      "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
    }
  }
}

```

```

    }
  }
},
"eventTime": "2019-02-07T23:53:54Z",
"eventSource": "iotevents.amazonaws.com",
"eventName": "ListDetectors",
"awsRegion": "us-east-1",
"sourceIPAddress": "192.168.0.1",
"userAgent": "aws-internal/3",
"requestParameters": {
  "detectorModelName": "batchputmessagedetectorinstancecreated",
  "stateName": "HIDDEN_DUE_TO_SECURITY_REASONS"
},
"responseElements": null,
"requestID": "4783666d-1e87-42a8-85f7-22d43068af94",
"eventID": "0d2b7e9b-afe6-4aef-afd2-a0bb1e9614a9",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Exemplo: ListInputs ação para CloudTrail

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a ListInputs ação.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      }
    },
    "sessionIssuer": {
      "type": "Role",
      "principalId": "AKIAI44QH8DHBEXAMPLE",
      "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456",
      "accountId": "123456789012",

```

```

    "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
  }
}
},
"eventTime": "2019-02-07T23:53:57Z",
"eventSource": "iotevents.amazonaws.com",
"eventName": "ListInputs",
"awsRegion": "us-east-1",
"sourceIPAddress": "192.168.0.1",
"userAgent": "aws-internal/3",
"requestParameters": {
  "nextToken": "CkhjYW5hcnlfdGVzdF9pbnB1dF9saXN0ZGV0ZWN0b3Jtb2R1bHN0ZXN0ZDU3OGZ",
  "maxResults": 3
},
"responseElements": null,
"requestID": "dd6762a1-1f24-4e63-a986-5ea3938a03da",
"eventID": "c500f6d8-e271-4366-8f20-da4413752469",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Exemplo: PutLoggingOptions ação para CloudTrail

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a PutLoggingOptions ação.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-02-07T22:22:30Z"
      }
    },
    "sessionIssuer": {
      "type": "Role",
      "principalId": "AKIAI44QH8DHBEXAMPLE",

```

```

    "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
    "accountId": "123456789012",
    "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
  }
},
"eventTime": "2019-02-07T23:56:43Z",
"eventSource": "iotevents.amazonaws.com",
"eventName": "PutLoggingOptions",
"awsRegion": "us-east-1",
"sourceIPAddress": "192.168.0.1",
"userAgent": "aws-internal/3",
"requestParameters": {
  "loggingOptions": {
    "roleArn": "arn:aws:iam::123456789012:role/logging__logging_role",
    "level": "INFO",
    "enabled": false
  }
},
"responseElements": null,
"requestID": "df570e50-fb19-4636-9ec0-e150a94bc52c",
"eventID": "3247f928-26aa-471e-b669-e4a9e6fbc42c",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Exemplo: UpdateDetectorModel ação para CloudTrail

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a UpdateDetectorModel ação.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {

```

```

    "mfaAuthenticated": "false",
    "creationDate": "2019-02-07T22:22:30Z"
  },
  "sessionIssuer": {
    "type": "Role",
    "principalId": "AKIAI44QH8DHBEXAMPLE",
    "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
    "accountId": "123456789012",
    "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
  }
}
},
"eventTime": "2019-02-07T23:55:51Z",
"eventSource": "iotevents.amazonaws.com",
"eventName": "UpdateDetectorModel",
"awsRegion": "us-east-1",
"sourceIPAddress": "192.168.0.1",
"userAgent": "aws-internal/3",
"requestParameters": {
  "detectorModelName": "myDetectorModel",
  "roleArn": "arn:aws:iam::123456789012:role/Events_action_execution_role"
},
"responseElements": null,
"requestID": "add29860-c1c5-4091-9917-d2ef13c356cf",
"eventID": "7baa9a14-6a52-47dc-aea0-3cace05147c3",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Exemplo: UpdateInput ação para CloudTrail

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a UpdateInput ação.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:IotEvents-EventsLambda",
    "arn": "arn:aws:sts::123456789012:assumed-role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456/IotEvents-EventsLambda",
    "accountId": "123456789012",

```

```

"accessKeyId": "AKIAI44QH8DHBEXAMPLE",
"sessionContext": {
  "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2019-02-07T22:22:30Z"
  },
  "sessionIssuer": {
    "type": "Role",
    "principalId": "AKIAI44QH8DHBEXAMPLE",
    "arn": "arn:aws:iam::123456789012:role/IotEventsLambda-RoleForIotEvents-
ABCD123DEF456",
    "accountId": "123456789012",
    "userName": "IotEventsLambda-RoleForIotEvents-ABCD123DEF456"
  }
},
"eventTime": "2019-02-07T23:53:00Z",
"eventSource": "iotevents.amazonaws.com",
"eventName": "UpdateInput",
"awsRegion": "us-east-1",
"sourceIPAddress": "192.168.0.1",
"userAgent": "aws-internal/3",
"errorCode": "ResourceNotFoundException",
"errorMessage": "Input of name: NoSuchInput not found",
"requestParameters": {
  "inputName": "NoSuchInput",
  "inputDescription": "this is a description of an input"
},
"responseElements": null,
"requestID": "58d5d2bb-4110-4c56-896a-ee9156009f41",
"eventID": "c2df241a-fd53-4fd0-936c-ba309e5dc62d",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Exemplo: BatchPutMessage ação para CloudTrail

AWS IoT Events pode usar uma CloudTrail integração para o registro da API do plano de dados. Este exemplo adiciona detalhes sobre eventos de dados por meio da BatchPutMessage ação.

```

{
  "eventVersion": "1.09",
  "userIdentity": {

```

```
"type": "AssumedRole",
"principalId": "AKIAI44QH8DHBEXAMPLE:PrincipalId",
"arn": "arn:aws:sts::123456789012:assumed-role/my-iam-role/my-iam-role-
entity",

"accountId": "123456789012",
"accessKeyId": "AKIAI44QH8DHBEXAMPLE",
"sessionContext": {
  "sessionIssuer": {
    "type": "Role",
    "principalId": "AKIAI44QH8DHBEXAMPLE",
    "arn": "arn:aws:iam::123456789012:role/my-iam-role",
    "accountId": "123456789012",
    "userName": "sample_user_name"
  },
  "attributes": {
    "creationDate": "2024-11-22T18:32:41Z",
    "mfaAuthenticated": "false"
  }
}
},
"eventTime": "2024-11-22T18:57:35Z",
"eventSource": "iotevents.amazonaws.com",
"eventName": "BatchPutMessage",
"awsRegion": "us-east-1",
"sourceIPAddress": "3.239.107.128",
"userAgent": "aws-internal/3",
"requestParameters": {
  "messages": [
    {
      "messageId": "e306d827-b2e4-4439-9c86-411d4242a397",
      "payload": "HIDDEN_DUE_TO_SECURITY_REASONS",
      "inputName": "my_input_name"
    }
  ]
},
"responseElements": {
  "batchPutMessageErrorEntries": []
},
"requestID": "cefc6b63-9ccf-4e31-9177-4aec8e701bfe",
"eventID": "b994b52c-6011-4e3c-ad5f-e784e732fde0",
"readOnly": false,
"resources": [
  {
    "accountId": "123456789012",
```

```
        "type": "AWS::IoTEvents::Input",
        "ARN": "arn:aws:iotevents:us-east-1:123456789012:input/
my_input_name"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "123456789012",
  "eventCategory": "Data",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "iotevents.us-east-1.amazonaws.com"
  }
},
```

Validação de conformidade para AWS IoT Events

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 em AWS IoT Events

A infraestrutura AWS global é construída em torno de AWS regiões e zonas de disponibilidade. AWS As regiões fornecem várias zonas de disponibilidade separadas e isoladas fisicamente, as quais são conectadas com baixa latência, alto throughput e redes 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 AWS regiões e zonas de disponibilidade, consulte [infraestrutura AWS global](#).

Segurança da infraestrutura em AWS IoT Events

Como serviço gerenciado, AWS IoT Events é protegido pela segurança de rede AWS global. Para obter informações sobre serviços AWS de segurança e como AWS proteger a infraestrutura, consulte [AWS Cloud Security](#). Para projetar seu AWS ambiente usando as melhores práticas de segurança de infraestrutura, consulte [Proteção](#) de infraestrutura no Security Pillar AWS Well-Architected Framework.

Você usa chamadas de API AWS publicadas para acessar AWS IoT Events pela rede. Os clientes devem oferecer compatibilidade com:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com perfect forward secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

AWS cotas de serviço para recursos AWS IoT Events

O Referência geral da AWS Guia fornece as cotas padrão AWS IoT Events para uma AWS conta. A menos que especificado, cada cota é por AWS região. Para obter mais informações, consulte [Endpoints e cotas do AWS IoT Events](#) e [AWS Service Quotas](#) no Guia Referência geral da AWS .

Para solicitar um aumento de Service Quotas, envie um caso de suporte no console da [Central de suporte](#). Para obter mais informações, consulte [Solicitando um Aumento de Cota](#) no Guia do usuário do Service Quotas.

Note

- Todos os nomes dos modelos e entradas de detectores devem ser exclusivos em uma conta.
- Não é possível alterar os nomes dos modelos e entradas dos detectores depois de criados.

Marcando seus recursos AWS IoT Events

Para ajudar no gerenciamento e na organização de suas instâncias do sistema, também é possível atribuir seus próprios metadados a cada um desses recursos na forma de tags. Esta seção descreve tags e mostra a você como criá-las.

Conceitos Básicos de Tags

As tags permitem que você categorize seus AWS IoT Events recursos de maneiras diferentes, por exemplo, por finalidade, proprietário ou ambiente. Isso é útil quando você tem muitos recursos do mesmo tipo. Identifique rapidamente um recurso específico com base nas tags atribuídas 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 as suas entradas que ajuda a rastrear os dispositivos que enviam essas entradas por seu 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.

É possível pesquisar e filtrar recursos com base nas tags adicionadas ou aplicadas, usar tags para categorizar e monitorar seus custos e também usar tags para controlar o acesso aos seus recursos, conforme descrito em [Uso de tags com políticas do IAM](#) no Guia do desenvolvedor do AWS IoT .

Para facilitar o uso, o Editor de tags no Console de gerenciamento da AWS fornece uma maneira central e unificada de criar e gerenciar suas tags. Para obter mais informações, consulte [Introdução ao Editor de tags](#) no Guia do usuário dos AWS Recursos de marcação e do editor de tags.

Você também pode trabalhar com tags usando a AWS CLI e a AWS IoT Events API. É possível associar tags a modelos de detector e entradas quando você as cria usando o campo "Tags" nos seguintes comandos:

- [CreateDetectorModel](#)
- [CreateInput](#)

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.

Para obter mais informações, consulte [Melhores práticas para marcar recursos AWS](#)

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
- Comprimento máximo do valor: 255 caracteres Unicode em UTF-8
- As chaves e valores das tags diferenciam maiúsculas de minúsculas.
- Não use o "aws :" prefixo nos nomes ou valores das tags porque ele 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.
- Caso seu esquema de marcação seja usado em vários serviços e recursos da , lembre-se de que outros serviços podem possuir restrições em caracteres permitidos. No geral, os caracteres permitidos são 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 Events . 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

Os valores e as chaves de contexto de condição em uma política do IAM se aplicam somente às ações do AWS IoT Events em que um identificador de um recurso que pode ser marcado com tags é um parâmetro obrigatório.

[Controle de acesso usando tags](#) no Guia do usuário do AWS Identity and Access Management tem informações adicionais sobre o uso de tags. 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.

A política de exemplo a seguir aplica duas restrições com base em tag. Um usuário restrito por essa política:

- Não é possível atribuir um recurso à tag "env=prod" (no exemplo, consulte a linha `"aws:RequestTag/env" : "prod"`)
- Não é possível modificar ou acessar um recurso que tenha uma tag "env=prod" (no exemplo, consulte a linha `"aws:ResourceTag/env" : "prod"`).

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
```

```

        "iotevents:CreateDetectorModel",
        "iotevents:CreateAlarmModel",
        "iotevents:CreateInput",
        "iotevents:TagResource"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/env": "prod"
        }
    }
},
{
    "Effect": "Deny",
    "Action": [
        "iotevents:DescribeDetectorModel",
        "iotevents:DescribeAlarmModel",
        "iotevents:UpdateDetectorModel",
        "iotevents:UpdateAlarmModel",
        "iotevents>DeleteDetectorModel",
        "iotevents>DeleteAlarmModel",
        "iotevents:ListDetectorModelVersions",
        "iotevents:ListAlarmModelVersions",
        "iotevents:UpdateInput",
        "iotevents:DescribeInput",
        "iotevents>DeleteInput",
        "iotevents:ListTagsForResource",
        "iotevents:TagResource",
        "iotevents:UntagResource",
        "iotevents:UpdateInputRouting"
    ],
    "Resource": "*",
    "Condition": {
        "StringLike": {
            "aws:ResourceTag/env": "prod"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "iotevents:*"
    ],
    "Resource": "*"
}

```

```
    ]  
  }  
}
```

Também é possível especificar vários valores de tags para uma determinada chave de tag, colocando-os em uma lista, conforme mostrado.

```
"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.

Solução de problemas AWS IoT Events

Este guia de solução de problemas fornece soluções para problemas comuns que você pode encontrar ao usar AWS IoT Events. Navegue pelos tópicos para identificar e resolver problemas com detecção de eventos, acesso a dados, permissões, integrações de serviços, configurações de dispositivos e muito mais. Com recomendações de solução de problemas para AWS IoT Events console, API, CLI, erros, latência e integrações, este guia tem como objetivo resolver rapidamente seus problemas para que você possa criar aplicativos orientados a eventos confiáveis e escaláveis.

Tópicos

- [AWS IoT Events Problemas e soluções comuns](#)
- [Solução de problemas de um modelo de detector executando análises em AWS IoT Events](#)

AWS IoT Events Problemas e soluções comuns

Consulte a seção a seguir para solucionar erros e encontrar possíveis soluções para resolver problemas com AWS IoT Events.

Erros

- [Erros na criação do modelo de detector](#)
- [Atualizações de um modelo de detector excluído](#)
- [Falha no gatilho da ação \(ao atender a uma condição\)](#)
- [Falha no gatilho de ação \(ao ultrapassar um limite\)](#)
- [Uso incorreto de estados](#)
- [Mensagem de conexão](#)
- [InvalidRequestException mensagem](#)
- [action.setTimerErros do Amazon CloudWatch Logs](#)
- [Erros de CloudWatch carga útil da Amazon](#)
- [Tipos de dados incompatíveis](#)
- [Falha ao enviar mensagem para AWS IoT Events](#)

Erros na criação do modelo do detector

Eu recebo erros quando tento criar um modelo de detector.

Solução

Ao criar um modelo de detector, considere as limitações a seguir.

- Somente uma ação é permitida em cada campo `action`.
- O `condition` é exigido para `transitionEvents`. É opcional para eventos `OnEnter`, `OnInput` e `OnExit`.
- Se o campo `condition` estiver vazio, o resultado avaliado da expressão da condição será equivalente a `true`.
- O resultado avaliado da expressão da condição deverá ser um valor booleano. Se o resultado não for um valor booleano, ele é equivalente ao `false` e não aciona o gatilho `actions` ou a transição para o `nextState` especificado no evento.

Para obter mais informações, consulte [AWS IoT Events restrições e limitações do modelo de detector](#).

Atualizações de um modelo de detector excluído

Eu atualizei ou excluí um modelo de detector há alguns minutos, mas ainda estou recebendo atualizações de estado do modelo de detector antigo por meio de mensagens MQTT ou alertas de SNS.

Solução

Se você atualizar, excluir ou recriar um modelo de detector (consulte [UpdateDetectorModel](#)), haverá um atraso até que todas as instâncias do detector sejam excluídas e o novo modelo seja usado. Durante esse período, as entradas podem continuar sendo processadas pelas instâncias da versão anterior do modelo do detector. Você pode continuar recebendo alertas definidos pelo modelo de detector anterior. Aguarde pelo menos sete minutos antes de verificar novamente a atualização ou relatar um erro.

Falha no gatilho da ação (ao atender a uma condição)

O detector falha em acionar uma ação ou transição para um novo estado quando a condição é atendida.

Solução

Verifique se o resultado avaliado da expressão condicional do detector é um valor booleano. Se o resultado não for um valor booleano, ele é equivalente ao `false` e não aciona o gatilho `action` ou a transição para o `nextState` especificado no evento. Para obter mais informações, consulte [Sintaxe de expressão condicional](#).

Falha no gatilho de ação (ao ultrapassar um limite)

O detector não aciona uma ação ou uma transição de evento quando a variável em uma expressão condicional atinge um valor especificado.

Solução

Se você atualizar `setVariable` para `onInput`, `onEnter` ou `onExit`, o novo valor não será usado ao avaliar nenhuma `condition` durante o ciclo de processamento corrente. Em vez disso, o valor original é usado até que o ciclo atual seja concluído. É possível alterar esse comportamento configurando o parâmetro `evaluationMethod` na definição de modelos de detectores. Quando `evaluationMethod` é definido para `SERIAL`, as variáveis são atualizadas e as condições do evento avaliadas na ordem em que os eventos são definidos. Quando `evaluationMethod` é definido para `BATCH` (o padrão), as variáveis são atualizadas e os eventos são executados somente após todas as condições do evento serem avaliadas.

Uso incorreto de estados

O detector entra nos estados errados quando eu tento enviar mensagens para as entradas usando `BatchPutMessage`.

Solução

Se você costuma [BatchPutMessage](#) enviar várias mensagens para entradas, a ordem na qual as mensagens ou entradas são processadas não é garantida. Para garantir o pedido, envie mensagens uma de cada vez e espere cada vez que o `BatchPutMessage` reconheça o sucesso.

Mensagem de conexão

Recebo um erro (`'Connection aborted.'`, `error(54, 'Connection reset by peer')`) quando tento chamar ou invocar uma API.

Solução

Verifique se o OpenSSL usa o TLS 1.1 ou uma versão posterior para estabelecer a conexão. Esse deve ser o padrão na maioria das distribuições Linux ou no Windows versão 7 e posterior. Os usuários do macOS talvez precisem atualizar o OpenSSL.

InvalidRequestException mensagem

Eu recebo `InvalidRequestException` quando tento ligar `CreateDetectorModel` `UpdateDetectorModel` APIs e.

Solução

Verifique o seguinte, para ajudar a resolver esse problema. Para obter mais informações, consulte [CreateDetectorModel](#) e [UpdateDetectorModel](#).

- Certifique-se de não usar `seconds` e `durationExpression` como parâmetros de `SetTimerAction` ao mesmo tempo.
- Certifique-se de que sua expressão de string para que `durationExpression` seja válida. A expressão em cadeia de caracteres pode conter números, variáveis (`$variable.<variable-name>`) ou valores de entrada (`$input.<input-name>.<path-to-datum>`).

action.setTimer Erros do Amazon CloudWatch Logs

Você pode configurar o Amazon CloudWatch Logs para monitorar instâncias do modelo de AWS IoT Events detector. A seguir estão os erros comuns gerados por AWS IoT Events, quando você usa `action.setTimer`.

- Erro: a sua expressão de duração para o temporizador chamado `<timer-name>` não pôde ser avaliada como um número.

Solução

Certifique-se de que sua expressão de string para `durationExpression` possa ser convertida em um número. Outros tipos de dados, como booleano, não são permitidos.

- Erro: o resultado avaliado da sua expressão de duração para o temporizador chamado `<timer-name>` é maior que 31622440. Para garantir a precisão, certifique-se de que a sua expressão de duração se refira a um valor entre 60-31622400.

Solução

Certifique-se de que a duração de seu temporizador seja menor ou igual a 31622400 segundos. O resultado avaliado da duração é arredondado para baixo para o número inteiro mais próximo.

- Erro: o resultado avaliado da sua expressão de duração para o temporizador chamado `<timer-name>` é menor que 60. Para garantir a precisão, certifique-se de que a sua expressão de duração se refira a um valor entre 60-31622400.

Solução

Certifique-se de que a duração de seu temporizador seja maior ou igual a 60 segundos. O resultado avaliado da duração é arredondado para baixo para o número inteiro mais próximo.

- Erro: sua expressão de duração para o temporizador chamado `<timer-name>` não pôde ser avaliada. Verifique os nomes das variáveis, nomes de entrada e caminhos para os dados para garantir que você se refira às variáveis e entradas existentes.

Solução

Certifique-se de que sua expressão de string se refira às variáveis e entradas existentes. A expressão de string pode conter números, variáveis (`$variable.variable-name`) e valores de entrada (`$input.input-name.path-to-datum`).

- Erro: falha ao definir o temporizador chamado `<timer-name>`. Confira a expressão de duração e tente novamente.

Solução

Veja a [SetTimerAction](#) ação para garantir que você especificou os parâmetros corretos e, em seguida, ajuste o cronômetro novamente.

Para obter mais informações, consulte [Habilitar o CloudWatch registro na Amazon ao desenvolver modelos AWS IoT Events de detectores](#).

Erros de CloudWatch carga útil da Amazon

Você pode configurar o Amazon CloudWatch Logs para monitorar instâncias do modelo de AWS IoT Events detector. A seguir estão os erros e avisos comuns gerados por AWS IoT Events, quando você configura a carga útil da ação.

- Erro: não foi possível avaliar sua expressão para a ação. Verifique se os nomes das variáveis, os nomes de entrada e os caminhos para os dados se referem às variáveis e aos valores de entrada existentes. Além disso, verifique se o tamanho da carga é menor que 1 KB, o tamanho máximo permitido de uma carga útil.

Solução

Certifique-se de inserir os nomes corretos das variáveis, nomes de entrada e caminhos para os dados. Você também pode receber essa mensagem de erro se a carga da ação for maior que 1 KB.

- Erro: não foi possível analisar sua expressão de conteúdo para a carga útil de `<action-type>`. Insira uma expressão de conteúdo com a sintaxe correta.

Solução

É possível usar uma expressão de string que contenha strings (`'string'`), variáveis (`$variable.variable-name`), valores de entrada (`$input.input-name.path-to-datum`), concatenações de string e strings que contêm `${}`.

- Erro: sua expressão de carga `{expression}` não é válida. O tipo de carga útil definido é JSON, então você deve especificar uma expressão que AWS IoT Events seria avaliada como uma string.

Solução

Se o tipo de carga útil especificado for JSON, AWS IoT Events primeiro verifique se o serviço pode avaliar sua expressão como uma string. O resultado avaliado não pode ser um booleano ou um número. Se a validação falhar, você poderá receber esse erro.

- Aviso: a ação foi executada, mas não foi possível avaliar sua expressão de conteúdo para que a carga da ação fosse um JSON válido. O tipo de carga útil definido é JSON.

Solução

Certifique-se de que AWS IoT Events pode avaliar sua expressão de conteúdo para a carga de ação como JSON válido, se você definir o tipo de carga como. JSON AWS IoT Events executa a ação mesmo que não AWS IoT Events consiga avaliar a expressão de conteúdo como JSON válido.

Para obter mais informações, consulte [Habilitar o CloudWatch registro na Amazon ao desenvolver modelos AWS IoT Events de detectores](#).

Tipos de dados incompatíveis

Mensagem: tipos de dados incompatíveis [<inferred-types>] encontrados para <reference> na seguinte expressão: <expression>

Solução

Você pode receber um erro por um dos motivos a seguir:

- Os resultados avaliados de suas referências não são compatíveis com outros operandos em suas expressões.
- O tipo do argumento passado para uma função não é compatível.

Ao usar referências em expressões, verifique o seguinte:

- Ao usar uma referência como operando com um ou mais operadores, verifique se todos os tipos de dados referenciados são compatíveis.

Por exemplo, na expressão a seguir, o inteiro 2 é um operando dos operadores == e &&. Para garantir que os operandos sejam compatíveis, `$variable.testVariable + 1` e `$variable.testVariable` devem referenciar um número inteiro ou decimal.

Além disso, o inteiro 1 é um operando do operador +. Portanto, `$variable.testVariable` deve fazer referência a um número inteiro ou decimal.

```
'$variable.testVariable + 1 == 2 && $variable.testVariable'
```

- Ao usar uma referência como argumento passado para uma função, verifique se a função é compatível com os tipos de dados aos quais você faz referência.

Por exemplo, a função a seguir `timeout("time-name")` requer uma string com aspas duplas como argumento. Se você usar uma referência para o `timer-name` valor, deverá referenciar uma string com aspas duplas.

```
timeout("timer-name")
```

Note

Para a `convert(type, expression)` função, se você usar uma referência para o *type* valor, o resultado avaliado da sua referência deverá ser `String`, `Decimal`, ou `Boolean`.

Para obter mais informações, consulte [AWS IoT Events referência para entradas e variáveis em expressões](#).

Falha ao enviar mensagem para AWS IoT Events

Mensagem: falha ao enviar mensagem para IoT Events

Solução

Você poderá ter esse erro pelos seguintes motivos:

- A carga útil de mensagem de entrada não contém `Input attribute Key`.
- O `Input attribute Key` não está no mesmo caminho JSON especificado na definição de entrada.
- A mensagem de entrada não corresponde ao esquema, conforme definido na AWS IoT Events entrada.

Note

A ingestão de dados de outros serviços também apresentará falhas.

Example

Por exemplo AWS IoT Core, em, a AWS IoT regra falhará com a seguinte mensagem `Verify the Input Attribute key`.

Para resolver isso, certifique-se de que o esquema da mensagem de carga útil de entrada esteja em conformidade com a definição de AWS IoT Events entrada e que a localização corresponda `Input attribute Key`. Para obter mais informações, consulte [Crie uma entrada para modelos em AWS IoT Events](#) para saber como definir AWS IoT Events entradas.

Solução de problemas de um modelo de detector executando análises em AWS IoT Events

AWS IoT Events pode analisar seu modelo de detector e gerar resultados de análise sem enviar dados de entrada para seu modelo de detector. AWS IoT Events executa uma série de análises descritas nesta seção para verificar seu modelo de detector. Essa solução avançada de solução de problemas também resume as informações de diagnóstico, incluindo o nível de gravidade e a localização, para que você possa encontrar e corrigir rapidamente possíveis problemas em seu modelo de detector. Para obter mais informações sobre tipos de erros de diagnóstico e mensagens para seu modelo de detector, consulte [Análise do modelo de detector e informações de diagnóstico para AWS IoT Events](#).

Você pode usar o AWS IoT Events console, a [API](#), [AWS Command Line Interface \(AWS CLI\)](#) ou o [AWS SDK](#) para visualizar mensagens de erro de diagnóstico da análise do seu modelo de detector.

Note

- Você deve corrigir todos os erros antes de publicar o seu modelo de detector.
- Recomendamos que você analise os avisos e tome as medidas necessárias antes de usar seu modelo de detector em ambientes de produção. Caso contrário, o modelo do detector pode não funcionar conforme o esperado.
- É possível ter até 10 análises no status RUNNING ao mesmo tempo.

Para saber como analisar seu modelo de detector, consulte [Análise um modelo de detector para AWS IoT Events \(Console\)](#) ou [Análise um modelo de detector em AWS IoT Events \(AWS CLI\)](#).

Tópicos

- [Análise do modelo de detector e informações de diagnóstico para AWS IoT Events](#)
- [Análise um modelo de detector para AWS IoT Events \(Console\)](#)
- [Análise um modelo de detector em AWS IoT Events \(AWS CLI\)](#)

Análise do modelo de detector e informações de diagnóstico para AWS IoT Events

As análises do modelo do detector reúnem as seguintes informações de diagnóstico:

- **Nível:** o nível de severidade do resultado da análise. Com base no nível de gravidade, os resultados da análise se enquadram em três categorias gerais:
 - **Informação (INFO):** um resultado de informação informa sobre um campo significativo em seu modelo de detector. Esse tipo de resultado geralmente não requer ação imediata.
 - **Aviso (WARNING):** um resultado de aviso chama atenção especial para campos que podem causar problemas em seu modelo de detector. Recomendamos que você analise os avisos e tome as medidas necessárias antes de usar seu modelo de detector em ambientes de produção. Caso contrário, o modelo do detector pode não funcionar conforme o esperado.
 - **Error (ERROR):** um resultado de erro notifica você sobre um problema encontrado em seu modelo de detector. O AWS IoT Events executa automaticamente esse conjunto de análises quando você tenta publicar o modelo do detector. Você deve corrigir todos os erros antes de publicar o modelo do detector.
- **Localização:** contém informações que podem ser usadas para localizar o campo em seu modelo de detector ao qual o resultado da análise faz referência. Um local normalmente inclui o nome do estado, o nome do evento de transição, o nome do evento e a expressão (por exemplo, `in state TemperatureCheck in onEnter in event Init in action setVariable`).
- **Tipo:** o tipo do resultado da análise. Os tipos de análise entram nas seguintes categorias:
 - **supported-actions**— AWS IoT Events pode invocar ações quando um evento específico ou evento de transição é detectado. É possível definir ações integradas para usar um temporizador, definir uma variável ou enviar dados para outros serviços da AWS. Você deve especificar ações que funcionem com outros AWS serviços em uma AWS região em que os AWS serviços estejam disponíveis.
 - **service-limits**— As cotas de serviço, também conhecidas como limites, são o número máximo ou mínimo de recursos ou operações de serviço para sua AWS conta. A menos que especificado de outra forma, cada cota é específica para a região. Dependendo das necessidades de sua empresa, é possível atualizar seu modelo de detector para evitar limites ou solicitar um aumento de cota. Você pode solicitar aumentos para algumas cotas e outras cotas não podem ser aumentadas. Para mais informações, consulte [Cotas do](#).
- **structure:** o modelo do detector deve ter todos os componentes necessários, como estados, e seguir uma estrutura compatível com o AWS IoT Events. Um modelo de detector deve ter

pelo menos um estado e uma condição que avalie os dados de entrada recebidos para detectar eventos significativos. Quando um evento é detectado, o modelo do detector passa para o próximo estado e pode invocar ações. Esses eventos são conhecidos como eventos de transição. Um evento de transição deve direcionar o próximo estado a ser inserido.

- **expression-syntax:** o AWS IoT Events fornece várias maneiras de especificar valores ao criar e atualizar modelos de detectores. É possível usar literais, operadores, funções, referências e modelos de substituição nas expressões. Você pode usar expressões para especificar valores literais ou AWS IoT Events avaliar as expressões antes de especificar valores específicos. Sua expressão deve seguir a sintaxe necessária. Para obter mais informações, consulte [Expressões para filtrar, transformar e processar dados de eventos](#).

As expressões do modelo Detector AWS IoT Events podem fazer referência a dados específicos ou a um recurso.

- **data-type:** o AWS IoT Events é compatível com tipos de dados inteiros, decimais, strings e booleanos. Se AWS IoT Events puder converter automaticamente os dados de um tipo de dados em outro durante a avaliação da expressão, esses tipos de dados são compatíveis.

Note

- Inteiro e decimal são os únicos tipos de dados compatíveis suportados pelo AWS IoT Events.
 - AWS IoT Events não é possível avaliar expressões aritméticas porque não é AWS IoT Events possível converter um inteiro em uma string.
- **referenced-data:** é preciso definir os dados referenciados em seu modelo de detector antes de poder usar os dados. Por exemplo, se quiser enviar dados para uma tabela do DynamoDB, você deve definir uma variável que faça referência ao nome da tabela antes de poder usar a variável em uma expressão (`$variable.TableName`).
 - **referenced-resource:** os recursos que o modelo do detector usa devem estar disponíveis. É preciso definir recursos para poder usá-los. Por exemplo, você deseja criar um modelo de detector para monitorar a temperatura de uma estufa. É preciso definir uma entrada (`$input.TemperatureInput`) para rotear os dados de temperatura recebidos para o modelo do detector antes de poder usar o `$input.TemperatureInput.sensorData.temperature` para referenciar a temperatura.

Consulte a seção a seguir para solucionar erros e encontrar possíveis soluções a partir da análise do seu modelo de detector.

Solucione erros do modelo do detector em AWS IoT Events

Os tipos de erros descritos acima fornecem informações de diagnóstico sobre um modelo de detector e correspondem às mensagens que você pode recuperar. Use essas mensagens e soluções sugeridas para solucionar erros com seu modelo de detector.

Mensagens e soluções

- [Location](#)
- [supported-actions](#)
- [service-limits](#)
- [structure](#)
- [expression-syntax](#)
- [data-type](#)
- [referenced-data](#)
- [referenced-resource](#)

Location

Um resultado de análise com informações sobre a `Location` corresponde à seguinte mensagem de erro:

- Mensagem: contém informações adicionais sobre o resultado da análise. Isso pode ser uma informação, um aviso ou uma mensagem de erro.

Solução: você pode receber essa mensagem de erro se tiver especificado uma ação que o AWS IoT Events atualmente não é compatível. Para ver uma de ações compatíveis, consulte [Ações suportadas para receber dados e acionar ações em AWS IoT Events](#).

supported-actions

Um resultado de análise com informações sobre `supported-actions`, corresponde às seguintes mensagens de erro:

- Mensagem: Tipo de ação inválido presente na definição da ação: *action-definition*.

Solução: você pode receber essa mensagem de erro se tiver especificado uma ação que o AWS IoT Events atualmente não é compatível. Para ver uma de ações compatíveis, consulte [Ações suportadas para receber dados e acionar ações em AWS IoT Events](#).

- Mensagem: a DetectorModel definição tem uma *aws-service* ação, mas o *aws-service* serviço não é suportado na região *region-name*.

Solução: Você pode receber essa mensagem de erro se a ação especificada for suportada pelo AWS IoT Events, mas a ação não estiver disponível na sua região atual. Isso pode ocorrer quando você tenta enviar dados para um AWS serviço que não está disponível na região. Você também deve escolher a mesma região para ambas AWS IoT Events e para os AWS serviços que está usando.

service-limits

Um resultado de análise com informações sobre `service-limits`, corresponde às seguintes mensagens de erro:

- Mensagem: A expressão de conteúdo permitida na carga útil excedeu o limite de *content-expression-size* bytes no estado do evento *event-name*. *state-name*

Solução: você poderá receber essa mensagem de erro se a expressão de conteúdo do sua carga útil for maior que 1024 bytes. O tamanho da expressão de conteúdo de uma carga pode ser de até 1024 bytes.

- Mensagem: O número de estados permitidos na definição do modelo do detector excedeu o limite *states-per-detector-model*.

Solução: você pode receber essa mensagem de erro se o modelo do detector tiver mais de 20 estados. Um modelo de detector pode ter até 20 estados.

- Mensagem: A duração do cronômetro *timer-name* deve ser de pelo menos *minimum-timer-duration* segundos.

Solução: você pode receber essa mensagem de erro se a duração do temporizador for inferior a 60 segundos. Recomendamos que a duração de um temporizador esteja entre 60 e 31622400 segundos. Se você especificar uma expressão para a duração do temporizador, o resultado avaliado da expressão de duração será arredondado para baixo para o número inteiro mais próximo.

- Mensagem: O número de ações permitidas por evento excedeu o limite *actions-per-event* na definição do modelo do detector

Solução: você pode receber essa mensagem de erro se o evento tiver mais de 10 ações. Você pode ter até dez ações para cada evento no seu modelo de detector.

- Mensagem: O número de eventos de transição permitidos por estado excedeu o limite *transition-events-per-state* na definição do modelo do detector.

Solução: você pode receber essa mensagem de erro se o estado tiver mais de 20 eventos de transição. Você pode ter até 20 eventos de transição para cada estado no seu modelo de detector.

- Mensagem: O número de eventos permitidos por estado excedeu o limite *events-per-state* na definição do modelo do detector

Solução: você pode receber essa mensagem de erro se o estado tiver mais de 20 eventos. Você pode ter até 20 eventos para cada estado no seu modelo de detector.

- Mensagem: o número máximo de modelos de detector que podem ser associados a uma única entrada. *input-name* A entrada é usada em rotas *detector-models-per-input* de modelos de detectores.

Solução: você pode receber essa mensagem de aviso se tentar rotear uma entrada para mais de 10 modelos de detectores. Você pode ter até 10 modelos de detectores diferentes associados a um único modelo de detector.

structure

Um resultado de análise com informações sobre `structure`, corresponde às seguintes mensagens de erro:

- Mensagem: As ações podem ter apenas um tipo definido, mas encontraram uma ação com *number-of-types* tipos. Divida em ações separadas.

Solução: você pode receber essa mensagem de erro se tiver especificado duas ou mais ações em um único campo usando operações de API para criar ou atualizar o seu modelo de detector. É possível definir uma matriz de objetos de `Action`. Certifique-se de definir cada ação como um objeto separado.

- Mensagem: As `TransitionEvent` *transition-event-name* transições para um estado inexistente. *state-name*

Solução: você pode receber essa mensagem de erro se não AWS IoT Events conseguir encontrar o próximo estado ao qual seu evento de transição fez referência. Verifique se o próximo estado está definido e se você inseriu o nome correto do estado.

- Mensagem: Eles DetectorModelDefinition tinham um nome de estado compartilhado: estado encontrado *state-name* com *number-of-states* repetições.

Solução: você pode receber essa mensagem de erro se usar o mesmo nome para um ou mais estados. Certifique-se de dar um nome exclusivo a cada estado em seu modelo de detector. Esse nome do estado deve ter de 1 a 128 caracteres. Os caracteres válidos são a-z, A-Z, 0-9, _ (sublinhado) e - (hífen).

- Mensagem: A definição initialStateName *initial-state-name* não correspondia a um estado definido.

Solução: você pode receber essa mensagem de erro se o nome do estado inicial estiver incorreto. O modelo do detector permanece no estado inicial (início) até que uma entrada chegue. Quando uma entrada chega, o modelo do detector passa imediatamente para o próximo estado. Verifique se o nome do estado inicial seja o nome de um estado definido e se você inseriu o nome correto.

- Mensagem: a definição do modelo do detector deve usar pelo menos uma entrada em uma condição.

Solução: você pode receber esse erro se não tiver especificado uma entrada em uma condição. É necessário usar pelo menos uma entrada em pelo menos uma condição. Caso contrário, AWS IoT Events não avalia os dados recebidos.

- Mensagem: Somente um de seconds e DurationExpression podem ser configurados. SetTimer

Solução: Você pode receber essa mensagem de erro se tiver usado seconds e durationExpression para o seu temporizador. Certifique-se de usar um seconds ou durationExpression como parâmetros de SetTimerAction. Para obter mais informações, consulte [SetTimerAction](#) na Referência de APIs do AWS IoT Events .

- Mensagem: uma ação em seu modelo de detector está inacessível. Verifique a condição que inicia a ação.

Solução: se uma ação em seu modelo de detector estiver inacessível, a condição do evento será avaliada como falsa. Verifique a condição do evento que contém a ação para garantir que ela seja avaliada como verdadeira. Quando a condição do evento é avaliada como verdadeira, a ação deve se tornar acessível.

- Mensagem: um atributo de entrada está sendo lido, mas isso pode ser causado pela expiração do temporizador.

Solução: o valor de um atributo de entrada pode ser lido quando ocorre uma das seguintes situações:

- Um novo valor de entrada foi recebido.
- Quando um temporizador no detector expirar.

Para garantir que um atributo de entrada seja avaliado somente quando o novo valor dessa entrada for recebido, inclua uma chamada para a função `triggerType("Message")` em sua condição da seguinte maneira:

A condição original que está sendo avaliada no modelo do detector:

```
if ($input.HeartBeat.status == "OFFLINE")
```

O relatório ficaria semelhante ao seguinte:

```
if ( triggerType("MESSAGE") && $input.HeartBeat.status == "OFFLINE")
```

onde uma chamada para a função `triggerType("Message")` vem antes da entrada inicial fornecida na condição. Ao usar essa técnica, a função `triggerType("Message")` será avaliada como verdadeira e satisfará a condição de receber um novo valor de entrada. Para obter mais informações sobre o uso da função `triggerType`, pesquise `triggerType` na seção [Expressões](#) no Guia do desenvolvedor do AWS IoT Events

- Mensagem: um estado em seu modelo de detector está inacessível. Verifique a condição que causará uma transição para o estado desejado.

Solução: se um estado em seu modelo de detector estiver inacessível, uma condição que causa uma transição de entrada para esse estado será avaliada como falsa. Verifique se as condições das transições de entrada para esse estado inacessível em seu modelo de detector são avaliadas como verdadeiras, para que o estado desejado possa se tornar acessível.

- Mensagem: um temporizador expirando pode causar o envio de uma quantidade inesperada de mensagens.

Solução: para evitar que o modelo do detector entre em um estado infinito de envio de uma quantidade inesperada de mensagens porque o temporizador expirou, considere usar uma

chamada para a função `triggerType("Message")`, nas condições do modelo do detector da seguinte forma:

A condição original que está sendo avaliada no modelo do detector:

```
if (timeout("awake"))
```

seria transformada em uma condição semelhante à seguinte:

```
if (triggerType("MESSAGE") && timeout("awake"))
```

onde uma chamada para a função `triggerType("Message")` vem antes da entrada inicial fornecida na condição.

Essa alteração impede o início de ações do temporizador em seu detector, evitando o envio de um loop infinito de mensagens. Para obter mais informações sobre como usar as ações do temporizador em seu detector, consulte a página [Usar ações integradas](#) do Guia do desenvolvedor do AWS IoT Events

expression-syntax

Um resultado de análise com informações sobre `expression-syntax`, corresponde às seguintes mensagens de erro:

- Mensagem: Sua expressão de carga `{expression}` não é válida. O tipo de carga útil definido é JSON, então você deve especificar uma expressão que AWS IoT Events seria avaliada como uma string.

Solução: se o tipo de carga útil especificado for JSON, AWS IoT Events primeiro verifique se o serviço pode avaliar sua expressão como uma string. O resultado avaliado não pode ser um booleano ou um número. Se a validação não for bem-sucedida, você poderá receber esse erro.

- Mensagem: `SetVariableAction.value` deve ser uma expressão. Falha ao analisar o valor "`variable-value`"

Solução: é possível usar `SetVariableAction` para definir uma variável com um `name` e `value`. `value` pode ser uma string, número ou valor booleano. Também é possível especificar uma expressão para o `value`. Para obter mais informações [SetVariableAction](#), consulte a Referência AWS IoT Events da API.

- Mensagem: Não foi possível analisar sua expressão dos atributos (*attribute-name*) para a ação do DynamoDB. Insira a expressão com a sintaxe correta.

Solução: é necessário usar expressões para todos os parâmetros nos modelos de substituição DynamoDBAction. Para obter mais informações, consulte [Dynamo DBAction](#) na Referência da AWS IoT Events API.

- Mensagem: Não foi possível analisar sua expressão do TableName para a ação do Dynamo. DBv2 Insira a expressão com a sintaxe correta.

Solução: a entrada tableName em DynamoDBv2Action deve ser uma string. Você deve usar uma expressão para o tableName. As expressões aceitam literais, operadores, funções, referências e modelos de substituição. Para obter mais informações, consulte [Dynamo DBv2 Action](#) na Referência da AWS IoT Events API.

- Mensagem: não foi possível avaliar a sua expressão como um JSON válido. A DBv2 ação do Dynamo só é compatível com o tipo de carga JSON.

Solução: o tipo de carga útil para DynamoDBv2 deve ser JSON. Certifique-se de que AWS IoT Events possa avaliar sua expressão de conteúdo para a carga útil em JSON válido. Para obter mais informações, consulte [Dynamo DBv2 Action](#), na Referência da AWS IoT Events API.

- Mensagem: Não foi possível analisar sua expressão de conteúdo para a carga de *action-type* Insira uma expressão de conteúdo com a sintaxe correta.

Solução: a expressão de conteúdo pode conter cadeias de caracteres (*'string'*), variáveis ($\$variable$. *variable-name*), valores de entrada ($\$input$. *input-name.path-to-datum*), concatenações de strings e strings que contêm. $\${}$

- Mensagem: as cargas personalizadas não devem estar vazias.

Solução: Você pode receber essa mensagem de erro se escolher Carga personalizada para sua ação e não inserir uma expressão de conteúdo no AWS IoT Events console. Se escolher Carga personalizada, deverá inserir uma expressão de conteúdo em Carga personalizada. Para obter mais informações, consulte [Carga útil](#) na Referência da API do AWS IoT Events .

- Mensagem: Falha ao analisar a expressão de duração '*duration-expression*' para o cronômetro ". *timer-name*

Solução: o resultado avaliado da sua expressão de duração para o temporizador deve ser um valor entre 60 e 31622400. O resultado avaliado da duração é arredondado para baixo para o número inteiro mais próximo.

- Mensagem: Falha ao analisar a expressão '*expression*' para *action-name*

Solução: você pode receber essa mensagem se a expressão da ação especificada tiver uma sintaxe incorreta. Certifique-se de inserir uma expressão com a sintaxe correta. Para obter mais informações, consulte [Sintaxe para filtrar dados do dispositivo e definir ações no AWS IoT Events](#).

- Mensagem: Seu *fieldName* formulário não IotSitewiseAction pôde ser analisado. Você deve usar a sintaxe correta na expressão.

Solução: Você pode receber esse erro se AWS IoT Events não conseguir analisar seu formulário *fieldName*. IotSitewiseAction Certifique-se de *fieldName* usar uma expressão que AWS IoT Events possa ser analisada. Para obter mais informações, consulte [lotSiteWiseAction](#) na Referência de APIs do AWS IoT Events .

data-type

Um resultado de análise com informações sobre data-type, corresponde às seguintes mensagens de erro:

- Mensagem: A expressão *duration-expression* de duração do cronômetro não *timer-name* é válida, ela deve retornar um número.

Solução: você pode receber essa mensagem de erro se AWS IoT Events não conseguir avaliar a expressão de duração do cronômetro para um número. Certifique-se de que seu *durationExpression* possa ser convertido em um número. Outros tipos de dados, como o booleano, não são compatíveis.

- Mensagem: *condition-expression* A expressão não é uma expressão de condição válida.

Solução: você pode receber essa mensagem de erro se não AWS IoT Events conseguir avaliá-la como um valor booleano. *condition-expression* O valor booleano deve ser TRUE ou FALSE. Certifique-se de que sua expressão de condição possa ser convertida em um valor booleano. Se o resultado não for um valor booleano, ele é equivalente a FALSE e não invoca as ações ou a transição para o *nextState* especificado no evento.

- Mensagem: Tipos de dados incompatíveis [*inferred-types*] encontrados *reference* na seguinte expressão: *expression*

Solução: Solução: todas as expressões para o mesmo atributo ou variável de entrada no modelo do detector devem fazer referência ao mesmo tipo de dados.

Use as informações a seguir para resolver o problema:

- Ao usar uma referência como operando com um ou mais operadores, verifique se todos os tipos de dados referenciados são compatíveis.

Por exemplo, na expressão a seguir, o inteiro 2 é um operando dos operadores == e &&. Para garantir que os operandos sejam compatíveis, `$variable.testVariable + 1` e `$variable.testVariable` devem referenciar um número inteiro ou decimal.

Além disso, o inteiro 1 é um operando do operador +. Portanto, `$variable.testVariable` deve fazer referência a um número inteiro ou decimal.

```
'$variable.testVariable + 1 == 2 && $variable.testVariable'
```

- Ao usar uma referência como argumento passado para uma função, verifique se a função é compatível com os tipos de dados aos quais você faz referência.

Por exemplo, a função a seguir `timeout("time-name")` requer uma string com aspas duplas como argumento. Se você usar uma referência para o `timer-name` valor, deverá referenciar uma string com aspas duplas.

```
timeout("timer-name")
```

Note

Para a `convert(type, expression)` função, se você usar uma referência para o `type` valor, o resultado avaliado da sua referência deverá ser `String`, `Decimal`, ou `Boolean`.

Para obter mais informações, consulte [AWS IoT Events referência para entradas e variáveis em expressões](#).

- Mensagem: Tipos de dados incompatíveis [*inferred-types*] usados com *reference*. Isso pode levar a um erro de runtime.

Solução: você pode receber essa mensagem de aviso se duas expressões para o mesmo atributo de entrada ou variável fizerem referência a dois tipos de dados. Certifique-se de que suas expressões para o mesmo atributo ou variável de entrada façam referência ao mesmo tipo de dados no modelo do detector.

- Mensagem: Os tipos de dados [*inferred-types*] que você inseriu para o operador [*operator*] não são compatíveis com a seguinte expressão: '*expression*'

Solução: você pode receber essa mensagem de erro se sua expressão combinar tipos de dados que não são compatíveis com um operador especificado. Por exemplo, na expressão a seguir, o operador + é compatível com os tipos de dados Inteiro, Decimal e String, mas não com operandos do tipo de dados booliano.

```
true + false
```

Você deve se certificar de que os tipos de dados usados com um operador sejam compatíveis.

- Mensagem: Os tipos de dados [*inferred-types*] encontrados *input-attribute* não são compatíveis e podem causar um erro de tempo de execução.

Solução: você pode receber essa mensagem de erro se duas expressões para o mesmo atributo de entrada fizerem referência a dois tipos de dados para o `OnEnterLifecycle` de um estado ou para o `OnInputLifecycle` e `OnExitLifecycle` de um estado. Certifique-se de que suas expressões em `OnEnterLifecycle` (ou, `OnInputLifecycle` e `OnExitLifecycle`) façam referência ao mesmo tipo de dados para cada estado do seu modelo de detector.

- Mensagem: A expressão da carga útil [*expression*] não é válida. Especifique uma expressão que seria avaliada como uma string em runtime porque o tipo de carga é o formato JSON.

Solução: você pode receber esse erro se o tipo de carga útil especificado for JSON, mas não AWS IoT Events conseguir avaliar sua expressão como uma String. Certifique-se de que o resultado avaliado seja uma string, não um booliano ou um número.

- Mensagem: Sua expressão interpolada {*interpolated-expression*} deve ser avaliada como um valor inteiro ou booleano em tempo de execução. Caso contrário, sua expressão de carga útil {*payload-expression*} não poderá ser analisada em tempo de execução como JSON válido.

Solução: você pode receber essa mensagem de erro se não AWS IoT Events conseguir avaliar sua expressão interpolada como um número inteiro ou um valor booleano. Certifique-se de que sua expressão interpolada possa ser convertida em um valor inteiro ou booliano, pois não há suporte para outros tipos de dados, como string.

- Mensagem: O tipo de expressão no `IotSitewiseAction` campo *expression* é definido como tipo *defined-type* e inferido como tipo *inferred-type*. O tipo definido e o tipo inferido devem ser iguais.

Solução: você pode receber essa mensagem de erro se sua expressão no `propertyValue` de `IotSightwiseAction` tiver um tipo de dados definido de forma diferente do tipo de dados inferido por AWS IoT Events. Certifique-se de usar o mesmo tipo de dados para todas as instâncias dessa expressão em seu modelo de detector.

- Mensagem: Os tipos de dados [*inferred-types*] usados para `setTimer` ação não são avaliados `Integer` para a seguinte expressão: *expression*

Solução: você pode receber essa mensagem de erro se o tipo de dados inferido para sua expressão de duração não for integral ou decimal. Verifique se o seu `durationExpression` pode ser convertido em um número. Outros armazenamentos de dados, como booleano e string, não são compatíveis.

- Mensagem: Os tipos de dados [*inferred-types*] usados com operandos do operador de comparação [*operator*] não são compatíveis na seguinte expressão: *expression*

Solução: Os tipos de dados inferidos para os operandos da expressão condicional (*expression*) do seu modelo de detector não *operator* coincidem. Os operandos devem ser usados com os tipos de dados correspondentes em todas as outras partes do seu modelo de detector.

Tip

Você pode usar `convert` para alterar o tipo de dados de uma expressão em seu modelo de detector. Para obter mais informações, consulte [Funções para usar em AWS IoT Events expressões](#).

referenced-data

Um resultado de análise com informações sobre `referenced-data`, corresponde às seguintes mensagens de erro:

- Mensagem: Detectado cronômetro quebrado: o cronômetro *timer-name* é usado em uma expressão, mas nunca é definido.

Solução: você pode receber essa mensagem de erro se usar um temporizador que não esteja definido. Você deve definir um temporizador antes de usá-lo em uma expressão. Além disso, certifique-se de inserir nome correto do temporizador.

- Mensagem: Variável quebrada detectada: a variável *variable-name* é usada em uma expressão, mas nunca é definida.

Solução: você pode receber essa mensagem de erro se usar uma variável que não esteja definida. Você deve definir uma variável antes de usá-la em uma expressão. Além disso, verifique se o nome correto da variável foi inserido.

- Mensagem: variável quebrada detectada: uma variável é usada em uma expressão antes de ser definida como um valor.

Solução: cada variável deve ser atribuída a um valor antes de poder ser avaliada em uma expressão. Defina o valor da variável antes de cada uso para que seu valor possa ser recuperado. Além disso, verifique se o nome correto da variável foi inserido.

referenced-resource

Um resultado de análise com informações sobre `referenced-resource`, corresponde às seguintes mensagens de erro:

- Mensagem: a definição do modelo do detector contém uma referência a uma entrada que não existe.

Solução: você pode receber essa mensagem de erro se usar expressões para referenciar uma entrada que não existe. Certifique-se de que sua expressão faça referência a uma entrada existente e insira o nome correto da entrada. Se não tiver uma entrada, crie uma primeiro.

- Mensagem: A definição do modelo do detector contém informações inválidas `InputName: input-name`

Solução: você pode receber essa mensagem de erro se o modelo do detector contiver um nome de entrada inválido. Certifique-se de que inseriu o nome da entrada correto. Esse nome deve ter de 1 a 128 caracteres. Os caracteres válidos são a-z, A-Z, 0-9, _ (sublinhado) e - (hífen).

Análise um modelo de detector para AWS IoT Events (Console)

AWS IoT Events permite monitorar e reagir aos dados de IoT detectando eventos e acionando ações com a API. AWS IoT Events As etapas a seguir usam o AWS IoT Events console para analisar um modelo de detector.

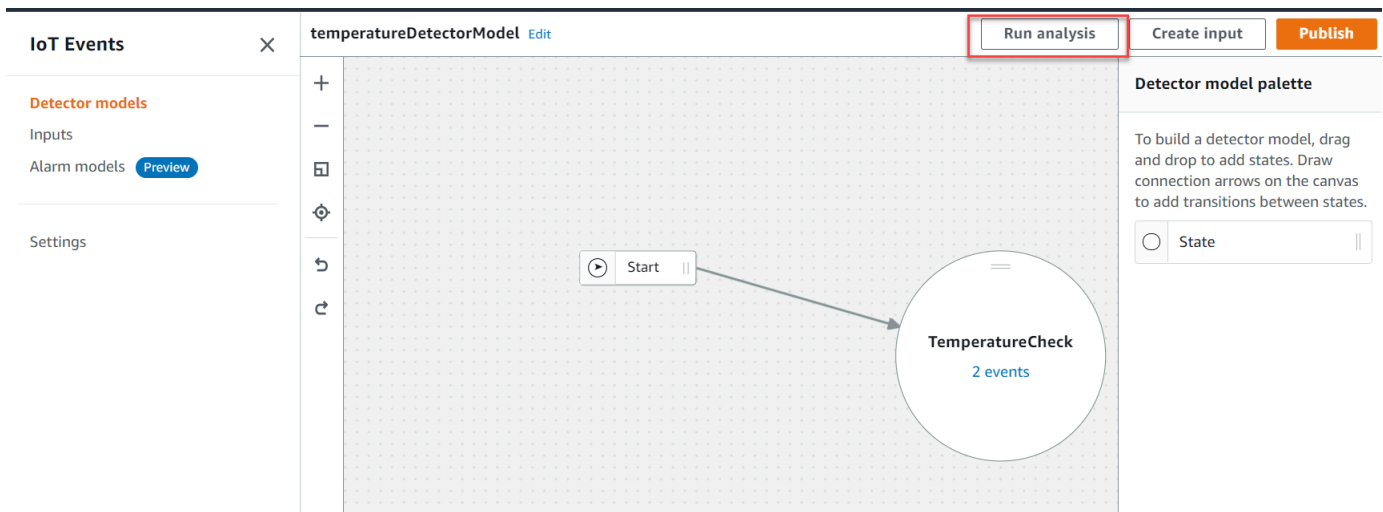
Note

Depois de AWS IoT Events começar a analisar seu modelo de detector, você tem até 24 horas para recuperar os resultados da análise.

A análise do modelo do detector pode ajudá-lo a otimizar seus modelos, identificar possíveis problemas e garantir que eles funcionem conforme o esperado. Por exemplo, em um parque eólico, a análise do modelo do detector pode revelar se o modelo identifica corretamente possíveis falhas na engrenagem com base em padrões de vibração anormais. Ou, se o modelo acionar alertas de manutenção com precisão quando as velocidades do vento excederem os limites operacionais seguros. Ao refinar um modelo com base na análise, você pode melhorar a manutenção preditiva, reduzir o tempo de inatividade e melhorar a eficiência geral da produção de energia.

Para analisar um modelo de detector

1. Faça login no [console do AWS IoT Events](#).
2. No painel de navegação, selecione Modelos de detector.
3. Em Modelos de detector, escolha o modelo do detector alvo.
4. Na página do modelo do detector, escolha Editar.
5. No canto superior direito, escolha Executar análise.



Veja a seguir um exemplo de resultado de análise no AWS IoT Events console.

The screenshot displays the AWS IoT Events console interface for editing a detector model named 'temperatureDetectorModel'. On the left, a sidebar shows navigation options: 'Detector models', 'Inputs', 'Alarm models' (with a 'Preview' button), and 'Settings'. The main canvas shows a state machine diagram with a 'Start' state and a 'TemperatureCheck' state (containing 2 events). A 'Detector model analysis' panel at the bottom provides a summary: (1) All, (0) Error, (0) Warning, and (1) Information. A detailed message indicates: 'Info: data-type Message: Inferred data types [Integer] for \$variable.temperatureChecked'. On the right, a 'Detector model palette' includes a 'State' component.

Analise um modelo de detector em AWS IoT Events (AWS CLI)

A análise programática de seus modelos de AWS IoT Events detectores fornece informações valiosas sobre sua estrutura, comportamento e desempenho. Essa abordagem baseada em API permite análise automatizada, integração com seus fluxos de trabalho existentes e a capacidade de realizar operações em massa em vários modelos de detectores. Ao aproveitar a [StartDetectorModelAnalysis](#) API, você pode iniciar exames detalhados de seus modelos, ajudando a identificar possíveis problemas, otimizar fluxos lógicos e garantir que o processamento de eventos de IoT esteja alinhado aos requisitos de sua empresa.

As etapas a seguir usam o AWS CLI para analisar um modelo de detector.

Para analisar um modelo de detector usando AWS CLI

1. Execute o comando a seguir para iniciar uma análise.

```
aws iotevents start-detector-model-analysis --cli-input-json file://file-name.json
```

Note

file-name Substitua pelo nome do arquivo que contém a definição do modelo do detector.

Example Definição do modelo de detector

```
{
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "TemperatureCheck",
        "onInput": {
          "events": [
            {
              "eventName": "Temperature Received",
              "condition":
"isNull($input.TemperatureInput.sensorData.temperature)==false",
              "actions": [
                {
                  "iotTopicPublish": {
                    "mqttTopic": "IoTEvents/Output"
                  }
                }
              ]
            }
          ],
          "transitionEvents": []
        },
        "onEnter": {
          "events": [
            {
              "eventName": "Init",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "temperatureChecked",
                    "value": "0"
                  }
                }
              ]
            }
          ]
        }
      }
    ]
  }
}
```

```

    ],
    "onExit": {
      "events": []
    }
  },
  "initialStateName": "TemperatureCheck"
}

```

Se você usar o AWS CLI para analisar um modelo de detector existente, escolha uma das opções a seguir para recuperar a definição do modelo de detector:

- Se você quiser usar o AWS IoT Events console, faça o seguinte:
 1. No painel de navegação à esquerda, escolha Modelos de detector.
 2. Em Modelos de detector, escolha o modelo do detector alvo.
 3. Escolha Exportar modelo de detector em Ação para baixar o modelo do detector. O modelo do detector é salvo em JSON.
 4. Abra o arquivo JSON do modelo do detector.
 5. Você só precisa do objeto `detectorModelDefinition`. Remova o seguinte:
 - O primeiro colchete (`{`) na parte superior da página
 - A linha `detectorModel`
 - O objeto `detectorModelConfiguration`
 - O último colchete (`}`) na parte inferior da página
 6. Salve o arquivo.
- Se você quiser usar o AWS CLI, faça o seguinte:
 1. Execute o comando a seguir em um terminal.

```
aws iotevents describe-detector-model --detector-model-name detector-model-name
```

2. *detector-model-name* Substitua pelo nome do modelo do seu detector.
3. Copie o objeto `detectorModelDefinition` em um editor de textos.

4. Adicione colchetes ({}) fora do `detectorModelDefinition`.
5. Salve o arquivo em JSON.

Example Exemplo de resposta

```
{
  "analysisId": "c1133390-14e3-4204-9a66-31efd92a4fed"
}
```

2. Copie a ID da análise da saída.
3. Execute o comando a seguir para recuperar o status da análise.

```
aws iotevents describe-detector-model-analysis --analysis-id "analysis-id"
```

Note

analysis-id Substitua pela ID de análise que você copiou.

Example Exemplo de resposta

```
{
  "status": "COMPLETE"
}
```

O status pode ser um dos valores a seguir:

- **RUNNING**— AWS IoT Events está analisando seu modelo de detector. O processo pode levar até um minuto para ser concluído.
 - **COMPLETE**— AWS IoT Events terminou de analisar seu modelo de detector.
 - **FAILED**— AWS IoT Events não foi possível analisar seu modelo de detector. Tente novamente mais tarde.
4. Execute o comando a seguir para recuperar um ou mais resultados de análise do modelo de detector.

Note

analysis-id Substitua pela ID de análise que você copiou.

```
aws iotevents get-detector-model-analysis-results --analysis-id "analysis-id"
```

Example Exemplo de resposta

```
{
  "analysisResults": [
    {
      "type": "data-type",
      "level": "INFO",
      "message": "Inferred data types [Integer] for
$variable.temperatureChecked",
      "locations": []
    },
    {
      "type": "referenced-resource",
      "level": "ERROR",
      "message": "Detector Model Definition contains reference to Input
'TemperatureInput' that does not exist.",
      "locations": [
        {
          "path": "states[0].onInput.events[0]"
        }
      ]
    }
  ]
}
```

Note

Depois de AWS IoT Events começar a analisar seu modelo de detector, você tem até 24 horas para recuperar os resultados da análise.

AWS IoT Events comandos

Este capítulo fornece um guia abrangente de todas as operações de API disponíveis em AWS IoT Events. Ele oferece explicações detalhadas, incluindo exemplos de solicitações, respostas e possíveis erros para cada operação nos protocolos de serviços da web compatíveis. Compreender essas operações de API ajuda você a se AWS IoT Events integrar com eficácia aos seus aplicativos de IoT e automatizar seus fluxos de trabalho de detecção e resposta a eventos.

AWS IoT Events ações

Você pode usar comandos de AWS IoT Events API para criar, ler, atualizar e excluir entradas e modelos de detectores e listar suas versões. Para obter mais informações, consulte as [ações](#) e [os tipos de dados](#) compatíveis com a Referência da AWS IoT Events API. AWS IoT Events

As [AWS IoT Events seções](#) na Referência de AWS CLI Comandos incluem os AWS CLI comandos que você pode usar para administrar e manipular AWS IoT Events.

AWS IoT Events dados

Você pode usar os comandos da API de AWS IoT Events dados para enviar entradas para detectores, listar detectores e visualizar ou atualizar o status de um detector. Para obter mais informações, consulte as [ações](#) e [os tipos de dados](#) compatíveis com AWS IoT Events os dados na Referência da AWS IoT Events API.

[As seções de AWS IoT Events dados](#) na Referência de AWS CLI Comandos incluem os AWS CLI comandos que você pode usar para processar AWS IoT Events dados.

Histórico do documento para AWS IoT Events

A tabela a seguir descreve as alterações importantes feitas no Guia do desenvolvedor do AWS IoT Events após 17 de setembro de 2020. Para obter mais informações sobre as atualizações desta documentação, você pode se tornar assinante de um feed RSS.

Alteração	Descrição	Data
Aviso de fim do suporte	Aviso de fim do suporte: em 20 de maio de 2026, AWS o suporte para o. AWS IoT Events Depois de 20 de maio de 2026, você não poderá mais acessar o AWS IoT Events console ou os AWS IoT Events recursos.	20 de maio de 2025
Lançamento regional	AWS IoT Events agora está disponível na região Ásia-Pacífico (Mumbai).	30 de setembro de 2021
Lançamento regional	AWS IoT Events agora está disponível na região AWS GovCloud (Oeste dos EUA).	22 de setembro de 2021
Solucione problemas em um modelo de detector executando análises	AWS IoT Events agora pode analisar seu modelo de detector e gerar resultados de análise que você pode usar para solucionar problemas em seu modelo de detector.	23 de fevereiro de 2021
Lançamento regional	Lançado AWS IoT Events na China (Pequim).	30 de setembro de 2020

[Uso de expressão](#)

Exemplos adicionados para mostrar como escrever expressões.

22 de setembro de 2020

[Monitorar com alarmes](#)

Os alarmes ajudam você a monitorar seus dados em busca de alterações. É possível criar alarmes que enviam notificações quando um limite é violado.

1 de junho de 2020

Atualizações anteriores

A tabela a seguir descreve alterações importantes feitas no Guia do desenvolvedor do AWS IoT Events antes de 18 de setembro de 2020.

Alteração	Descrição	Data
Validação de tipo adicionada à referência de expressões	Informações de validação de tipo adicionadas à referência de expressões.	3 de agosto de 2020
Aviso de região adicionado para outros serviços	Foi adicionado um aviso sobre a seleção da mesma região AWS IoT Events e de outros AWS serviços.	7 de maio de 2020
Adições, atualizações	<ul style="list-style-type: none"> Recurso de personalização de carga Novas ações de eventos: Amazon DynamoDB e AWS IoT SiteWise 	27 de abril de 2020
Funções integradas adicionadas para expressões condicionais do modelo de detector	Funções integradas adicionadas para expressões condicionais do modelo detector.	10 de setembro de 2019

Alteração	Descrição	Data
Exemplos de modelos de detectores adicionados	Exemplos adicionados para o modelo do detector.	5 de agosto de 2019
Adicionadas novas ações de eventos	Foram adicionadas novas ações de eventos para: <ul style="list-style-type: none"> • Lambda • Amazon SQS • Kinesis Data Firehose • AWS IoT Events entrada 	19 de julho de 2019
Adições, correções	<ul style="list-style-type: none"> • Descrição atualizada da <code>timeout()</code> função. • Foi adicionada a melhor prática em relação à inatividade da conta. 	11 de junho de 2019
Política de permissões atualizada e opções de depuração do console	<ul style="list-style-type: none"> • Atualizou a política de permissões do console. • Imagem atualizada da página de opções de depuração do console. 	5 de junho de 2019
Atualizações	AWS IoT Events serviço aberto à disponibilidade geral.	30 de maio de 2019
Adições, atualizações	<ul style="list-style-type: none"> • Informações de segurança atualizadas. • Foi adicionado um exemplo de modelo de detector anotado. 	22 de maio de 2019

Alteração	Descrição	Data
Exemplos adicionados e permissões necessárias	Exemplos de carga útil do Amazon SNS adicionados; acréscimos às permissões necessárias para. <code>CreateDetectorModel</code>	17 de maio de 2019
Informações de segurança adicionais adicionadas	Informações adicionadas à seção de segurança.	9 de maio de 2019
Versão pré-visualização limitada	Versão inicial limitada da documentação da versão.	28 de março de 2019