



Manual do usuário

AWS Serviço de injeção de falhas



AWS Serviço de injeção de falhas: Manual do usuário

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

O que é o AWS FIS?	1
Conceitos	1
Ações	2
Destinos	2
Condições de parada	2
Suportado Serviços da AWS	3
Acesse o AWS FIS	3
Preços	3
Planejar experimentos	4
Princípios e diretrizes básicos	4
Diretrizes para planejamento de experimentos	5
Componentes do modelo de experimento	8
Sintaxe do modelo	8
Conceitos básicos	9
Ações	9
Sintaxe da ação	10
Identificadores de ação	11
Parâmetros de ação	11
Destinos da ação	12
Duração da ação	13
Exemplo de ações	13
Destinos	16
Sintaxe de destino	17
Resource types	18
Identificar recursos de destino	19
Modo de seleção	24
Exemplos de destinos	25
Exemplo de filtros	26
Condições de parada	31
Sintaxe da condição de parada	31
Saiba mais	32
Perfil do experimento	32
Pré-requisitos	33
Opção 1: criar uma função experimental e anexar uma política AWS gerenciada	34

Opção 2: criar um perfil experimental e adicionar um documento de política em linha	35
Configuração do relatório do experimento	37
Sintaxe de configuração do relatório do experimento	39
Permissões do relatório de experimentos	41
Práticas recomendadas para relatórios de experimentos	43
Opções do experimento	43
Segmentação de conta	44
Modo de resolução de destino vazio	46
Modo de ações	46
Referência das ações	48
Ações de injeção de falhas	49
aws:fis:inject-api-internal-error	49
aws:fis:inject-api-throttle-error	50
aws:fis:inject-api-unavailable-error	51
Ação de recuperação	51
aws:arc:start-zonal-autoshift	51
Ação de espera	53
aws:fis:wait	53
CloudWatch Ações da Amazon	53
aws:cloudwatch:assert-alarm-state	53
Ações do Amazon DynamoDB	54
aws:dynamodb:global-table-pause-replication	54
Ações do Amazon Aurora DSQL	58
aws:dsq:cluster-connection-failure	58
Ações do Amazon EBS	59
aws:ebs:pause-volume-io	59
aws:ebs:volume-io-latency	60
Ações do Amazon EC2	61
aws:ec2:api-insufficient-instance-capacity-error	61
aws:ec2:asg-insufficient-instance-capacity-error	62
aws:ec2:reboot-instances	63
aws:ec2:send-spot-instance-interruptions	64
aws:ec2:stop-instances	65
aws:ec2:terminate-instances	66
Ações do Amazon ECS	66
aws:ecs:drain-container-instances	67

aws:ecs:stop-task	68
aws:ecs:task-cpu-stress	68
aws:ecs:task-io-stress	69
aws:ecs:task-kill-process	70
aws:ecs:task-network-blackhole-port	71
aws:ecs:task-network-latency	72
aws:ecs:task-network-packet-loss	73
Ações do Amazon EKS	74
aws:eks:inject-kubernetes-custom-resource	75
aws:eks:pod-cpu-stress	76
aws:eks:pod-delete	77
aws:eks:pod-io-stress	78
aws:eks:pod-memory-stress	79
aws:eks:pod-network-blackhole-port	81
aws:eks:pod-network-latency	82
aws:eks:pod-network-packet-loss	83
aws:eks:terminate-nodegroup-instances	85
ElastiCache Ações da Amazon	85
aws:elasticache:replicationgroup-interrupt-az-power	85
Ações do Amazon Kinesis Data Streams	86
aws:kinesis:stream-provisioned-throughput-exception	87
aws:kinesis:stream-expired-iterator-exception	87
AWS Lambda ações	88
aws:lambda:invocation-add-delay	88
aws:lambda:invocation-error	89
aws:lambda:invocation-http-integration-response	90
Ação Amazon MemoryDB	91
aws:memorydb:multi-region-cluster-pause-replication	91
Ações de rede	91
aws:network:disrupt-connectivity	92
aws:network:route-table-disrupt-cross-region-connectivity	93
aws:network:transit-gateway-disrupt-cross-region-connectivity	95
aws:network:disrupt-vpc-endpoint	95
Ações do Amazon RDS	96
aws:rds:failover-db-cluster	97
aws:rds:reboot-db-instances	97

Ações do Amazon S3	98
aws:s3:bucket-pause-replication	98
Ações do Systems Manager	99
aws:ssm:send-command	99
aws:ssm:start-automation-execution	100
AWS Direct Connect ações	101
aws:directconnect:virtual-interface-disconnect	101
Ações de documentos do SSM	102
Usar a ação aws:ssm:send-command	103
Documentos AWS FIS SSM pré-configurados	104
Exemplos	114
Limitações	114
Scripts de reversão	114
Solução de problemas	116
Ações de tarefa do ECS	116
Ações	117
Limitações	117
Requisitos	118
Versão de referência do script	121
Exemplo de modelo de experimento	123
Ações do EKS Pod	124
Ações	125
Limitações	126
Requisitos	127
Criar um perfil de experimento	127
Configuração da conta de serviço do Kubernetes	127
Conceda aos usuários e funções do IAM acesso ao Kubernetes APIs	129
Imagens do contêiner do pod	130
Exemplo de modelo de experimento	132
AWS Lambda ações	133
Ações	134
Limitações	134
Pré-requisitos	134
Configurar funções do Lambda	136
Configurar um AWS FIS experimento	136
Registro em log	137

Tópicos avançados	138
AWS FIS Versões da extensão Lambda	144
Gerenciar modelos de experimento	148
Criar um modelo de experimento	148
Visualizar modelos de experimentos	151
Gerar uma pré-visualização de destinos	152
Iniciar um experimento a partir de um modelo	153
Atualizar um modelo de experimento	153
Marcar modelos de experimentos	154
Excluir um modelo de experimento	154
Exemplos de modelos do	155
Interrompa EC2 instâncias com base em filtros	156
Interromper um número específico de EC2 instâncias	157
Execute um documento AWS FIS SSM pré-configurado	158
Executar um runbook predefinido de automação	159
Ações da API Throttle em EC2 instâncias com a função de destino do IAM	160
Teste de estresse da CPU de pods em um cluster do Kubernetes	161
Exceção de taxa de transferência provisionada para um número especificado de Kinesis Data Streams	163
Experimente um exemplo de permissões de função	164
Gerenciar experimentos	166
Iniciar um experimento	166
Visualizar seus experimentos	167
Estados do experimento	168
Estados da ação	168
Marcar um experimento	168
Interromper um experimento	169
Listar destinos resolvidos	169
Tutoriais	171
Testar início e interrupção da instância	171
Pré-requisitos	171
Etapa 1: criar um modelo de experimento	171
Etapa 2: iniciar o experimento	175
Etapa 3: acompanhar o progresso do experimento	175
Etapa 4: verificar o resultado do experimento	175
Etapa 5: limpar	176

Executar o estresse da CPU em uma instância	177
Pré-requisitos	177
Etapa 1: criar um CloudWatch alarme para uma condição de parada	178
Etapa 2: criar um modelo de experimento	178
Etapa 3: iniciar o experimento	181
Etapa 4: acompanhar o progresso do experimento	181
Etapa 5: verificar os resultados do experimento	182
Etapa 6: limpar	176
Testar interrupções da instância spot	184
Pré-requisitos	184
Etapa 1: criar um modelo de experimento	186
Etapa 2: iniciar o experimento	188
Etapa 3: acompanhar o progresso do experimento	189
Etapa 4: verificar o resultado do experimento	189
Etapa 5: limpar	190
Simular um evento de conectividade	191
Pré-requisitos	192
Etapa 1: criar um modelo de experimento AWS FIS	192
Etapa 2: fazer ping em um endpoint do Amazon S3	194
Etapa 3: inicie seu experimento AWS FIS	195
Etapa 4: Acompanhe o progresso do seu experimento AWS FIS	195
Etapa 5: verificar interrupção na rede do Amazon S3	195
Etapa 5: limpar	196
Programar um experimento recorrente	196
Pré-requisitos	197
Etapa 1: criar uma política e um perfil do IAM	197
Etapa 2: criar um Amazon EventBridge agendador	199
Etapa 3: verificar seu experimento	201
Etapa 4: limpar	201
Trabalhar com a biblioteca de cenários	202
Visualizar um cenário	202
Usar um cenário	203
Exportação de um cenário	204
Referência de cenários	204
AZ Availability: Power Interruption	208
AZ: Application Slowdown	223

Cross-AZ: Traffic Slowdown	230
Cross-Region: Connectivity	237
Trabalhar com experimentos em várias contas	252
Conceitos	253
Práticas recomendadas	253
Pré-requisitos	253
Permissões	254
Interromper condições (opcional)	257
Medidas de segurança para experimentos em várias contas (opcional)	257
Criar um modelo de experimento com várias contas	257
Atualizar uma configuração de conta de destino	259
Excluir uma configuração da conta de destino	260
Agendar experimentos	261
Criar um perfil de agendador	261
Criar um agendamento de experimento	265
Para atualizar uma programação usando o console	266
Atualizar um agendamento de experimento	266
Desabilitar ou excluir um agendamento de experimento	267
Medidas de segurança	268
Conceitos para medidas de segurança	268
Recurso das medidas de segurança	269
Trabalhar com medidas de segurança	269
Visualizar uma medida de segurança	269
Acionar uma medida de segurança	270
Desacionar uma medida de segurança	270
Monitorar experimentos	272
Monitore usando CloudWatch	273
Monitore os AWS experimentos do FIS	273
AWS Métricas de uso do FIS	274
Monitore usando EventBridge	275
Registro em log do experimento	277
Permissões	277
Esquema de logs	277
Destinos de logs	279
Exemplo de registros de log	279
Habilitar registro em log de experimento	284

Desabilitar registro em log	285
Registre chamadas de API com AWS CloudTrail	285
Use CloudTrail	286
Entenda as AWS entradas do arquivo de log do FIS	287
Solução de problemas	291
Códigos de erro	291
Segurança	294
Proteção de dados	294
Criptografia em repouso	296
Criptografia em trânsito	296
Gerenciamento de identidade e acesso	296
Público	297
Autenticação com identidades	297
Gerenciar o acesso usando políticas	298
Como o AWS Fault Injection Service funciona com o IAM	300
Exemplos de políticas	306
Usar perfis vinculados a serviços	316
AWS políticas gerenciadas	319
Segurança da infraestrutura	324
AWS PrivateLink	325
Considerações	325
Criar um VPC endpoint de interface	325
Criar uma política de endpoint da VPC	325
Marcando seus Recursos	328
Restrições de marcação	328
Trabalhar com tags	328
Cotas e limitações	330
Histórico do documento	345
.....	cccliii

O que é o serviço de injeção de AWS falhas?

AWS O AWS Fault Injection Service (FIS) é um serviço gerenciado que permite realizar experimentos de injeção de falhas em suas AWS cargas de trabalho. A injeção de falhas é baseada nos princípios da engenharia do caos. Esses experimentos estressam um aplicativo criando eventos disruptivos para que você possa observar como seu aplicativo responde. Em seguida, você pode usar essas informações para melhorar o desempenho e a resiliência de seus aplicativos para que eles se comportem conforme o esperado.

Para usar o AWS FIS, você configura e executa experimentos que ajudam a criar as condições reais necessárias para descobrir problemas de aplicativos que, de outra forma, seriam difíceis de encontrar. AWS O FIS fornece modelos que geram interrupções e os controles e proteções necessários para realizar experimentos na produção, como reverter ou interromper automaticamente o experimento se condições específicas forem atendidas.

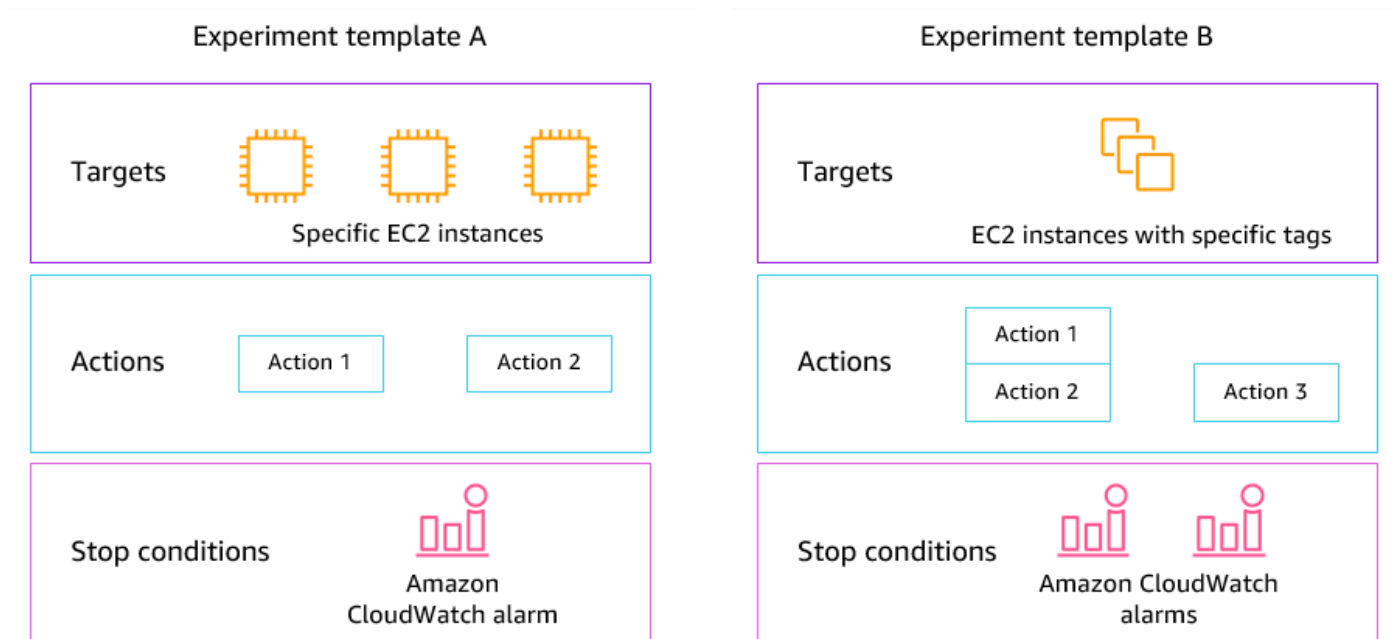
Important

AWS O FIS realiza ações reais em AWS recursos reais em seu sistema. Portanto, antes de usar o AWS FIS para executar experimentos em produção, é altamente recomendável que você conclua uma fase de planejamento e execute os experimentos em um ambiente de pré-produção.

Para obter mais informações sobre como planejar seu experimento, consulte [Confiabilidade do teste e Planejando seus AWS experimentos FIS](#). Para obter mais informações sobre o AWS FIS, consulte [AWS Fault Injection Service](#).

AWS Conceitos do FIS

Para usar o AWS FIS, você executa experimentos em seus AWS recursos para testar sua teoria de como um aplicativo ou sistema funcionará em condições de falha. Para realizar experimentos, primeiro você cria um modelo de experimento. Um modelo de experimento é o esquema do seu experimento. Ele contém as ações, destinos e condições de parada do experimento. Depois de criar um modelo de experimento, você pode usá-lo para executar um experimento. Enquanto seu experimento está sendo executado, você pode acompanhar seu progresso e ver seu status. Um experimento é concluído quando todas as ações do experimento tiverem sido executadas.



Ações

Uma ação é uma atividade que o AWS FIS executa em um AWS recurso durante um experimento. O AWS FIS fornece um conjunto de ações pré-configuradas com base no tipo de AWS recurso. Cada ação é executada por um período específico durante um experimento ou até você interromper o experimento. As ações podem ser executadas sequencialmente ou simultaneamente (em paralelo).

Destinos

Um alvo é um ou mais AWS recursos nos quais o AWS FIS executa uma ação durante um experimento. Você pode escolher recursos específicos ou selecionar um grupo de recursos com base em critérios específicos, como tags ou estado.

Condições de parada

O AWS FIS fornece os controles e as proteções de que você precisa para executar experimentos com segurança em suas cargas de trabalho. Uma condição de parada é um mecanismo para interromper um experimento se ele atingir um limite que você define como um alarme de CloudWatch da Amazon. Se uma condição de parada for acionada durante a execução do experimento, o AWS FIS interrompe o experimento.

Suportado Serviços da AWS

AWS O FIS fornece ações pré-configuradas para tipos específicos de alvos em todos AWS os serviços. Para obter a lista de serviços suportados e suas ações, consulte a [referência de ações do AWS FIS](#).

Para experimentos com uma única conta, os recursos-alvo devem estar nos Conta da AWS mesmos do experimento. Você pode executar experimentos do AWS FIS que tenham como alvo recursos em uma Conta da AWS conta diferente usando os experimentos de várias contas AWS do FIS.

Para obter mais informações, consulte [Ações para o AWS FIS](#).

Acesse o AWS FIS

Você pode trabalhar com o AWS FIS de qualquer uma das seguintes formas:

- Console de gerenciamento da AWS— Fornece uma interface web que você pode usar para acessar o AWS FIS. Para obter mais informações, consulte [Como trabalhar com o Console de gerenciamento da AWS](#).
- AWS Command Line Interface (AWS CLI) — Fornece comandos para um amplo conjunto de AWS serviços, incluindo AWS FIS, e é compatível com Windows, macOS e Linux. Para obter mais informações, consulte [AWS Command Line Interface](#). Para obter mais informações sobre os comandos do AWS FIS, consulte [fis na Referência](#) de AWS CLI Comandos.
- AWS CloudFormation— Crie modelos que descrevam seus AWS recursos. Você usa os modelos para provisionar e gerenciar esses recursos como uma só unidade. Para obter mais informações, consulte a [referência do tipo de recurso do AWS Fault Injection Service](#).
- AWS SDKs— fornece informações específicas para o idioma APIs e cuida de muitos detalhes da conexão, como calcular assinaturas, lidar com novas tentativas de solicitação e lidar com erros. Para obter mais informações, consulte [AWS SDKs](#).
- API HTTPS: fornece ações de API de nível inferior que você chama usando solicitações HTTPS. Para obter mais informações, consulte a [referência de API do AWS Fault Injection Service](#).

Preços do AWS FIS

A cobrança é feita por minuto de execução de uma ação, do início ao fim, com base no número de contas-alvo do experimento. Para obter mais informações, consulte [Preços do AWS FIS](#).

Planejando seus AWS experimentos FIS

A injeção de falhas é o processo de sobrecarregar um aplicativo em ambientes de teste ou produção criando eventos disruptivos, como interrupções no servidor ou controle de utilização da API. Ao observar como o sistema responde, você pode implementar melhorias. Quando você executa experimentos em seu sistema, isso pode ajudá-lo a identificar fraquezas sistêmicas de forma controlada antes que essas fraquezas afetem os clientes que dependem do seu sistema. Depois, você pode resolver os problemas de forma proativa para ajudar a evitar resultados imprevisíveis.

Antes de começar a realizar experimentos de injeção de falhas usando o AWS FIS, recomendamos que você se familiarize com os princípios e diretrizes a seguir.

Important

AWS O FIS realiza ações reais em AWS recursos reais em seu sistema. Portanto, antes de começar a usar o AWS FIS para realizar experimentos, é altamente recomendável que você primeiro conclua uma fase de planejamento e um teste em um ambiente de pré-produção ou teste.

Conteúdo

- [Princípios e diretrizes básicos](#)
- [Diretrizes para planejamento de experimentos](#)

Princípios e diretrizes básicos

Antes de iniciar os experimentos com o AWS FIS, siga as seguintes etapas:

1. Identifique a implantação de destino para o experimento — Comece identificando a implantação de destino. Se esse for seu primeiro experimento, recomendamos começar em um ambiente de pré-produção ou teste.
2. Revise a arquitetura do aplicativo — Você deve garantir que identificou todos os componentes, dependências e procedimentos de recuperação do aplicativo para cada componente. Comece analisando a arquitetura do aplicativo. Dependendo do aplicativo, consulte o [AWS Well-Architected Framework](#).

3. Defina o comportamento estável — Defina o comportamento estável do seu sistema em termos de métricas técnicas e comerciais importantes, como latência, carga da CPU, falhas de login por minuto, número de novas tentativas ou velocidade de carregamento da página.
4. Forme uma hipótese — Forme uma hipótese de como você espera que o comportamento do sistema mude durante o experimento. A definição de uma hipótese segue esse formato:

Se *fault injection action* for executado, o não *business or technical metric impact* deve exceder *value*.

Por exemplo, uma hipótese para um serviço de autenticação pode ser a seguinte: “Se a latência da rede aumentar em 10%, haverá um aumento de menos de 1% nas falhas de login”. Depois que o experimento for concluído, você avalia se a resiliência do aplicativo está alinhada às suas expectativas comerciais e técnicas.

Também recomendamos seguir estas diretrizes ao trabalhar com o AWS FIS:

- Sempre comece a experimentar o AWS FIS em um ambiente de teste. Nunca comece com um ambiente de produção. À medida que avança em seus experimentos de injeção de falhas, você pode experimentar em outros ambientes controlados além do ambiente de teste.
- Aumente a confiança da sua equipe na resiliência do seu aplicativo começando com experimentos pequenos e simples, como executar a ação `aws:ec2:stop-instances` em um destino.
- A injeção de falhas pode causar problemas reais. Prossiga com cuidado e certifique-se de que suas primeiras injeções de falha sejam em instâncias de teste para que nenhum cliente seja afetado.
- Teste, teste e teste um pouco mais. A injeção de falhas deve ser implementada em um ambiente controlado com experimentos bem planejados. Isso permite que você ganhe confiança nas habilidades de seu aplicativo e suas ferramentas para resistir a condições turbulentas.
- É altamente recomendável que você tenha um excelente programa de monitoramento e alerta antes de começar. Sem isso, você não conseguirá entender ou medir o impacto de seus experimentos, o que é fundamental para práticas sustentáveis de injeção de falhas.

Diretrizes para planejamento de experimentos

Com o AWS FIS, você executa experimentos em seus AWS recursos para testar sua teoria de como um aplicativo ou sistema funcionará em condições de falha.

A seguir estão as diretrizes recomendadas para planejar seus experimentos AWS FIS.

- Revise o histórico de interrupções — Analise as interrupções e eventos anteriores do seu sistema. Isso pode ajudá-lo a criar uma imagem da integridade geral e da resiliência do seu sistema. Antes de começar a executar experimentos em seu sistema, você deve abordar problemas e fraquezas conhecidos em seu sistema.
- Identifique os serviços com o maior impacto — Analise seus serviços e identifique aqueles que têm o maior impacto sobre seus usuários finais ou clientes se eles falharem ou não funcionarem corretamente.
- Identifique o sistema de destino — O sistema de destino é o sistema no qual você executará experimentos. Se você AWS é novo no FIS ou nunca realizou experimentos de injeção de falhas antes, recomendamos que comece executando experimentos em um sistema de pré-produção ou teste.
- Consulte sua equipe — Pergunte com o que eles estão preocupados. Você pode formar uma hipótese para provar ou refutar suas preocupações. Você também pode perguntar à sua equipe com o que eles não estão preocupados. Essa pergunta pode revelar duas falácias comuns: a falácia do custo irrecuperável e a falácia do viés de confirmação. Formar uma hipótese com base nas respostas da sua equipe pode ajudar a fornecer mais informações sobre a realidade do estado do seu sistema.
- Revise a arquitetura do aplicativo — Conduza uma revisão do seu sistema ou aplicativo e certifique-se de ter identificado todos os componentes do aplicativo, dependências e procedimentos de recuperação para cada componente.

Recomendamos que você revise o AWS Well-Architected Framework. O framework pode ajudar você a construir uma infraestrutura segura, de alto desempenho, resiliente e eficiente para seus aplicativos e workloads. Para obter mais informações, consulte [AWS Well-Architected](#).

- Identifique as métricas aplicáveis — Você pode monitorar o impacto de um experimento em seus AWS recursos usando CloudWatch as métricas da Amazon. Você pode usar essas métricas para determinar a linha de base ou o “estado estável” quando seu aplicativo está funcionando de maneira ideal. Em seguida, você pode monitorar essas métricas durante ou após o experimento para determinar o impacto. Para obter mais informações, consulte [Monitore AWS as métricas de uso do FIS usando a Amazon CloudWatch](#).
- Defina um limite de desempenho aceitável para seu sistema — Identifique a métrica que representa um estado estável aceitável para seu sistema. Você usará essa métrica para criar um ou mais CloudWatch alarmes que representem uma condição de parada para seu experimento.

Se o alarme for acionado, o experimento será interrompido automaticamente. Para obter mais informações, consulte [Condições de parada para o AWS FIS](#).

AWS Componentes do modelo de experimento FIS

Use os seguintes componentes para construir modelos de experimento:

Ações

As [ações do AWS FIS](#) que você deseja executar. As ações podem ser executadas em uma ordem definida que você especifica ou podem ser executadas simultaneamente. Para obter mais informações, consulte [Ações](#).

Destinos

Os AWS recursos sobre os quais uma ação específica é realizada. Para obter mais informações, consulte [Destinos](#).

Condições de parada

Os CloudWatch alarmes que definem um limite no qual o desempenho do seu aplicativo não é aceitável. Se uma condição de parada for acionada durante a execução de um experimento, o AWS FIS interrompe o experimento. Para obter mais informações, consulte [Condições de parada](#).

Perfil do experimento

Uma função do IAM que concede à AWS FIS as permissões necessárias para que ela possa realizar experimentos em seu nome. Para obter mais informações, consulte [Perfil do experimento](#).

Configuração do relatório do experimento

A configuração para habilitar relatórios de experimentos. Para obter mais informações, consulte [Configurações de relatórios de experimentos para AWS FIS](#).

Opções do experimento

Opções do modelo de experimento. Para obter mais informações, consulte [Opções de experimentos para AWS FIS](#).

Sua conta tem cotas relacionadas ao AWS FIS. Por exemplo, há uma cota para o número de ações por modelo de experimento. Para obter mais informações, consulte [Cotas e limitações](#).

Sintaxe do modelo

A seguir está a sintaxe para um modelo de experimento.

```
{
  "description": "string",
  "targets": {},
  "actions": {},
  "stopConditions": [],
  "roleArn": "arn:aws:iam::123456789012:role/AllowFISActions",
  "experimentReportConfiguration": {},
  "experimentOptions": {},
  "tags": {}
}
```

Para obter exemplos, consulte [Exemplos de modelos do](#) .

Conceitos básicos

Para criar um modelo de experimento usando o Console de gerenciamento da AWS, consulte [Criar um modelo de experimento](#).

Para criar um modelo de experimento usando o AWS CLI, consulte [Exemplos de AWS modelos de experimentos FIS](#).

Ações para o AWS FIS

Para criar um modelo de experimento, você deve definir uma ou mais ações. Para obter uma lista de ações predefinidas fornecidas pelo AWS FIS, consulte [Referência das ações](#)

Você pode executar uma ação somente uma vez durante um experimento. Para executar a mesma ação do AWS FIS mais de uma vez no mesmo experimento, adicione-a ao modelo várias vezes usando nomes diferentes.

Conteúdo

- [Sintaxe da ação](#)
- [Identificadores de ação](#)
- [Parâmetros de ação](#)
- [Destinos da ação](#)
- [Duração da ação](#)

- [Exemplo de ações](#)

Sintaxe da ação

A seguir está a sintaxe para uma ação.

```
{
  "actions": {
    "action_name": {
      "actionId": "aws:service:action-type",
      "description": "string",
      "parameters": {
        "name": "value"
      },
      "startAfter": ["action_name", ...],
      "targets": {
        "ResourceType": "target_name"
      }
    }
  }
}
```

Quando você define uma ação, fornece o seguinte:

action_name

Um nome para a ação.

actionId

O [identificador da ação](#).

description

Uma descrição opcional.

parameters

Qualquer [parâmetro de ação](#).

startAfter

Qualquer ação que precise ser concluída antes que essa ação possa ser iniciada. Caso contrário, a ação será executada no início do experimento.

targets

Qualquer [destino de ação](#).

Para obter exemplos, consulte [the section called “Exemplo de ações”](#).

Identificadores de ação

Cada ação AWS do FIS tem um identificador com o seguinte formato:

```
aws:service-name:action-type
```

Por exemplo, a ação a seguir interrompe as instâncias de destino do Amazon EC2:

```
aws:ec2:stop-instances
```

Para obter uma lista completa de ações, consulte [AWS FIS Referência de ações](#).

Parâmetros de ação

Algumas ações do AWS FIS têm parâmetros adicionais específicos para a ação. Esses parâmetros são usados para passar informações para o AWS FIS quando a ação é executada.

AWS O FIS oferece suporte a tipos de falha personalizados usando a `aws:ssm:send-command` ação, que usa o agente SSM e um documento de comando SSM para criar a condição de falha nas instâncias de destino. A ação do `aws:ssm:send-command` inclui um parâmetro `documentArn` que usa o nome do recurso da Amazon (ARN) de um documento do SSM como um valor. Você especifica valores para os parâmetros ao adicionar a ação ao seu modelo de experimento.

Para obter mais informações sobre como especificar os parâmetros para a ação do `aws:ssm:send-command`, consulte [Usar a ação aws:ssm:send-command](#).

Sempre que possível, você pode inserir uma configuração de reversão (também chamada de ação posterior) dentro dos parâmetros da ação. Uma ação posterior retorna o destino ao estado em que estava antes da execução da ação. A ação posterior é executada após o tempo especificado na duração da ação. Nem todas as ações podem oferecer suporte a ações posteriores. Por exemplo, se a ação encerrar uma instância do Amazon EC2, você não poderá recuperar a instância depois que ela for encerrada.

Destinos da ação

Uma ação é executada nos recursos de destino que você especifica. Depois de definir um destino, você pode especificar seu nome ao definir uma ação.

```
"targets": {  
  "ResourceType": "resource_name"  
}
```

AWS As ações do FIS oferecem suporte aos seguintes tipos de recursos para metas de ação:

- AutoScalingGroups— Grupos do Amazon EC2 Auto Scaling
- Buckets: buckets do Amazon S3
- Cluster – clusters do Amazon EKS
- Clusters — Amazon ECS, Aurora DSQL ou clusters de banco de dados Amazon Aurora
- DBInstances— Instâncias de banco de dados Amazon RDS
- Funções — AWS Lambda funções
- Instances – instâncias do Amazon EC2
- KinesisStreams— Streams de dados do Kinesis
- ManagedResources— Clusters do Amazon EKS, balanceadores de carga de aplicativos e redes do Amazon EC2 e grupos do Amazon EC2 Auto Scaling habilitados para mudança de zona ARC.
- MultiRegionClusters— Clusters multirregionais Amazon MemoryDB
- Nodegroups – grupos de nós do Amazon EKS
- Pods – pods do Kubernetes no Amazon EKS
- ReplicationGroups— ElastiCache Grupos de replicação
- Perfis – perfis do IAM
- SpotInstances— Instâncias spot do Amazon EC2
- Sub-redes: sub-redes da VPC
- Tabelas — Tabelas globais multirregionais fortes do Amazon DynamoDB e eventualmente consistentes
- Tasks – tarefas do Amazon ECS
- TransitGateways— Gateways de trânsito

- [VirtualInterfaces](#)— Interfaces Direct Connect virtuais
- [Volumes](#) – volumes do Amazon EBS
- [VPCEndpoints](#)— Endpoints Amazon VPC

Para obter exemplos, consulte [the section called “Exemplo de ações”](#).

Duração da ação

Se uma ação incluir um parâmetro que você possa usar para especificar a duração da ação, por padrão, a ação será considerada concluída somente após o término da duração especificada. Se você tiver definido a opção de experimento `emptyTargetResolutionMode` como `skip`, a ação será concluída imediatamente com o status “ignorado” quando nenhum alvo for resolvido. Por exemplo, se você especificar uma duração de 5 minutos, o AWS FIS considera a ação concluída após 5 minutos. Em seguida, ele inicia a próxima ação, até que todas as ações sejam concluídas.

A duração pode ser o período em que uma condição de ação é mantida ou o período durante o qual as métricas são monitoradas. Por exemplo, a latência é injetada durante o tempo especificado. Para tipos de ação quase instantâneos, como encerrar uma instância, as condições de parada são monitoradas pelo tempo especificado.

Se uma ação incluir uma ação posterior nos parâmetros da ação, a ação posterior será executada após a conclusão da ação. O tempo necessário para concluir a ação posterior pode causar um atraso entre a duração da ação especificada e o início da próxima ação (ou o final do experimento, se todas as outras ações forem concluídas).

Exemplo de ações

Os seguintes são exemplos de ações.

Exemplos

- [Interromper instâncias do EC2](#)
- [Interromper instâncias spot](#)
- [Interromper tráfego de rede](#)
- [Encerrar funcionários do EKS](#)
- [Inicie o deslocamento automático zonal ARC](#)

Exemplo: interromper instâncias do EC2

A ação a seguir interrompe as instâncias do EC2 identificadas usando o destino chamado *targetInstances*. Depois de dois minutos, ele reinicia as instâncias de destino.

```
"actions": {
  "stopInstances": {
    "actionId": "aws:ec2:stop-instances",
    "parameters": {
      "startInstancesAfterDuration": "PT2M"
    },
    "targets": {
      "Instances": "targetInstances"
    }
  }
}
```

Exemplo: interromper instâncias spot

A ação a seguir interrompe as Instâncias Spot identificadas usando o destino chamado *targetSpotInstances*. Ele espera dois minutos antes de interromper a instância spot.

```
"actions": {
  "interruptSpotInstances": {
    "actionId": "aws:ec2:send-spot-instance-interruptions",
    "parameters": {
      "durationBeforeInterruption": "PT2M"
    },
    "targets": {
      "SpotInstances": "targetSpotInstances"
    }
  }
}
```

Exemplo: interromper tráfego de rede

A ação a seguir nega tráfego entre as sub-redes de destino e sub-redes em outras zonas de disponibilidade.

```
"actions": {
```

```

    "disruptAZConnectivity": {
      "actionId": "aws:network:disrupt-connectivity",
      "parameters": {
        "scope": "availability-zone",
        "duration": "PT5M"
      },
      "targets": {
        "Subnets": "targetSubnets"
      }
    }
  }
}

```

Exemplo: encerrar funcionários do EKS

A ação a seguir encerra 50% das instâncias do EC2 no cluster EKS identificadas usando o destino chamado. *targetNodeGroups*

```

"actions": {
  "terminateWorkers": {
    "actionId": "aws:eks:terminate-nodegroup-instances",
    "parameters": {
      "instanceTerminationPercentage": "50"
    },
    "targets": {
      "Nodegroups": "targetNodeGroups"
    }
  }
}

```

Exemplo: Iniciar o deslocamento automático zonal ARC

A ação a seguir inicia um deslocamento automático ARC Zonal que afasta os recursos gerenciados de for. *az-in-parameters duration-in-parameteres* O tipo de recurso ManagedResources é usado como uma chave para o nome do alvo no modelo de experimento AWS FIS.

```

{
  "description": "aaa",
  "targets": {
    "ManagedResources-Target-1": {
      "resourceType": "aws:arc:zonal-shift-managed-resource",

```

```
        "resourceArns": [
            "arn:aws:elasticloadbalancing:us-east-1:0124567890:loadbalancer/app/
application/11223312312516",
        ],
        "selectionMode": "ALL"
    }
},
"actions": {
    "arc": {
        "actionId": "aws:arc:start-zonal-autoshift",
        "parameters": {
            "availabilityZoneIdentifier": "us-east-1a",
            "duration": "PT1M"
        },
        "targets": {
            "ManagedResources": "ManagedResources-Target-1"
        }
    }
},
"stopConditions": [
    {
        "source": "none"
    }
],
"roleArn": "arn:aws:iam::718579638765:role/fis",
"tags": {},
"experimentOptions": {
    "accountTargeting": "single-account",
    "emptyTargetResolutionMode": "fail"
}
}
```

Metas para o AWS FIS

Um alvo é um ou mais AWS recursos nos quais uma ação é executada pelo AWS AWS Fault Injection Service (FIS) durante um experimento. Os destinos podem estar na mesma conta da AWS do experimento, ou em uma conta diferente usando um experimento com várias contas. Para saber mais sobre como segmentar recursos em uma conta diferente, consulte [Trabalhar com experimentos em várias contas](#).

Você define destinos ao [criar um modelo de experimento](#). Você pode usar o mesmo destino para várias ações em seu modelo de experimento.

AWS O FIS identifica todos os alvos no início do experimento, antes de iniciar qualquer uma das ações no conjunto de ações. AWS O FIS usa os recursos-alvo que seleciona para todo o experimento. Se nenhum destino for encontrado, o experimento falhará.

Sumário

- [Sintaxe de destino](#)
- [Resource types](#)
- [Identificar recursos de destino](#)
 - [Filtros de recursos](#)
 - [Parâmetros de recurso](#)
- [Modo de seleção](#)
- [Exemplos de destinos](#)
- [Exemplo de filtros](#)

Sintaxe de destino

A seguir está a sintaxe para um destino.

```
{
  "targets": {
    "target_name": {
      "resourceType": "resource-type",
      "resourceArns": [
        "resource-arn"
      ],
      "resourceTags": {
        "tag-key": "tag-value"
      },
      "parameters": {
        "parameter-name": "parameter-value"
      },
      "filters": [
        {
          "path": "path-string",
          "values": ["value-string"]
        }
      ],
      "selectionMode": "value"
    }
  }
}
```

```
    }  
  }  
}
```

Quando você define um destino, fornece o seguinte:

target_name

Um nome para o destino.

resourceType

O [tipo de recurso](#).

resourceArns

O nome do recurso da Amazon (ARN) dos recursos específicos.

resourceTags

As tags aplicadas a recursos específicos.

parameters

Os [parâmetros](#) que identificam destinos usando atributos específicos.

filters

O [recurso filtra](#) o escopo dos recursos de destino identificados usando atributos específicos.

selectionMode

O [modo de seleção](#) dos recursos identificados.

Para obter exemplos, consulte [the section called “Exemplos de destinos”](#).

Resource types

Cada ação AWS do FIS é executada em um tipo de AWS recurso específico. Quando você define um destino, deve especificar exatamente um tipo de recurso. Quando você especifica um destino para uma ação, o destino deve ser o tipo de recurso compatível com a ação.

Os seguintes tipos de recursos são compatíveis com o AWS FIS:

- `aws:arc: zonal-shift-managed-resource` — Um AWS recurso registrado com ARC zonal shift

- `aws:directconnect:virtual-interface` — Uma interface virtual Direct Connect
- `aws:dsq:cluster` — Um cluster Amazon Aurora DSQL
- `aws:dynamodb:global-table` — Uma tabela global multirregional do Amazon DynamoDB
- `aws:ec2:autoscaling-group` — Um grupo do Amazon Auto Scaling EC2
- `aws:ec2:ebs-volume` – Um volume do Amazon EBS
- `aws:ec2:instance` — Uma instância da Amazon EC2
- `aws:ec2:spot-instance` — Uma instância spot da Amazon EC2
- `aws:ec2:subnet` – Uma sub-rede do Amazon VPC
- `aws:ec2:transit-gateway`: um gateway de trânsito
- `aws:ec2:vpc-endpoint` — Um endpoint Amazon VPC
- `aws:ecs:cluster` – Um cluster do Amazon ECS
- `aws:ecs:task` – Uma tarefa do Amazon ECS
- `aws:eks:cluster` – Um cluster do Amazon EKS
- `aws:eks:nodegroup` – Um grupo de nós do Amazon EKS
- `aws:eks:pod` – Um pod do Kubernetes
- `aws:elasticache:replicationgroup` — Um grupo de replicação ElastiCache
- `aws:iam:role` — um perfil do IAM
- `aws:kinesis:stream` — Um stream de dados do Amazon Kinesis
- `aws:lambda:function` — Uma função AWS Lambda
- `aws:memorydb:multi-region-cluster` Um cluster multirregional Amazon MemoryDB
- `aws:rds:cluster` – Um cluster do Amazon Aurora DB
- `aws:rds:db` – Uma instância do Amazon RDS DB
- `aws:s3:bucket`: um bucket do Amazon S3

Identificar recursos de destino

Ao definir um alvo no console do AWS FIS, você pode escolher AWS recursos específicos (de um tipo de recurso específico) para segmentar. Ou você pode permitir que o AWS FIS identifique um grupo de recursos com base nos critérios que você fornece.

Para identificar seus recursos de destino, você pode especificar o seguinte:

- Recurso IDs — O recurso IDs de AWS recursos específicos. Todos os recursos IDs devem representar o mesmo tipo de recurso.
- Tags de recursos — As tags aplicadas a AWS recursos específicos.
- Filtros de recursos – O caminho e os valores que representam recursos com atributos específicos. Para obter mais informações, consulte [Filtros de recursos](#).
- Parâmetros de recursos – Os parâmetros que representam recursos que atendem a critérios específicos. Para obter mais informações, consulte [Parâmetros de recurso](#).

Considerações

- Você não pode especificar um ID de recurso e uma tag de recurso para o mesmo destino.
- Você não pode especificar um ID de recurso e um filtro de recurso para o mesmo destino.
- Se você especificar uma tag de recurso com um valor de tag vazio, isso não é equivalente a um caractere curinga. Ele combina recursos que têm uma tag com a chave de tag especificada e um valor de tag vazio.
- Se você especificar mais de uma tag, todas as tags especificadas deverão estar presentes no recurso de destino para que ele seja selecionado (AND).

Filtros de recursos

Os filtros de recursos são consultas que identificam os recursos de destino de acordo com atributos específicos. AWS O FIS aplica a consulta à saída de uma ação de API que contém a descrição canônica do AWS recurso, de acordo com o tipo de recurso que você especificar. Os recursos que têm atributos que correspondem à consulta são incluídos na definição de destino.

Cada filtro é expresso como um caminho de atributo e valores possíveis. Um caminho é uma sequência de elementos, separados por pontos, que descrevem o caminho para alcançar um atributo na saída da ação Descrever de um recurso. Cada período representa a expansão de um elemento. Cada elemento deve ser expresso em letras maiúsculas e minúsculas, mesmo que a saída da ação Descrever de um recurso esteja em maiúsculas e minúsculas. Por exemplo, você deve usar `AvailabilityZone`, não `availabilityZone` como um elemento de atributo.

```
"filters": [  
  {  
    "path": "Component.Component.Component",  
    "values": [  
      "string"  
    ]  
  }  
]
```

```

    ]
  }
],

```

A lógica a seguir se aplica a todos os filtros de recursos:

- Se vários filtros forem fornecidos, incluindo filtros com o mesmo caminho, todos os filtros precisarão ser combinados para que um recurso seja selecionado — AND
- Se vários valores forem fornecidos para um único filtro, qualquer valor precisará ser correspondido para que um recurso seja selecionado — OR
- Se vários valores forem encontrados no local do caminho da chamada de API de descrição, qualquer valor precisará ser correspondido para que um recurso seja selecionado — OR
- Para combinar os key/value pares de tags, você deve selecionar os recursos de destino por tags (veja acima).

A tabela a seguir inclui as ações e AWS CLI os comandos da API que você pode usar para obter as descrições canônicas de cada tipo de recurso. AWS O FIS executa essas ações em seu nome para aplicar os filtros que você especifica. A documentação correspondente descreve os recursos incluídos nos resultados por padrão. Por exemplo, a documentação `DescribeInstances` indica que instâncias encerradas recentemente podem aparecer nos resultados.

Tipo de atributo	Ação da API	AWS CLI comando
aws:arc:zonal-shift-managed-resource	ListManagedResources	list-managed-resources
aws:directconnect:virtual-interface	DescribeVirtualInterfaces	describe-virtual-interfaces
aws:ec2:autoscaling-group	DescribeAutoScalingGroups	describe-auto-scaling-groups
aws:ec2:ebs-volume	DescribeVolumes	describe-volumes
aws:ec2:instance	DescribeInstances	describe-instances
aws:ec2:subnet	DescribeSubnets	describe-subnets
aws:ec2:transit-gateway	DescribeTransitGateways	describe-transit-gateways

Tipo de atributo	Ação da API	AWS CLI comando
aws:ec2:vpc-endpoint	DescribeVpcEndpoints	describe-vpc-endpoints
aws:ecs:cluster	DescribeClusters	describe-clusters
aws:ecs:task	DescribeTasks	describe-tasks
aws:eks:cluster	DescribeClusters	describe-clusters
aws:eks:nodegroup	DescribeNodegroup	describe-nodegroup
aws:elasticache:replication group	DescribeReplicationGroups	describe-replication-groups
aws:iam:role	ListRoles	list-roles
aws:kinesis:stream	DescribeStreamSummary	describe-stream-summary
aws:lambda:function	ListFunctions	funções de lista
aws:memorydb:multi-region-cluster	DescribeMultiRegionClusters	describe-multi-region-clusters
aws:rds:cluster	DescribeDBClusters	describe-db-clusters
aws:rds:db	DescribeDBInstances	describe-db-instances
aws:s3:bucket	ListBuckets	list-buckets
aws:dynamodb:global-table	DescribeTable	tabela de descrição
aws:dsq:cluster	GetCluster	get-cluster

Para obter exemplos, consulte [the section called “Exemplo de filtros”](#).

Parâmetros de recurso

Os parâmetros dos recursos identificam os recursos de destino de acordo com critérios específicos.

O tipo de recurso a seguir oferece suporte a parâmetros.

aws:ec2:ebs-volume

- `availabilityZoneIdentifier` – O código (por exemplo, us-east-1a) da zona de disponibilidade que contém os volumes de destino.

aws:ec2:subnet

- `availabilityZoneIdentifier` – O código (por exemplo, us-east-1a) ou o ID da AZ (por exemplo, use1-az1) da zona de disponibilidade que contém as sub-redes de destino.
- `vpc` – A VPC que contém as sub-redes de destino. Não é permitido mais de uma VPC por conta.

aws:ecs:task

- `cluster` – O cluster que contém as tarefas de destino.
- `service` – O cluster que contém as tarefas de destino.

aws:eks:pod

- `availabilityZoneIdentifier`: opcional. A zona de disponibilidade que contém os pods de destino. Por exemplo, `.us-east-1d` Determinamos a zona de disponibilidade de um pod comparando seu HostIP e o CIDR da sub-rede do cluster.
- `clusterIdentifier` – obrigatório. O nome ou o ARN do cluster do EKS de destino.
- `namespace` – obrigatório. O namespace do Kubernetes dos pods de destino.
- `selectorType` – obrigatório. O tipo de seletor. Os valores possíveis são `labelSelector`, `deploymentName` e `podName`.
- `selectorValue` – obrigatório. O valor do seletor. Esse valor depende do valor de `selectorType`.
- `targetContainerName`: opcional. O nome do contêiner de destino, conforme a especificação do pod. O padrão é o primeiro contêiner definido em cada especificação do pod de destino.

aws:lambda:function

- `functionQualifier`: opcional. A versão ou o alias da função a ser segmentada. Se nenhum qualificador for especificado, todas as invocações serão consideradas para segmentação. Se um alias com várias versões for especificado, todas as versões incluídas no alias serão consideradas para segmentação, desde que sejam invocadas usando um ARN contendo o alias. Se o alias especial `$LATEST` for usado, as invocações para a função base ARN e as invocações incluídas `$LATEST` no ARN serão consideradas para injeção de falhas. Para obter mais informações sobre as versões do Lambda, consulte Gerenciar versões [da função Lambda no guia do usuário](#).AWS Lambda

aws:rds:cluster

- `writerAvailabilityZoneIdentifiers`: opcional. As zonas de disponibilidade do gravador do cluster de banco de dados. Os valores possíveis são: uma lista separada por vírgulas de identificadores de zona de disponibilidade, `all`.

aws:rds:db

- `availabilityZoneIdentifiers`: opcional. As zonas de disponibilidade da instância de banco de dados a serem afetadas. Os valores possíveis são: uma lista separada por vírgulas de identificadores de zona de disponibilidade, `all`.

aws:elasticache:replicationgroup

- `availabilityZoneIdentifier` – obrigatório. O código (por exemplo, `us-east-1a`) ou a ID da AZ (por exemplo, `use1-az1`) da zona de disponibilidade que contém os nós de destino.

Modo de seleção

Você define o escopo dos recursos identificados especificando um modo de seleção. AWS O FIS suporta os seguintes modos de seleção:

- `ALL` – Executar a ação em todos os destinos.
- `COUNT(n)` – Executar a ação no número especificado de destinos escolhidos aleatoriamente entre os destinos identificados. Por exemplo, `COUNT(1)` seleciona um dos destinos identificados.
- `PERCENT(n)` – Executar a ação no percentual especificado de destinos escolhidos aleatoriamente entre os destinos identificados. Por exemplo, `PERCENT(25)` seleciona 25% dos destinos identificados.

Se você tiver um número ímpar de recursos e especificar 50%, o AWS FIS arredonda para baixo. Por exemplo, se você adicionar cinco EC2 instâncias da Amazon como metas e o escopo até 50%, o AWS FIS arredonda para duas instâncias. Você não pode especificar uma porcentagem menor que um recurso. Por exemplo, se você adicionar quatro EC2 instâncias da Amazon e o escopo chegar a 5%, o AWS FIS não poderá selecionar uma instância.

Se você definir vários alvos usando o mesmo tipo de recurso de destino, o AWS FIS poderá selecionar o mesmo recurso várias vezes.

Independentemente do modo de seleção usado, se o escopo especificado não identificar recursos, o experimento falhará.

Exemplos de destinos

Os seguintes são exemplos de destinos.

Exemplos

- [Instâncias na VPC especificada com as tags especificadas](#)
- [Tarefas com os parâmetros especificados](#)

Exemplo: instâncias na VPC especificada com as tags especificadas

Os destinos possíveis para este exemplo são EC2 instâncias da Amazon na VPC especificada com a tag. `env=prod` O modo de seleção especifica que o AWS FIS escolha um desses alvos aleatoriamente.

```
{
  "targets": {
    "randomInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceTags": {
        "env": "prod"
      },
      "filters": [
        {
          "path": "VpcId",
          "values": [
            "vpc-aabbcc11223344556"
          ]
        }
      ],
      "selectionMode": "COUNT(1)"
    }
  }
}
```

Exemplo: tarefas com os parâmetros especificados

Os destinos possíveis para este exemplo são tarefas do Amazon ECS com o cluster e o serviço especificados. O modo de seleção especifica que o AWS FIS escolha um desses alvos aleatoriamente.

```
{
  "targets": {
    "randomTask": {
      "resourceType": "aws:ecs:task",
      "parameters": {
        "cluster": "myCluster",
        "service": "myService"
      },
      "selectionMode": "COUNT(1)"
    }
  }
}
```

Exemplo de filtros

Os seguintes são exemplos de filtros.

Exemplos

- [EC2 Instâncias](#)
- [Clusters do banco de dados](#)

Exemplo: EC2 instâncias

Quando você especifica um filtro para uma ação compatível com o tipo de recurso `aws:ec2:instance`, o AWS FIS usa o comando EC2 `describe-instances` Amazon e aplica o filtro para identificar os destinos.

O comando `describe-instances` retorna a saída JSON em que cada instância é uma estrutura do `Instances`. A seguir está uma saída parcial que inclui campos marcados com *italics*. Forneceremos exemplos que usam esses campos para especificar um caminho de atributo a partir da estrutura da saída JSON.

```
{
  "Reservations": [
    {
      "Groups": [],
      "Instances": [
        {
          "ImageId": "ami-0011111111111111",
```

```
"InstanceId": "i-00aaaaaaaaaaaaaaaa",
"InstanceType": "t2.micro",
"KeyName": "virginia-kp",
"LaunchTime": "2020-09-30T11:38:17.000Z",
"Monitoring": {
  "State": "disabled"
},
"Placement": {
  "AvailabilityZone": "us-east-1a",
  "GroupName": "",
  "Tenancy": "default"
},
"PrivateDnsName": "ip-10-0-1-240.ec2.internal",
"PrivateIpAddress": "10.0.1.240",
"ProductCodes": [],
"PublicDnsName": "ec2-203-0-113-17.compute-1.amazonaws.com",
"PublicIpAddress": "203.0.113.17",
"State": {
  "Code": 16,
  "Name": "running"
},
"StateTransitionReason": "",
"SubnetId": "subnet-aabbcc11223344556",
"VpcId": "vpc-00bbbbbbbbbbbbbbbb",
...
"NetworkInterfaces": [
{
  ...
  "Groups": [
    {
      "GroupName": "sec-group-1",
      "GroupId": "sg-a0011223344556677"
    },
    {
      "GroupName": "sec-group-1",
      "GroupId": "sg-b9988776655443322"
    }
  ],
  ...
},
...
],
...
{
```

```

        ...
    }
  ],
  "OwnerId": "123456789012",
  "ReservationId": "r-aaaaaabbbbb111111"
},
...
]
}

```

Para selecionar instâncias em uma zona de disponibilidade específica usando um filtro de recursos, especifique o caminho do atributo para `AvailabilityZone` e o código da zona de disponibilidade como valor. Por exemplo:

```

"filters": [
  {
    "path": "Placement.AvailabilityZone",
    "values": [ "us-east-1a" ]
  }
],

```

Para selecionar instâncias em uma sub-rede específica usando um filtro de recurso, especifique o caminho do atributo para `SubnetId` e o ID da sub-rede como o valor. Por exemplo:

```

"filters": [
  {
    "path": "SubnetId",
    "values": [ "subnet-aabbcc11223344556" ]
  }
],

```

Para selecionar instâncias que estão em um estado de instância específico, especifique o caminho do atributo para `Name` e um dos seguintes nomes de estado como o valor: `pending` | `running` | `shutting-down` | `terminated` | `stopping` | `stopped`. Por exemplo:

```

"filters": [
  {
    "path": "State.Name",
    "values": [ "running" ]
  }
],

```

Para selecionar instâncias que tenham vários grupos de segurança anexados, especifique um único filtro com o caminho do atributo `GroupId` e vários grupos de segurança IDs. Por exemplo:

```
"filters": [
  {
    "path": "NetworkInterfaces.Groups.GroupId",
    "values": [
      "sg-a0011223344556677",
      "sg-f1100110011001100"
    ]
  }
],
```

Para selecionar instâncias que tenham vários grupos de segurança anexados, especifique vários filtros com o caminho do atributo `GroupId` e um único ID do grupo de segurança para cada filtro. Por exemplo:

```
"filters": [
  {
    "path": "NetworkInterfaces.Groups.GroupId",
    "values": [
      "sg-a0011223344556677"
    ]
  },
  {
    "path": "NetworkInterfaces.Groups.GroupId",
    "values": [
      "sg-b9988776655443322"
    ]
  }
],
```

Exemplo: cluster Amazon RDS (cluster do banco de dados)

Quando você especifica um filtro para uma ação que suporta o tipo de recurso `aws:rds:cluster`, o FIS AWS executa o `describe-db-clusters` comando Amazon RDS e aplica o filtro para identificar os destinos.

O comando `describe-db-clusters` retorna uma saída JSON semelhante à seguinte para cada cluster de banco de dados. A seguir está uma saída parcial que inclui campos marcados com *italics*.

Forneceremos exemplos que usam esses campos para especificar um caminho de atributo a partir da estrutura da saída JSON.

```
[
  {
    "AllocatedStorage": 1,
    "AvailabilityZones": [
      "us-east-2a",
      "us-east-2b",
      "us-east-2c"
    ],
    "BackupRetentionPeriod": 7,
    "DatabaseName": "",
    "DBClusterIdentifier": "database-1",
    "DBClusterParameterGroup": "default.aurora-postgresql11",
    "DBSubnetGroup": "default-vpc-01234567abc123456",
    "Status": "available",
    "EarliestRestorableTime": "2020-11-13T15:08:32.211Z",
    "Endpoint": "database-1.cluster-example.us-east-2.rds.amazonaws.com",
    "ReaderEndpoint": "database-1.cluster-ro-example.us-east-2.rds.amazonaws.com",
    "MultiAZ": false,
    "Engine": "aurora-postgresql",
    "EngineVersion": "11.7",
    ...
  }
]
```

Para aplicar um filtro de recursos que retorna somente os clusters de banco de dados que usam um mecanismo de banco de dados específico, especifique o caminho do atributo para `Engine` e o valor como `aurora-postgresql` conforme mostrado no exemplo a seguir.

```
"filters": [
  {
    "path": "Engine",
    "values": [ "aurora-postgresql" ]
  }
],
```

Para aplicar um filtro de recurso que retorna apenas os clusters de banco de dados em uma zona de disponibilidade específica, especifique o caminho do atributo e o valor conforme mostrado no exemplo a seguir.

```
"filters": [
  {
    "path": "AvailabilityZones",
    "values": [ "us-east-2a" ]
  }
],
```

Condições de parada para o AWS FIS

AWS O AWS Fault Injection Service (FIS) fornece controles e proteções para que você execute experimentos com segurança em cargas de trabalho. AWS Uma condição de parada é um mecanismo para interromper um experimento se ele atingir um limite que você define como um CloudWatch alarme da Amazon. Se uma condição de parada for acionada durante um experimento, o AWS FIS interrompe o experimento. Você não pode retomar um experimento interrompido.

Para criar uma condição de parada, primeiro defina o estado estável do seu aplicativo ou serviço. O estado estável é quando seu aplicativo está funcionando de maneira ideal, definido em termos de métricas comerciais ou técnicas. Por exemplo, latência, carga da CPU ou número de novas tentativas. Você pode usar o estado estacionário para criar um CloudWatch alarme que pode ser usado para interromper um experimento se seu aplicativo ou serviço atingir um estado em que seu desempenho não seja aceitável. Para obter mais informações, consulte [Usando CloudWatch alarmes da Amazon](#) no Guia do CloudWatch usuário da Amazon.

Sua conta da tem uma cota quanto ao número de condições de parada que você pode especificar em um modelo de experimento. Para obter mais informações, consulte [Cotas e limitações para o serviço de injeção de AWS falhas](#).

Sintaxe da condição de parada

Ao criar um modelo de experimento, você especifica uma ou mais condições de parada especificando os CloudWatch alarmes que você criou.

```
{
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "arn:aws:cloudwatch:region:123456789012:alarm:alarm-name"
    }
  ]
}
```

```
]
}
```

O exemplo a seguir indica que o modelo do experimento não especifica uma condição de parada.

```
{
  "stopConditions": [
    {
      "source": "none"
    }
  ]
}
```

Saiba mais

Para ver um tutorial que demonstra como criar um CloudWatch alarme e adicionar uma condição de parada a um modelo de experimento, consulte [Executar o estresse da CPU em uma instância](#).

Para obter mais informações sobre as CloudWatch métricas disponíveis para os tipos de recursos suportados pelo AWS FIS, consulte o seguinte:

- [Monitore suas instâncias usando CloudWatch](#)
- [Métricas do Amazon ECS CloudWatch](#)
- [Monitoramento de métricas do Amazon RDS usando CloudWatch](#)
- [Monitorando métricas do Run Command usando CloudWatch](#)

Funções do IAM para experimentos AWS do FIS

AWS Identity and Access Management (IAM) é um AWS serviço que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Para usar o AWS FIS, você deve criar uma função do IAM que conceda ao AWS FIS as permissões necessárias para que o AWS FIS possa realizar experimentos em seu nome. Você especifica esse perfil do experimento ao criar um modelo de experimento. Para um experimento de conta única, a política do IAM para o perfil do experimento deve conceder permissão para modificar os recursos que você especifica como destinos no modelo de experimento. Para um experimento com várias contas, o perfil do experimento deve conceder permissão ao perfil de orquestrador para assumir o perfil do IAM em cada conta de destino. Para obter mais informações, consulte [Permissões para experimentos com várias contas](#).

Recomendamos seguir a prática de segurança padrão de conceder privilégio mínimo. Você pode fazer isso especificando recursos ARNs ou tags específicos em suas políticas.

Para ajudar você a começar a usar o AWS FIS rapidamente, fornecemos políticas AWS gerenciadas que você pode especificar ao criar uma função experimental. Como alternativa, você também pode usar essas políticas como modelo ao criar seus próprios documentos de política em linha.

Conteúdos

- [Pré-requisitos](#)
- [Opção 1: criar uma função experimental e anexar uma política AWS gerenciada](#)
- [Opção 2: criar um perfil experimental e adicionar um documento de política em linha](#)

Pré-requisitos

Antes de começar, instale AWS CLI e crie a política de confiança necessária.

Instale o AWS CLI

Antes de começar, instale e configure a AWS CLI. Ao configurar o AWS CLI, você é solicitado a fornecer as AWS credenciais. Os exemplos neste procedimento pressupõem que você também tenha configurado uma região padrão. Caso contrário, adicione a opção `--region` para cada comando. Para obter informações, consulte [Instalação e configuração da AWS CLI](#) e [Configuração da AWS CLI](#).

Criar uma política de relacionamento de confiança

Uma função experimental deve ter uma relação de confiança que permita que o serviço AWS FIS assuma a função. Crie um arquivo de texto chamado `fis-role-trust-policy.json` e adicione a política de relacionamento de confiança a seguir.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "fis.amazonaws.com"
        ]
      }
    }
  ]
}
```

```

    ]
  },
  "Action": "sts:AssumeRole"
}
]
}

```

Recomendamos o uso das chaves de condição `aws:SourceAccount` e `aws:SourceArn` para se proteger contra o [problema confused deputy](#). A conta de origem é o proprietário do experimento e o ARN de origem é o ARN do experimento. Por exemplo, você deveria adicionar o bloco de condições a seguir à política de confiança.

```

"Condition": {
  "StringEquals": {
    "aws:SourceAccount": "account_id"
  },
  "ArnLike": {
    "aws:SourceArn": "arn:aws:fis:region:account_id:experiment/*"
  }
}

```

Adicione permissões para assumir os perfil da conta-alvo (somente experimentos com várias contas)

Para experimentos com várias contas, você precisa de permissões para que a conta do orquestrador assuma os perfis da conta de destino. É possível modificar o exemplo a seguir e adicioná-lo como um documento de política em linha para assumir os perfis da conta de destino:

```

{
  "Effect": "Allow",
  "Action": "sts:AssumeRole",
  "Resource": [
    "arn:aws:iam::target_account_id:role/role_name"
  ]
}

```

Opção 1: criar uma função experimental e anexar uma política AWS gerenciada

Use uma das políticas AWS gerenciadas do AWS FIS para começar rapidamente.

Para criar uma função experimental e anexar uma política AWS gerenciada

1. Verifique se há uma política gerenciada para as ações do AWS FIS em seu experimento. Caso contrário, você precisará criar seu próprio documento de política em linha. Para obter mais informações, consulte [the section called “AWS políticas gerenciadas”](#).
2. Use o comando [create-role](#) a seguir para criar um perfil e adicionar a política de confiança que você criou nos pré-requisitos.

```
aws iam create-role --role-name my-fis-role --assume-role-policy-document  
file://fis-role-trust-policy.json
```

3. Use o [attach-role-policy](#) comando a seguir para anexar a política AWS gerenciada.

```
aws iam attach-role-policy --role-name my-fis-role --policy-arn fis-policy-arn
```

Onde *fis-policy-arn* está uma das seguintes opções:

- arn:aws:iam::aws:policy/service-role/AWSFaultInjectionSimulatorEC2Access
- arn:aws:iam::aws:policy/service-role/AWSFaultInjectionSimulatorECSAccess
- arn:aws:iam::aws:policy/service-role/AWSFaultInjectionSimulatorEKSAccess
- arn:aws:iam::aws:policy/service-role/AWSFaultInjectionSimulatorNetworkAccess
- arn:aws:iam::aws:policy/service-role/AWSFaultInjectionSimulatorRDSAccess
- arn:aws:iam::aws:policy/service-role/AWSFaultInjectionSimulatorSSMAccess

Opção 2: criar um perfil experimental e adicionar um documento de política em linha

Use essa opção para ações que não tenham uma política gerenciada ou para incluir somente as permissões necessárias para seu experimento específico.

Para criar um experimento e adicionar um documento de política em linha

1. Use o comando [create-role](#) a seguir para criar um perfil e adicionar a política de confiança que você criou nos pré-requisitos.

```
aws iam create-role --role-name my-fis-role --assume-role-policy-document  
file://fis-role-trust-policy.json
```

2. Crie um arquivo de texto chamado `fis-role-permissions-policy.json` e adicione uma política de permissões. Para obter um exemplo que você pode usar como ponto de partida, consulte o seguinte.

- Ações de injeção de falhas: comece com a política a seguir.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowFISExperimentRoleFaultInjectionActions",
      "Effect": "Allow",
      "Action": [
        "fis:InjectApiInternalError",
        "fis:InjectApiThrottleError",
        "fis:InjectApiUnavailableError"
      ],
      "Resource": "arn:*:fis:*:*:experiment/*"
    }
  ]
}
```

- Ações do Amazon EBS: comece com a política a seguir.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeVolumes"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:PauseVolumeIO"
      ],

```

```

    "Resource": "arn:aws:ec2:*:*:volume/*"
  }
]
}

```

- Ações do Amazon EC2 — Comece com a política de [AWSFaultInjectionSimulatorEC2acesso](#).
 - Ações do Amazon ECS — Comece com a [AWSFaultInjectionSimulatorECSAccess](#) política.
 - Ações do Amazon EKS — Comece com a [AWSFaultInjectionSimulatorEKSAccess](#) política.
 - Ações de rede — Comece com a [AWSFaultInjectionSimulatorNetworkAccess](#) política.
 - Ações do Amazon RDS — Comece com a [AWSFaultInjectionSimulatorRDSAccess](#) política.
 - Ações do Systems Manager — Comece com a [AWSFaultInjectionSimulatorSSMAccess](#) política.
3. Use o [put-role-policy](#) comando a seguir para adicionar a política de permissões que você criou na etapa anterior.

```
aws iam put-role-policy --role-name my-fis-role --policy-name my-fis-policy --
policy-document file://fis-role-permissions-policy.json
```

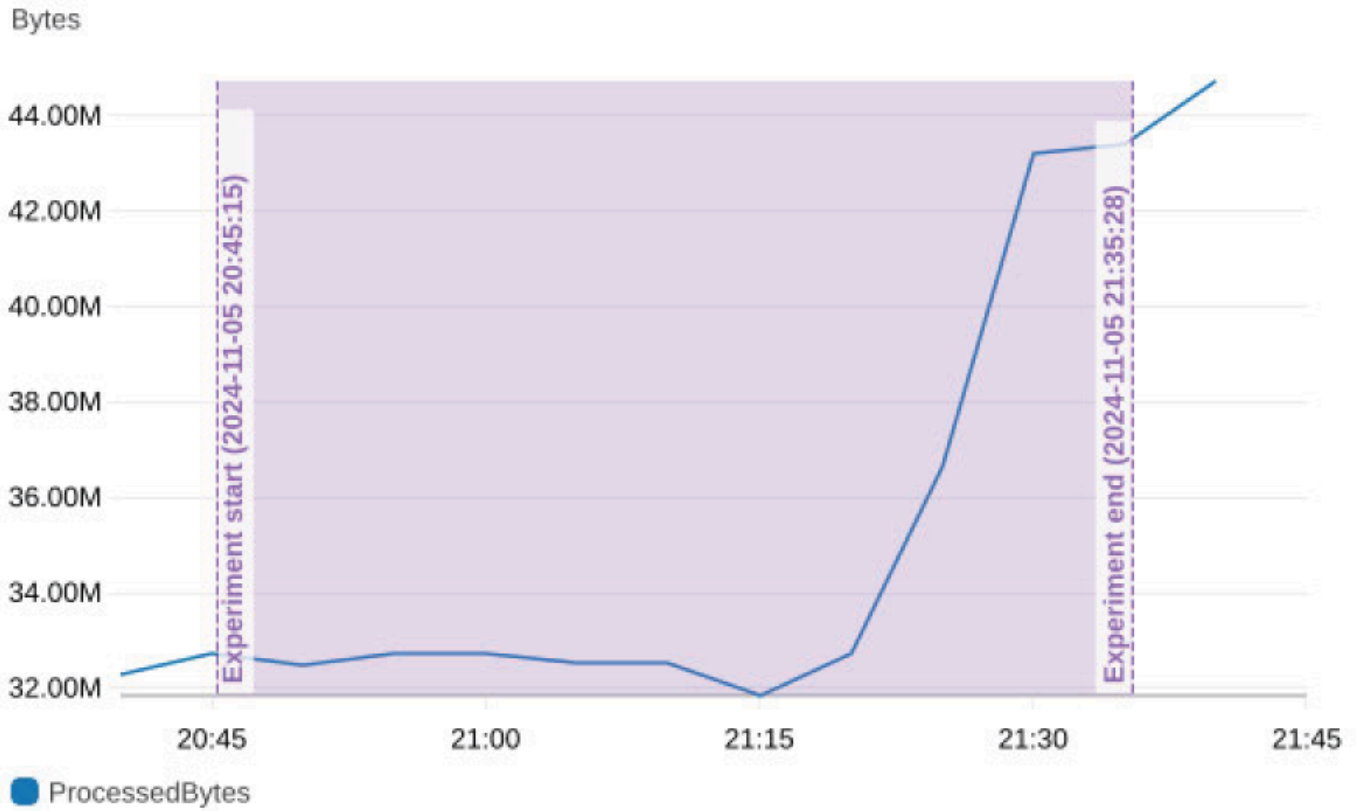
Configurações de relatórios de experimentos para AWS FIS

Você pode ativar o AWS Fault Injection Service (FIS) para gerar relatórios para experimentos, facilitando a produção de evidências de testes de resiliência. O relatório do experimento é um documento PDF que resume as ações do experimento e, opcionalmente, captura a resposta do aplicativo a partir de um CloudWatch painel especificado por você. Para ver um exemplo de relatório de experimento, baixe o arquivo zip [aqui](#).

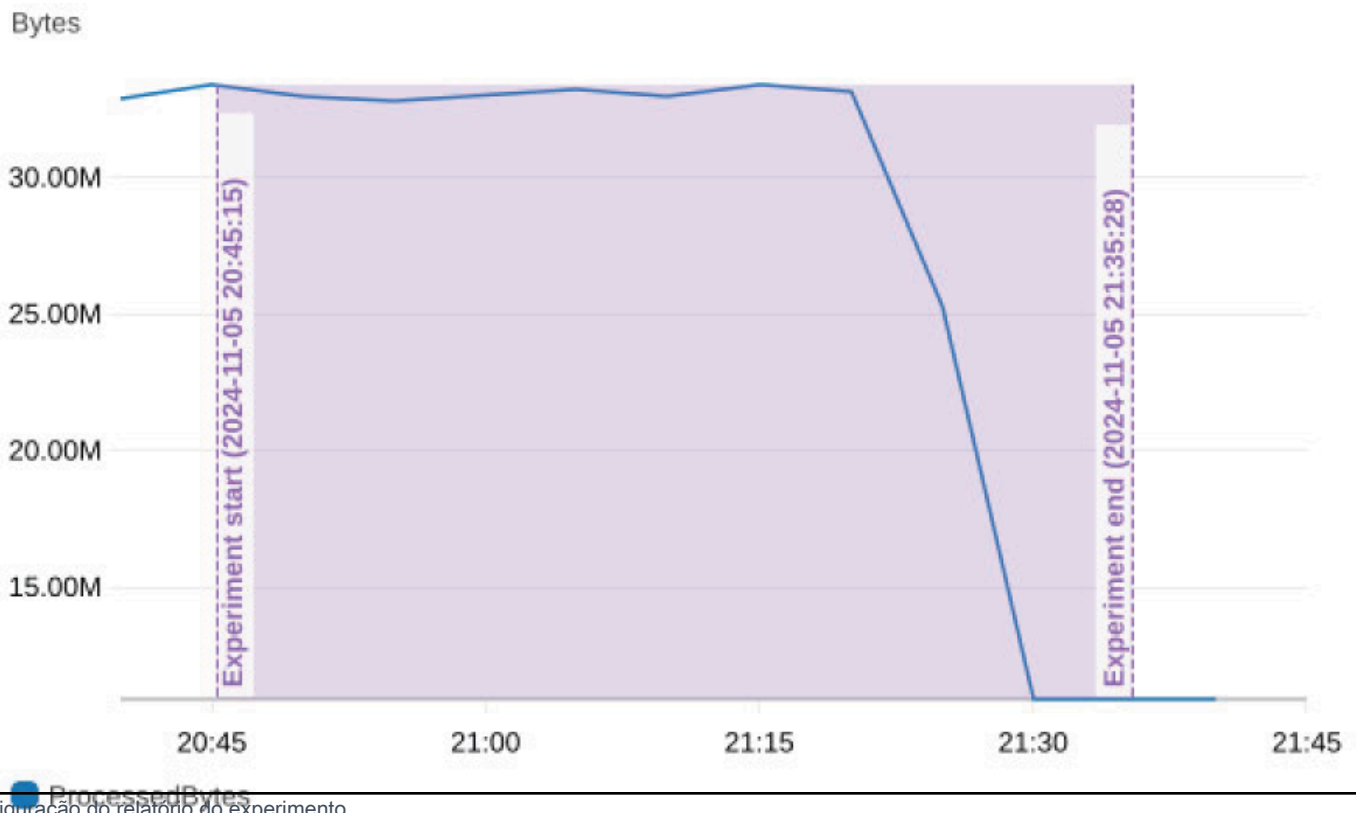
Para ativar e configurar o conteúdo do relatório gerado para o experimento, você define a configuração do relatório do experimento para o modelo do experimento. Quando você especifica um CloudWatch painel, o AWS FIS inclui um gráfico instantâneo de todos os widgets no painel fornecido, anotado com a hora de início e término do experimento durante um período especificado, conforme mostrado no exemplo abaixo.

Este exemplo demonstra o impacto de um experimento de perda de pacotes em uma zona de disponibilidade (AZ). Quando a perda de pacotes é introduzida no AZ use1-az6, o tráfego muda de use1-az6 para use1-az4, de modo que o número de bytes processados pelo balanceador de carga nesse AZ diminui.

NLB ProcessedBytes use1-az4



NLB ProcessedBytes use1-az6



Quando o experimento termina, o relatório pode ser baixado do console do AWS FIS e também armazenado em um bucket do Amazon S3. Se você incluir um CloudWatch painel na configuração do relatório, imagens de cada widget também serão entregues. Os relatórios não são gerados para experimentos que são cancelled ou são executados como parte da visualização de destino (com o ActionsMode definido como). skip-all Quando o experimento exceder o limite de retenção de dados do experimento, o relatório estará disponível somente no bucket do Amazon S3. AWS As cobranças do FIS se aplicam a cada relatório entregue, exceto aqueles que falham com erros internos. Para obter mais informações, consulte [Preços do AWS Fault Injection Service Cotas e limitações para o serviço de injeção de AWS falhas](#) e. Taxas de ingestão e armazenamento do Amazon S3 CloudWatch e taxas de API GetDashboard GetMetricWidgetImagesolicitações podem ser aplicadas. [Para obter mais informações, consulte os preços e CloudWatch preços do Amazon S3.](#)

Conteúdo

- [Síntaxe de configuração do relatório do experimento](#)
- [Permissões do relatório de experimentos](#)
- [Práticas recomendadas para relatórios de experimentos](#)

Síntaxe de configuração do relatório do experimento

A seguir está a síntaxe da configuração do relatório do experimento, uma seção opcional do modelo do experimento.

```
{
  "experimentReportConfiguration": {
    "outputs": {
      "s3Configuration": {
        "bucketName": "my-bucket-name",
        "prefix": "report-storage-prefix"
      }
    },
    "dataSources": {
      "cloudWatchDashboards": [
        {
          "dashboardIdentifier": "arn:aws:cloudwatch::123456789012:dashboard/MyDashboard"
        }
      ]
    }
  },
}
```

```
        "preExperimentDuration": "PT20M",  
        "postExperimentDuration": "PT20M"  
    }  
}
```

Usando `experimentReportConfiguration`, você pode personalizar o destino de saída, os dados de entrada e as janelas de tempo para que os dados sejam incluídos no relatório do experimento, o que pode ajudá-lo a entender melhor o impacto e os resultados de seus experimentos AWS FIS. Ao definir a configuração do relatório do experimento, você fornece o seguinte:

outputs

Seção do `experimentReportConfiguration` que especifica onde o relatório do experimento será entregue. Em `outputs`, você especifica o `s3Configuration` fornecendo o seguinte:

- `bucketName`- O nome do bucket do Amazon S3 em que o relatório será armazenado. O balde deve estar na mesma região do experimento.
- `prefix`(Opcional) - Um prefixo dentro do bucket do Amazon S3 onde o relatório será armazenado. Esse campo é altamente recomendado para que você possa limitar o acesso somente ao prefixo.

Fontes de dados

Seção opcional do `experimentReportConfiguration` que especifica as fontes de dados adicionais que serão incluídas no relatório do experimento.

- `cloudWatchDashboards`- Uma matriz de CloudWatch painéis que serão incluídos no relatório. Limitado a um CloudWatch painel.
- `dashboardIdentifier`- O ARN do painel. CloudWatch Gráficos instantâneos de cada widget com o tipo `metric` desse painel serão incluídos no relatório, com exceção das métricas entre regiões.

preExperimentDuration

Seção opcional do `experimentReportConfiguration` que define a duração do pré-experimento para que as métricas CloudWatch do painel sejam incluídas no relatório, em até 30 minutos. Esse deve ser um período que represente o estado estável do seu aplicativo. Por exemplo, uma duração pré-experimento de 5 minutos significa que os gráficos instantâneos incluirão métricas 5 minutos antes do início do experimento. O formato da duração é ISO 8601 e o padrão é 20 minutos.

postExperimentDuration

Seção opcional do `experimentReportConfiguration` que define a duração pós-experimento das métricas do CloudWatch painel a serem incluídas no relatório, em até 2 horas. Essa deve ser uma duração que represente o estado estável ou o período de recuperação do aplicativo. Por exemplo, se você especificar uma duração pós-experimento de 5 minutos, os gráficos instantâneos incluirão métricas até 5 minutos após o término do experimento. O formato da duração é ISO 8601 e o padrão é 20 minutos.

Permissões do relatório de experimentos

Para permitir que o AWS FIS gere e armazene o relatório do experimento, você precisa permitir as seguintes operações da sua função do IAM do experimento AWS FIS:

- `cloudwatch:GetDashboard`
- `cloudwatch:GetMetricWidgetImage`
- `s3:GetObject`
- `s3:PutObject`

Recomendamos que você siga as melhores práticas de AWS segurança e restrinja a função do experimento ao intervalo e ao prefixo. Veja a seguir um exemplo de declaração de política que restringe o acesso à função do experimento.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:PutObject",
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::my-experiment-report-bucket/my-prefix/*",
      "Effect": "Allow"
    }
  ]
}
```

```

    {
      "Action": [
        "cloudwatch:GetDashboard"
      ],
      "Resource": "arn:aws:cloudwatch::012345678912:dashboard/my-
experiment-report-dashboard",
      "Effect": "Allow"
    },
    {
      "Action": [
        "cloudwatch:GetMetricWidgetImage"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}

```

Permissões adicionais para relatórios entregues aos buckets do Amazon S3 criptografados com chaves gerenciadas pelo cliente (CMK)

Se o bucket do Amazon S3 que você especificar `S3Configuration` for criptografado com CMK, você precisará conceder as seguintes permissões adicionais à função do experimento FIS em sua política de chaves do KMS:

- `kms:GenerateDataKey`
- `kms:Decrypt`

Veja a seguir um exemplo de declaração de política de chaves do KMS que permite que a função experimental do FIS grave relatórios em buckets criptografados:

```

{
  "Sid": "Allow FIS experiment report",
  "Effect": "Allow",
  "Principal":
  {
    "AWS": [
      "arn:aws:iam::012345678912:role/FISExperimentRole",
    ]
  },
}

```

```
"Action": [  
    "kms:Decrypt",  
    "kms:GenerateDataKey"  
  ],  
"Resource": "*" } }
```

Práticas recomendadas para relatórios de experimentos

A seguir estão as melhores práticas para usar a configuração do relatório do experimento AWS FIS:

- Antes de começar um experimento, gere uma prévia do alvo para verificar se o modelo do experimento está configurado conforme o esperado. A visualização prévia do alvo fornecerá informações sobre os alvos esperados do seu experimento. Para saber mais, consulte [Gerar uma pré-visualização de destinos com base em um modelo de experimento](#).
- O relatório não deve ser usado para solucionar problemas em experimentos que falharam. Em vez disso, use registros de experimentos para solucionar erros de experimentos. Recomendamos que você confie no relatório somente para experimentos que você já executou e concluiu com êxito.
- Restrinja a função do IAM do experimento e obtenha acesso ao objeto ao bucket e prefixo de destino do S3. Recomendamos que você dedique o prefixo bucket/somente aos relatórios de experimentos do AWS FIS e não conceda a outros AWS serviços acesso a esse intervalo e prefixo.
- Use o Amazon S3 Object Lock para evitar que o relatório seja excluído ou sobrescrito por um período fixo ou indefinidamente. Para saber mais, consulte [Bloquear objetos com o Object Lock](#).
- Se seu CloudWatch painel estiver em uma conta separada na mesma região, você poderá usar a observabilidade CloudWatch entre contas para habilitar sua conta do orquestrador AWS FIS como conta de monitoramento e a conta separada como conta de origem a partir do CloudWatch console ou dos comandos do Observability Access Manager na API e. AWS CLI Para saber mais, consulte [CloudWatch observabilidade entre contas](#).

Opções de experimentos para AWS FIS

As opções de experimento são configurações opcionais para um experimento. Você pode definir determinadas opções de experimento no modelo de experimento. As opções adicionais de experimento são definidas quando você inicia o experimento.

Veja a seguir a sintaxe para opções de experimento que você define no modelo de experimento.

```
{
  "experimentOptions": {
    "accountTargeting": "single-account | multi-account",
    "emptyTargetResolutionMode": "fail | skip"
  }
}
```

Se você não especificar nenhuma opção de experimento ao criar o modelo de experimento, será usado o padrão de cada opção.

Veja a seguir a sintaxe para opções de experimento que você define quando inicia o experimento.

```
{
  "experimentOptions": {
    "actionsMode": "run-all | skip-all"
  }
}
```

Se você não especificar nenhuma opção de experimento ao iniciar o experimento, a opção padrão `run-all` será usada.

Conteúdo

- [Segmentação de conta](#)
- [Modo de resolução de destino vazio](#)
- [Modo de ações](#)

Segmentação de conta

Se você tiver várias AWS contas com recursos que deseja segmentar em um experimento, você pode definir um experimento com várias contas usando a opção de experimento de segmentação por conta. Você executa experimentos com várias contas por uma conta de orquestrador que afeta os recursos em várias contas de destino. A conta do orquestrador é proprietária do modelo e do AWS FIS experimento do experimento. Uma conta alvo é uma conta individual da AWS com recursos que podem ser afetados por um AWS FIS experimento. Para obter mais informações, consulte [Trabalhando com experimentos com várias contas para AWS FIS](#).

Você usa a segmentação por conta para indicar a localização dos recursos de destino. Você pode fornecer dois valores para a segmentação da conta:

- **conta única: padrão.** O experimento só terá como alvo os recursos da AWS conta em que o AWS FIS experimento é executado.
- **várias contas:** o experimento pode se voltar para recursos em várias contas da AWS.

Configurações de conta de destino

Para realizar um experimento com várias contas, você deve definir uma ou mais configurações de conta de destino. A configuração da conta de destino especifica `accountId`, `roleArn` e a descrição de cada conta com recursos direcionados ao experimento. A conta IDs das configurações da conta de destino para um modelo de experimento deve ser exclusiva.

Quando você cria um modelo de experimento com várias contas, ele retornará um campo somente para leitura, `targetAccountConfigurationsCount`, que é uma contagem de todas as configurações da conta de destino para o modelo de experimento.

Veja a seguir a sintaxe para uma configuração de conta de destino.

```
{
  accountId: "123456789012",
  roleArn: "arn:aws:iam::123456789012:role/AllowFISActions",
  description: "fis-ec2-test"
}
```

Quando você cria uma configuração de conta de destino, fornece o seguinte:

`accountId`

ID de 12 dígitos da conta de destino da AWS.

`roleArn`

Uma função do IAM que concede AWS FIS permissões para realizar ações na conta de destino.

`description`

Uma descrição opcional.

Para saber mais sobre como trabalhar com as configurações das contas de destino, consulte [Trabalhando com experimentos com várias contas para AWS FIS](#).

Modo de resolução de destino vazio

Esse modo oferece a opção de permitir que os experimentos sejam concluídos mesmo quando um recurso de destino não for resolvido.

- **falha: padrão.** Se nenhum recurso for resolvido para o destino, o experimento será encerrado imediatamente com o status de `failed`.
- **ignorar:** se nenhum recurso for resolvido para o destino, o experimento continuará e todas as ações sem destinos resolvidas serão ignoradas. Ações com alvos definidos usando identificadores exclusivos, como ARNs, não podem ser ignoradas. Se um destino definido usando um identificador exclusivo não for encontrado, o experimento será encerrado imediatamente com o status de `failed`.

Modo de ações

O modo de ações é um parâmetro opcional que você pode especificar ao iniciar um experimento. Você pode definir o modo de ações como `skip-all` para gerar uma pré-visualização de destinos antes de injetar falhas nos recursos de destino. A pré-visualização de destinos permite que você verifique o seguinte:

- Que você configurou o modelo de experimento para atingir os recursos esperados. Os recursos reais que são definidos como destino quando você inicia esse experimento podem ser diferentes da pré-visualização, pois os recursos podem ser removidos, atualizados ou amostrados aleatoriamente.
- Que as configurações de registro em log estão definidas corretamente.
- Que, para experimentos em várias contas, você definiu corretamente um perfil do IAM para as configurações de cada uma das contas de destino.

Note

O `skip-all` modo não permite verificar se você tem as permissões necessárias para executar o AWS FIS experimento e realizar ações em seus recursos.

O parâmetro do modo de ações aceita os seguintes valores:

- `run-all` (padrão): o experimento executará ações nos recursos de destino.

- `skip-all`: o experimento ignorará todas as ações nos recursos de destino.

Para saber mais sobre como configurar o parâmetro do modo de ações ao iniciar um experimento, consulte [Gerar uma pré-visualização de destinos com base em um modelo de experimento](#).

AWS FIS Referência de ações

Uma ação é a atividade de injeção de falhas que você executa em um alvo usando AWS Fault Injection Service (AWS FIS). AWS FIS fornece ações pré-configuradas para tipos específicos de alvos em todos os AWS serviços. Você adiciona ações a um modelo de experimento, que depois usa para realizar experimentos.

Essa referência descreve as ações comuns em AWS FIS, incluindo informações sobre os parâmetros de ação e as permissões necessárias do IAM. Você também pode listar as AWS FIS ações suportadas usando o AWS FIS console ou o comando [list-actions](#) do AWS Command Line Interface (CLI). Depois de ter o nome de uma ação específica, você pode ver informações detalhadas sobre a ação usando o comando [get-action](#). Para obter mais informações sobre o uso de AWS FIS comandos com o AWS CLI, consulte o [Guia AWS Command Line Interface do usuário](#) e [corrija](#) a Referência de AWS CLI comandos.

Para obter mais informações sobre como AWS FIS as ações funcionam, consulte [Ações para o AWS FIS Como o AWS Fault Injection Service funciona com o IAM](#) e.

Ações

- [Ações de injeção de falhas](#)
- [Ação de recuperação](#)
- [Ação de espera](#)
- [CloudWatch Ações da Amazon](#)
- [Ações do Amazon DynamoDB](#)
- [Ações do Amazon Aurora DSQL](#)
- [Ações do Amazon EBS](#)
- [Ações do Amazon EC2](#)
- [Ações do Amazon ECS](#)
- [Ações do Amazon EKS](#)
- [ElastiCache Ações da Amazon](#)
- [Ações do Amazon Kinesis Data Streams](#)
- [AWS Lambda ações](#)
- [Ação Amazon MemoryDB](#)

- [Ações de rede](#)
- [Ações do Amazon RDS](#)
- [Ações do Amazon S3](#)
- [Ações do Systems Manager](#)
- [AWS Direct Connect ações](#)
- [Use documentos SSM do Systems Manager com AWS o FIS](#)
- [Use as ações do AWS FIS `aws:ecs:task`](#)
- [Use as ações do AWS FIS `aws:eks:pod`](#)
- [Use as AWS FIS ações `aws:lambda:function`](#)

Ações de injeção de falhas

AWS FIS suporta as seguintes ações de injeção de falhas.

Ações

- [aws:fis:inject-api-internal-error](#)
- [aws:fis:inject-api-throttle-error](#)
- [aws:fis:inject-api-unavailable-error](#)

aws:fis:inject-api-internal-error

Injeta erros internos nas solicitações feitas pelo perfil do IAM de destino. A resposta específica depende de cada serviço e API. Para ter mais informações, revise a documentação do SDK e da API do seu serviço.

Tipo de atributo

- `aws:iam:role`

Parâmetros

- `duration` – A duração, de um minuto a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, `PT1 M` representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.

- `service`— O namespace AWS da API de destino. O valor suportado é `ec2 kinesis e`.
- `percentage` – A porcentagem (1-100) de chamadas para injetar a falha.
- `operations` – As operações para injetar a falha, separadas por vírgulas. Para obter uma lista das ações de API para o `ec2` namespace, consulte [Amazon EC2 API Reference](#) e [Amazon Kinesis Data Streams API Reference](#).

Permissões

- `fis:InjectApiInternalError`

aws:fis:inject-api-throttle-error

Injeta erros de controle de utilização nas solicitações feitas pelo perfil do IAM de destino. A resposta específica depende de cada serviço e API. Para ter mais informações, revise a documentação do SDK e da API do seu serviço.

Tipo de atributo

- `aws:iam:role`

Parâmetros

- `duration` – A duração, de um minuto a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, `PT1 M` representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.
- `service`— O namespace AWS da API de destino. O valor suportado é `ec2 kinesis e`.
- `percentage` – A porcentagem (1-100) de chamadas para injetar a falha.
- `operations` – As operações para injetar a falha, separadas por vírgulas. Para obter uma lista das ações de API para o `ec2` namespace, consulte [Amazon EC2 API Reference](#) e [Amazon Kinesis Data Streams API Reference](#).

Permissões

- `fis:InjectApiThrottleError`

aws:fis:inject-api-unavailable-error

Injeta erros de indisponibilidade nas solicitações feitas pelo perfil do IAM de destino. A resposta específica depende de cada serviço e API. Para ter mais informações, revise a documentação do SDK e da API do seu serviço.

Tipo de atributo

- `aws:iam:role`

Parâmetros

- `duration` – A duração, de um minuto a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, `PT1 M` representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.
- `service`— O namespace AWS da API de destino. O valor suportado é `ec2 kinesis e`.
- `percentage` – A porcentagem (1-100) de chamadas para injetar a falha.
- `operations` – As operações para injetar a falha, separadas por vírgulas. Para obter uma lista das ações de API para o `ec2` namespace, consulte [Amazon EC2 API Reference](#) e [Amazon Kinesis Data Streams API Reference](#).

Permissões

- `fis:InjectApiUnavailableError`

Ação de recuperação

As ações de recuperação são realizadas para reduzir o risco ou proteger os aplicativos após a deficiência.

AWS FIS suporta as seguintes ações de recuperação.

aws:arc:start-zonal-autoshift

Desloca automaticamente o tráfego dos recursos suportados de uma zona de disponibilidade (AZ) potencialmente comprometida e os redireciona para uma área íntegra AZs na mesma região da AWS. Isso permite experimentar o deslocamento automático zonal por meio do FIS. O deslocamento

automático zonal é um recurso do Amazon Application Recovery Controller (ARC) que permite AWS transferir o tráfego de um recurso para fora de uma AZ, em seu nome, quando AWS determina que há uma deficiência que poderia afetar clientes na AZ.

Quando você executa a `aws:arc:start-zonal-autoshift` ação, AWS FIS gerencia a mudança zonal usando o `StartZonalShift UpdateZonalShift`, e `CancelZonalShift` APIs com o `expiresIn` campo para essas solicitações definido como 1 minuto como mecanismo de segurança. Isso permite AWS FIS reverter rapidamente a mudança de zona no caso de eventos inesperados, como interrupções na rede ou problemas no sistema. No console ARC, o campo de tempo de expiração será exibido AWS FIS-managed, e a expiração real esperada será determinada pela duração especificada na ação de mudança zonal.

Tipo de atributo

- `aws:arc:zonal-shift-managed-resource`

Os recursos gerenciados por turnos zonais são tipos de recursos que incluem clusters Amazon EKS, Amazon EC2 Application and Network Load Balancers e grupos do Amazon EC2 Auto Scaling que podem ser habilitados para o deslocamento automático zonal ARC. Para obter mais informações, consulte [os recursos suportados](#) e a [ativação de recursos de mudança automática zonal no Guia](#) do desenvolvedor do ARC.

Parâmetros

- `duration`— O período de tempo durante o qual o tráfego será deslocado. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, `PT1 M` representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.
- `availabilityZoneIdentifier`— O tráfego se afasta dessa AZ. Isso pode ser um nome de AZ (`us-east-1a`) ou ID de AZ (`use1-az1`).
- `managedResourceTypes`— Os tipos de recursos dos quais o tráfego será deslocado, separados por vírgulas. As opções possíveis são ASG (Auto Scaling Group), ALB (Application Load Balancer)NLB, (Network Load Balancer) e EKS (Amazon EKS).
- `zonalAutoshiftStatus`— O `zonalAutoshiftStatus` status dos recursos que você deseja segmentar. As opções possíveis são `ENABLEDDISABLED`, `ANY` e. O padrão é `ENABLED`.

Permissões

- arc-zonal-shift:StartZonalShift
- arc-zonal-shift:GetManagedResource
- arc-zonal-shift:UpdateZonalShift
- arc-zonal-shift:CancelZonalShift
- arc-zonal-shift>ListManagedResources
- escalonamento automático: DescribeTags
- etiqueta: GetResources

Ação de espera

AWS FIS suporta a seguinte ação de espera.

aws:fis:wait

Executa a ação de AWS FIS espera.

Parâmetros

- duration – A duração, de um minuto a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT1 M representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.

Permissões

- Nenhum

CloudWatch Ações da Amazon

AWS FIS apoia a seguinte CloudWatch ação da Amazon.

aws:cloudwatch:assert-alarm-state

Verifica se os alarmes especificados estão em um dos estados de alarme especificados.

Tipo de atributo

- Nenhum

Parâmetros

- `alarmArns`— O ARNs dos alarmes, separados por vírgulas. Você pode especificar até cinco alarmes.
- `alarmStates` – Os estados dos alarmes, separados por vírgulas. Os possíveis estados do volume são OK, ALARM e INSUFFICIENT_DATA.

Permissões

- `cloudwatch:DescribeAlarms`

Ações do Amazon DynamoDB

AWS FIS suporta a seguinte ação do Amazon DynamoDB.

`aws:dynamodb:global-table-pause-replication`

Pausa a replicação de tabela global multirregional do Amazon DynamoDB para qualquer tabela de réplica. As tabelas podem continuar sendo replicadas por até cinco minutos após o início da ação.

Tabelas globais altamente consistentes em várias regiões (MRSC)

As instruções a seguir serão anexadas dinamicamente à política da tabela global de destino do DynamoDB MRSC:

```
{
  "Statement": [
    {
      "Sid": "DoNotModifyFisDynamoDbPauseReplicationEXPxxxxxxxxxxxxxxxx",
      "Effect": "Deny",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "dynamodb:UpdateTable"
      ]
    }
  ]
}
```

```

    ],
    "Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/ExampleGlobalTable",
    "Condition": {
      "DateLessThan": {
        "aws:CurrentTime": "2024-04-10T09:51:41.511Z"
      },
      "ArnEquals": {
        "aws:PrincipalArn": "arn:aws:iam::123456789012:role/aws-service-role/
replication.dynamodb.amazonaws.com/AWSServiceRoleForDynamoDBReplication"
      }
    }
  },
  {
    "Sid":
"DoNotModifyFisDynamoDbPauseReplicationEXPxxxxxxxxxxxxxxxxxxxxForApplicationAutoScaling",
    "Effect": "Deny",
    "Principal": {
      "AWS": "*"
    },
    "Action": [
      "dynamodb:DescribeTable",
      "dynamodb:UpdateTable"
    ],
    "Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/ExampleGlobalTable",
    "Condition": {
      "DateLessThan": {
        "aws:CurrentTime": "2024-04-10T09:51:41.511Z"
      },
      "ArnEquals": {
        "aws:PrincipalArn": "arn:aws:iam::123456789012:role/
aws-service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable"
      }
    }
  }
]
}

```

Se uma tabela de destino não tiver nenhuma política de recursos anexada, uma política de recursos será criada durante o experimento e excluída automaticamente quando o experimento terminar. Caso contrário, a declaração de falha será inserida em uma política existente, sem nenhuma modificação adicional nas declarações de política existentes. A declaração de falha será removida da política no final do experimento.

As tabelas globais de destino do Amazon DynamoDB MRSC estão sujeitas a uma cota adicional. Essa cota determina que nenhuma mesa possa estar sujeita a mais de 5.040 minutos de redução ao valor recuperável em uma janela contínua de 7 dias.

Tabelas globais multirregionais eventualmente consistentes (MREC)

A declaração a seguir será anexada dinamicamente à política da tabela global MREC do DynamoDB de destino:

```
{
  "Statement": [
    {
      "Sid": "DoNotModifyFisDynamoDbPauseReplicationEXPxxxxxxxxxxxxxxxx",
      "Effect": "Deny",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb>DeleteItem",
        "dynamodb:DescribeTable",
        "dynamodb:UpdateTable",
        "dynamodb:Scan",
        "dynamodb:DescribeTimeToLive",
        "dynamodb:UpdateTimeToLive"
      ],
      "Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/ExampleGlobalTable",
      "Condition": {
        "DateLessThan": {
          "aws:CurrentTime": "2024-04-10T09:51:41.511Z"
        },
        "ArnEquals": {
          "aws:PrincipalArn": "arn:aws:iam::123456789012:role/aws-service-role/replication.dynamodb.amazonaws.com/AWSServiceRoleForDynamoDBReplication"
        }
      }
    }
  ]
}
```

A declaração a seguir será anexada dinamicamente à política de stream da tabela global MREC do DynamoDB de destino:

```
{
  "Statement": [
    {
      "Sid": "DoNotModifyFisDynamoDbPauseReplicationEXPxxxxxxxxxxxxxxxx",
      "Effect": "Deny",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "dynamodb:GetRecords",
        "dynamodb:DescribeStream",
        "dynamodb:GetShardIterator"
      ],
      "Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/ExampleGlobalTable/stream/2023-08-31T09:50:24.025",
      "Condition": {
        "DateLessThan": {
          "aws:CurrentTime": "2024-04-10T09:51:41.511Z"
        },
        "ArnEquals": {
          "aws:PrincipalArn": "arn:aws:iam::123456789012:role/aws-service-role/replication.dynamodb.amazonaws.com/AWSServiceRoleForDynamoDBReplication"
        }
      }
    }
  ]
}
```

Se um fluxo ou uma tabela de destino não tiverem nenhuma política de recursos anexada, uma política de recursos será criada durante o experimento e excluída automaticamente quando o experimento terminar. Caso contrário, a declaração de falha será inserida em uma política existente, sem nenhuma modificação adicional nas declarações de política existentes. A declaração de falha será removida da política no final do experimento.

Tipo de atributo

- aws:dynamodb:global-table

Parâmetros

- **duration**— Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT1 M representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.

Permissões

- `dynamodb:PutResourcePolicy`
- `dynamodb>DeleteResourcePolicy`
- `dynamodb:GetResourcePolicy`
- `dynamodb:DescribeTable`
- `tag:GetResources`
- `dynamodb:InjectError` *

* A permissão só é necessária se você estiver visando tabelas globais do MRSC

Ações do Amazon Aurora DSQL

AWS FIS suporta as seguintes ações do Amazon Aurora DSQL.

`aws:dsql:cluster-connection-failure`

Cria falhas de conexão controladas em um cluster Aurora DSQL por um período especificado para testar a resiliência do aplicativo.

Tipo de atributo

- `aws:dsql:cluster`

Parâmetros

- **duration** – A duração, de um minuto a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT1 M representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.
- **percentage** – A porcentagem (1-100) de chamadas para injetar a falha.

Permissões

- `dsql:InjectError`
- `dsql:GetCluster`
- `tag:GetResources`

Para iniciar o experimento com o Aurora DSQL, [consulte Teste de injeção de falhas](#) no Guia do usuário do Aurora DSQL.

Ações do Amazon EBS

AWS FIS suporta a seguinte ação do Amazon EBS.

Ações

- [aws:ebs:pause-volume-io](#)
- [aws:ebs:volume-io-latency](#)

aws:ebs:pause-volume-io

Pausa I/O as operações nos volumes de destino do EBS. Os volumes de destino devem estar na mesma zona de disponibilidade e anexados a instâncias criadas no Nitro System. Os volumes não podem ser anexados a instâncias em um Outpost.

Para iniciar o experimento usando o console do Amazon EC2, consulte [Teste de falhas no Amazon EBS](#) no Guia do usuário do Amazon EC2.

Tipo de atributo

- `aws:ec2:ebs-volume`

Parâmetros

- `duration` – A duração, de um segundo a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT1 M representa um minuto, PT5 S representa cinco segundos e PT6 H representa seis horas. No AWS FIS console, você insere o número de segundos, minutos ou horas. Se a duração for pequena, como PT5 S, ela I/O será pausada pela duração

especificada, mas pode levar mais tempo para que o experimento seja concluído devido ao tempo necessário para inicializá-lo.

Permissões

- `ec2:DescribeVolumes`
- `ec2:PauseVolumeIO`
- `tag:GetResources`

aws:ebs:volume-io-latency

Injeta latência nas I/O operações dos volumes de destino do EBS. Os volumes de destino devem estar na mesma zona de disponibilidade e devem estar vinculados às [instâncias baseadas em Nitro](#). Os volumes não podem ser anexados a instâncias em um Posto Avançado.

Para iniciar o experimento usando o console do Amazon EC2, [consulte Teste de falhas no Amazon EBS no Guia do usuário do Amazon EBS](#).

Tipo de atributo

- `aws:ec2:ebs-volume`

Parâmetros

- `readIOPercentage`— A porcentagem de I/O operações de leitura nas quais a latência será injetada, de 0,1% para 100,%. Essa é a porcentagem de todas as I/O operações de leitura no volume que serão afetadas durante o experimento. O padrão é 100.
- `readIOLatencyMilliseconds`— A quantidade de latência injetada nas I/O operações de leitura em milissegundos, de 1 ms (volumes io2) ou 10 ms (volumes não io2) a 60 segundos. Esse é o valor de latência que será observado na porcentagem especificada da leitura I/O durante o experimento. O padrão é 100.
- `writelIOPercentage`— A porcentagem de I/O operações de gravação nas quais a latência será injetada, de 0,1% a 100,%. Essa é a porcentagem de todas as I/O operações de gravação no volume que serão afetadas durante o experimento. O padrão é 100.
- `writelIOLatencyMilliseconds`— A quantidade de latência injetada nas I/O operações de gravação em milissegundos, de 1 ms (volumes io2) ou 10 ms (volumes não io2) a 60 segundos. Esse é

o valor de latência que será observado na porcentagem especificada da leitura I/O durante o experimento. O padrão é 100.

- `duration`— A duração pela qual a latência será injetada, de 1 segundo a 12 horas.

Permissões

- `ec2:DescribeVolumes`
- `ec2:InjectVolumeIOLatency`
- `tag:GetResources`

Ações do Amazon EC2

AWS FIS suporta as seguintes ações do Amazon EC2.

Ações

- [aws:ec2:api-insufficient-instance-capacity-error](#)
- [aws:ec2:asg-insufficient-instance-capacity-error](#)
- [aws:ec2:reboot-instances](#)
- [aws:ec2:send-spot-instance-interruptions](#)
- [aws:ec2:stop-instances](#)
- [aws:ec2:terminate-instances](#)

AWS FIS também suporta ações de injeção de falhas por meio do Agente AWS Systems Manager SSM. O Systems Manager utiliza um documento do SSM que define as ações a serem realizadas nas instâncias do EC2. Você pode usar seu próprio documento para injetar falhas personalizadas ou usar documentos do SSM pré-configurados. Para obter mais informações, consulte [the section called “Ações de documentos do SSM”](#).

`aws:ec2:api-insufficient-instance-capacity-error`

Injeta respostas de erro `InsufficientInstanceCapacity` nas solicitações feitas pelos perfis do IAM de destino. As operações suportadas são `RunInstances`, `CreateCapacityReservation`, `StartInstances`, `CreateFleet` chamadas. Solicitações que incluem pedidos de capacidade em várias zonas de disponibilidade não são compatíveis. Essa ação não é compatível com a definição de alvos usando tags de recursos, filtros ou parâmetros.

Para a LaunchInstances operação de Auto Scaling, os `InsufficientInstanceCapacity` erros serão retornados no `errors` campo da resposta, mas a capacidade desejada pelo grupo de Auto Scaling ainda será atualizada, permitindo que o processo de escalabilidade assíncrona possa iniciar instâncias. Para testes mais amplos de gerenciamento de capacidade insuficiente com LaunchInstances, considere usar essa ação junto com [the section called “aws:ec2:asg-insufficient-instance-capacity-error”](#).

Tipo de atributo

- `aws:iam:role`

Parâmetros

- `duration`— Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, `PT1 M` representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.
- `availabilityZonesIdentifiers`: a lista separada por vírgulas das zonas de disponibilidade. Suporta zonas IDs (por exemplo `"use1-az1, use1-az2"`) e nomes de zonas (por exemplo `"us-east-1a"`).
- `percentage` – A porcentagem (1-100) de chamadas para injetar a falha.

Permissões

- `ec2:InjectApiError` com o valor `ec2:FisActionId` da chave de condição definido como `aws:ec2:api-insufficient-instance-capacity-error` e a chave de condição `ec2:FisTargetArns` definida para os perfis do IAM de destino.

Para visualizar um exemplo de política, consulte [Exemplo: use chaves de condição para ec2:InjectApiError](#).

`aws:ec2:asg-insufficient-instance-capacity-error`

Injeta respostas de erro `InsufficientInstanceCapacity` nas solicitações feitas pelos grupos do Auto Scaling de destino. Essa ação só é compatível com grupos do Auto Scaling que usam modelos de execução. Para saber mais sobre erros de capacidade de instância insuficiente, consulte o [Guia do usuário do Amazon EC2](#).

Tipo de atributo

- `aws:ec2:autoscaling-group`

Parâmetros

- `duration`— Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, `PT1 M` representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.
- `availabilityZoneIdentifiers`: a lista separada por vírgulas das zonas de disponibilidade. Suporta zonas IDs (por exemplo `"use1-az1, use1-az2"`) e nomes de zonas (por exemplo `"us-east-1a"`).
- `percentage`: opcional. A porcentagem (1-100) das solicitações de inicialização do grupo do Auto Scaling de destino para injetar a falha. O padrão é 100.

Permissões

- `ec2:InjectApiError` com chave de condição `ec2:FisActionId` valor definido como `aws:ec2:asg-insufficient-instance-capacity-error` e chave de `ec2:FisTargetArns` condição definida para grupos de destino do Auto Scaling.
- `autoscaling:DescribeAutoScalingGroups`

Para visualizar um exemplo de política, consulte [Exemplo: use chaves de condição para `ec2:InjectApiError`](#).

`aws:ec2:reboot-instances`

Executa a ação da API do Amazon EC2 [RebootInstances](#) nas instâncias EC2 de destino.

Tipo de atributo

- `aws:ec2:instance`

Parâmetros

- Nenhum

Permissões

- `ec2:RebootInstances`
- `ec2:DescribeInstances`

AWS política gerenciada

- [AWSFaultInjectionSimulatorEC2Acesso](#)

`aws:ec2:send-spot-instance-interruptions`

Interrompe as instâncias spot de destino. Envia um [aviso de interrupção da Instância spot](#) para as Instâncias spot de destino dois minutos antes de interrompê-las. O tempo de interrupção é determinado pelo `durationBeforeInterruption` parâmetro especificado. Dois minutos após o tempo de interrupção, as Instâncias spot são encerradas ou interrompidas, dependendo do comportamento de interrupção. Uma instância spot que foi interrompida pelo AWS FIS permanece parada até ser reiniciada.

Imediatamente após o início da ação, a instância de destino recebe uma [recomendação de rebalanceamento da instância do EC2](#). Se você especificou `durationBeforeInterruption`, pode haver um atraso entre a recomendação de rebalanceamento e o aviso de interrupção.

Para obter mais informações, consulte [the section called “Testar interrupções da instância spot”](#). Como alternativa, para iniciar o experimento usando o console do Amazon EC2, consulte [Iniciar uma interrupção de instância spot](#) no Guia do usuário do Amazon EC2.

Tipo de atributo

- `aws:ec2:spot-instance`

Parâmetros

- `durationBeforeInterruption` – O tempo de espera antes de interromper a instância, de 2 a 15 minutos. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT2 M representa dois minutos. No AWS FIS console, você insere o número de minutos.

Permissões

- `ec2:SendSpotInstanceInterruptions`
- `ec2:DescribeInstances`

AWS política gerenciada

- [AWSFaultInjectionSimulatorEC2Acesso](#)

`aws:ec2:stop-instances`

Executa a ação da API do Amazon EC2 [StopInstances](#) nas instâncias EC2 de destino.

Tipo de atributo

- `aws:ec2:instance`

Parâmetros

- `startInstancesAfterDuration`: opcional. O tempo de espera antes de iniciar a instância, de um minuto a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT1 M representa um minuto. No console do AWS FIS , você insere o número de segundos, minutos ou horas. Se a instância tiver um volume do EBS criptografado, você deverá conceder AWS FIS permissão à chave KMS usada para criptografar o volume ou adicionar a função experimental à política de chaves do KMS.
- `completeIfInstancesTerminated`: opcional. Se verdadeiro, e se `startInstancesAfterDuration` também for verdadeiro, essa ação não falhará quando as instâncias do EC2 de destino forem encerradas por uma solicitação separada fora do FIS e não puderem ser reiniciadas. Por exemplo, grupos do Auto Scaling podem encerrar instâncias do EC2 interrompidas sob seu controle antes que essa ação seja concluída. O padrão é falso.

Permissões

- `ec2:StopInstances`
- `ec2:StartInstances`
- `ec2:DescribeInstances`: opcional. Obrigatório com `completeIfInstancesTerminated` para validar o estado da instância no final da ação.

- `kms:CreateGrant`: opcional. Obrigatório com `startInstancesAfterDuração` para reiniciar instâncias com volumes criptografados.

AWS política gerenciada

- [AWSFaultInjectionSimulatorEC2Acesso](#)

aws:ec2:terminate-instances

Executa a ação da API do Amazon EC2 [TerminateInstances](#) nas instâncias EC2 de destino.

Tipo de atributo

- `aws:ec2:instance`

Parâmetros

- Nenhum

Permissões

- `ec2:TerminateInstances`
- `ec2:DescribeInstances`

AWS política gerenciada

- [AWSFaultInjectionSimulatorEC2Acesso](#)

Ações do Amazon ECS

AWS FIS suporta as seguintes ações do Amazon ECS.

Ações

- [aws:ecs:drain-container-instances](#)
- [aws:ecs:stop-task](#)
- [aws:ecs:task-cpu-stress](#)

- [aws:ecs:task-io-stress](#)
- [aws:ecs:task-kill-process](#)
- [aws:ecs:task-network-blackhole-port](#)
- [aws:ecs:task-network-latency](#)
- [aws:ecs:task-network-packet-loss](#)

aws:ecs:drain-container-instances

Executa a ação da API do Amazon ECS [UpdateContainerInstancesState](#) para drenar a porcentagem especificada de instâncias subjacentes do Amazon EC2 nos clusters de destino.

Tipo de atributo

- aws:ecs:cluster

Parâmetros

- drainagePercentage – A porcentagem (1-100).
- duration – A duração, de um minuto a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT1 M representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.

Permissões

- ecs:DescribeClusters
- ecs:UpdateContainerInstancesState
- ecs>ListContainerInstances
- tag:GetResources

AWS política gerenciada

- [AWSFaultInjectionSimulatorECSAccess](#)

aws:ecs:stop-task

Executa a ação da API do Amazon ECS [StopTask](#) para interromper a tarefa de destino.

Tipo de atributo

- aws:ecs:task

Parâmetros

- Nenhum

Permissões

- ecs:DescribeTasks
- ecs:ListTasks
- ecs:StopTask
- tag:GetResources

AWS política gerenciada

- [AWSFaultInjectionSimulatorECSAccess](#)

aws:ecs:task-cpu-stress

Executa o estresse da CPU nas tarefas de destino. Usa o documento [AWSFIS-RunSSM -CPU-Stress](#). As tarefas devem ser gerenciadas por AWS Systems Manager. Para obter mais informações, consulte [Ações de tarefa do ECS](#).

Tipo de atributo

- aws:ecs:task

Parâmetros

- duration – A duração do teste de estresse, no formato ISO 8601.

- `percent`: opcional. A porcentagem da carga de destino, de 0 (sem carga) a 100 (carga total). O padrão é 100.
- `workers`: opcional. O número de estressores a usar. O padrão é 0, que usa todos os estressores.
- `installDependencies`: opcional. Se esse valor for `True`, o Systems Manager instalará as dependências necessárias no contêiner auxiliar do Atendente SSM, caso elas ainda não estejam instaladas. O padrão é `True`. A dependência é `stress-ng`.

Permissões

- `ecs:DescribeTasks`
- `ssm:SendCommand`
- `ssm:ListCommands`
- `ssm:CancelCommand`

`aws:ecs:task-io-stress`

I/O Exerce estresse nas tarefas desejadas. Usa o documento [AWSFIS-RunSSM -IO-Stress](#). As tarefas devem ser gerenciadas por AWS Systems Manager. Para obter mais informações, consulte [Ações de tarefa do ECS](#).

Tipo de atributo

- `aws:ecs:task`

Parâmetros

- `duration` – A duração do teste de estresse, no formato ISO 8601.
- `percent`: opcional. A porcentagem de espaço livre no sistema de arquivos a ser usada durante o teste de estresse. O padrão é 80%.
- `workers`: opcional. O número de funcionários. Os trabalhadores realizam uma combinação de read/write operações sequenciais, aleatórias e mapeadas na memória, sincronização forçada e descarte de cache. Vários processos secundários realizam I/O operações diferentes no mesmo arquivo. O padrão é um.

- `installDependencies`: opcional. Se esse valor for `True`, o Systems Manager instalará as dependências necessárias no contêiner auxiliar do Atendente SSM, caso elas ainda não estejam instaladas. O padrão é `True`. A dependência é `stress-ng`.

Permissões

- `ecs:DescribeTasks`
- `ssm:SendCommand`
- `ssm:ListCommands`
- `ssm:CancelCommand`

aws:ecs:task-kill-process

Interrompe o processo especificado nas tarefas usando o comando `killall`. Usa o documento [AWSFIS-RunSSM -Kill-Process](#). A definição da tarefa deve ter `pidMode` definido como `task`. As tarefas devem ser gerenciadas por AWS Systems Manager. Para obter mais informações, consulte [Ações de tarefa do ECS](#).

Tipo de atributo

- `aws:ecs:task`

Parâmetros

- `processName` – O nome do processo a ser interrompido.
- `signal`: opcional. O sinal a ser enviado junto com o comando. Os valores possíveis são `SIGTERM` (que o receptor pode optar por ignorar) e `SIGKILL` (que não pode ser ignorado). O padrão é `SIGTERM`.
- `installDependencies`: opcional. Se esse valor for `True`, o Systems Manager instalará as dependências necessárias no contêiner auxiliar do Atendente SSM, caso elas ainda não estejam instaladas. O padrão é `True`. A dependência é `killall`.

Permissões

- `ecs:DescribeTasks`
- `ssm:SendCommand`

- `ssm:ListCommands`
- `ssm:CancelCommand`

`aws:ecs:task-network-blackhole-port`

Reduz o tráfego de entrada ou saída para o protocolo e a porta especificados, usando os endpoints do [Amazon ECS Fault Injection](#). Usa o documento SSM [AWSFIS-Run-Network-BlackHole-Port-ECS](#). A definição da tarefa deve ter `pidMode` definido como `task`. As tarefas devem ser gerenciadas por AWS Systems Manager. Você não pode definir `networkMode` como `bridge` na definição da tarefa. Para obter mais informações, consulte [Ações de tarefa do ECS](#).

Quando `useEcsFaultInjectionEndpoints` está definida como `false`, a falha usa a `iptables` ferramenta e usa o documento SSM [AWSFIS-Run-Network-Blackhole-Port](#).

Tipo de atributo

- `aws:ecs:task`

Parâmetros

- `duration` – A duração do teste, no formato ISO 8601.
- `port` – O número da porta.
- `trafficType` – O tipo de tráfego. Os valores possíveis são `ingress` e `egress`.
- `protocol`: opcional. O protocolo. Os valores possíveis são `tcp` e `udp`. O padrão é `tcp`.
- `installDependencies`: opcional. Se esse valor for `True`, o Systems Manager instalará as dependências necessárias no contêiner auxiliar do Atendente SSM, caso elas ainda não estejam instaladas. O padrão é `True`. As dependências são `atdcurl-minimal`, `dig` e `jq`.
- `useEcsFaultInjectionEndpoints`: opcional. Se definido como verdadeiro, a injeção de falhas do Amazon ECS APIs será usada. O padrão é falso.

Permissões

- `ecs:DescribeTasks`
- `ssm:SendCommand`
- `ssm:ListCommands`
- `ssm:CancelCommand`

aws:ecs:task-network-latency

Adiciona latência e instabilidade à interface de rede para tráfego de saída para fontes específicas, usando os endpoints do [Amazon ECS](#) Fault Injection. Use o documento SSM [AWSFIS-Run-Network-Latency-ECS](#). A definição da tarefa deve ter `pidMode` definido como `task`. As tarefas devem ser gerenciadas por AWS Systems Manager. Você não pode definir `networkMode` como `bridge` na definição da tarefa. Para obter mais informações, consulte [Ações de tarefa do ECS](#).

Quando `useEcsFaultInjectionEndpoints` está definida como `false`, a falha usa a `tc` ferramenta e usa o documento SSM [AWSFIS-Run-Network-Latency-Sources](#).

Use o `flowsPercent` parâmetro para adicionar latência em uma porcentagem das conexões. Para usar o `flowsPercent` parâmetro, a versão do agente ECS deve ser igual `1.100.0` ou superior.

Para usar nomes de AZ ou AZ IDs no `sources` parâmetro, todos os destinos da ação devem estar na mesma VPC.

Tipo de atributo

- `aws:ecs:task`

Parâmetros

- `duration` – A duração do teste, no formato ISO 8601.
- `delayMilliseconds`: opcional. O atraso, em milissegundos. O padrão é 200.
- `jitterMilliseconds`: opcional. O jitter, em milissegundos. O padrão é 10.
- `flowsPercent`: opcional. A porcentagem de fluxos de rede que serão afetados pela ação. O padrão para é de 100%.
- `sources`: opcional. As fontes, separadas por vírgulas, sem espaços. Os valores possíveis são: um IPv4 endereço, um bloco IPv4 CIDR, um nome de domínio, um nome de AZ (us-east-1a), um ID de AZ (use1-az1), ALL, e DYNAMODB S3. Se você especificar DYNAMODB ou S3, isso se aplica somente ao endpoint regional na região atual. O padrão é ALL, que corresponde a todo IPv4 o tráfego.
- `installDependencies`: opcional. Se esse valor for `True`, o Systems Manager instalará as dependências necessárias no contêiner auxiliar do Atendente SSM, caso elas ainda não estejam instaladas. O padrão é `True`. As dependências são `atdcurl-minimal`, `dig`, `jq` e `Isosf`.
- `useEcsFaultInjectionEndpoints`: opcional. Se definido como verdadeiro, a injeção de falhas do Amazon ECS APIs será usada. O padrão é falso.

Permissões

- `ecs:DescribeTasks`
- `ecs:DescribeContainerInstances`
- `ec2:DescribeInstances`
- `ec2:DescribeSubnets`
- `ssm:SendCommand`
- `ssm:ListCommands`
- `ssm:CancelCommand`

`aws:ecs:task-network-packet-loss`

Adiciona perda de pacotes à interface de rede para tráfego de saída para fontes específicas, usando os endpoints do [Amazon ECS Fault Injection](#). Use o documento SSM [AWSFIS-Run-Network-Packet-loss-ECS](#). A definição da tarefa deve ter `pidMode` definido como `task`. As tarefas devem ser gerenciadas por AWS Systems Manager. Você não pode definir `networkMode` como `bridge` na definição da tarefa. Para obter mais informações, consulte [Ações de tarefa do ECS](#).

Quando `useEcsFaultInjectionEndpoints` está definida como `false`, a falha usa a `tc` ferramenta e usa o documento SSM [AWSFIS-Run-Network-Packet-Loss-Sources](#).

Use o `flowsPercent` parâmetro para injetar perda de pacotes em uma porcentagem das conexões. Para usar o `flowsPercent` parâmetro, a versão do agente ECS deve ser igual `1.100.0` ou superior.

Para usar nomes de AZ ou AZ IDs no `sources` parâmetro, todos os destinos da ação devem estar na mesma VPC.

Tipo de atributo

- `aws:ecs:task`

Parâmetros

- `duration` – A duração do teste, no formato ISO 8601.
- `lossPercent`: opcional. A porcentagem de perda de pacotes. O padrão é 7%.

- `flowsPercent`: opcional. A porcentagem de fluxos de rede que serão afetados pela ação. O padrão para é de 100%.
- `sources`: opcional. As fontes, separadas por vírgulas, sem espaços. Os valores possíveis são: um IPv4 endereço, um bloco IPv4 CIDR, um nome de domínio, um nome de AZ (us-east-1a), um ID de AZ (use1-az1), ALL, e DYNAMODB S3. Se você especificar DYNAMODB ou S3, isso se aplica somente ao endpoint regional na região atual. O padrão é ALL, que corresponde a todo IPv4 o tráfego.
- `installDependencies`: opcional. Se esse valor for `True`, o Systems Manager instalará as dependências necessárias no contêiner auxiliar do Atendente SSM, caso elas ainda não estejam instaladas. O padrão é `True`. As dependências são `atdcurl-minimal`, `dig`, `jq` e `lsfd`.
- `useEcsFaultInjectionEndpoints`: opcional. Se definido como verdadeiro, a injeção de falhas do Amazon ECS APIs será usada. O padrão é falso.

Permissões

- `ecs:DescribeTasks`
- `ecs:DescribeContainerInstances`
- `ec2:DescribeInstances`
- `ec2:DescribeSubnets`
- `ssm:SendCommand`
- `ssm:ListCommands`
- `ssm:CancelCommand`

Ações do Amazon EKS

AWS FIS suporta as seguintes ações do Amazon EKS.

Ações

- [aws:eks:inject-kubernetes-custom-resource](#)
- [aws:eks:pod-cpu-stress](#)
- [aws:eks:pod-delete](#)
- [aws:eks:pod-io-stress](#)
- [aws:eks:pod-memory-stress](#)
- [aws:eks:pod-network-blackhole-port](#)

- [aws:eks:pod-network-latency](#)
- [aws:eks:pod-network-packet-loss](#)
- [aws:eks:terminate-nodegroup-instances](#)

aws:eks:inject-kubernetes-custom-resource

Executa um experimento ChaosMesh ou Litmus em um único cluster de destino. Você deve instalar ChaosMesh o Litmus no cluster de destino.

Ao criar um modelo de experimento e definir um alvo do tipo `aws:eks:cluster`, você deve direcionar essa ação para um único nome do recurso da Amazon (ARN). Essa ação não é compatível com a definição de alvos usando tags de recursos, filtros ou parâmetros.

Ao instalar ChaosMesh, você deve especificar o tempo de execução apropriado do contêiner. A partir da versão 1.23 do Amazon EKS, o runtime padrão mudou de Docker para containerd. A partir da versão 1.24, o Docker foi removido.

Tipo de atributo

- `aws:eks:cluster`

Parâmetros

- `kubernetesApiVersion` – A versão da API do [recurso personalizado do Kubernetes](#). Os valores possíveis são `chaos-mesh.org/v1alpha1` | `litmuschaos.io/v1alpha1`.
- `kubernetesKind` – O tipo de recurso personalizado do Kubernetes. O valor depende da versão da API.
 - `chaos-mesh.org/v1alpha1` – Os valores possíveis são `AWSChaos` | `DNSChaos` | `GCPChaos` | `HTTPChaos` | `I0Chaos` | `JVMChaos` | `KernelChaos` | `NetworkChaos` | `PhysicalMachineChaos` | `PodChaos` | `PodHttpChaos` | `PodIOChaos` | `PodNetworkChaos` | `Schedule` | `StressChaos` | `TimeChaos` |
 - `litmuschaos.io/v1alpha1` – O valor possível é `ChaosEngine`.
- `kubernetesNamespace` – O [namespace do Kubernetes](#).
- `kubernetesSpec` – A seção `spec` do recurso personalizado do Kubernetes, em formato JSON.
- `maxDuration` – O tempo máximo permitido para a execução da automação ser concluído, de um minuto a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, `PT1`

M representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.

Permissões

Nenhuma permissão de AWS Identity and Access Management (IAM) é necessária para essa ação. As permissões necessárias para usar essa ação são controladas pelo Kubernetes usando a autorização RBAC. Para obter mais informações, consulte [Usar a autorização RBAC](#) na documentação oficial do Kubernetes. Para obter mais informações sobre o Chaos Mesh, consulte a [documentação oficial do Chaos Mesh](#). Para obter mais informações sobre o Litmus, consulte a [documentação oficial do Litmus](#).

aws:eks:pod-cpu-stress

Executa o estresse da CPU nos pods de destino. Para obter mais informações, consulte [Ações do EKS Pod](#).

Tipo de atributo

- aws:eks:pod

Parâmetros

- duration – A duração do teste de estresse, no formato ISO 8601.
- percent: opcional. A porcentagem da carga de destino, de 0 (sem carga) a 100 (carga total). O padrão é 100.
- workers: opcional. O número de estressores a usar. O padrão é 0, que usa todos os estressores.
- kubernetesServiceAccount – A conta de serviço do Kubernetes. Para obter informações sobre as permissões necessárias, consulte [the section called “Configuração da conta de serviço do Kubernetes”](#).
- fisPodContainerImage: opcional. A imagem do contêiner usada para criar o pod do injetor de falhas. O padrão é usar as imagens fornecidas pelo AWS FIS. Para obter mais informações, consulte [the section called “Imagens do contêiner do pod”](#).
- maxErrorsPercent: opcional. A porcentagem de destinos que podem falhar antes que a injeção de falhas falhe. O padrão é 0.
- fisPodLabels: opcional. Os rótulos do Kubernetes anexados ao pod de orquestração de falhas criado pelo FIS.

- `fisPodAnnotations`: opcional. As anotações do Kubernetes anexadas ao pod de orquestração de falhas criado pelo FIS.
- `fisPodSecurityPolicy`: opcional. A política de [padrões de segurança do Kubernetes](#) a ser usada no pod de orquestração de falhas criado pelo FIS e nos contêineres temporários. Os possíveis valores são `privileged`, `baseline` e `restricted`. Essa ação é compatível com todos os níveis de política.

Permissões

- `eks:DescribeCluster`
- `ec2:DescribeSubnets`
- `tag:GetResources`

AWS política gerenciada

- [AWSFaultInjectionSimulatorEKSAccess](#)

`aws:eks:pod-delete`

Exclui os pods de destino. Para obter mais informações, consulte [Ações do EKS Pod](#).

Tipo de atributo

- `aws:eks:pod`

Parâmetros

- `gracePeriodSeconds`: opcional. A duração, em segundos, para esperar que o pod termine dentro do prazo. Se o valor for 0, executaremos a ação imediatamente. Se o valor for nulo, usaremos o período de carência padrão para o pod.
- `kubernetesServiceAccount` – A conta de serviço do Kubernetes. Para obter informações sobre as permissões necessárias, consulte [the section called “Configuração da conta de serviço do Kubernetes”](#).
- `fisPodContainerImage`: opcional. A imagem do contêiner usada para criar o pod do injetor de falhas. O padrão é usar as imagens fornecidas pelo AWS FIS. Para obter mais informações, consulte [the section called “Imagens do contêiner do pod”](#).

- `maxErrorsPercent`: opcional. A porcentagem de destinos que podem falhar antes que a injeção de falhas falhe. O padrão é 0.
- `fisPodLabels`: opcional. Os rótulos do Kubernetes anexados ao pod de orquestração de falhas criado pelo FIS.
- `fisPodAnnotations`: opcional. As anotações do Kubernetes anexadas ao pod de orquestração de falhas criado pelo FIS.
- `fisPodSecurityPolicy`: opcional. A política de [padrões de segurança do Kubernetes](#) a ser usada no pod de orquestração de falhas criado pelo FIS e nos contêineres temporários. Os possíveis valores são `privileged`, `baseline` e `restricted`. Essa ação é compatível com todos os níveis de política.

Permissões

- `eks:DescribeCluster`
- `ec2:DescribeSubnets`
- `tag:GetResources`

AWS política gerenciada

- [AWSFaultInjectionSimulatorEKSAccess](#)

`aws:eks:pod-io-stress`

Exerce I/O estresse nas cápsulas alvo. Para obter mais informações, consulte [Ações do EKS Pod](#).

Tipo de atributo

- `aws:eks:pod`

Parâmetros

- `duration` – A duração do teste de estresse, no formato ISO 8601.
- `workers`: opcional. O número de funcionários. Os trabalhadores realizam uma combinação de `read/write` operações sequenciais, aleatórias e mapeadas na memória, sincronização forçada e descarte de cache. Vários processos secundários realizam I/O operações diferentes no mesmo arquivo. O padrão é um.

- `percent`: opcional. A porcentagem de espaço livre no sistema de arquivos a ser usada durante o teste de estresse. O padrão é 80%.
- `kubernetesServiceAccount` – A conta de serviço do Kubernetes. Para obter informações sobre as permissões necessárias, consulte [the section called “Configuração da conta de serviço do Kubernetes”](#).
- `fisPodContainerImage`: opcional. A imagem do contêiner usada para criar o pod do injetor de falhas. O padrão é usar as imagens fornecidas pelo AWS FIS. Para obter mais informações, consulte [the section called “Imagens do contêiner do pod”](#).
- `maxErrorsPercent`: opcional. A porcentagem de destinos que podem falhar antes que a injeção de falhas falhe. O padrão é 0.
- `fisPodLabels`: opcional. Os rótulos do Kubernetes anexados ao pod de orquestração de falhas criado pelo FIS.
- `fisPodAnnotations`: opcional. As anotações do Kubernetes anexadas ao pod de orquestração de falhas criado pelo FIS.
- `fisPodSecurityPolicy`: opcional. A política de [padrões de segurança do Kubernetes](#) a ser usada no pod de orquestração de falhas criado pelo FIS e nos contêineres temporários. Os possíveis valores são `privileged`, `baseline` e `restricted`. Essa ação é compatível com todos os níveis de política.

Permissões

- `eks:DescribeCluster`
- `ec2:DescribeSubnets`
- `tag:GetResources`

AWS política gerenciada

- [AWSFaultInjectionSimulatorEKSAccess](#)

`aws:eks:pod-memory-stress`

Exerce estresse de memória nos pods de destino. Para obter mais informações, consulte [Ações do EKS Pod](#).

Tipo de atributo

- `aws:eks:pod`

Parâmetros

- `duration` – A duração do teste de estresse, no formato ISO 8601.
- `workers`: opcional. O número de estressores a usar. O padrão é um.
- `percent`: opcional. A porcentagem de memória virtual a ser usada durante o teste de estresse. O padrão é 80%.
- `kubernetesServiceAccount` – A conta de serviço do Kubernetes. Para obter informações sobre as permissões necessárias, consulte [the section called “Configuração da conta de serviço do Kubernetes”](#).
- `fisPodContainerImage`: opcional. A imagem do contêiner usada para criar o pod do injetor de falhas. O padrão é usar as imagens fornecidas pelo AWS FIS. Para obter mais informações, consulte [the section called “Imagens do contêiner do pod”](#).
- `maxErrorsPercent`: opcional. A porcentagem de destinos que podem falhar antes que a injeção de falhas falhe. O padrão é 0.
- `fisPodLabels`: opcional. Os rótulos do Kubernetes anexados ao pod de orquestração de falhas criado pelo FIS.
- `fisPodAnnotations`: opcional. As anotações do Kubernetes anexadas ao pod de orquestração de falhas criado pelo FIS.
- `fisPodSecurityPolicy`: opcional. A política de [padrões de segurança do Kubernetes](#) a ser usada no pod de orquestração de falhas criado pelo FIS e nos contêineres temporários. Os possíveis valores são `privileged`, `baseline` e `restricted`. Essa ação é compatível com todos os níveis de política.

Permissões

- `eks:DescribeCluster`
- `ec2:DescribeSubnets`
- `tag:GetResources`

AWS política gerenciada

- [AWSFaultInjectionSimulatorEKSAccess](#)

aws:eks:pod-network-blackhole-port

Elimina o tráfego de entrada ou saída para o protocolo e a porta especificados. Compatível apenas com a política `privileged` dos [padrões de segurança do Kubernetes](#). Para obter mais informações, consulte [Ações do EKS Pod](#).

Tipo de atributo

- `aws:eks:pod`

Parâmetros

- `duration` – A duração do teste, no formato ISO 8601.
- `protocol`: o protocolo. Os valores possíveis são `tcp` e `udp`.
- `trafficType` – O tipo de tráfego. Os valores possíveis são `ingress` e `egress`.
- `port` – O número da porta.
- `kubernetesServiceAccount` – A conta de serviço do Kubernetes. Para obter informações sobre as permissões necessárias, consulte [the section called “Configuração da conta de serviço do Kubernetes”](#).
- `fisPodContainerImage`: opcional. A imagem do contêiner usada para criar o pod do injetor de falhas. O padrão é usar as imagens fornecidas pelo AWS FIS. Para obter mais informações, consulte [the section called “Imagens do contêiner do pod”](#).
- `maxErrorsPercent`: opcional. A porcentagem de destinos que podem falhar antes que a injeção de falhas falhe. O padrão é 0.
- `fisPodLabels`: opcional. Os rótulos do Kubernetes anexados ao pod de orquestração de falhas criado pelo FIS.
- `fisPodAnnotations`: opcional. As anotações do Kubernetes anexadas ao pod de orquestração de falhas criado pelo FIS.

Permissões

- `eks:DescribeCluster`

- `ec2:DescribeSubnets`
- `tag:GetResources`

AWS política gerenciada

- [AWSFaultInjectionSimulatorEKSAccess](#)

aws:eks:pod-network-latency

Adiciona latência e instabilidade à interface de rede usando a ferramenta `tc` para tráfego de ou para fontes específicas. Compatível apenas com a política `privileged` dos [padrões de segurança do Kubernetes](#). Para obter mais informações, consulte [Ações do EKS Pod](#).

Use o `flowsPercent` parâmetro para adicionar latência em uma porcentagem das conexões.

Tipo de atributo

- `aws:eks:pod`

Parâmetros

- `duration` – A duração do teste, no formato ISO 8601.
- `interface`: opcional. As interfaces de rede, separadas por vírgulas. Os valores `ALL` e `DEFAULT` são suportados. O padrão é `DEFAULT`, que terá como alvo a interface de rede primária do sistema operacional.
- `delayMilliseconds`: opcional. O atraso, em milissegundos. O padrão é 200.
- `jitterMilliseconds`: opcional. O jitter, em milissegundos. O padrão é 10.
- `flowsPercent`: opcional. A porcentagem de fluxos de rede que serão afetados pela ação. O padrão para é de 100%.
- `sources`: opcional. As fontes, separadas por vírgulas, sem espaços. Os valores possíveis são: um IPv4 endereço, um bloco IPv4 CIDR, um nome de domínio, um nome de AZ (`us-east-1a`), um ID de AZ (`use1-az1`), `ALL`, e `DYNAMODB S3`. Se você especificar `DYNAMODB` ou `S3`, isso se aplica somente ao endpoint regional na região atual. Para nomes de domínio, são feitas 10 tentativas de resolução de DNS para coletar endereços IP. Devido ao balanceamento e à rotação de carga do DNS, essa ação pode não prejudicar todos os endereços IP possíveis para os quais o domínio poderia resolver. O padrão é `ALL`, que corresponde a todo IPv4 o tráfego.

- `kubernetesServiceAccount` – A conta de serviço do Kubernetes. Para obter informações sobre as permissões necessárias, consulte [the section called “Configuração da conta de serviço do Kubernetes”](#).
- `fisPodContainerImage`: opcional. A imagem do contêiner usada para criar o pod do injetor de falhas. O padrão é usar as imagens fornecidas pelo AWS FIS. Para obter mais informações, consulte [the section called “Imagens do contêiner do pod”](#).
- `maxErrorsPercent`: opcional. A porcentagem de destinos que podem falhar antes que a injeção de falhas falhe. O padrão é 0.
- `fisPodLabels`: opcional. Os rótulos do Kubernetes anexados ao pod de orquestração de falhas criado pelo FIS.
- `fisPodAnnotations`: opcional. As anotações do Kubernetes anexadas ao pod de orquestração de falhas criado pelo FIS.

Permissões

- `eks:DescribeCluster`
- `ec2:DescribeSubnets`
- `tag:GetResources`

AWS política gerenciada

- [AWSFaultInjectionSimulatorEKSAccess](#)

`aws:eks:pod-network-packet-loss`

Adiciona perda de pacotes à interface de rede usando a ferramenta `tc`. Compatível apenas com a política `privileged` dos [padrões de segurança do Kubernetes](#). Para obter mais informações, consulte [Ações do EKS Pod](#).

Use o `flowsPercent` parâmetro para injetar perda de pacotes em uma porcentagem das conexões.

Tipo de atributo

- `aws:eks:pod`

Parâmetros

- `duration` – A duração do teste, no formato ISO 8601.
- `interface`: opcional. As interfaces de rede, separadas por vírgulas. Os valores ALL e DEFAULT são suportados. O padrão é DEFAULT, que terá como alvo a interface de rede primária do sistema operacional.
- `lossPercent`: opcional. A porcentagem de perda de pacotes. O padrão é 7%.
- `flowsPercent`: opcional. A porcentagem de fluxos de rede que serão afetados pela ação. O padrão para é de 100%.
- `sources`: opcional. As fontes, separadas por vírgulas, sem espaços. Os valores possíveis são: um IPv4 endereço, um bloco IPv4 CIDR, um nome de domínio, um nome de AZ (us-east-1a), um ID de AZ (use1-az1), ALL, e DYNAMODB S3. Se você especificar DYNAMODB ou S3, isso se aplica somente ao endpoint regional na região atual. Para nomes de domínio, são feitas 10 tentativas de resolução de DNS para coletar endereços IP. Devido ao balanceamento e à rotação de carga do DNS, essa ação pode não prejudicar todos os endereços IP possíveis para os quais o domínio poderia resolver. O padrão é ALL, que corresponde a todo IPv4 o tráfego.
- `kubernetesServiceAccount` – A conta de serviço do Kubernetes. Para obter informações sobre as permissões necessárias, consulte [the section called “Configuração da conta de serviço do Kubernetes”](#).
- `fisPodContainerImage`: opcional. A imagem do contêiner usada para criar o pod do injetor de falhas. O padrão é usar as imagens fornecidas pelo AWS FIS. Para obter mais informações, consulte [the section called “Imagens do contêiner do pod”](#).
- `maxErrorsPercent`: opcional. A porcentagem de destinos que podem falhar antes que a injeção de falhas falhe. O padrão é 0.
- `fisPodLabels`: opcional. Os rótulos do Kubernetes anexados ao pod de orquestração de falhas criado pelo FIS.
- `fisPodAnnotations`: opcional. As anotações do Kubernetes anexadas ao pod de orquestração de falhas criado pelo FIS.

Permissões

- `eks:DescribeCluster`
- `ec2:DescribeSubnets`
- `tag:GetResources`

AWS política gerenciada

- [AWSFaultInjectionSimulatorEKSAccess](#)

aws:eks:terminate-nodegroup-instances

Executa a ação da API do Amazon EC2 [TerminateInstances](#) no grupo de nós de destino. Compatível somente com grupos de nós gerenciados pelo Amazon EKS. Não há suporte para grupos de nós autogerenciados. Para obter mais informações, consulte [EKS manage compute](#).

Tipo de atributo

- aws:eks:nodegroup

Parâmetros

- instanceTerminationPercentage – A porcentagem (1-100) de instâncias a encerrar.

Permissões

- ec2:DescribeInstances
- ec2:TerminateInstances
- eks:DescribeNodegroup
- tag:GetResources

AWS política gerenciada

- [AWSFaultInjectionSimulatorEKSAccess](#)

ElastiCache Ações da Amazon

AWS FIS suporta a seguinte ElastiCache ação.

aws:elasticache:replicationgroup-interrupt-az-power

Interrompe a alimentação dos nós na zona de disponibilidade especificada para grupos de ElastiCache replicação de destino com o Multi-AZ ativado. Somente uma zona de disponibilidade por

grupo de replicação pode ser afetada por vez. Quando um nó primário é o alvo, a réplica de leitura correspondente com o menor atraso de replicação é promovida para primária. As substituições de réplicas de leitura na zona de disponibilidade especificada são bloqueadas durante essa ação, o que significa que os grupos de replicação de destino operam com capacidade reduzida. O alvo dessa ação é compatível com os mecanismos Redis e Valkey. A ação não oferece suporte à opção de implantação “sem servidor”.

Tipo de atributo

- `aws:elasticache:replicationgroup`

Parâmetros

- `duration` – A duração, de um minuto a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, `PT1 M` representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.

Permissões

- `elasticache:InterruptClusterAzPower`
- `elasticache:DescribeReplicationGroups`
- `tag:GetResources`

Note

A ação de ElastiCache interrupção de alimentação do AZ agora oferece suporte a todos os tipos de grupos de replicação, incluindo Valkey e Redis. Para melhor representar essa funcionalidade, a ação foi renomeada. Se você estiver usando atualmente `aws:elasticache:interrupt-cluster-az-power`, recomendamos que você migre para a nova ação `aws:elasticache:replicationgroup-interrupt-az-power` para aproveitar os recursos mais recentes.

Ações do Amazon Kinesis Data Streams

O Amazon Kinesis Data Streams oferece suporte às seguintes ações do Kinesis.

Ações

- [aws:kinesis:stream-provisioned-throughput-exception](#)
- [aws:kinesis:stream-expired-iterator-exception](#)

aws:kinesis:stream-provisioned-throughput-exception

Injeta respostas `ProvisionedThroughputExceededException` de erro nas solicitações dos Kinesis Data Streams de destino. As operações suportadas incluem: `GetRecords`, `GetShardIteratorPutRecord`, `PutRecords` e.

Tipo de atributo

- `aws:kinesis:stream`

Parâmetros

- `duration` — A duração, que varia de um minuto a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, `PT1 M` representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.
- `percentage` — A porcentagem (1-100) de chamadas nas quais a falha deve ser injetada.

Permissões

- `kinesis:InjectApiError`

aws:kinesis:stream-expired-iterator-exception

Injeta respostas `ExpiredIteratorException` de erro para `GetRecords` chamadas direcionadas a Kinesis Data Streams específicos.

Tipo de atributo

- `aws:kinesis:stream`

Parâmetros

- **duração** — A duração, que varia de um minuto a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT1 M representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.
- **porcentagem** — A porcentagem (1-100) de chamadas nas quais a falha deve ser injetada.

Permissões

- `kinesis:InjectApiError`

AWS Lambda ações

AWS Lambda suporta as seguintes ações do Lambda

Ações

- [aws:lambda:invocation-add-delay](#)
- [aws:lambda:invocation-error](#)
- [aws:lambda:invocation-http-integration-response](#)

aws:lambda:invocation-add-delay

Atrasa o início de uma função por um número de milissegundos especificado por você. O efeito dessa ação é semelhante ao das partidas a frio do Lambda, mas o tempo adicional é gasto como parte da duração da cobrança e é aplicado a todos os ambientes de execução, em vez de afetar apenas os novos ambientes de execução. Isso significa que você pode experimentar tanto uma partida a frio do Lambda quanto esse atraso. Ao definir um valor de latência maior do que o tempo limite configurado na função Lambda, essa ação também fornecerá acesso a um evento de tempo limite de alta fidelidade.

Tipo de atributo

- `aws:lambda:function`

Parâmetros

- **duração** — A duração da ação. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT1 M representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.
- **InvocationPercentage** — Opcional. A porcentagem (1-100) de invocações de função nas quais injetar a falha. O padrão é 100.
- **startupDelayMilliseconds**: opcional. A quantidade de tempo em milissegundos (0-900.000) de espera entre a invocação e a execução do código da função. O padrão é 1000.

Permissões

- `s3:PutObject`
- `s3>DeleteObject`
- `lambda:GetFunction`
- `tag:GetResources`

aws:lambda:invocation-error

Marca as invocações da função Lambda como falhadas. Essa ação é útil para testar mecanismos de tratamento de erros, como alarmes e configurações de repetição. Ao usar essa ação, você seleciona se deseja ou não executar o código da função antes de retornar um erro.

Tipo de atributo

- `aws:lambda:function`

Parâmetros

- **duração** — A duração da ação. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT1 M representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.
- **InvocationPercentage** — Opcional. A porcentagem (1-100) de invocações de função nas quais injetar a falha. O padrão é 100.
- **preventExecution** — Se o valor for verdadeiro, a ação retornará o erro sem executar a função.

Permissões

- `s3:PutObject`
- `s3:DeleteObject`
- `lambda:GetFunction`
- `tag:GetResources`

`aws:lambda:invocation-http-integration-response`

Modifica o comportamento da função. Você seleciona um tipo de conteúdo e um código de resposta HTTP para suportar integrações com ALB, API-GW e VPC Lattice. Para permitir o impacto seletivo das integrações upstream ou downstream, você pode escolher se deseja retornar diretamente a resposta modificada ou se deseja executar a função e substituir a resposta após a conclusão da execução da função.

Tipo de atributo

- `aws:lambda:function`

Parâmetros

- `contentTypeHeader`— Valor da string do cabeçalho do tipo de conteúdo HTTP a ser retornado da função Lambda.
- `duração` — A duração da ação. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT1 M representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.
- `InvocationPercentage` — Opcional. A porcentagem (1-100) de invocações de função nas quais injetar a falha. O padrão é 100.
- `preventExecution` — Se o valor for verdadeiro, a ação retornará a resposta sem executar a função.
- `StatusCode` — Valor do código de status HTTP (000-999) a ser retornado da função Lambda.

Permissões

- `s3:PutObject`
- `s3:DeleteObject`

- `lambda:GetFunction`
- `tag:GetResources`

Ação Amazon MemoryDB

AWS FIS suporta a seguinte ação MemoryDB.

`aws:memorydb:multi-region-cluster-pause-replication`

Pausa a replicação entre um cluster regional e todos os outros clusters regionais dentro do cluster multirregional. O cluster regional alvo é o cluster na região em que o experimento FIS está sendo executado. Enquanto a replicação está pausada, o cluster multirregional não pode ser atualizado. Depois que a ação for concluída, pode levar alguns minutos para que o cluster multirregional retorne a um estado disponível. Para saber mais sobre o Amazon MemoryDB Multi-Region, consulte o [Amazon MemoryDB Multi-Region Developer Guide](#). Para obter informações sobre a disponibilidade da região, consulte [Pré-requisitos e limitações do MemoryDB Multi-Region](#).

Tipo de atributo

- `aws:memorydb:multi-region-cluster`

Parâmetros

- `duration` – A duração, de um minuto a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT1 M representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.

Permissões

- `memorydb:DescribeMultiRegionClusters`
- `memorydb:PauseMultiRegionClusterReplication`
- `tag:GetResources`

Ações de rede

AWS FIS suporta as seguintes ações de rede.

Ações

- [aws:network:disrupt-connectivity](#)
- [aws:network:route-table-disrupt-cross-region-connectivity](#)
- [aws:network:transit-gateway-disrupt-cross-region-connectivity](#)
- [aws:network:disrupt-vpc-endpoint](#)

aws:network:disrupt-connectivity

Nega o tráfego especificado para as sub-redes de destino clonando temporariamente a lista de controle de acesso à rede original (ACL de rede) associada à sub-rede de destino. O FIS adiciona regras de negação à ACL de rede clonada, que tem uma tag `managedByFIS=true`, e a associa à sub-rede durante a ação. Ao concluir a ação, o FIS exclui a ACL de rede clonada e restaura a associação de ACL de rede original.

Tipo de atributo

- `aws:ec2:subnet`

Parâmetros

- `scope` – O tipo de tráfego a negar. Quando o escopo não é `all`, o número máximo de entradas na rede ACLs é 20. Os valores possíveis são:
 - `all` – Nega todo o tráfego que entra e sai da sub-rede. Observe que essa opção permite tráfego dentro da sub-rede, incluindo tráfego de e para interfaces de rede na sub-rede.
 - `availability-zone` – Nega tráfego intra-VPC de e para sub-redes em outras zonas de disponibilidade. O número máximo de sub-redes que podem ser definidas como destino em uma VPC é 30.
 - `dynamodb` – Nega tráfego de e para o endpoint regional do DynamoDB na região atual.
 - `prefix-list` – Nega tráfego de e para a lista de prefixos especificada.
 - `s3` – Nega tráfego de e para o endpoint regional do Amazon S3 na região atual.
 - `s3express`— Nega tráfego de e para o endpoint zonal do Amazon S3 Express One Zone na AZ das sub-redes de destino. As sub-redes de destino devem residir AZs onde o S3 Express One Zone está disponível atualmente. Para obter mais informações, consulte [Zonas e regiões de disponibilidade do S3 Express One Zone](#).
 - `vpc` – Nega todo o tráfego que entra e sai da VPC.

- `duration` – A duração, de um minuto a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, `PT1 M` representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.
- `prefixListIdentifier` – Se o escopo for `prefix-list`, este será o identificador da lista de prefixos gerenciada pelo cliente. Você pode especificar um nome, um ID ou um ARN. A lista de prefixos pode ter no máximo 10 entradas.

Permissões

- `ec2:CreateNetworkAcl` – Cria a ACL de rede com a tag `managedByFIS=true`.
- `ec2:CreateNetworkAclEntry` – A ACL de rede deve ter a tag `managedByFIS=true`.
- `ec2:CreateTags`
- `ec2>DeleteNetworkAcl` – A ACL de rede deve ter a tag `managedByFIS=true`.
- `ec2:DescribeManagedPrefixLists`
- `ec2:DescribeNetworkAcls`
- `ec2:DescribeSubnets`
- `ec2:DescribeVpcs`
- `ec2:GetManagedPrefixListEntries`
- `ec2:ReplaceNetworkAclAssociation`

AWS política gerenciada

- [AWSFaultInjectionSimulatorNetworkAccess](#)

`aws:network:route-table-disrupt-cross-region-connectivity`

Bloqueia o tráfego que se origina nas sub-redes de destino e é destinado à região especificada. Cria tabelas de rotas que incluem todas as rotas para a região isolar. Para permitir que o FIS crie essas tabelas de rotas, aumente a cota da Amazon VPC `routes per route table` para 250 (ou 350 se o `region` parâmetro for `us-east-1`) mais o número de rotas em suas tabelas de rotas existentes.

Tipo de atributo

- `aws:ec2:subnet`

Parâmetros

- `region`: o código da região a ser isolada (por exemplo, eu-west-1).
- `duration`: o tempo de duração da ação. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT1 M representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.

Permissões

- `ec2:AssociateRouteTable`
- `ec2:CreateManagedPrefixList` †
- `ec2:CreateNetworkInterface` †
- `ec2:CreateRoute` †
- `ec2:CreateRouteTable` †
- `ec2:CreateTags` †
- `ec2>DeleteManagedPrefixList` †
- `ec2>DeleteNetworkInterface` †
- `ec2>DeleteRouteTable` †
- `ec2:DescribeManagedPrefixLists`
- `ec2:DescribeNetworkInterfaces`
- `ec2:DescribeRouteTables`
- `ec2:DescribeSubnets`
- `ec2:DescribeVpcPeeringConnections`
- `ec2:DescribeVpcs`
- `ec2:DisassociateRouteTable`
- `ec2:GetManagedPrefixListEntries`
- `ec2:ModifyManagedPrefixList` †
- `ec2:ModifyVpcEndpoint`
- `ec2:ReplaceRouteTableAssociation`

† Escopo definido usando a tag `managedByFIS=true`. Você não precisa gerenciar essa tag. AWS FIS adiciona e remove essa tag durante o experimento.

AWS política gerenciada

- [AWSFaultInjectionSimulatorNetworkAccess](#)

aws:network:transit-gateway-disrupt-cross-region-connectivity

Bloqueia o tráfego do gateway de trânsito de destino emparelhando anexos destinados à região especificada.

Tipo de atributo

- aws:ec2:transit-gateway

Parâmetros

- `region`: o código da região a ser isolada (por exemplo, eu-west-1).
- `duration`: o tempo de duração da ação. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT1 M representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.

Permissões

- `ec2:AssociateTransitGatewayRouteTable`
- `ec2:DescribeTransitGatewayAttachments`
- `ec2:DescribeTransitGatewayPeeringAttachments`
- `ec2:DescribeTransitGateways`
- `ec2:DisassociateTransitGatewayRouteTable`

AWS política gerenciada

- [AWSFaultInjectionSimulatorNetworkAccess](#)

aws:network:disrupt-vpc-endpoint

Bloqueia o tráfego de entrada e saída dos endpoints VPC da interface de destino. O FIS cria um grupo de segurança gerenciado com regras vazias e substitui temporariamente os grupos de

segurança dos endpoints VPC de destino por esse grupo de segurança gerenciado. Se forem feitas modificações nos recursos de destino durante a execução da ação, a ação falhará e os recursos não serão restaurados ao estado anterior ao experimento. Além disso, se um grupo de segurança gerenciado pelo FIS for modificado durante a execução da ação, ele não será excluído pelo FIS.

Tipo de atributo

- `aws:ec2:vpc-endpoint`

Parâmetros

- `duration`: o tempo de duração da ação. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT1 M representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.

Permissões

- `ec2:DescribeVpcEndpoints`
- `ec2:DescribeSecurityGroups`
- `ec2:ModifyVpcEndpoint`
- `ec2:CreateSecurityGroup`
- `ec2>DeleteSecurityGroup`
- `ec2:RevokeSecurityGroupEgress`
- `ec2:CreateTags`
- `vpce:AllowMultiRegion *`

* A permissão só é necessária se você estiver visando endpoints VPC entre regiões

Ações do Amazon RDS

AWS FIS suporta as seguintes ações do Amazon RDS.

Ações

- [aws:rds:failover-db-cluster](#)
- [aws:rds:reboot-db-instances](#)

aws:rds:failover-db-cluster

Executa a ação de [failover](#) da API do Amazon RDS DBCluster no cluster de banco de dados Aurora de destino. Há suporte para clusters RDS e clusters DocumentDB.

Tipo de atributo

- aws:rds:cluster

Parâmetros

- Nenhum

Permissões

- rds:FailoverDBCluster
- rds:DescribeDBClusters
- tag:GetResources

AWS política gerenciada

- [AWSFaultInjectionSimulatorRDSAccess](#)

aws:rds:reboot-db-instances

Executa a ação [Reboot](#) da API do Amazon RDS DBInstance na instância de banco de dados de destino. Há suporte para clusters RDS e clusters DocumentDB.

Tipo de atributo

- aws:rds:db

Parâmetros

- forceFailover: opcional. Se o valor for verdadeiro e as instâncias forem de várias zonas de disponibilidade (Multi-AZ), força o failover para outra zona de disponibilidade. O padrão é falso.

Permissões

- `rds:RebootDBInstance`
- `rds:DescribeDBInstances`
- `tag:GetResources`

AWS política gerenciada

- [AWSFaultInjectionSimulatorRDSAccess](#)

Ações do Amazon S3

AWS FIS suporta a seguinte ação do Amazon S3.

Ações

- [aws:s3:bucket-pause-replication](#)

aws:s3:bucket-pause-replication

Pausa a replicação de buckets-alvo de origem para buckets de destino. Os buckets de destino podem estar em diferentes regiões da AWS ou na mesma região que o bucket de origem. Os objetos existentes podem continuar sendo replicados por até uma hora após o início da ação. Essa ação só é compatível com a segmentação por tags. Para saber mais sobre a replicação do Amazon S3, consulte o [Guia do usuário do Amazon S3](#).

Tipo de atributo

- `aws:s3:bucket`

Parâmetros

- `duration` – A duração, de um minuto a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, `PT1 M` representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.
- `region`: a região da AWS onde os buckets de destino estão localizados.
- `destinationBuckets`: opcional. Lista separada por vírgulas dos buckets do S3 de destino.

- **prefixes:** opcional. Lista separada por vírgulas de prefixos de chave de objeto do S3 dos filtros de regras de replicação. As regras de replicação dos buckets de destino com um filtro baseado nos prefixos serão pausadas.

Permissões

- `S3:PutReplicationConfiguration` com a chave de condição `S3:IsReplicationPauseRequest` definida como `True`
- `S3:GetReplicationConfiguration` com a chave de condição `S3:IsReplicationPauseRequest` definida como `True`
- `S3:PauseReplication`
- `S3:ListAllMyBuckets`
- `tag:GetResources`

Para visualizar um exemplo de política, consulte [Exemplo: use chaves de condição para `aws:s3:bucket-pause-replication`](#).

Ações do Systems Manager

AWS FIS suporta as seguintes ações do Systems Manager.

Ações

- [aws:ssm:send-command](#)
- [aws:ssm:start-automation-execution](#)

aws:ssm:send-command

Executa a ação da API Systems Manager [SendCommand](#) nas instâncias EC2 de destino. O documento do Systems Manager (documento do SSM) define as ações que o Systems Manager realiza nas suas instâncias. Para obter mais informações, consulte [Usar a ação `aws:ssm:send-command`](#).

Tipo de atributo

- `aws:ec2:instance`

Parâmetros

- `documentArn` – O nome do recurso da Amazon (ARN) do documento. No console, esse parâmetro será preenchido se você escolher um valor do Tipo de ação que corresponda a um dos documentos [AWS FIS SSM pré-configurados](#).
- `documentVersion`: opcional. A versão do documento. Se estiver vazia, a versão padrão será executada.
- `documentParameters` – Condicional. Os parâmetros necessários e opcionais que o documento aceita. O formato é um objeto JSON com chaves que são strings e valores que são strings ou arrays de strings.
- `duration` – A duração, de um minuto a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT1 M representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.

Permissões

- `ssm:SendCommand`
- `ssm:ListCommands`
- `ssm:CancelCommand`

AWS política gerenciada

- [AWSFaultInjectionSimulatorEC2Acesso](#)

`aws:ssm:start-automation-execution`

Executa a ação da API Systems Manager [StartAutomationExecution](#).

Tipo de atributo

- Nenhum

Parâmetros

- `documentArn` – O nome do recurso da Amazon (ARN) do documento de automação.

- `documentVersion`: opcional. A versão do documento. Se estiver vazia, a versão padrão será executada.
- `documentParameters` – Condicional. Os parâmetros necessários e opcionais que o documento aceita. O formato é um objeto JSON com chaves que são strings e valores que são strings ou arrays de strings.
- `maxDuration` – O tempo máximo permitido para a execução da automação ser concluído, de um minuto a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT1M representa um minuto. No AWS FIS console, você insere o número de segundos, minutos ou horas.

Permissões

- `ssm:GetAutomationExecution`
- `ssm:StartAutomationExecution`
- `ssm:StopAutomationExecution`
- `iam:PassRole`: opcional. Obrigatório se o documento de automação assumir uma função.

AWS política gerenciada

- [AWSFaultInjectionSimulatorSSMAccess](#)

AWS Direct Connect ações

AWS FIS suporta a seguinte AWS Direct Connect ação.

Ações

- [aws:directconnect:virtual-interface-disconnect](#)

aws:directconnect:virtual-interface-disconnect

Testa a resiliência da AWS Direct Connect conexão interrompendo temporariamente as sessões do Border Gateway Protocol (BGP) entre as redes locais e os pares associados às interfaces virtuais de destino (). VIFs Antes de iniciar o experimento, o FIS verifica se todos os VIFs alvos do experimento estão em um estado “disponível” e cada VIF tem todos os pares de BGP com estado “disponível” e status BGP “ativo”. Durante o experimento, as sessões de emparelhamento do BGP para as

interfaces virtuais de destino serão colocadas no estado inativo. Para obter informações detalhadas sobre o teste de failover do Direct Connect, consulte a [AWS Direct Connect documentação](#).

Tipo de atributo

- `aws: conexão direta: interface virtual`

Parâmetros

- `duration`— A duração, de 10 minutos a 12 horas. Na AWS FIS API, o valor é uma string no formato ISO 8601. Por exemplo, PT10 M representa dez minutos. No console do AWS FIS, você insere o número de segundos, minutos ou horas.

Permissões

- `directconnect:DescribeVirtualInterfaces`
- `directconnect:StartBgpFailoverTest`
- `directconnect:ListVirtualInterfaceTestHistory`
- `directconnect:StopBgpFailoverTest`
- `tag:GetResources`

Use documentos SSM do Systems Manager com AWS o FIS

AWS O FIS oferece suporte a tipos de falha personalizados por meio do Agente AWS Systems Manager SSM e da ação AWS FIS. [aws:ssm:send-command](#) Documentos SSM pré-configurados do Systems Manager (documentos SSM) que podem ser usados para criar ações comuns de injeção de falhas estão disponíveis como AWS documentos públicos que começam com o AWSFIS prefixo -.

O SSM Agent é um software da Amazon que pode ser instalado e configurado em instâncias do Amazon EC2, servidores locais ou máquinas virtuais (VMs). Isso permite que o Systems Manager gerencie esses recursos. O agente processa solicitações do Systems Manager e, em seguida, as executa conforme especificado na solicitação. Você pode incluir seu próprio documento do SSM para injetar falhas personalizadas ou fazer referência a um dos documentos públicos de propriedade da Amazon.

Requisitos

Para ações que exigem que o Atendente SSM execute a ação no destino, você deve garantir o seguinte:

- O agente está instalado no destino. O SSM Agent é instalado por padrão em algumas Amazon Machine Images (AMIs). Caso contrário, você pode instalar o Atendente SSM em suas instâncias. Para obter mais informações, consulte [Instalar manualmente o Atendente SSM em instâncias do EC2](#) no Guia do usuário do AWS Systems Manager .
- O Systems Manager tem permissão para executar ações em suas instâncias. Você concede acesso usando um perfil da instância do IAM. Para obter mais informações, consulte [Criar um perfil de instância do IAM para o Systems Manager](#) e [Anexar um perfil de instância do IAM a uma instância do EC2](#) no Guia do usuário do AWS Systems Manager .

Usar a ação `aws:ssm:send-command`

Um documento do SSM define as ações que o Systems Manager realiza nas suas instâncias gerenciadas. O Systems Manager inclui vários documentos pré-configurados, ou você pode criar seus próprios. Para obter mais informações sobre como criar seu próprio documento do SSM, consulte [Criar documentos do Systems Manager](#) no Guia do usuário do AWS Systems Manager . Para obter mais informações sobre documentos do SSM em geral, consulte [Documentos do AWS Systems Manager](#) no Guia do usuário do AWS Systems Manager .

AWS O FIS fornece documentos SSM pré-configurados. [Você pode visualizar os documentos SSM pré-configurados em Documentos no AWS Systems Manager console: https://console.aws.amazon.com/systems-manager/ documentos.](#) Você também pode escolher entre uma seleção de documentos pré-configurados no console AWS FIS. Para obter mais informações, consulte [Documentos AWS FIS SSM pré-configurados.](#)

Para usar um documento SSM em seus experimentos do AWS FIS, você pode usar a [aws:ssm:send-command](#) ação. Essa ação busca e executa o documento do SSM especificado em suas instâncias de destino.

Ao usar a ação `aws:ssm:send-command` em seu modelo de experimento, você deve especificar parâmetros adicionais para a ação, incluindo o seguinte:

- `documentArn` – obrigatório. O nome do recurso da Amazon (ARN) é o nome do documento do SSM.

- `documentParameters` – Condicional. Os parâmetros necessários e opcionais que o documento do SSM aceita. O formato é um objeto JSON com chaves que são strings e valores que são strings ou arrays de strings.
- `documentVersion`: opcional. A versão do documento do SSM a ser executada.

Você pode visualizar as informações de um documento do SSM (incluindo os parâmetros do documento) usando o console do Systems Manager ou a linha de comando.

Para visualizar informações sobre um documento do SSM usando o console

1. Abra o AWS Systems Manager console em <https://console.aws.amazon.com/systems-manager/>.
2. No painel de navegação, escolha Documents.
3. Selecione o documento e escolha a guia Detalhes.

Para visualizar informações sobre um documento do SSM usando a linha de comando

Use o comando [describe-document](#) do SSM.

Saiba mais sobre o estado de ação

O estado de ação do SSM é determinado pelos status dos [comandos do SSM](#).

Documentos AWS FIS SSM pré-configurados

Você pode usar documentos AWS FIS SSM pré-configurados com a `aws:ssm:send-command` ação em seus modelos de experimento.

Requisitos

- Os documentos SSM pré-configurados fornecidos pela AWS FIS são suportados somente nos seguintes sistemas operacionais:
 - Amazon Linux 2023, Amazon Linux 2
 - Ubuntu
 - RODA 8, 9
 - CentOS 9
- Os documentos SSM pré-configurados fornecidos pela AWS FIS são suportados somente em instâncias EC2. Eles não são compatíveis com outros tipos de nós gerenciados, como servidores on-premises.

Para usar esses documentos do SSM em experimentos em tarefas do ECS, use o [the section called “Ações do Amazon ECS”](#) correspondente. Por exemplo, a ação `aws:ecs:task-cpu-stress` usa o documento `AWSFIS-Run-CPU-Stress`.

Documentos

- [AWSFIS-Run-CPU-Stress](#)
- [AWSFIS-Run-Disk-Fill](#)
- [AWSFIS-Run-IO-Stress](#)
- [AWSFIS-Run-Kill-Process](#)
- [AWSFIS-Run-Memory-Stress](#)
- [AWSFIS-Run-Network-Blackhole-Port](#)
- [AWSFIS-Run-Network-Latency](#)
- [AWSFIS-Run-Network-Latency-Sources](#)
- [AWSFIS-Run-Network-Packet-Loss](#)
- [AWSFIS-Run-Network-Packet-Loss-Sources](#)

Diferença entre a duração da ação e DurationSeconds nos documentos AWS FIS SSM

Alguns documentos SSM limitam seu próprio tempo de execução, por exemplo, o `DurationSeconds` parâmetro é usado por alguns dos documentos SSM AWS FIS pré-configurados. Como resultado, você precisa especificar duas durações independentes na definição da ação do AWS FIS:

- **Action duration:** para experimentos com uma única ação, a duração da ação é equivalente à duração do experimento. Com várias ações, a duração do experimento depende das durações individuais das ações e da ordem em que elas são executadas. AWS O FIS monitora cada ação até que a duração da ação termine.
- **Parâmetro de documento DurationSeconds:** a duração, especificada em segundos, pela qual o documento do SSM será executado.

Você pode escolher valores diferentes para os dois tipos de duração:

- **Action duration exceeds DurationSeconds:** a execução do documento SSM termina antes que a ação seja concluída. AWS O FIS espera até que a duração da ação termine antes que as ações subsequentes sejam iniciadas.

- **Action duration is shorter than DurationSeconds:** o documento do SSM continua a execução após a conclusão da ação. Se a execução do documento SSM ainda estiver em andamento e a duração da ação tiver expirado, o status da ação será definido como Concluído. AWS O FIS monitora apenas a execução até que a duração da ação tenha decorrido.

Observe que alguns documentos do SSM têm durações variáveis. Por exemplo, documentos AWS FIS SSM têm a opção de instalar pré-requisitos, o que pode estender a duração geral da execução além do parâmetro especificado. **DurationSeconds** Portanto, se você definir a duração da ação **DurationSeconds** com o mesmo valor, é possível que o script SSM seja executado por mais tempo do que a duração da ação.

AWSFIS-Run-CPU-Stress

Executa o estresse da CPU em uma instância usando a ferramenta `stress-ng`. Usa o documento [AWSFIS-RunSSM -CPU-Stress](#).

Tipo de ação (somente console)

`aws:ssm:send-command/AWSFIS-Run-CPU-Stress`

ARN

`arn:aws:ssm:region::document/AWSFIS-Run-CPU-Stress`

Parâmetros do documento

- **DurationSeconds** – obrigatório. A duração do teste de estresse da CPU, em segundos.
- **CPU**: opcional. O número de estressores da CPU a usar. O padrão é 0, que usa todos os estressores da CPU.
- **LoadPercent**: opcional. A porcentagem da carga da CPU de destino, de 0 (sem carga) a 100 (carga total). O padrão é 100.
- **InstallDependencies**: opcional. Se o valor for `True`, o Systems Manager instalará as dependências necessárias nas instâncias de destino, caso elas ainda não estejam instaladas. O padrão é `True`. A dependência é `stress-ng`.

Veja a seguir um exemplo da string que você pode inserir no console.

```
{"DurationSeconds":"60", "InstallDependencies":"True"}
```

AWSFIS-Run-Disk-Fill

Aloca espaço em disco no volume raiz de uma instância para simular uma falha no disco cheio. Usa o documento [AWSFIS-RunSSM -Disk-Fill](#).

Se o experimento de injeção dessa falha for interrompido, manualmente ou por meio de uma condição de parada, o AWS FIS tentará reverter cancelando o documento SSM em execução. No entanto, se o disco estiver 100% cheio, devido à falha ou à falha mais a atividade do aplicativo, o Systems Manager talvez não consiga concluir a operação de cancelamento. Portanto, se você precisar interromper o experimento, certifique-se de que o disco não fique 100% cheio.

Tipo de ação (somente console)

`aws:ssm:send-command/AWSFIS-Run-Disk-Fill`

ARN

`arn:aws:ssm:region::document/AWSFIS-Run-Disk-Fill`

Parâmetros do documento

- `DurationSeconds` – obrigatório. A duração do teste de preenchimento de disco, em segundos.
- `Percent`: opcional. A porcentagem do disco para alocar durante o teste de preenchimento do disco. O padrão é 95%.
- `InstallDependencies`: opcional. Se o valor for `True`, o Systems Manager instalará as dependências necessárias nas instâncias de destino, caso elas ainda não estejam instaladas. O padrão é `True`. As dependências são `atd kmod e. fallocate`

Veja a seguir um exemplo da string que você pode inserir no console.

```
{"DurationSeconds":"60", "InstallDependencies":"True"}
```

AWSFIS-Run-IO-Stress

Executa o estresse de E/S em uma instância usando a ferramenta `stress-ng`. Usa o documento [AWSFIS-RunSSM -IO-Stress](#).

Tipo de ação (somente console)

`aws:ssm:send-command/AWSFIS-Run-IO-Stress`

ARN

arn:aws:ssm:region::document/AWSFIS-Run-IO-Stress

Parâmetros do documento

- **DurationSeconds** – obrigatório. A duração do teste de estresse de E/S, em segundos.
- **Workers**: opcional. O número de trabalhadores que realizam uma combinação de read/write operações sequenciais, aleatórias e mapeadas na memória, sincronização forçada e descarte de cache. Vários processos secundários realizam I/O operações diferentes no mesmo arquivo. O padrão é um.
- **Percent**: opcional. A porcentagem de espaço livre no sistema de arquivos a ser usada durante o teste de estresse de E/S. O padrão é 80%.
- **InstallDependencies**: opcional. Se o valor for `True`, o Systems Manager instalará as dependências necessárias nas instâncias de destino, caso elas ainda não estejam instaladas. O padrão é `True`. A dependência é `stress-ng`.

Veja a seguir um exemplo da string que você pode inserir no console.

```
{"Workers":"1", "Percent":"80", "DurationSeconds":"60", "InstallDependencies":"True"}
```

AWSFIS-Run-Kill-Process

Interrompe o processo especificado na instância usando o comando `killall`. Usa o documento [AWSFIS-RunSSM -Kill-Process](#).

Tipo de ação (somente console)

aws:ssm:send-command/AWSFIS-Run-Kill-Process

ARN

arn:aws:ssm:region::document/AWSFIS-Run-Kill-Process

Parâmetros do documento

- **ProcessName** – obrigatório. O nome do processo a interromper.
- **Signal**: opcional. O sinal a ser enviado junto com o comando. Os valores possíveis são `SIGTERM` (que o receptor pode optar por ignorar) e `SIGKILL` (que não pode ser ignorado). O padrão é `SIGTERM`.

- `InstallDependencies`: opcional. Se o valor for `True`, o Systems Manager instalará as dependências necessárias nas instâncias de destino, caso elas ainda não estejam instaladas. O padrão é `True`. A dependência é `killall`.

Veja a seguir um exemplo da string que você pode inserir no console.

```
{"ProcessName":"myapplication", "Signal":"SIGTERM"}
```

AWSFIS-Run-Memory-Stress

Executa o estresse de memória em uma instância usando a ferramenta `stress-ng`. Usa o documento [AWSFIS-RunSSM -Memory-Stress](#).

Tipo de ação (somente console)

`aws:ssm:send-command/AWSFIS-Run-Memory-Stress`

ARN

`arn:aws:ssm:region::document/AWSFIS-Run-Memory-Stress`

Parâmetros do documento

- `DurationSeconds` – obrigatório. A duração do teste de estresse de memória, em segundos.
- `Workers`: opcional. O número de estressores da memória virtual. O padrão é um.
- `Percent` – obrigatório. A porcentagem de memória virtual a ser usada durante o teste de estresse de memória.
- `InstallDependencies`: opcional. Se o valor for `True`, o Systems Manager instalará as dependências necessárias nas instâncias de destino, caso elas ainda não estejam instaladas. O padrão é `True`. A dependência é `stress-ng`.

Veja a seguir um exemplo da string que você pode inserir no console.

```
{"Percent":"80", "DurationSeconds":"60", "InstallDependencies":"True"}
```

AWSFIS-Run-Network-Blackhole-Port

Reduz o tráfego de entrada ou saída do protocolo e da porta usando a ferramenta `iptables`. Usa o documento SSM [AWSFIS-Run-Network-Blackhole-Port](#).

Tipo de ação (somente console)

aws:ssm:send-command/AWSFIS-Run-Network-Blackhole-Port

ARN

arn:aws:ssm:region::document/AWSFIS-Run-Network-Blackhole-Port

Parâmetros do documento

- Protocol – obrigatório. O protocolo. Os valores possíveis são tcp e udp.
- Port – obrigatório. O número da porta.
- TrafficType: opcional. O tipo de tráfego. Os valores possíveis são ingress e egress. O padrão é ingress.
- DurationSeconds – obrigatório. A duração do teste de buraco negro da rede, em segundos.
- InstallDependencies: opcional. Se o valor for True, o Systems Manager instalará as dependências necessárias nas instâncias de destino, caso elas ainda não estejam instaladas. O padrão é True. As dependências são atd, dig, lsof e iptables.

Veja a seguir um exemplo da string que você pode inserir no console.

```
{"Protocol":"tcp", "Port":"8080", "TrafficType":"egress", "DurationSeconds":"60", "InstallDependencies":"True"}
```

AWSFIS-Run-Network-Latency

Adiciona latência à interface de rede usando a ferramenta tc. Usa o documento [AWSFIS-RunSSM - Network-Latency](#).

Tipo de ação (somente console)

aws:ssm:send-command/AWSFIS-Run-Network-Latency

ARN

arn:aws:ssm:region::document/AWSFIS-Run-Network-Latency

Parâmetros do documento

- Interface: opcional. A interface de rede. O padrão é eth0.

- `DelayMilliseconds`: opcional. O atraso, em milissegundos. O padrão é 200.
- `DurationSeconds` – obrigatório. A duração do teste de latência da rede, em segundos.
- `InstallDependencies`: opcional. Se o valor for `True`, o Systems Manager instalará as dependências necessárias nas instâncias de destino, caso elas ainda não estejam instaladas. O padrão é `True`. As dependências são `atd`, `dig` e `tc`.

Veja a seguir um exemplo da string que você pode inserir no console.

```
{"DelayMilliseconds":"200", "Interface":"eth0", "DurationSeconds":"60",  
"InstallDependencies":"True"}
```

AWSFIS-Run-Network-Latency-Sources

Adiciona latência e instabilidade à interface de rede usando a ferramenta `tc` para tráfego de ou para fontes específicas. Usa o documento SSM [AWSFIS-Run-Network-Latency-Sources](#).

Use o `FlowsPercent` parâmetro para adicionar latência em uma porcentagem das conexões.

Tipo de ação (somente console)

`aws:ssm:send-command/AWSFIS-Run-Network-Latency-Sources`

ARN

`arn:aws:ssm:region::document/AWSFIS-Run-Network-Latency-Sources`

Parâmetros do documento

- `Interface`: opcional. As interfaces de rede, separadas por vírgulas. Os valores `ALL` e `DEFAULT` são suportados. O padrão é `DEFAULT`, que terá como alvo a interface de rede primária do sistema operacional.
- `DelayMilliseconds`: opcional. O atraso, em milissegundos. O padrão é 200.
- `JitterMilliseconds`: opcional. O jitter, em milissegundos. O padrão é 10.
- `FlowsPercent`: opcional. A porcentagem de fluxos de rede que serão afetados pela ação. O padrão para é de 100%.
- `Sources` – obrigatório. As fontes, separadas por vírgulas, sem espaços. Os valores possíveis são: um IPv4 endereço, um bloco IPv4 CIDR, um nome de domínio, um nome de AZ (`us-east-1a`), um ID de AZ (`use1-az1`), `ALL`, e `DYNAMODB S3`. Se você especificar `DYNAMODB` ou `S3`, isso se aplica somente ao endpoint regional na região atual.

- **TrafficType**: opcional. O tipo de tráfego. Os valores possíveis são `ingress` e `egress`. O padrão é `ingress`.
- **DurationSeconds** – obrigatório. A duração do teste de latência da rede, em segundos.
- **InstallDependencies**: opcional. Se o valor for `True`, o Systems Manager instalará as dependências necessárias nas instâncias de destino, caso elas ainda não estejam instaladas. O padrão é `True`. As dependências são `atddig`, `jq`, `lsuf`, e `tc`.

Ao usar este documento, a função do experimento exige as seguintes permissões:

- `ec2:DescribeInstances`
- `ec2:DescribeSubnets`

Veja a seguir um exemplo da string que você pode inserir no console.

```
{"DelayMilliseconds":"200", "JitterMilliseconds":"15",  
  "Sources":"S3,www.example.com,72.21.198.67", "Interface":"eth0",  
  "TrafficType":"egress", "DurationSeconds":"60", "InstallDependencies":"True"}
```

AWSFIS-Run-Network-Packet-Loss

Adiciona perda de pacotes à interface de rede usando a ferramenta `tc`. Usa o documento SSM [AWSFIS-Run-Network-Packet-Loss](#).

Tipo de ação (somente console)

`aws:ssm:send-command/AWSFIS-Run-Network-Packet-Loss`

ARN

`arn:aws:ssm:region::document/AWSFIS-Run-Network-Packet-Loss`

Parâmetros do documento

- **Interface**: opcional. A interface de rede. O padrão é `eth0`.
- **LossPercent**: opcional. A porcentagem de perda de pacotes. O padrão é 7%.
- **DurationSeconds** – obrigatório. A duração do teste de perda de pacotes de rede, em segundos.
- **InstallDependencies**: opcional. Se o valor for `True`, o Systems Manager instalará as dependências necessárias nas instâncias de destino. O padrão é `True`. As dependências são `atd`, `lsuf`, `dig` e `tc`.

Veja a seguir um exemplo da string que você pode inserir no console.

```
{"LossPercent": "15", "Interface": "eth0", "DurationSeconds": "60",  
  "InstallDependencies": "True"}
```

AWSFIS-Run-Network-Packet-Loss-Sources

Adiciona perda de pacotes à interface de rede usando a ferramenta tc para tráfego de ou para fontes específicas. Usa o documento SSM [AWSFIS-Run-Network-Packet-Loss-Sources](#).

Use o `FlowsPercent` parâmetro para injetar perda de pacotes em uma porcentagem das conexões.

Tipo de ação (somente console)

`aws:ssm:send-command/AWSFIS-Run-Network-Packet-Loss-Sources`

ARN

`arn:aws:ssm:region::document/AWSFIS-Run-Network-Packet-Loss-Sources`

Parâmetros do documento

- `Interface`: opcional. As interfaces de rede, separadas por vírgulas. Os valores `ALL` e `DEFAULT` são suportados. O padrão é `DEFAULT`, que terá como alvo a interface de rede primária do sistema operacional.
- `LossPercent`: opcional. A porcentagem de perda de pacotes. O padrão é 7%.
- `FlowsPercent`: opcional. A porcentagem de fluxos de rede que serão afetados pela ação. O padrão para é de 100%.
- `Sources` – obrigatório. As fontes, separadas por vírgulas, sem espaços. Os valores possíveis são: um IPv4 endereço, um bloco IPv4 CIDR, um nome de domínio, um nome de AZ (`us-east-1a`), um ID de AZ (`use1-az1`), `ALL`, e `DYNAMODB S3`. Se você especificar `DYNAMODB` ou `S3`, isso se aplica somente ao endpoint regional na região atual.
- `TrafficType`: opcional. O tipo de tráfego. Os valores possíveis são `ingress` e `egress`. O padrão é `ingress`.
- `DurationSeconds` – obrigatório. A duração do teste de perda de pacotes de rede, em segundos.
- `InstallDependencies`: opcional. Se o valor for `True`, o Systems Manager instalará as dependências necessárias nas instâncias de destino. O padrão é `True`. As dependências são `atdig`, `jq`, `lsof`, e `tc`.

Ao usar este documento, a função do experimento exige as seguintes permissões:

- `ec2:DescribeInstances`
- `ec2:DescribeSubnets`

Veja a seguir um exemplo da string que você pode inserir no console.

```
{"LossPercent": "15", "Sources": "S3, www.example.com, 72.21.198.67", "Interface": "eth0", "TrafficType": "egress", "DurationSeconds": "60", "InstallDependencies": "True"}
```

Exemplos

Para obter um exemplo de modelo de experimento, consulte [the section called “Execute um documento AWS FIS SSM pré-configurado”](#).

Para obter um tutorial de exemplo, consulte [Executar o estresse da CPU em uma instância](#).

Limitações

- Os seguintes documentos não podem ser executados paralelamente:
 - `AWSFIS-Run-Network-Blackhole-Port`
 - `AWSFIS-Run-Network-Latency`
 - `AWSFIS-Run-Network-Latency-Sources`
 - `AWSFIS-Run-Network-Packet-Loss`
 - `AWSFIS-Run-Network-Packet-Loss-Sources`

Scripts de reversão

AWS Os documentos do FIS SSM criam automaticamente scripts de reversão como um mecanismo de segurança para restaurar o estado do sistema após experimentos de injeção de falhas. Esses scripts garantem que as falhas injetadas sejam removidas, mesmo que a ação falhe ou seja encerrada inesperadamente.

Criação de script de reversão

Os scripts de reversão são criados automaticamente quando os experimentos de injeção de falhas começam.

Detalhes da criação

- Localização — Os scripts são criados no `/var/lib/amazon/ssm/` diretório.
- Padrão de nomenclatura — `FAULT_NAME-FAULT_IDENTIFIER-Rollback.sh` onde `FAULT_IDENTIFIER` está uma string de 32 caracteres gerada aleatoriamente
- Tempo — Criado no início de cada experimento de injeção de falhas, antes do início da injeção de falhas.
- Conteúdo — contém todas as variáveis e comandos de ambiente necessários para reverter a falha específica.

Por exemplo, um experimento de latência de rede pode criar um script de reversão em `/var/lib/amazon/ssm/NetworkLatency-abc123-Rollback.sh`

Registro de reversão

Os scripts de reversão implementam o registro duplo para capturar todas as atividades de reversão para fins de solução de problemas e auditoria.

Locais de arquivos de log

Quando um script de reversão é executado, ele cria registros em dois locais:

- Arquivos temporários — `/tmp/aws-fis-rollback-TIMESTAMP-PID.log`
- Registros do sistema — Enviados para o syslog com facilidade `local0.info`

Nomeação do arquivo de log

Os arquivos de log temporários usam a seguinte convenção de nomenclatura:

```
/tmp/aws-fis-rollback-YYYY-MM-DDTHH:MM:SSZ-PID.log
```

Onde `YYYY-MM-DDTHH:MM:SSZ` está o timestamp UTC e `PID` o ID do processo do script de reversão.

Configuração do syslog

Os registros de reversão são enviados ao syslog com a seguinte configuração:

- Etiqueta — `aws-fis-rollback`

- Prioridade: local0.info
- Formato — [YYYY-MM-DDTHH:MM:SSZ] *log_message*

Para ver os registros de reversão

Use o comando a seguir para visualizar todos os registros de reversão do diário do systemd:

```
sudo journalctl -t aws-fis-rollback
```

Solução de problemas

Use o procedimento a seguir para solucionar problemas.

Para solucionar problemas com documentos do SSM

1. Abra o AWS Systems Manager console em <https://console.aws.amazon.com/systems-manager/>.
2. No painel de navegação, em Gerenciamento de nós, Executar comando.
3. Na guia Histórico de comandos, use os filtros para localizar a execução do documento.
4. Escolha o ID do comando para abrir sua página de detalhes.
5. Escolha o ID da instância. Analise a saída e os erros de cada etapa.

Use as ações do AWS FIS `aws:ecs:task`

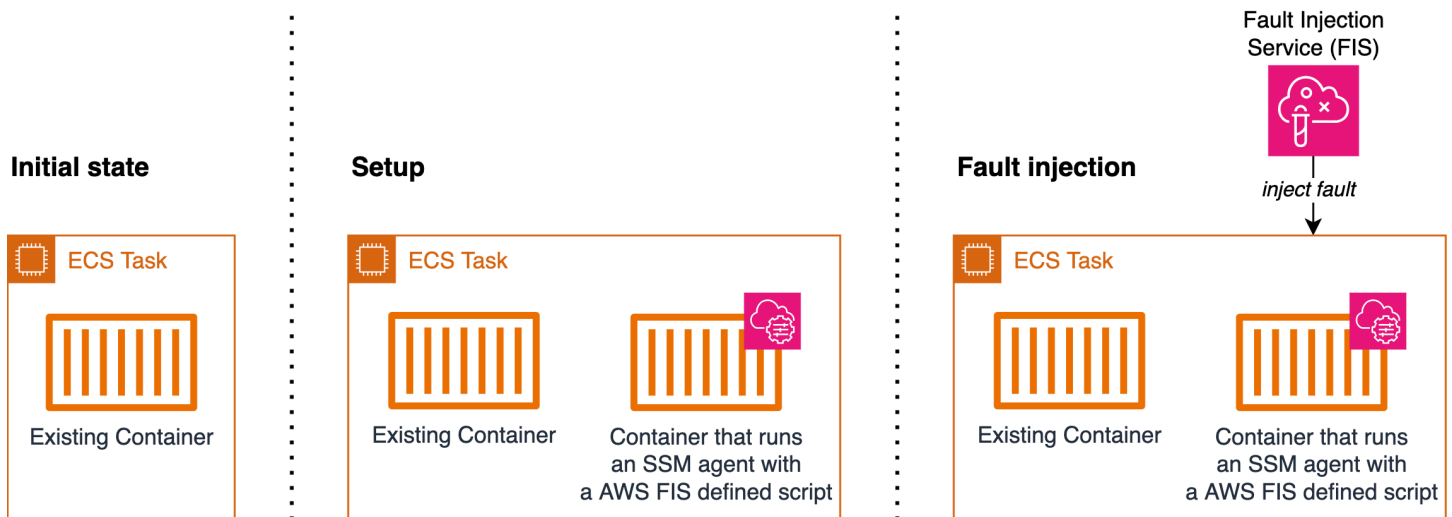
Você pode usar as ações `aws:ecs:task` para injetar falhas em suas tarefas do Amazon ECS. Os tipos de capacidade Amazon EC2 e Fargate são compatíveis.

Essas ações usam [documentos do AWS Systems Manager \(SSM\)](#) para injetar falhas. Para usar `aws:ecs:task` ações, você precisará adicionar um contêiner com um agente SSM à sua definição de tarefa do Amazon Elastic Container Service (Amazon ECS). O contêiner executa um [script definido pelo AWS FIS](#) que registra a tarefa do Amazon ECS como instância gerenciada no serviço SSM. Além disso, o script recupera os metadados da tarefa para adicionar etiquetas à instância gerenciada. A configuração permitirá que o AWS FIS resolva a tarefa alvo. Este parágrafo se refere a Configuração no diagrama abaixo.

Quando você executa a segmentação de um experimento do AWS FIS `aws:ecs:task`, o AWS FIS mapeia as tarefas alvo do Amazon ECS que você especifica em um modelo de experimento

do AWS FIS para um conjunto de instâncias gerenciadas pelo SSM usando uma tag de recurso, ECS_TASK_ARN. O valor da etiqueta é o ARN da tarefa associada do Amazon ECS na qual os documentos do SSM devem ser executados. Este parágrafo se refere a Injeção de falhas no diagrama abaixo.

O diagrama a seguir exemplifica a configuração e a injeção de falhas em uma tarefa com um contêiner existente.



Ações

- [the section called “aws:ecs:task-cpu-stress”](#)
- [the section called “aws:ecs:task-io-stress”](#)
- [the section called “aws:ecs:task-kill-process”](#)
- [the section called “aws:ecs:task-network-blackhole-port”](#)
- [the section called “aws:ecs:task-network-latency”](#)
- [the section called “aws:ecs:task-network-packet-loss”](#)

Limitações

- As ações a seguir não podem ser executadas em paralelo:
 - aws:ecs:task-network-blackhole-port
 - aws:ecs:task-network-latency
 - aws:ecs:task-network-packet-loss
- Se você habilitou o Amazon ECS Exec, deverá desativá-lo antes de poder usar essas ações.

- A execução do documento do SSM pode ter o status Cancelado mesmo que o experimento tenha o estado Concluído. Ao executar ações do Amazon ECS, a duração fornecida pelo cliente é usada tanto para a duração da ação no experimento quanto para a duração do documento do Amazon EC2 Systems Manager (SSM). Depois que a ação é iniciada, leva algum tempo para que o documento do SSM comece a ser executado. Consequentemente, quando a duração da ação especificada for atingida, o documento do SSM ainda poderá ter alguns segundos restantes para concluir sua execução. Quando a duração da ação do experimento é atingida, a ação é interrompida e a execução do documento do SSM é cancelada. A injeção de falhas foi bem-sucedida.

Requisitos

- Adicione as seguintes permissões à [função do experimento AWS FIS](#):
 - `ecs:DescribeTasks`
 - `ssm:SendCommand`
 - `ssm:ListCommands`
 - `ssm:CancelCommand`
- Adicione as permissões a seguir ao [perfil do IAM de tarefa](#) do Amazon ECS.
 - `ssm:CreateActivation`
 - `ssm:AddTagsToResource`
 - `iam:PassRole`

Observe que você pode especificar o ARN da função de instância gerenciada como recurso para `iam:PassRole`.

- Crie uma [função IAM de execução de tarefas](#) do Amazon ECS e adicione a política ECSTask ExecutionRolePolicy gerenciada [da Amazon](#).
- Na definição de tarefa, defina a variável de ambiente `MANAGED_INSTANCE_ROLE_NAME` como o nome do [perfil de instância gerenciada](#). Esse é o perfil que será anexado às tarefas registradas como instâncias gerenciadas no SSM.
- Adicione as seguintes permissões ao perfil de instância gerenciada:
 - `ssm>DeleteActivation`
 - `ssm:DeregisterManagedInstance`
- Adicione a política `SSManaged InstanceCore` gerenciada da [Amazon](#) à função de instância gerenciada.

- Adicione um contêiner do agente SSM à definição de tarefas do Amazon ECS. O script de comando registra as tarefas do Amazon ECS como instâncias gerenciadas.

```
{
  "name": "amazon-ssm-agent",
  "image": "public.ecr.aws/amazon-ssm-agent/amazon-ssm-agent:latest",
  "cpu": 0,
  "links": [],
  "portMappings": [],
  "essential": false,
  "entryPoint": [],
  "command": [
    "/bin/bash",
    "-c",
    "set -e; dnf upgrade -y; dnf install jq procps awscli -y; term_handler()
    { echo \"Deleting SSM activation $ACTIVATION_ID\"; if ! aws ssm delete-
    activation --activation-id $ACTIVATION_ID --region $ECS_TASK_REGION; then
    echo \"SSM activation $ACTIVATION_ID failed to be deleted\" 1>&2; fi;
    MANAGED_INSTANCE_ID=$(jq -e -r .ManagedInstanceID /var/lib/amazon/ssm/registration);
    echo \"Deregistering SSM Managed Instance $MANAGED_INSTANCE_ID\"; if ! aws
    ssm deregister-managed-instance --instance-id $MANAGED_INSTANCE_ID --region
    $ECS_TASK_REGION; then echo \"SSM Managed Instance $MANAGED_INSTANCE_ID
    failed to be deregistered\" 1>&2; fi; kill -SIGTERM $$SSM_AGENT_PID; }; trap
    term_handler SIGTERM SIGINT; if [[ -z $MANAGED_INSTANCE_ROLE_NAME ]]; then
    echo \"Environment variable MANAGED_INSTANCE_ROLE_NAME not set, exiting\"
    1>&2; exit 1; fi; if ! ps ax | grep amazon-ssm-agent | grep -v grep > /dev/
    null; then if [[ -n $ECS_CONTAINER_METADATA_URI_V4 ]] ; then echo \"Found ECS
    Container Metadata, running activation with metadata\"; TASK_METADATA=$(curl
    \"${ECS_CONTAINER_METADATA_URI_V4}/task\"); ECS_TASK_AVAILABILITY_ZONE=$(echo
    $TASK_METADATA | jq -e -r '.AvailabilityZone'); ECS_TASK_ARN=$(echo $TASK_METADATA
    | jq -e -r '.TaskARN'); ECS_TASK_REGION=$(echo $ECS_TASK_AVAILABILITY_ZONE | sed
    's/.$//'); ECS_TASK_AVAILABILITY_ZONE_REGEX='^(af|ap|ca|cn|eu|me|sa|us|us-gov)-
    (central|north|(north(east|west))|south|south(east|west)|east|west)-[0-9]{1}[a-z]
    {1}$'; if ! [[ $ECS_TASK_AVAILABILITY_ZONE =~ $ECS_TASK_AVAILABILITY_ZONE_REGEX ]];
    then echo \"Error extracting Availability Zone from ECS Container Metadata,
    exiting\" 1>&2; exit 1; fi; ECS_TASK_ARN_REGEX='^arn:(aws|aws-cn|aws-us-gov):ecs:
    [a-z0-9-]+:[0-9]{12}:task/[a-zA-Z0-9-]+/[a-zA-Z0-9]+$'; if ! [[ $ECS_TASK_ARN
    =~ $ECS_TASK_ARN_REGEX ]]; then echo \"Error extracting Task ARN from ECS
    Container Metadata, exiting\" 1>&2; exit 1; fi; CREATE_ACTIVATION_OUTPUT=
    $(aws ssm create-activation --iam-role $MANAGED_INSTANCE_ROLE_NAME --
    tags Key=ECS_TASK_AVAILABILITY_ZONE,Value=$ECS_TASK_AVAILABILITY_ZONE
    Key=ECS_TASK_ARN,Value=$ECS_TASK_ARN Key=FAULT_INJECTION_SIDEDECAR,Value=true --
    region $ECS_TASK_REGION); ACTIVATION_CODE=$(echo $CREATE_ACTIVATION_OUTPUT | jq
```

```

-e -r .ActivationCode); ACTIVATION_ID=$(echo $CREATE_ACTIVATION_OUTPUT | jq -e
-r .ActivationId); if ! amazon-ssm-agent -register -code $ACTIVATION_CODE -id
$ACTIVATION_ID -region $ECS_TASK_REGION; then echo \"Failed to register with AWS
Systems Manager (SSM), exiting\" 1>&2; exit 1; fi; amazon-ssm-agent & SSM_AGENT_PID=
$!; wait $$SSM_AGENT_PID; else echo \"ECS Container Metadata not found, exiting\"
1>&2; exit 1; fi; else echo \"SSM agent is already running, exiting\" 1>&2; exit 1;
fi"
],
"environment": [
  {
    "name": "MANAGED_INSTANCE_ROLE_NAME",
    "value": "SSMManagedInstanceRole"
  }
],
"environmentFiles": [],
"mountPoints": [],
"volumesFrom": [],
"secrets": [],
"dnsServers": [],
"dnsSearchDomains": [],
"extraHosts": [],
"dockerSecurityOptions": [],
"dockerLabels": {},
"ulimits": [],
"logConfiguration": {},
"systemControls": []
}

```

Para obter uma versão mais legível do script, consulte [the section called “Versão de referência do script”](#).

- Ative a injeção APIs de falhas do Amazon ECS definindo o `enableFaultInjection` campo na definição da tarefa do Amazon ECS:

```
"enableFaultInjection": true,
```

- Ao usar as `aws:ecs:task-network-packet-loss` ações `aws:ecs:task-network-blackhole-port`, `aws:ecs:task-network-latency`, ou nas tarefas do Fargate, a ação deve ter o `useEcsFaultInjectionEndpoints` parâmetro definido como `true`
- Ao usar as `aws:ecs:task-network-packet-loss` ações `aws:ecs:task-kill-process`, `aws:ecs:task-network-blackhole-port`, `aws:ecs:task-network-latency`, ou, a definição da tarefa do Amazon ECS deve ter sido `pidMode` definida como `task`

- Ao usar as `aws:ecs:task-network-packet-loss` ações `aws:ecs:task-network-blackhole-port`, `aws:ecs:task-network-latency`, ou em tarefas com o tipo de inicialização do EC2, a [opção de rede na definição da tarefa](#) deve ser definida como `awsvpc` ou `host`.

Versão de referência do script

A seguir está uma versão mais legível do script na seção Requisitos, para sua referência.

```
#!/usr/bin/env bash

# This is the activation script used to register ECS tasks as Managed Instances in SSM
# The script retrieves information from the ECS task metadata endpoint to add three
# tags to the Managed Instance
# - ECS_TASK_AVAILABILITY_ZONE: To allow customers to target Managed Instances / Tasks
# in a specific Availability Zone
# - ECS_TASK_ARN: To allow customers to target Managed Instances / Tasks by using the
# Task ARN
# - FAULT_INJECTION_SIDE CAR: To make it clear that the tasks were registered as
# managed instance for fault injection purposes. Value is always 'true'.
# The script will leave the SSM Agent running in the background
# When the container running this script receives a SIGTERM or SIGINT signal, it will
# do the following cleanup:
# - Delete SSM activation
# - Deregister SSM managed instance

set -e # stop execution instantly as a query exits while having a non-zero

dnf upgrade -y
dnf install jq procps awscli -y

term_handler() {
    echo "Deleting SSM activation $ACTIVATION_ID"
    if ! aws ssm delete-activation --activation-id $ACTIVATION_ID --region
$ECS_TASK_REGION; then
        echo "SSM activation $ACTIVATION_ID failed to be deleted" 1>&2
    fi
}

MANAGED_INSTANCE_ID=$(jq -e -r .ManagedInstanceID /var/lib/amazon/ssm/registration)
echo "Deregistering SSM Managed Instance $MANAGED_INSTANCE_ID"
if ! aws ssm deregister-managed-instance --instance-id $MANAGED_INSTANCE_ID --region
$ECS_TASK_REGION; then
```

```

    echo "SSM Managed Instance $MANAGED_INSTANCE_ID failed to be deregistered" 1>&2
fi

kill -SIGTERM $SSM_AGENT_PID
}
trap term_handler SIGTERM SIGINT

# check if the required IAM role is provided
if [[ -z $MANAGED_INSTANCE_ROLE_NAME ]] ; then
    echo "Environment variable MANAGED_INSTANCE_ROLE_NAME not set, exiting" 1>&2
    exit 1
fi

# check if the agent is already running (it will be if ECS Exec is enabled)
if ! ps ax | grep amazon-ssm-agent | grep -v grep > /dev/null; then

# check if ECS Container Metadata is available
if [[ -n $ECS_CONTAINER_METADATA_URI_V4 ]] ; then

    # Retrieve info from ECS task metadata endpoint
    echo "Found ECS Container Metadata, running activation with metadata"
    TASK_METADATA=$(curl "${ECS_CONTAINER_METADATA_URI_V4}/task")
    ECS_TASK_AVAILABILITY_ZONE=$(echo $TASK_METADATA | jq -e -r '.AvailabilityZone')
    ECS_TASK_ARN=$(echo $TASK_METADATA | jq -e -r '.TaskARN')
    ECS_TASK_REGION=$(echo $ECS_TASK_AVAILABILITY_ZONE | sed 's/.$//')

    # validate ECS_TASK_AVAILABILITY_ZONE
    ECS_TASK_AVAILABILITY_ZONE_REGEX='^(af|ap|ca|cn|eu|me|sa|us|us-gov)-(central|north|
(north(east|west))|south|south(east|west)|east|west)-[0-9]{1}[a-z]{1}$'
    if ! [[ $ECS_TASK_AVAILABILITY_ZONE =~ $ECS_TASK_AVAILABILITY_ZONE_REGEX ]] ; then
        echo "Error extracting Availability Zone from ECS Container Metadata, exiting"
1>&2
        exit 1
    fi

    # validate ECS_TASK_ARN
    ECS_TASK_ARN_REGEX='^arn:(aws|aws-cn|aws-us-gov):ecs:[a-z0-9-]+:[0-9]{12}:task/[a-
zA-Z0-9-_-]+/[a-zA-Z0-9]+$'
    if ! [[ $ECS_TASK_ARN =~ $ECS_TASK_ARN_REGEX ]] ; then
        echo "Error extracting Task ARN from ECS Container Metadata, exiting" 1>&2
        exit 1
    fi

    # Create activation tagging with Availability Zone and Task ARN

```

```

CREATE_ACTIVATION_OUTPUT=$(aws ssm create-activation \
  --iam-role $MANAGED_INSTANCE_ROLE_NAME \
  --tags Key=ECS_TASK_AVAILABILITY_ZONE,Value=$ECS_TASK_AVAILABILITY_ZONE
Key=ECS_TASK_ARN,Value=$ECS_TASK_ARN Key=FAULT_INJECTION_SIDEAR,Value=true \
  --region $ECS_TASK_REGION)

ACTIVATION_CODE=$(echo $CREATE_ACTIVATION_OUTPUT | jq -e -r .ActivationCode)
ACTIVATION_ID=$(echo $CREATE_ACTIVATION_OUTPUT | jq -e -r .ActivationId)

# Register with AWS Systems Manager (SSM)
if ! amazon-ssm-agent -register -code $ACTIVATION_CODE -id $ACTIVATION_ID -region
$ECS_TASK_REGION; then
  echo "Failed to register with AWS Systems Manager (SSM), exiting" 1>&2
  exit 1
fi

# the agent needs to run in the background, otherwise the trapped signal
# won't execute the attached function until this process finishes
amazon-ssm-agent &
SSM_AGENT_PID=$!

# need to keep the script alive, otherwise the container will terminate
wait $SSM_AGENT_PID

else
  echo "ECS Container Metadata not found, exiting" 1>&2
  exit 1
fi

else
  echo "SSM agent is already running, exiting" 1>&2
  exit 1
fi

```

Exemplo de modelo de experimento

Veja a seguir um exemplo de modelo de experimento para a ação [the section called “aws:ecs:task-cpu-stress”](#).

```

{
  "description": "Run CPU stress on the target ECS tasks",
  "targets": {
    "myTasks": {

```

```

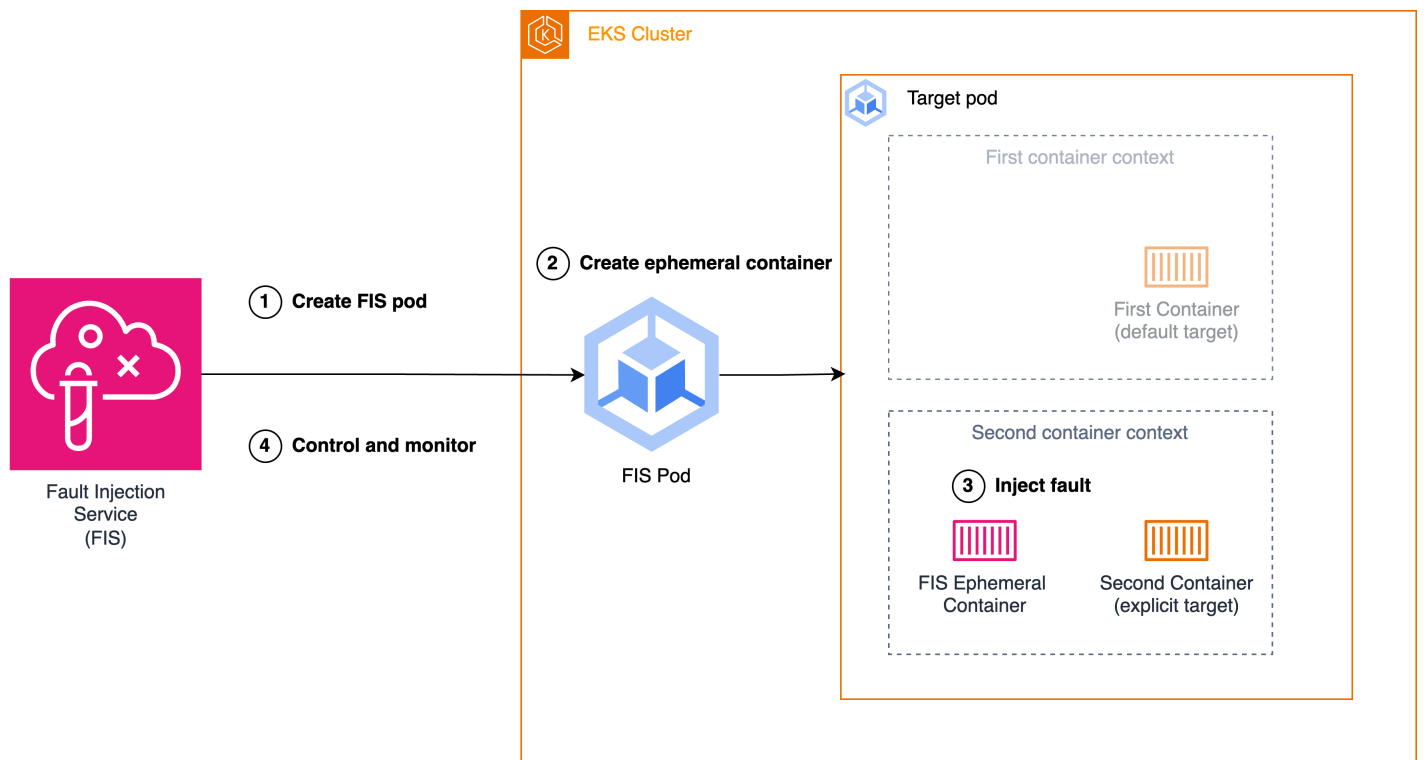
        "resourceType": "aws:ecs:task",
        "resourceArns": [
            "arn:aws:ecs:us-east-1:111122223333:task/my-
cluster/09821742c0e24250b187dfed8EXAMPLE"
        ],
        "selectionMode": "ALL"
    }
},
"actions": {
    "EcsTask-cpu-stress": {
        "actionId": "aws:ecs:task-cpu-stress",
        "parameters": {
            "duration": "PT1M"
        },
        "targets": {
            "Tasks": "myTasks"
        }
    }
},
"stopConditions": [
    {
        "source": "none",
    }
],
"roleArn": "arn:aws:iam::111122223333:role/fis-experiment-role",
"tags": {}
}

```

Use as ações do AWS FIS aws:eks:pod

Você pode usar as ações `aws:eks:pod` para injetar falhas nos pods do Kubernetes em execução nos clusters do EKS.

Quando uma ação é iniciada, o FIS recupera a [imagem do contêiner de pod do FIS](#). Essa imagem é então usada para criar um pod no cluster EKS de destino. O Pod recém-criado é responsável por injetar, controlar e monitorar a falha. Para todas as ações do FIS EKS, exceto [aws:eks:pod-delete](#), a injeção de falhas é obtida por meio do uso de [contêineres efêmeros, um recurso do Kubernetes que permite a criação de contêineres](#) temporários em um pod existente. O contêiner temporário é iniciado no mesmo namespace do contêiner de destino e executa as tarefas de injeção de falhas desejadas. Se nenhum contêiner de destino for especificado, o primeiro contêiner na especificação do pod será selecionado como destino.



1. O FIS cria o Pod FIS no cluster de destino especificado no modelo do experimento.
2. O Pod FIS cria um contêiner efêmero no Pod de destino no mesmo namespace do contêiner de destino.
3. O contêiner efêmero injeta falhas no namespace do contêiner de destino.
4. O pod do FIS controla e monitora a injeção de falhas do contêiner efêmero e o FIS controla e monitora o pod do FIS.

Após a conclusão do experimento, ou em caso de erro, o contêiner efêmero e o pod do FIS serão removidos.

Ações

- [the section called “aws:eks:pod-cpu-stress”](#)
- [the section called “aws:eks:pod-delete”](#)
- [the section called “aws:eks:pod-io-stress”](#)
- [the section called “aws:eks:pod-memory-stress”](#)
- [the section called “aws:eks:pod-network-blackhole-port”](#)

- [the section called “aws:eks:pod-network-latency”](#)
- [the section called “aws:eks:pod-network-packet-loss”](#)

Limitações

- As ações a seguir não funcionam com AWS Fargate:
 - aws:eks:pod-network-blackhole-port
 - aws:eks:pod-network-latency
 - aws:eks:pod-network-packet-loss
- As ações a seguir não são compatíveis com o [modo de rede](#) bridge:
 - aws:eks:pod-network-blackhole-port
 - aws:eks:pod-network-latency
 - aws:eks:pod-network-packet-loss
- As ações a seguir exigem permissões de root no contêiner temporário.
 - aws:eks:pod-network-blackhole-port
 - aws:eks:pod-network-latency
 - aws:eks:pod-network-packet-loss

O contêiner temporário herdará suas permissões do contexto de segurança do pod de destino. Se precisar executar os contêineres no pod como usuário não root, você pode definir contextos de segurança separados para os contêineres no pod de destino.

- Você não pode identificar alvos do tipo aws:eks:pod em seu modelo de experimento usando tags de recursos ou de recursos ARNs . Você deve identificar alvos usando os parâmetros de recursos necessários.
- As ações aws:eks: pod-network-latency e aws:eks: não pod-network-packet-loss devem ser executadas paralelamente e ter como alvo o mesmo Pod. Dependendo do valor do parâmetro `maxErrors` especificado, a ação pode terminar no estado concluído ou com falha:
 - Se `maxErrorsPercent` for 0 (padrão), a ação terminará no estado de falha.
 - Caso contrário, a falha aumentará até o orçamento `maxErrorsPercent`. Se o número de injeções com falha não atingir o `maxErrors` fornecido, a ação terminará no estado concluído.
 - Você pode identificar essas falhas nos registros do contêiner efêmero injetado no pod de destino. Ele falhará com `Exit Code: 16`.

- A ação `aws:eks: não pod-network-blackhole-port` deve ser executada paralelamente a outras ações que tenham como alvo o mesmo Pod e usem o mesmo. `trafficType` Ações paralelas usando diferentes tipos de tráfego são compatíveis.
- O FIS só pode monitorar o status da injeção de falhas quando o `securityContext` dos pods de destino está definido como. `readOnlyRootFilesystem: false` Sem essa configuração, todas as ações do EKS Pod falharão.

Requisitos

- Instale o AWS CLI no seu computador. Isso é necessário somente se você usar a AWS CLI para criar perfis do IAM. Para obter mais informações, consulte [Instalar ou atualizar a AWS CLI](#).
- Instalar o kubectl em seu computador. Isso é necessário apenas para interagir com o cluster do EKS para configurar ou monitorar o aplicativo de destino. Para obter mais informações, consulte <https://kubernetes.io/docs/tasks/tools/>.
- A versão mínima compatível do EKS é 1.23.

Criar um perfil de experimento

Para realizar um experimento, você precisa configurar um perfil do IAM para o experimento. Para obter mais informações, consulte [the section called “Perfil do experimento”](#). As permissões necessárias para esse perfil dependem da ação que você está usando. Consulte as [ações do AWS FIS que têm `aws:eks:pod` como destino](#) para encontrar as permissões necessárias para sua ação.

Configuração da conta de serviço do Kubernetes

Configure uma conta de serviço do Kubernetes para realizar experimentos com destinos no namespace especificado do Kubernetes. No exemplo a seguir, a conta de serviço é `myserviceaccount` e o namespace é. `default` Observe que esse default é um dos namespaces padrão do Kubernetes.

Para configurar sua conta de serviço do Kubernetes

1. Crie um arquivo chamado `rbac.yaml` e adicione o seguinte.

```
kind: ServiceAccount
apiVersion: v1
metadata:
```

```
namespace: default
name: myserviceaccount

---
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: role-experiments
rules:
- apiGroups: ["" ]
  resources: ["configmaps"]
  verbs: [ "get", "create", "patch", "delete" ]
- apiGroups: ["" ]
  resources: ["pods"]
  verbs: ["create", "list", "get", "delete", "deletecollection"]
- apiGroups: ["" ]
  resources: ["pods/ephemeralcontainers"]
  verbs: ["update"]
- apiGroups: ["" ]
  resources: ["pods/exec"]
  verbs: ["create"]
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["get"]

---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: bind-role-experiments
  namespace: default
subjects:
- kind: ServiceAccount
  name: myserviceaccount
  namespace: default
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: fis-experiment
roleRef:
  kind: Role
  name: role-experiments
  apiGroup: rbac.authorization.k8s.io
```

2. Execute o comando a seguir.

```
kubectl apply -f rbac.yaml
```

Conceda aos usuários e funções do IAM acesso ao Kubernetes APIs

Siga as etapas explicadas em [Associe identidades do IAM a permissões do Kubernetes](#) na documentação do EKS.

Opção 1: criar entradas de acesso

Recomendamos usar Access Entries. Você pode usar o comando a seguir para criar uma entrada de acesso que associe a função do IAM ao usuário do Kubernetes. *fis-experiment* Para obter mais informações, consulte [Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso do EKS](#).

```
aws eks create-access-entry \  
    --principal-arn arn:aws:iam::123456789012:role/fis-experiment-role \  
    --username fis-experiment \  
    --cluster-name my-cluster
```

Important

Para aproveitar as entradas de acesso, o modo de autenticação do cluster do EKS deve ser configurado para o modo API ou API_AND_CONFIG_MAP.

Opção 2: adicionar entradas ao aws-auth ConfigMap

Você também pode usar o comando a seguir para criar um mapeamento de identidades. Para obter mais informações, consulte [Gerenciar usuários e perfis do IAM](#) na documentação do eksctl.

```
eksctl create iamidentitymapping \  
    --arn arn:aws:iam::123456789012:role/fis-experiment-role \  
    --username fis-experiment \  
    --cluster my-cluster
```

⚠ Important

Aproveitar o kit de ferramentas eksctl para configurar mapeamentos de identidade resultará na criação de entradas no. `aws-auth ConfigMap` É importante observar que essas entradas geradas não oferecem suporte à inclusão de um componente de caminho. Consequentemente, o ARN fornecido como entrada não deve conter um segmento de caminho (por exemplo, `arn:aws:iam::123456789012:role/service-role/fis-experiment-role`).

Imagens do contêiner do pod

As imagens do contêiner do pod fornecidas pela AWS FIS são hospedadas no Amazon ECR. Ao fazer referência a uma imagem do Amazon ECR, você deverá usar a URI completa da imagem.

A imagem do contêiner do pod também está disponível na [Galeria Pública do AWS ECR](#).

Região da AWS	URI da imagem
Leste dos EUA (Ohio)	<code>051821878176.dkr.ecr.us-east-2.amazonaws.com/aws-fis-pod:0.1</code>
Leste dos EUA (Norte da Virgínia)	<code>731367659002.dkr.ecr.us-east-1.amazonaws.com/aws-fis-pod:0.1</code>
Oeste dos EUA (N. da Califórnia)	<code>080694859247.dkr.ecr.us-west-1.amazonaws.com/aws-fis-pod:0.1</code>
Oeste dos EUA (Oregon)	<code>864386544765.dkr.ecr.us-west-2.amazonaws.com/aws-fis-pod:0.1</code>
África (Cidade do Cabo)	<code>056821267933.dkr.ecr.af-south-1.amazonaws.com/aws-fis-pod:0.1</code>
Ásia-Pacífico (Hong Kong)	<code>246405402639.dkr.ecr.ap-east-1.amazonaws.com/aws-fis-pod:0.1</code>
Ásia-Pacífico (Mumbai)	<code>524781661239.dkr.ecr.ap-south-1.amazonaws.com/aws-fis-pod:0.1</code>

Região da AWS	URI da imagem
Ásia-Pacífico (Osaka)	148336246925.dkr.ecr.ap-northeast-3.amazonaws.com/aws-fis-pod:0.1
Ásia-Pacífico (Seul)	526524659354.dkr.ecr.ap-northeast-2.amazonaws.com/aws-fis-pod:0.1
Ásia-Pacífico (Singapura)	316401638346.dkr.ecr.ap-southeast-1.amazonaws.com/aws-fis-pod:0.1
Ásia-Pacífico (Sydney)	488104106298.dkr.ecr.ap-southeast-2.amazonaws.com/aws-fis-pod:0.1
Ásia-Pacífico (Tóquio)	635234321696.dkr.ecr.ap-northeast-1.amazonaws.com/aws-fis-pod:0.1
Canadá (Central)	490658072207.dkr.ecr.ca-central-1.amazonaws.com/aws-fis-pod:0.1
Europa (Frankfurt)	713827034473.dkr.ecr.eu-central-1.amazonaws.com/aws-fis-pod:0.1
Europa (Irlanda)	205866052826.dkr.ecr.eu-west-1.amazonaws.com/aws-fis-pod:0.1
Europa (Londres)	327424803546.dkr.ecr.eu-west-2.amazonaws.com/aws-fis-pod:0.1
Europa (Milão)	478809367036.dkr.ecr.eu-south-1.amazonaws.com/aws-fis-pod:0.1
Europe (Paris)	154605889247.dkr.ecr.eu-west-3.amazonaws.com/aws-fis-pod:0.1
Europa (Espanha)	395402409451.dkr.ecr.eu-south-2.amazonaws.com/aws-fis-pod:0.1
Europa (Estocolmo)	263175118295.dkr.ecr.eu-north-1.amazonaws.com/aws-fis-pod:0.1

Região da AWS	URI da imagem
Europa (Zurique)	604225987275.dkr.ecr.eu-central-2.amazonaws.com/ aws-fis-pod:0.1
Oriente Médio (Bahrein)	065825543785.dkr.ecr.me-south-1.amazonaws.com/ aws-fis-pod:0.1
Oriente Médio (Emirados Árabes Unidos)	438374459301.dkr.ecr.me-central-1.amazonaws.com/ aws-fis-pod:0.1
América do Sul (São Paulo)	767113787785.dkr.ecr.sa-east-1.amazonaws.com/ aws-fis-pod:0.1
AWS GovCloud (Leste dos EUA)	246533647532.dkr.ecr.us-gov-east-1.amazonaws.com/ aws-fis-pod:0.1
AWS GovCloud (Oeste dos EUA)	246529956514.dkr.ecr.us-gov-west-1.amazonaws.com/ aws-fis-pod:0.1

Exemplo de modelo de experimento

Veja a seguir um exemplo de modelo de experimento para a ação [the section called “aws:eks:pod-network-latency”](#).

```
{
  "description": "Add latency and jitter to the network interface for the target EKS
  Pods",
  "targets": {
    "myPods": {
      "resourceType": "aws:eks:pod",
      "parameters": {
        "clusterIdentifier": "mycluster",
        "namespace": "default",
        "selectorType": "labelSelector",
        "selectorValue": "mylabel=mytarget"
      },
      "selectionMode": "COUNT(3)"
    }
  }
}
```

```

},
"actions": {
  "EksPod-latency": {
    "actionId": "aws:eks:pod-network-latency",
    "description": "Add latency",
    "parameters": {
      "kubernetesServiceAccount": "myserviceaccount",
      "duration": "PT5M",
      "delayMilliseconds": "200",
      "jitterMilliseconds": "10",
      "sources": "0.0.0.0/0"
    },
    "targets": {
      "Pods": "myPods"
    }
  }
},
"stopConditions": [
  {
    "source": "none",
  }
],
"roleArn": "arn:aws:iam::111122223333:role/fis-experiment-role",
"tags": {
  "Name": "EksPodNetworkLatency"
}
}

```

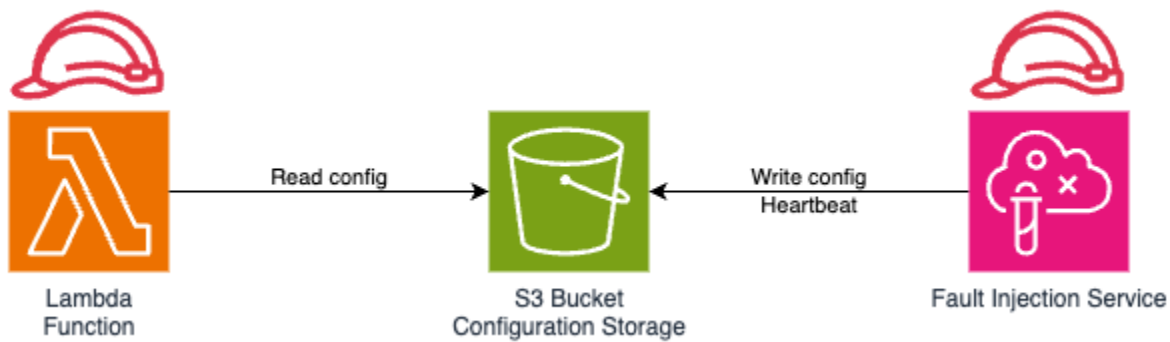
Use as AWS FIS ações `aws:lambda:function`

Você pode usar as ações `aws:lambda:function` para injetar falhas nas invocações de suas funções.

AWS Lambda

Essas ações usam uma extensão AWS FIS gerenciada para injetar falhas. Para usar as ações `aws:lambda:function`, você precisará anexar a extensão como uma camada às suas funções do Lambda e configurar um bucket do Amazon S3 para se comunicar entre e a extensão. AWS FIS

Quando você executa um AWS FIS experimento visando `aws:lambda:function`, lê a configuração do AWS FIS Amazon S3 a partir da sua função Lambda e grava as informações de injeção de falhas no local especificado do Amazon S3, conforme mostrado no diagrama abaixo.



Ações

- [the section called “aws:lambda:invocation-add-delay”](#)
- [the section called “aws:lambda:invocation-error”](#)
- [the section called “aws:lambda:invocation-http-integration-response”](#)

Limitações

- A extensão AWS FIS Lambda não pode ser usada com funções que usam streaming de resposta. Mesmo quando nenhuma falha for aplicada, a extensão AWS FIS Lambda suprimirá as configurações de streaming. Para obter mais informações, consulte [Streaming de resposta para funções Lambda](#) no guia do AWS Lambda usuário.

Pré-requisitos

Antes de usar as ações do AWS FIS Lambda, certifique-se de ter concluído essas tarefas únicas:

- Crie um bucket do Amazon S3 na região da qual você planeja iniciar um experimento - Você pode usar um único bucket do Amazon S3 para vários experimentos e compartilhar o bucket entre várias contas. AWS No entanto, você deve ter um bucket separado para cada um Região da AWS.
- Crie uma política do IAM para conceder acesso de leitura à extensão Lambda no bucket do Amazon S3 - No modelo a seguir, `my-config-distribution-bucket` substitua pelo nome do bucket do Amazon S3 que você criou acima `FisConfigs` e pelo nome de uma pasta no bucket do Amazon S3 que você deseja usar.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListingConfigLocation",
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3::my-config-distribution-bucket"],
      "Condition": {
        "StringLike": {
          "s3:prefix": ["FisConfigs/*"]
        }
      }
    },
    {
      "Sid": "AllowReadingObjectFromConfigLocation",
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": ["arn:aws:s3::my-config-distribution-bucket/FisConfigs/
*"]
    }
  ]
}

```

- Crie uma política do IAM para conceder acesso de gravação para o AWS FIS experimento no bucket do Amazon S3 - No modelo a seguir, my-config-distribution-bucket substitua pelo nome do bucket do Amazon S3 que você criou acima FisConfigs e pelo nome de uma pasta no bucket do Amazon S3 que você deseja usar.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowFisToWriteAndDeleteFaultConfigurations",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3::my-config-distribution-bucket/FisConfigs/*"
    }
  ]
}

```

```
    },
    {
      "Sid": "AllowFisToInspectLambdaFunctions",
      "Effect": "Allow",
      "Action": [
        "lambda:GetFunction"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowFisToDoTagLookups",
      "Effect": "Allow",
      "Action": [
        "tag:GetResources"
      ],
      "Resource": "*"
    }
  ]
}
```

Configurar funções do Lambda

Siga as etapas abaixo para cada função do Lambda que você deseja impactar:

1. Anexe a política de acesso de leitura do Amazon S3 criada acima à função Lambda.
2. Anexe a AWS FIS extensão como uma camada à função. Para obter mais informações sobre a camada ARNs, consulte [Versões disponíveis da AWS FIS extensão para Lambda](#).
3. Defina a `AWS_FIS_CONFIGURATION_LOCATION` variável como o ARN da pasta de configuração do Amazon S3, por exemplo. `arn:aws:s3:::my-config-distribution-bucket/FisConfigs/`
4. Defina a variável `AWS_LAMBDA_EXEC_WRAPPER` como `/opt/aws-fis/bootstrap`.

Configurar um AWS FIS experimento

Antes de executar seu experimento, certifique-se de ter anexado a política de acesso de gravação do Amazon S3 que você criou nos pré-requisitos às funções do experimento que usarão ações do Lambda. AWS FIS Para obter mais informações sobre como configurar um AWS FIS experimento, consulte [Gerenciando modelos AWS de experimentos do FIS](#).

Registro em log

A extensão AWS FIS Lambda grava registros no console e CloudWatch nos registros. O registro pode ser configurado usando a `AWS_FIS_LOG_LEVEL` variável. Os valores compatíveis são `INFO`, `WARN` e `ERROR`. Os registros serão gravados no formato de log configurado para sua função Lambda.

Veja a seguir um exemplo de um registro em formato de texto:

```
2024-08-09T18:51:38.599984Z INFO AWS FIS EXTENSION - extension enabled 1.0.1
```

Veja a seguir um exemplo de um registro no formato JSON:

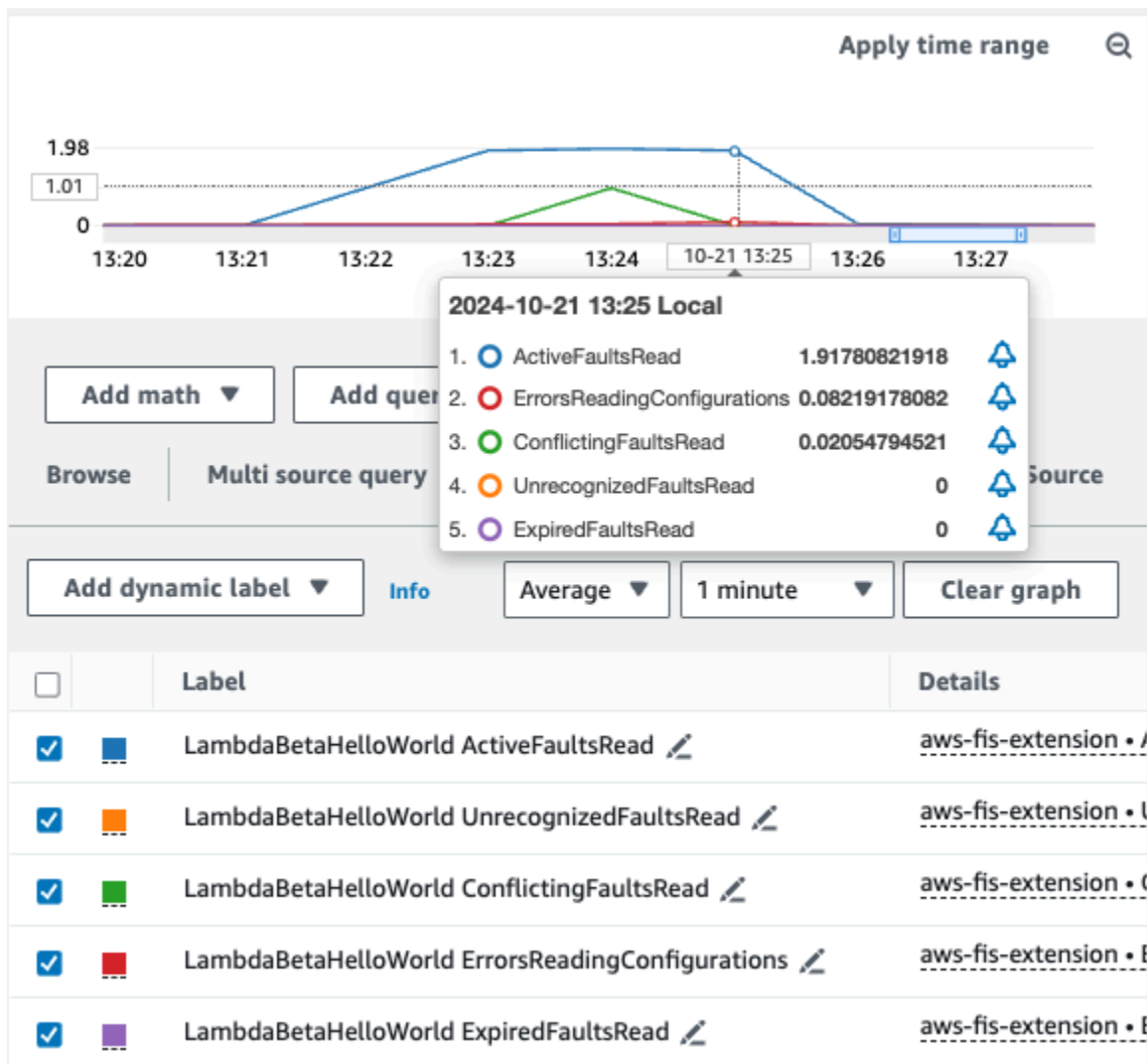
```
{
  "timestamp": "2024-10-08T17:15:36.953905Z",
  "level": "INFO",
  "fields": {
    "message": "AWS FIS EXTENSION - adding 5000 milliseconds of latency to function invocation",
    "requestId": "0608bf70-908f-4a17-bbfe-3782cd783d8b"
  }
}
```

Os registros emitidos podem ser usados com filtros CloudWatch métricos da Amazon para gerar métricas personalizadas. Para obter mais informações sobre filtros de métricas, consulte [Criação de métricas a partir de eventos de log usando filtros](#) no guia do usuário do Amazon CloudWatch Logs.

Usando o formato métrico CloudWatch incorporado (EMF)

Você pode configurar a extensão AWS FIS Lambda para emitir registros EMF definindo a variável como `AWS_FIS_EXTENSION_METRICS all`. Por padrão, a extensão não emite registros EMF, e o `AWS_FIS_EXTENSION_METRICS` padrão é `none`. Os registros do EMF são publicados `aws-fis-extension` namespace no CloudWatch console.

Dentro do `aws-fis-extension` namespace, você pode selecionar determinadas métricas para serem exibidas em um gráfico. O exemplo abaixo mostra algumas das métricas disponíveis no `aws-fis-extension` namespace.



Tópicos avançados

Esta seção fornece informações adicionais sobre como AWS FIS funciona com a extensão Lambda e casos de uso especiais.

Tópicos

- [Entendendo a pesquisa](#)
- [Entendendo a simultaneidade](#)
- [Entendendo a porcentagem de invocação](#)
- [Considerações especiais para SnapStart](#)
- [Considerações especiais para funções rápidas e pouco frequentes](#)
- [Configurando várias extensões usando o proxy da API Lambda Runtime](#)

- [Usando AWS FIS com tempos de execução de contêineres](#)
- [AWS FIS Variáveis de ambiente Lambda](#)

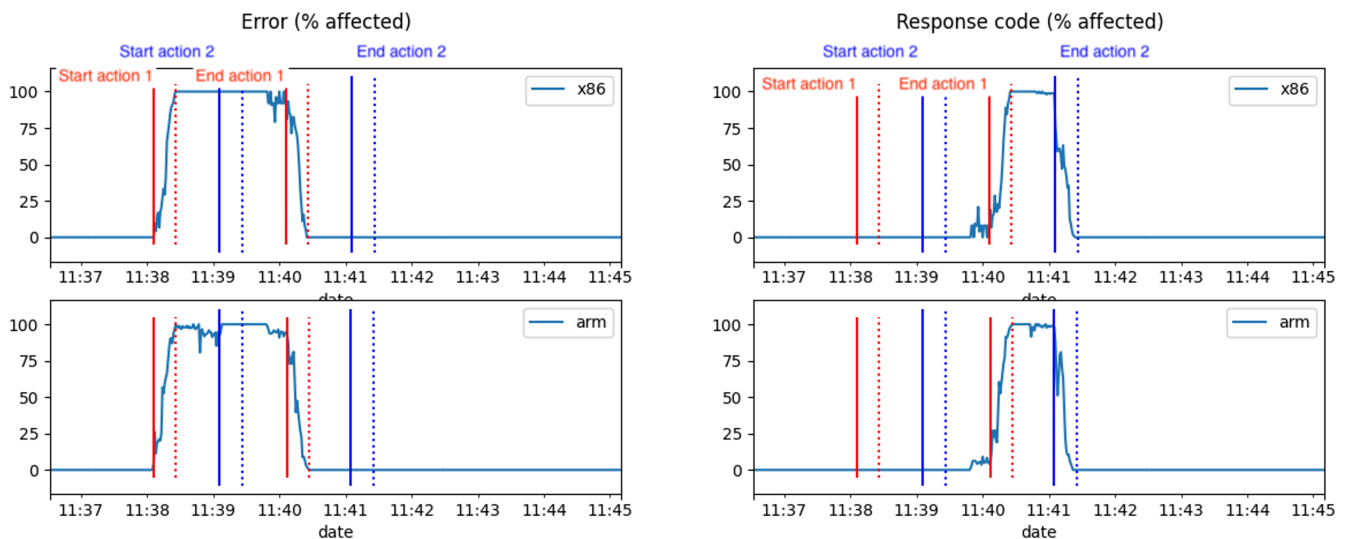
Entendendo a pesquisa

Você pode observar um período de aceleração de até 60s antes que as falhas comecem a afetar todas as invocações. Isso ocorre porque a extensão Lambda pesquisa informações de configuração com pouca frequência enquanto espera o início de um experimento. Você pode ajustar o intervalo de pesquisa definindo a variável de `AWS_FIS_SLOW_POLL_INTERVAL_SECONDS` ambiente (padrão 60s). Um valor menor será pesquisado com mais frequência, mas gerará maior impacto e custo no desempenho. Você também pode notar um período de redução de até 20s após a falha ter sido injetada. Isso ocorre porque a extensão faz pesquisas com mais frequência durante a execução dos experimentos.

Entendendo a simultaneidade

Você pode direcionar as mesmas funções do Lambda com várias ações simultaneamente. Se as ações forem todas diferentes umas das outras, todas as ações serão aplicadas. Por exemplo, você pode adicionar um atraso inicial antes de retornar um erro. Se duas ações idênticas ou conflitantes forem aplicadas à mesma função, somente a ação com a data de início mais antiga será aplicada.

A figura abaixo mostra duas ações conflitantes, `aws:lambda:invocation-error` e `aws:lambda:invocation-http-integration-response` sobrepostas. Inicialmente, `aws:lambda:invocation-error` aumenta às 11:38 e funciona por 2 minutos. Então, `aws:lambda:invocation-http-integration-response` tenta começar às 11:39, mas não entra em vigor até 11:40 após a conclusão da primeira ação. Para manter o tempo do experimento, `aws:lambda:invocation-http-integration-response` ainda termina no horário originalmente previsto de 11:41.



Entendendo a porcentagem de invocação

As ações do AWS Fault Injection Service Lambda usam um destino `aws:lambda:function` que permite selecionar uma ou mais funções. AWS Lambda ARNs Com elas ARNs, as ações do AWS Fault Injection Service Lambda podem injetar falhas em cada invocação da função Lambda selecionada. Para permitir que você injete falhas em apenas uma fração das invocações, cada ação permite especificar um `invocationPercentage` parâmetro com valores de 0 a 100. Usando o `invocationPercentage` parâmetro, você pode garantir que as ações sejam simultâneas mesmo para porcentagens de invocação abaixo de 100%.

Considerações especiais para SnapStart

AWS Lambda as funções SnapStart ativadas terão uma probabilidade maior de esperar por toda a duração `AWS_FIS_SLOW_POLL_INTERVAL_SECONDS` antes de obter a primeira configuração de falha, mesmo que um experimento já esteja em execução. Isso ocorre porque o Lambda SnapStart usa um único instantâneo como estado inicial para vários ambientes de execução e persiste no armazenamento temporário. Para a extensão AWS Fault Injection Service Lambda, ela persistirá na frequência de pesquisa e ignorará a verificação de configuração inicial na inicialização do ambiente de execução. Para obter mais informações sobre o Lambda SnapStart, consulte [Melhorando o desempenho de startups com o Lambda SnapStart](#) no guia do usuário.AWS Lambda

Considerações especiais para funções rápidas e pouco frequentes

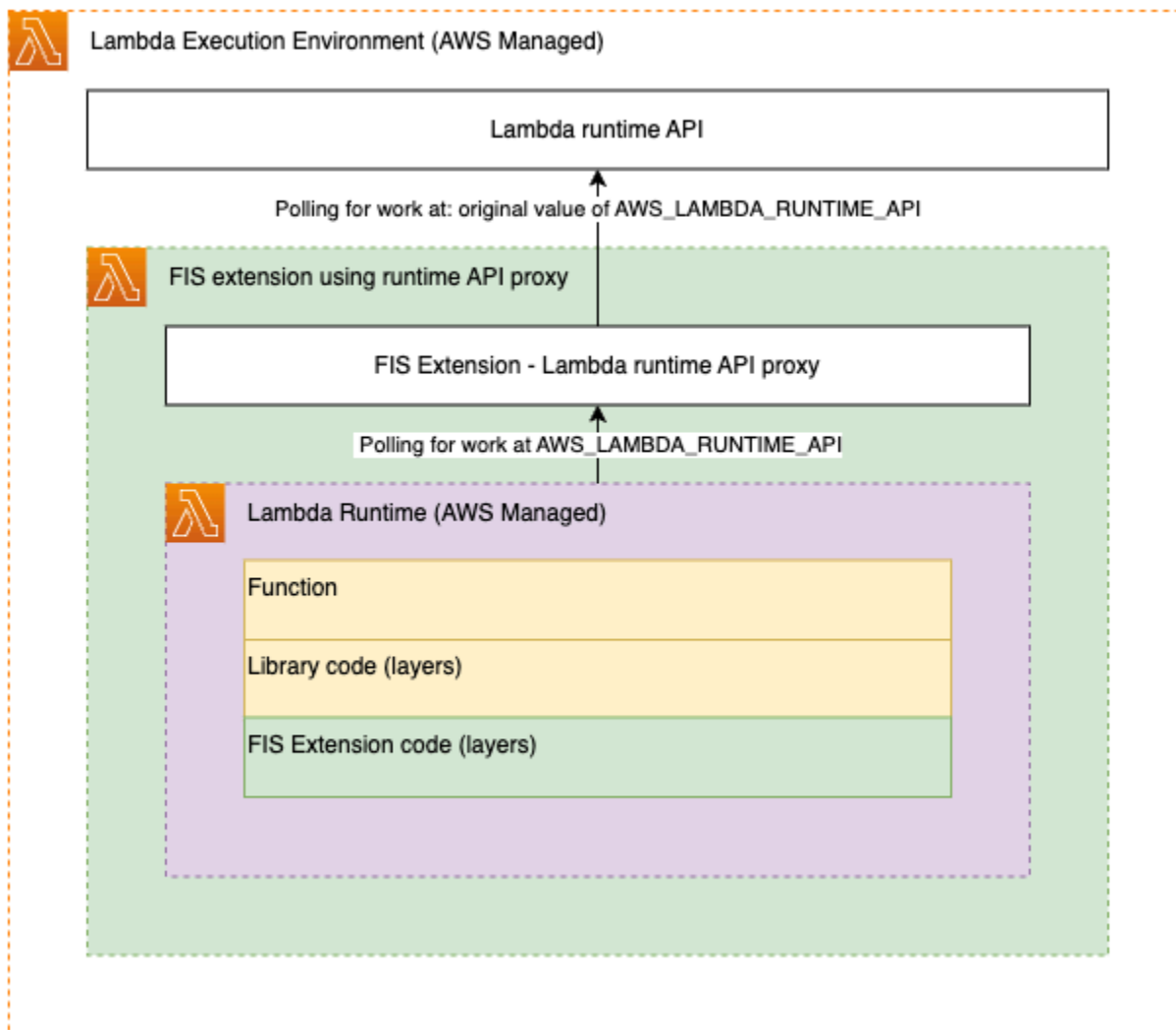
Se sua função Lambda for executada por menos do que a duração média da pesquisa de 70 milissegundos, o encadeamento de pesquisa poderá precisar de várias invocações para obter configurações de falha. Se a função for executada com pouca frequência, por exemplo, uma vez a

cada 15 minutos, a pesquisa nunca será concluída. Para garantir que o encadeamento de pesquisa possa ser concluído, defina o `AWS_FIS_POLL_MAX_WAIT_MILLISECONDS` parâmetro. A extensão aguardará até a duração que você definiu para que uma enquete em voo termine antes de iniciar a função. Observe que isso aumentará a duração da função faturada e resultará em um atraso adicional em algumas invocações.

Configurando várias extensões usando o proxy da API Lambda Runtime

A extensão Lambda usa o proxy da API AWS Lambda Runtime para interceptar invocações de funções antes que elas atinjam o tempo de execução. Isso é feito expondo um proxy da API AWS Lambda Runtime ao tempo de execução e anunciando sua localização na `AWS_LAMBDA_RUNTIME_API` variável.

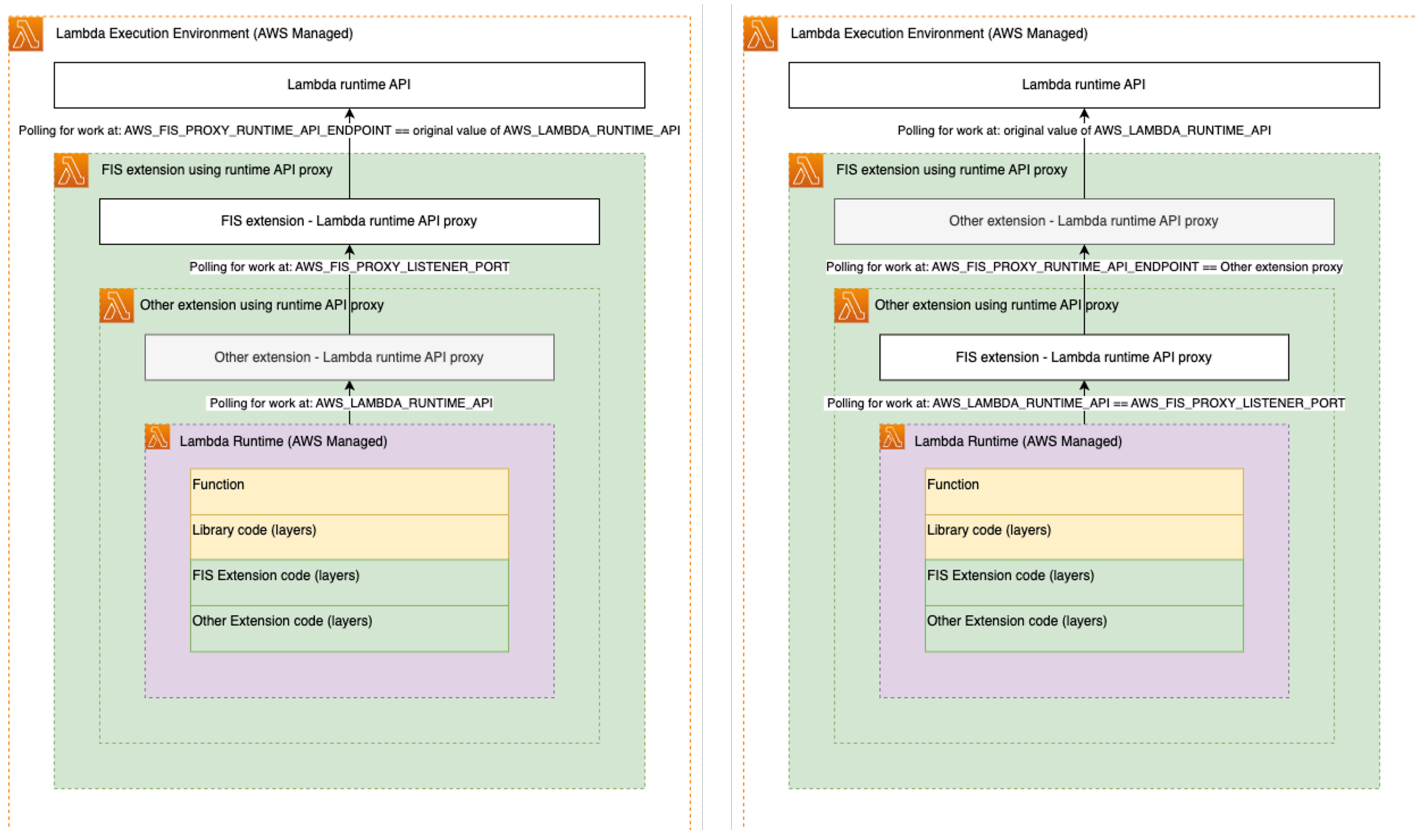
O diagrama a seguir mostra a configuração de uma única extensão usando o proxy da API Lambda Runtime:



Para usar a extensão AWS FIS Lambda com outra extensão usando o padrão de proxy da API AWS Lambda Runtime, você precisará encadear os proxies usando um script de bootstrap personalizado. A extensão AWS FIS Lambda aceita as seguintes variáveis de ambiente:

- `AWS_FIS_PROXY_RUNTIME_API_ENDPOINT`- Usa uma string no formato `127.0.0.1:9876` que representa o IP local e a porta do ouvinte para a API AWS Lambda Runtime. Esse pode ser o valor original `AWS_LAMBDA_RUNTIME_API` ou a localização de outro proxy.
- `AWS_FIS_PROXY_LISTENER_PORT`- Obtém um número de porta na qual a AWS FIS extensão deve iniciar seu próprio proxy, por padrão `9100`.

Com essas configurações, você pode encadear a AWS FIS extensão com outra extensão usando o proxy da API Lambda Runtime em duas ordens diferentes.



Para obter mais informações sobre o proxy da AWS Lambda Runtime API, consulte [Como aprimorar a segurança e a governança do tempo de AWS Lambda execução com a extensão de proxy da Runtime API](#) e [Usar a API de tempo de execução Lambda para tempos de execução personalizados](#) no AWS Lambda guia do usuário.

Usando AWS FIS com tempos de execução de contêineres

Para AWS Lambda funções que usam imagens de contêiner que aceitam a variável de ambiente `AWS_LAMBDA_RUNTIME_API`, você pode empacotar a extensão AWS FIS Lambda em sua imagem de contêiner seguindo as etapas abaixo:

1. Determine o ARN da camada da qual extrair a extensão. Para obter mais informações sobre como encontrar o ARN, consulte [Configurar funções do Lambda](#)
2. Use a AWS Command Line Interface (CLI) para solicitar detalhes sobre a extensão. `aws lambda get-layer-version-by-arn --arn fis-extension-arn` A resposta conterá um `Location` campo contendo uma URL pré-assinada a partir da qual você pode baixar a extensão FIS como um arquivo ZIP.
3. Descompacte o conteúdo da extensão em seu sistema `/opt` de arquivos Docker. Veja a seguir um exemplo de Dockerfile baseado no tempo de execução do NodeJS Lambda:

```
# extension installation #
FROM amazon/aws-lambda-nodejs:12 AS builder
COPY extension.zip extension.zip
RUN yum install -y unzip
RUN mkdir -p /opt
RUN unzip extension.zip -d /opt
RUN rm -f extension.zip
FROM amazon/aws-lambda-nodejs:12
WORKDIR /opt
COPY --from=builder /opt .
# extension installation finished #
# JS example. Modify as required by your runtime
WORKDIR ${LAMBDA_TASK_ROOT}
COPY index.js package.json .
RUN npm install
CMD [ "index.handler" ]
```

Para obter mais informações sobre imagens de contêiner, consulte [Criar uma função Lambda usando uma imagem de contêiner](#) no guia do AWS Lambda usuário.

AWS FIS Variáveis de ambiente Lambda

A seguir está uma lista de variáveis de ambiente para a extensão AWS FIS Lambda

- `AWS_FIS_CONFIGURATION_LOCATION`- Obrigatório. Local onde AWS FIS gravará as configurações de falha ativas e a extensão lerá as configurações de falha. Os locais devem estar no formato ARN do Amazon S3, incluindo um bucket e um caminho. Por exemplo, `.arn:aws:s3:::my-fis-config-bucket/FisConfigs/`
- `AWS_LAMBDA_EXEC_WRAPPER`- Obrigatório. Localização do [script do AWS Lambda wrapper](#) usado para configurar a extensão AWS FIS Lambda. Isso deve ser definido para o `/opt/aws-fis/bootstrap` script incluído na extensão.
- `AWS_FIS_LOG_LEVEL`- Opcional. Nível de log para mensagens emitidas pela extensão AWS FIS Lambda. Os valores compatíveis são `INFO`, `WARN` e `ERROR`. Se não for definida, a AWS FIS extensão assumirá como padrão `INFO`.
- `AWS_FIS_EXTENSION_METRICS`- Opcional. Os possíveis valores são `all` e `none`. Se definido para `all` a extensão, emitirá métricas EMF sob o `aws-fis-extension` namespace
- `AWS_FIS_SLOW_POLL_INTERVAL_SECONDS`- Opcional. Se definido, substituirá o intervalo de pesquisa (em segundos) enquanto a extensão não estiver injetando falhas e aguardando que uma configuração de falha seja adicionada ao local de configuração. O padrão é `60`.
- `AWS_FIS_PROXY_RUNTIME_API_ENDPOINT`- Opcional. Se definido, substituirá o valor de `AWS_LAMBDA_RUNTIME_API` para definir onde a AWS FIS extensão interage com a API de tempo de AWS Lambda execução para controlar a invocação da função. Espera `IP:PORT`, por exemplo, `127.0.0.1:9000` Para obter mais informações `AWS_LAMBDA_RUNTIME_API`, consulte [Como usar a API de tempo de execução do Lambda para tempos de execução personalizados no guia do AWS Lambda usuário](#).
- `AWS_FIS_PROXY_LISTENER_PORT`- Opcional. Define a porta na qual a extensão AWS FIS Lambda expõe um proxy de API AWS Lambda de tempo de execução que pode ser usado por outra extensão ou pelo tempo de execução. O padrão é `9100`.
- `AWS_FIS_POLL_MAX_WAIT_MILLISECONDS`- Opcional. Se definida como um valor diferente de zero, essa variável define o número de milissegundos em que a extensão aguardará a conclusão de uma pesquisa assíncrona em andamento antes de avaliar as configurações de falha e iniciar a invocação do tempo de execução. O padrão é `0`.

Versões disponíveis da AWS FIS extensão para Lambda

Esta seção inclui informações sobre as versões da extensão AWS FIS Lambda. A extensão suporta funções Lambda desenvolvidas para as plataformas x86-64 e ARM64 (Graviton2). Sua função Lambda deve ser configurada para usar o Amazon Resource Name (ARN) específico para o Região

da AWS local em que está atualmente hospedada. Você pode ver Região da AWS os detalhes do ARN abaixo.

Tópicos

- [AWS FIS Notas de lançamento da extensão Lambda](#)
- [Guia de acesso para a extensão Lambda ARNs](#)
- [Como encontrar o número da versão da sua extensão do Lambda](#)

AWS FIS Notas de lançamento da extensão Lambda

A tabela a seguir descreve as alterações feitas nas versões recentes da extensão AWS FIS Lambda

Versão	Data de lançamento	Observações
1.0.0	2024-10-29	Lançamento inicial

Guia de acesso para a extensão Lambda ARNs

Você deve ter pelo menos um parâmetro em seu Conta da AWS e Região da AWS antes de poder pesquisar parâmetros públicos usando o console. Para descobrir parâmetros públicos, consulte [Descobrimdo parâmetros públicos no Parameter Store](#).

Acesso ao console:

1. Abra o AWS Systems Manager console em <https://console.aws.amazon.com/systems-manager/>.
2. No painel de navegação, selecione Repositório de parâmetros.
3. Selecione a guia Public parameters (Parâmetros públicos).
4. Selecione a lista suspensa Select a service (Selecionar um serviço). Nas opções do menu suspenso, escolha. `fis`
5. (Opcional) Filtre os parâmetros selecionados inserindo mais informações na barra de pesquisa. Para arquiteturas arm64, filtre os parâmetros inserindo "arm64". Para arquiteturas x86_64, filtre os parâmetros inserindo "x86_64".
6. Escolha o parâmetro público que você quiser usar.
7. Nos detalhes do parâmetro, localize o valor do ARN. Copie o ARN para usar na configuração de extensões de camada em suas funções Lambda de destino.

AWS CLI Acesso:

Nomes de parâmetros SSM

Os seguintes nomes de parâmetros SSM estão disponíveis para diferentes arquiteturas:

1. arm64 diz: `/aws/service/fis/lambda-extension/AWS-FIS-extension-arm64/1.x.x`
2. x86_64: `/aws/service/fis/lambda-extension/AWS-FIS-extension-x86_64/1.x.x`

AWS CLI Formato de comando

Para recuperar a extensão ARNs, use o seguinte formato de AWS CLI comando, em que `parameterName` é o nome da arquitetura e a região é o destino: Região da AWS

```
aws ssm get-parameter --name parameterName --region region
```

Exemplo de uso

```
aws ssm get-parameter --name /aws/service/fis/lambda-extension/AWS-FIS-extension-x86_64/1.x.x --region ap-southeast-2
```

Formato de resposta

O comando retorna um objeto JSON contendo os detalhes do parâmetro, conforme mostrado a seguir. O ARN da camada Lambda está incluído no campo Valor do objeto Parâmetro. Copie o ARN para usar na configuração de extensões de camada em suas funções Lambda de destino.

```
{
  "Parameter": {
    "Name": "/aws/service/fis/lambda-extension/AWS-FIS-extension-x86_64/1.x.x",
    "Type": "String",
    "Value": "arn:aws:lambda:ap-southeast-2:211125361907:layer:aws-fis-extension-x86_64:9",
    "Version": 1,
    "LastModifiedDate": "2025-01-02T15:13:54.465000-05:00",
    "ARN": "arn:aws:ssm:ap-southeast-2::parameter/aws/service/fis/lambda-extension/AWS-FIS-extension-x86_64/1.x.x",
    "DataType": "text"
  }
}
```

```
}  
}  
}
```

Acesso programático:

Recupere esses parâmetros públicos programaticamente ao criar ou configurar suas funções do Lambda usando Infrastructure as Code (IaC). Essa abordagem ajuda a manter suas funções do Lambda com o ARN da versão mais recente da camada sem exigir atualizações manuais de código que seriam necessárias se o ARN da AWS FIS camada de extensão fosse codificado permanentemente. Os recursos a seguir mostram como recuperar parâmetros públicos usando plataformas comuns de IaC:

- [Obtenha parâmetros públicos usando o AWS SDK](#)
- [Obtenha parâmetros públicos do AWS Systems Manager Parameter Store usando o AWS CDK](#)
- [Obtenha parâmetros públicos usando o Terraform](#)

Como encontrar o número da versão da sua extensão do Lambda

Use o procedimento a seguir para localizar o número da versão da sua extensão AWS FIS Lambda atualmente configurada.

1. Abra o AWS Lambda console em <https://console.aws.amazon.com/lambda/>.
2. Escolha a função do Lambda na qual deseja adicionar a camada AWS-FIS-Extension.
3. Na seção Camadas, escolha Editar.
4. Na seção Editar camadas, escolha Adicionar uma camada.
5. Na seção Escolha uma camada, escolha Especificar um ARN.
6. Insira o ARN da camada de AWS FIS extensão que corresponde à sua arquitetura Região da AWS . Você pode encontrar o ARN usando o console ou os métodos AWS CLI de acesso programático descritos nas seções anteriores.
7. Escolha Verificar para confirmar se o ARN da camada é válido e, em seguida, escolha Adicionar.
8. Use a guia Testar para testar a função.
9. Depois que o teste for concluído, visualize a saída de logs. Localize a versão AWS FIS da extensão Lambda na seção Detalhes da execução.

Gerenciando modelos AWS de experimentos do FIS

Você pode criar e gerenciar modelos de experimentos usando o console AWS FIS ou a linha de comando. Um modelo de experimento contém uma ou mais ações a serem executadas em alvos específicos durante um experimento. Ele também contém as condições de parada que evitam que o experimento ultrapasse os limites. Para ter mais informações sobre os componentes de um modelo de experimento, consulte [Componentes do modelo de experimento](#). Depois de criar um modelo de experimento, você pode usá-lo para executar um experimento.

Tarefas

- [Criar um modelo de experimento](#)
- [Visualizar modelos de experimentos](#)
- [Gerar uma pré-visualização de destinos com base em um modelo de experimento](#)
- [Iniciar um experimento a partir de um modelo](#)
- [Atualizar um modelo de experimento](#)
- [Marcar modelos de experimentos](#)
- [Excluir um modelo de experimento](#)
- [Exemplos de AWS modelos de experimentos FIS](#)

Criar um modelo de experimento

Antes de começar, conclua as seguintes tarefas:

- [Planejar seu experimento](#).
- Crie uma função do IAM que conceda ao serviço AWS FIS permissão para realizar ações em seu nome. Para obter mais informações, consulte [Funções do IAM para experimentos AWS do FIS](#).
- Certifique-se de ter acesso ao AWS FIS. Para obter mais informações, consulte [Exemplos de políticas do AWS FIS](#).

Para criar um modelo de experimento usando o console

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Modelos de experimento.
3. Escolha Criar modelo de experimento.

4. Para a Etapa 1, Especificar detalhes do modelo, faça o seguinte:
 - a. Em Descrição e nome, insira uma descrição para o modelo, como `Amazon S3 Network Disrupt Connectivity`.
 - b. (Opcional) Em Segmentação de contas, escolha Várias contas para configurar um modelo de experimento com várias contas.
 - c. Escolha Avançar e vá para a Etapa 2, Especifique ações e alvos.
5. Em Ações, especifique o conjunto de ações para o modelo. Para cada ação, escolha Adicionar ação e conclua o seguinte:
 - Em Nome, insira um nome para a ação.

Os caracteres permitidos são caracteres alfanuméricos, hífen (-) e sublinhados (_). O nome deve iniciar com uma letra. Espaços não são permitidos. Cada nome de ação deve ser exclusivo neste modelo.
 - (Opcional) Para Descrição, insira uma descrição para a ação. O tamanho máximo é de 512 caracteres.
 - (Opcional) Para Iniciar depois, selecione outra ação definida neste modelo que deve ser concluída antes do início da ação atual. Caso contrário, a ação será executada no início do experimento.
 - Em Tipo de ação, escolha a ação AWS FIS.
 - Para Destino, escolha um destino que você definiu na seção Destinos. Se você ainda não definiu um alvo para essa ação, o AWS FIS cria um novo alvo para você.
 - Em Parâmetros de ação, especifique os parâmetros da ação. Essa seção aparece somente se a ação AWS FIS tiver parâmetros.
 - Escolha Salvar.
6. Para Destinos, defina os recursos de destino nos quais deseja realizar as ações. Você deve especificar pelo menos um ID de recurso ou uma tag de recurso como destino. Escolha Editar para editar o alvo que o AWS FIS criou para você na etapa anterior ou escolha Adicionar destino. Para cada destino, faça o seguinte:
 - Para Nome, insira um nome para o destino.

Os caracteres permitidos são caracteres alfanuméricos, hífen (-) e sublinhados (_). O nome deve iniciar com uma letra. Espaços não são permitidos. Cada nome de destino deve ser exclusivo neste modelo.

- Para Tipo de recurso, escolha um tipo de recurso compatível com a ação.
 - Para Método do destino, siga um destes procedimentos:
 - Escolha Recurso IDs e, em seguida, escolha ou adicione o recurso IDs.
 - Escolha tags, filtros e parâmetros de recursos e, em seguida, adicione as tags e os filtros necessários. Para obter mais informações, consulte [the section called “Identificar recursos de destino”](#).
 - Para Modo de seleção, escolha Contagem para executar a ação no número especificado de destinos identificados ou escolha Porcentagem para executar a ação na porcentagem especificada de destinos identificados. Por padrão, a ação é executada em todos os destinos identificados.
 - Escolha Salvar.
7. Para atualizar uma ação com o destino que você criou, localize a ação em Ações, escolha Editar e, em seguida, atualize o Destino. Você pode usar o mesmo destino para várias ações.
 8. (Opcional) Para opções de experimento, selecione o comportamento do modo de resolução de alvo vazio.
 9. Escolha Avançar para ir para a Etapa 3, Configurar o acesso ao serviço.
 10. Em Acesso ao serviço, escolha Usar um perfil do IAM existente e, em seguida, escolha o perfil do IAM que você criou conforme descrito nos pré-requisitos deste tutorial. Se sua função não for exibida, verifique se ela tem a relação de confiança necessária. Para obter mais informações, consulte [the section called “Perfil do experimento”](#).
 11. (Somente experimentos com várias contas) Em Configurações da conta de destino, adicione um ARN do perfil e uma descrição opcional para cada conta de destino. Para carregar a função da conta de destino ARNs com um arquivo CSV, escolha Carregar função ARNs para todas as contas de destino e, em seguida, escolha Escolher arquivo.CSV
 12. Escolha Avançar para ir para a Etapa 4, Definir configurações opcionais.
 13. (Opcional) Para condições de parada, selecione os CloudWatch alarmes da Amazon para as condições de parada. Para obter mais informações, consulte [Condições de parada para o AWS FIS](#).
 14. (Opcional) Para Logs, configure a opção de destino. Para enviar logs para um bucket do S3, escolha Enviar para um bucket do Amazon S3 e insira o nome e o prefixo do bucket. Para enviar registros para CloudWatch registros, escolha Enviar para CloudWatch registros e entre no grupo de registros.

15. (Opcional) Em Tags, escolha Adicionar nova tag e especifique uma chave de tag e um valor de tag. As tags que você adiciona são aplicadas ao seu modelo de experimento, não aos experimentos que são executados usando o modelo.
16. Escolha Avançar para ir para a Etapa 5, Revisar e criar.
17. Revise o modelo e escolha Criar modelo de experimento. Quando a confirmação for solicitada, insira `create`, em seguida, escolha Criar modelo de experimento.

Para criar um modelo de experimento usando a CLI

Use o comando [create-experiment-template](#).

Você pode carregar um modelo de experimento a partir de um arquivo JSON.

Use o parâmetro `--cli-input-json`.

```
aws fis create-experiment-template --cli-input-json fileb://<path-to-json-file>
```

Para obter mais informações, consulte [Gerar um modelo de esqueleto de CLI](#) no Guia do usuário do AWS Command Line Interface . Para obter modelos de exemplo, consulte [Exemplos de AWS modelos de experimentos FIS](#).

Visualizar modelos de experimentos

Você pode ver os modelos de experimento que você criou.

Para visualizar um modelo de experimento usando o console

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Modelos de experimento.
3. Para visualizar informações sobre um modelo específico, selecione o ID do modelo de experimento.
4. Na seção Detalhes, você pode ver a descrição e as condições de interrupção do modelo.
5. Para visualizar as ações do modelo de experimento, escolha Ações.
6. Para visualizar os destinos do modelo de experimento, escolha Destinos.
7. Para visualizar as tags do modelo de experimento, escolha Tags.

Para visualizar um modelo de experimento usando a CLI

Use o [list-experiment-templates](#) comando para obter uma lista de modelos de experimento e use o [get-experiment-template](#) comando para obter informações sobre um modelo de experimento específico.

Gerar uma pré-visualização de destinos com base em um modelo de experimento

Antes de iniciar um experimento, você pode gerar uma pré-visualização de destinos para verificar se o modelo do experimento está configurado para atingir os recursos esperados. Os recursos que são definidos como destino quando você inicia o experimento real podem ser diferentes dos recursos da pré-visualização, pois eles podem ser removidos, atualizados ou amostrados aleatoriamente. Ao gerar uma pré-visualização de destinos, você inicia um experimento que ignora todas as ações.

Note

A geração de uma pré-visualização de destinos não permite verificar se você tem as permissões necessárias para realizar ações nos recursos.

Como iniciar uma pré-visualização de destinos usando o console

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Modelos de experimento.
3. Para visualizar os destinos do modelo de experimento, escolha Destinos.
4. Para verificar os recursos de destino para o modelo de experimento, escolha Gerar pré-visualização. Quando você executa um experimento, essa pré-visualização de destinos é atualizada automaticamente com os destinos do experimento mais recente.

Como iniciar uma pré-visualização de destinos usando a CLI

- Execute o comando [start-experiment](#) a seguir. Substitua os valores em itálico por seus próprios valores.

```
aws fis start-experiment \  
  --experiment-options actionsMode=skip-all \  
  --experiment-name <nome do experimento>
```

```
--experiment-template-id EXTxxxxxxxx
```

Iniciar um experimento a partir de um modelo

Depois de criar um modelo de experimento, você pode começar os experimentos usando esse modelo.

Quando você inicia um experimento, criamos um snapshot do modelo especificado e usamos esse snapshot para executar o experimento. Portanto, se o modelo do experimento for atualizado ou excluído durante a execução do experimento, essas alterações não terão impacto no experimento em execução.

Quando você inicia um experimento, o AWS FIS cria uma função vinculada ao serviço em seu nome. Para obter mais informações, consulte [Use funções vinculadas ao serviço para o AWS Fault Injection Service](#).

Depois de iniciar o experimento, você pode interrompê-lo a qualquer momento. Para obter mais informações, consulte [Interromper um experimento](#).

Para iniciar um experimento usando o console

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Modelos de experimento.
3. Selecione o modelo de experimento e escolha Iniciar experimento.
4. (Opcional) Para adicionar uma tag ao experimento, escolha Adicionar nova tag e insira uma chave de tag e um valor de tag.
5. Escolha Start experiment (Iniciar experimento). Quando a confirmação for solicitada, insira **start** e escolha Iniciar experimento.

Para iniciar um experimentos usando a CLI

Use o comando [start-experiment](#).

Atualizar um modelo de experimento

Você pode atualizar um modelo de experimento existente. Quando você atualiza um modelo de experimento, as alterações não afetam nenhum experimento em execução que use o modelo.

Para atualizar um modelo de experimento usando o console

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Modelos de experimento.
3. Selecione o modelo do experimento e escolha Ações, Atualizar modelo de experimento.
4. Modifique os detalhes do modelo conforme necessário e escolha Atualizar modelo de experimento.

Para atualizar um modelo de experimento usando a CLI

Use o comando [update-experiment-template](#).

Marcar modelos de experimentos

Você pode aplicar suas próprias tags aos modelos de experimento para ajudar na organização. Você também pode implementar [políticas do IAM baseadas em tags](#) para controlar o acesso aos modelos de experimentos.

Para marcar um modelo de experimento usando o console

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Modelos de experimento.
3. Selecione o modelo de experimento e escolha Ações, Gerenciar tags.
4. Para adicionar uma nova tag, escolha Adicionar nova tag e especifique uma chave e valor.

Para remover uma tag, selecione Remover para a tag.

5. Escolha Salvar.

Para marcar um modelo de experimento usando a CLI

Use o comando [tag-resource](#).

Excluir um modelo de experimento

Se você não precisar mais de um modelo de experimento, poderá excluí-lo. Quando você exclui um modelo de experimento, quaisquer experimentos em execução que usem o modelo não são

afetados. O experimento continua em andamento até ser concluído ou interrompido. No entanto, os modelos de experimentos excluídos não estão disponíveis para visualização na página Experimentos no console.

Para excluir um modelo de experimento usando o console

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Modelos de experimento.
3. Selecione o modelo do experimento e escolha Ações, Excluir modelo de experimento.
4. Quando a confirmação for solicitada, insira **delete** e escolha Excluir experimento.

Para excluir um modelo de experimento usando a CLI

Use o comando [delete-experiment-template](#).

Exemplos de AWS modelos de experimentos FIS

Se você estiver usando a API AWS FIS ou uma ferramenta de linha de comando para criar um modelo de experimento, poderá criar o modelo em JavaScript Object Notation (JSON). Para obter mais informações sobre os componentes de um modelo de experimento, consulte [AWS Componentes do modelo de experimento FIS](#).

Para criar um experimento usando um dos modelos de exemplo, salve-o em um arquivo JSON (por exemplo, `my-template.json`), substitua os valores do espaço reservado por seus próprios valores e execute o comando a seguir [create-experiment-template](#). *italics*

```
aws fis create-experiment-template --cli-input-json file://my-template.json
```

Exemplos de modelos do

- [Interrompa EC2 instâncias com base em filtros](#)
- [Interromper um número específico de EC2 instâncias](#)
- [Execute um documento AWS FIS SSM pré-configurado](#)
- [Executar um runbook predefinido de automação](#)
- [Ações da API Throttle em EC2 instâncias com a função de destino do IAM](#)
- [Teste de estresse da CPU de pods em um cluster do Kubernetes](#)

- [Exceção de taxa de transferência provisionada para um número especificado de Kinesis Data Streams](#)
- [Experimente um exemplo de permissões de função](#)

Interrompa EC2 instâncias com base em filtros

O exemplo a seguir interrompe todas as EC2 instâncias da Amazon em execução na região especificada com a tag especificada na VPC especificada. Ele as reinicia após dois minutos.

```
{
  "tags": {
    "Name": "StopEC2InstancesWithFilters"
  },
  "description": "Stop and restart all instances in us-east-1b with the tag env=prod in the specified VPC",
  "targets": {
    "myInstances": {
      "resourceType": "aws:ec2:instance",
      "resourceTags": {
        "env": "prod"
      },
      "filters": [
        {
          "path": "Placement.AvailabilityZone",
          "values": ["us-east-1b"]
        },
        {
          "path": "State.Name",
          "values": ["running"]
        },
        {
          "path": "VpcId",
          "values": [ "vpc-aabbcc11223344556" ]
        }
      ],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "StopInstances": {
      "actionId": "aws:ec2:stop-instances",
      "description": "stop the instances",

```

```

    "parameters": {
      "startInstancesAfterDuration": "PT2M"
    },
    "targets": {
      "Instances": "myInstances"
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "arn:aws:cloudwatch:us-east-1:111122223333:alarm:alarm-name"
    }
  ],
  "roleArn": "arn:aws:iam::111122223333:role/role-name"
}

```

Interromper um número específico de EC2 instâncias

O exemplo a seguir interrompe três instâncias com a tag especificada. AWS O FIS seleciona as instâncias específicas para parar aleatoriamente. Ele reinicia essas instâncias após dois minutos.

```

{
  "tags": {
    "Name": "StopEC2InstancesByCount"
  },
  "description": "Stop and restart three instances with the specified tag",
  "targets": {
    "myInstances": {
      "resourceType": "aws:ec2:instance",
      "resourceTags": {
        "env": "prod"
      },
      "selectionMode": "COUNT(3)"
    }
  },
  "actions": {
    "StopInstances": {
      "actionId": "aws:ec2:stop-instances",
      "description": "stop the instances",
      "parameters": {
        "startInstancesAfterDuration": "PT2M"
      },
    },
  },
}

```

```

    "targets": {
      "Instances": "myInstances"
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "arn:aws:cloudwatch:us-east-1:111122223333:alarm:alarm-name"
    }
  ],
  "roleArn": "arn:aws:iam::111122223333:role/role-name"
}

```

Execute um documento AWS FIS SSM pré-configurado

[O exemplo a seguir executa uma injeção de falha de CPU por 60 segundos na EC2 instância especificada usando um documento AWS FIS SSM pré-configurado, -CPU-stress. AWSFIS-Run](#)
 AWS O FIS monitora o experimento por dois minutos.

```

{
  "tags": {
    "Name": "CPUStress"
  },
  "description": "Run a CPU fault injection on the specified instance",
  "targets": {
    "myInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["arn:aws:ec2:us-east-1:111122223333:instance/instance-id"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "CPUStress": {
      "actionId": "aws:ssm:send-command",
      "description": "run cpu stress using ssm",
      "parameters": {
        "duration": "PT2M",
        "documentArn": "arn:aws:ssm:us-east-1::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\": \"60\"",
        "\"InstallDependencies\": \"True\", \"CPU\": \"0\"}"
      },
    },
  },
}

```

```

        "targets": {
            "Instances": "myInstance"
        }
    },
    "stopConditions": [
        {
            "source": "aws:cloudwatch:alarm",
            "value": "arn:aws:cloudwatch:us-east-1:111122223333:alarm:alarm-name"
        }
    ],
    "roleArn": "arn:aws:iam::111122223333:role/role-name"
}

```

Executar um runbook predefinido de automação

[O exemplo a seguir publica uma notificação para o Amazon SNS usando um runbook fornecido pelo Systems Manager, AWS-Publish. SNSNotification](#) A função deve ter permissões para publicar notificações no tópico do SNS especificado.

```

{
    "description": "Publish event through SNS",
    "stopConditions": [
        {
            "source": "none"
        }
    ],
    "targets": {
    },
    "actions": {
        "sendToSns": {
            "actionId": "aws:ssm:start-automation-execution",
            "description": "Publish message to SNS",
            "parameters": {
                "documentArn": "arn:aws:ssm:us-east-1::document/AWS-PublishSNSNotification",
                "documentParameters": "{\"Message\": \"Hello, world\", \"TopicArn\": \"arn:aws:sns:us-east-1:111122223333:topic-name\"}",
                "maxDuration": "PT1M"
            },
            "targets": {
            }
        }
    }
}

```

```

    },
    "roleArn": "arn:aws:iam::111122223333:role/role-name"
  }

```

Ações da API Throttle em EC2 instâncias com a função de destino do IAM

O exemplo a seguir controla a utilização de 100% das chamadas de API especificadas na definição da ação para chamadas de API feitas pelos perfis do IAM especificados na definição de destinos.

Note

Se você quiser segmentar EC2 instâncias que sejam membros de um grupo de Auto Scaling, use a ação `aws:ec2: asg-insufficient-instance-capacity -error` e, em vez disso, segmente por grupo de Auto Scaling. Para obter mais informações, consulte [aws:ec2:asg-insufficient-instance-capacity-error](#).

```

{
  "tags": {
    "Name": "ThrottleEC2APIActions"
  },
  "description": "Throttle the specified EC2 API actions on the specified IAM role",
  "targets": {
    "myRole": {
      "resourceType": "aws:iam:role",
      "resourceArns": ["arn:aws:iam::111122223333:role/role-name"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "ThrottleAPI": {
      "actionId": "aws:fis:inject-api-throttle-error",
      "description": "Throttle APIs for 5 minutes",
      "parameters": {
        "service": "ec2",
        "operations": "DescribeInstances,DescribeVolumes",
        "percentage": "100",
        "duration": "PT2M"
      },
      "targets": {
        "Roles": "myRole"
      }
    }
  }
}

```

```

    }
  }
},
"stopConditions": [
  {
    "source": "aws:cloudwatch:alarm",
    "value": "arn:aws:cloudwatch:us-east-1:111122223333:alarm:alarm-name"
  }
],
"roleArn": "arn:aws:iam::111122223333:role/role-name"
}

```

Teste de estresse da CPU de pods em um cluster do Kubernetes

O exemplo a seguir usa o Chaos Mesh para testar o estresse da CPU de pods em um cluster do Amazon EKS Kubernetes por um minuto.

```

{
  "description": "ChaosMesh StressChaos example",
  "targets": {
    "Cluster-Target-1": {
      "resourceType": "aws:eks:cluster",
      "resourceArns": [
        "arn:aws:eks:arn:aws::111122223333:cluster/cluster-id"
      ],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "TestCPUStress": {
      "actionId": "aws:eks:inject-kubernetes-custom-resource",
      "parameters": {
        "maxDuration": "PT2M",
        "kubernetesApiVersion": "chaos-mesh.org/v1alpha1",
        "kubernetesKind": "StressChaos",
        "kubernetesNamespace": "default",
        "kubernetesSpec": "{\"selector\":{\"namespaces\":[\"default\"],\n\"labelSelectors\":{\"run\":{\"nginx\"}},\"mode\":{\"all\"},\"stressors\":{\"cpu\":{\"workers\":1,\"load\":50}},\"duration\":{\"1m\"}}",
        "targets": {
          "Cluster": "Cluster-Target-1"
        }
      }
    }
  }
}

```

```

    }
  },
  "stopConditions": [{
    "source": "none"
  }],
  "roleArn": "arn:aws:iam::111122223333:role/role-name",
  "tags": {}
}

```

O exemplo a seguir utiliza o Litmus para realizar um teste de estresse na CPU de pods em um cluster do Kubernetes do Amazon EKS por um minuto.

```

{
  "description": "Litmus CPU Hog",
  "targets": {
    "MyCluster": {
      "resourceType": "aws:eks:cluster",
      "resourceArns": [
        "arn:aws:eks:arn:aws::111122223333:cluster/cluster-id"
      ],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "MyAction": {
      "actionId": "aws:eks:inject-kubernetes-custom-resource",
      "parameters": {
        "maxDuration": "PT2M",
        "kubernetesApiVersion": "litmuschaos.io/v1alpha1",
        "kubernetesKind": "ChaosEngine",
        "kubernetesNamespace": "litmus",
        "kubernetesSpec": "{\n  \"engineState\": \"active\",\n  \"appinfo\":\n  {\n    \"appns\": \"default\",\n    \"applabel\": \"run=nginx\",\n    \"appkind\": \"deployment\",\n    \"chaosServiceAccount\": \"litmus-admin\",\n    \"experiments\": [\n      {\n        \"name\": \"pod-cpu-hog\",\n        \"spec\": {\n          \"components\": {\n            \"env\": [\n              {\n                \"name\": \"TOTAL_CHAOS_DURATION\",\n                \"value\": \"60\",\n              },\n              {\n                \"name\": \"CPU_CORES\",\n                \"value\": \"1\",\n              },\n              {\n                \"name\": \"PODS_AFFECTED_PERC\",\n                \"value\": \"100\",\n              },\n              {\n                \"name\": \"CONTAINER_RUNTIME\",\n                \"value\": \"docker\",\n              },\n              {\n                \"name\": \"SOCKET_PATH\",\n                \"value\": \"/var/run/docker.sock\",\n              }\n            ],\n            \"probe\": []\n          }\n        }\n      }\n    ],\n    \"annotationCheck\": \"false\"\n  }\n}"
      },
      "targets": {
        "Cluster": "MyCluster"
      }
    }
  }
}

```

```

    }
  },
  "stopConditions": [{
    "source": "none"
  }],
  "roleArn": "arn:aws:iam::111122223333:role/role-name",
  "tags": {}
}

```

Exceção de taxa de transferência provisionada para um número especificado de Kinesis Data Streams

O exemplo a seguir gera uma exceção de taxa de transferência provisionada para 100% das solicitações de até cinco Kinesis Data Streams com a tag especificada. AWS O FIS seleciona os fluxos a serem afetados aleatoriamente. Após 5 minutos, a falha é removida.

```

{
  "description": "Kinesis stream experiment",
  "targets": {
    "KinesisStreams-Target-1": {
      "resourceType": "aws:kinesis:stream",
      "resourceTags": {
        "tag-key": "tag-value"
      },
      "selectionMode": "COUNT(5)"
    }
  },
  "actions": {
    "kinesis": {
      "actionId": "aws:kinesis:stream-provisioned-throughput-exception",
      "description": "my-stream",
      "parameters": {
        "duration": "PT5M",
        "percentage": "100",
        "service": "kinesis"
      },
      "targets": {
        "KinesisStreams": "KinesisStreams-Target-1"
      }
    }
  },
  "stopConditions": [

```

```

    {
      "source": "none"
    }
  ],
  "roleArn": "arn:aws:iam::111122223333:role/role-name",
  "tags": {},
  "experimentOptions": {
    "accountTargeting": "single-account",
    "emptyTargetResolutionMode": "fail"
  }
}

```

Experimente um exemplo de permissões de função

A permissão a seguir permite que você execute as `aws:kinesis:stream-expired-iterator-exception` ações `aws:kinesis:stream-provisioned-throughput-exception` e em um stream específico, afetando 50% das solicitações.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:InjectApiError",
      "Resource": "*"
      "Condition": {
        "ForAllValues:StringEquals": {
          "kinesis:FisActionId": [
            "aws:kinesis:stream-provisioned-throughput-exception",
            "aws:kinesis:stream-expired-iterator-exception"
          ],
          "kinesis:FisTargetArns": [
            "arn:aws:kinesis:us-east-1:111122223333:stream/stream-name"
          ],
        },
        "NumericEquals": {
          "kinesis:FisInjectPercentage": "50"
        }
      }
    },
    {
      "Action": [
        "kinesis:DescribeStreamSummary",

```

```
    ],  
    "Resource": "*",  
    "Effect": "Allow"  
  }  
]  
}
```

Gerenciando seus AWS experimentos FIS

AWS O FIS permite que você realize experimentos de injeção de falhas em suas AWS cargas de trabalho. Para começar, crie um [modelo de experimento](#). Depois de criar um modelo de experimento, você pode usá-lo para iniciar um experimento.

Um experimento é concluído quando uma das seguintes condições ocorrer:

- Todas as [ações](#) no modelo foram concluídas com êxito.
- Uma [condição de parada](#) é acionada.
- Uma ação não pode ser concluída devido a um erro. Por exemplo, se o [destino](#) não puder ser encontrado.
- O experimento é [interrompido manualmente](#).

Você não pode retomar um experimento interrompido ou malsucedido. Você também não pode repetir um experimento concluído. No entanto, você pode iniciar um novo experimento usando o mesmo modelo de experimento. Opcionalmente, você pode atualizar o modelo do experimento antes de especificá-lo novamente em um novo experimento.

Tarefas

- [Iniciar um experimento](#)
- [Visualizar seus experimentos](#)
- [Marcar um experimento](#)
- [Interromper um experimento](#)
- [Listar destinos resolvidos](#)

Iniciar um experimento

Você inicia um experimento a partir de um modelo de experimento. Para obter mais informações, consulte [Iniciar um experimento a partir de um modelo](#).

Você pode programar seus experimentos como uma tarefa única ou tarefas recorrentes usando o Amazon EventBridge. Para obter mais informações, consulte [Tutorial: programar um experimento recorrente](#).

Você pode monitorar seu experimento usando qualquer um dos seguintes recursos:

- Veja seus experimentos no console do AWS FIS. Para obter mais informações, consulte [Visualizar seus experimentos](#).
- Veja CloudWatch as métricas da Amazon para os recursos-alvo em seus experimentos ou visualize as métricas de uso do AWS FIS. Para obter mais informações, consulte [Monitore usando CloudWatch](#).
- Habilite o registro de experimentos para capturar informações detalhadas sobre seu experimento à medida que ele é executado. Para obter mais informações, consulte [Registro em log do experimento](#).

Visualizar seus experimentos

Você pode ver o progresso de um experimento em execução e ver os experimentos que foram concluídos, interrompidos ou falharam.

Experimentos interrompidos, concluídos e malsucedidos são automaticamente removidos da sua conta após 120 dias.

Para visualizar os experimentos usando o console

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Experimentos.
3. Escolha o ID do experimento para abrir sua página de detalhes.
4. Siga um dos procedimentos abaixo:
 - Marque Detalhes, Estado para ver o [estado do experimento](#).
 - Escolha a guia Ações para obter informações sobre as ações do experimento.
 - Escolha a guia Destinos para obter informações sobre os destinos do experimento.
 - Escolha a guia Cronograma para uma representação visual das ações com base nos horários de início e término.

Para visualizar os experimentos usando a CLI

Use o comando [list-experiments](#) para obter uma lista de experimentos e use o comando [get-experiment](#) para obter informações sobre um experimento específico.

Estados do experimento

Um experimento pode estar em um dos seguintes estados:

- pendente – O experimento está pendente.
- iniciando – O experimento está em preparação para início.
- em execução – O experimento está sendo executado.
- concluído – Todas as ações do experimento foram concluídas com sucesso.
- interrompendo – A condição de parada foi acionada ou o experimento foi interrompido manualmente.
- parada – Todas as ações em execução ou pendentes no experimento são interrompidas.
- falha – O experimento falhou devido a um erro, como permissões insuficientes ou sintaxe incorreta.
- cancelado: o experimento foi interrompido ou impedido de iniciar devido a uma medida de segurança acionada.

Estados da ação

Uma ação pode estar em um dos seguintes estados:

- pendente – A ação está pendente, seja porque o experimento não foi iniciado ou porque a ação deve ser iniciada posteriormente no experimento.
- iniciando – A ação está em preparação para início.
- em execução – A ação está sendo executada.
- concluída – A ação foi concluída com sucesso.
- cancelada – O experimento foi interrompido antes do início da ação.
- ignorado: a ação foi ignorada.
- interrompendo – A ação está parando.
- parada – Todas as ações em execução ou pendentes no experimento são interrompidas.
- falha – A ação falhou devido a um erro, como permissões insuficientes ou sintaxe incorreta.

Marcar um experimento

Você pode aplicar tags a experimentos para ajudá-lo a organizá-los. Você também pode implementar [políticas do IAM baseadas em tags](#) para controlar o acesso aos experimentos.

Para marcar um experimentos usando o console

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Experimentos.
3. Selecione o experimento e escolha Ações, Gerenciar tags.
4. Para adicionar uma nova tag, escolha Adicionar nova tag e especifique uma chave e o valor.

Para remover uma tag, selecione Remover para a tag.
5. Escolha Salvar.

Para marcar um experimentos usando a CLI

Use o comando [tag-resource](#).

Interromper um experimento

É possível interromper um experimento em execução a qualquer momento. Quando você interrompe um experimento, todas as ações posteriores que não foram concluídas para uma ação são concluídas antes que o experimento seja interrompido. Você não pode retomar um experimento interrompido.

Para marcar um experimento usando o console

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Experimentos.
3. Selecione o experimento e escolha Parar experimento.
4. Na caixa de diálogo de confirmação, escolha Interromper experimento.

Para interromper um experimentos usando a CLI

Use o comando [stop-experiment](#).

Listar destinos resolvidos

Você pode visualizar as informações dos destinos resolvidos de um experimento após o término da resolução deles.

Para visualizar destinos resolvidos usando o console

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Experimentos.
3. Selecione o experimento e escolha Relatório.
4. Veja as informações do destino resolvido em Recursos.

Para visualizar destinos resolvidos usando a CLI

Use o comando [list-experiment-resolved-targets](#).

Tutoriais para o serviço de injeção de AWS falhas

Os tutoriais a seguir mostram como criar e executar experimentos usando o AWS Fault Injection Service (AWS FIS).

Tutoriais

- [Tutorial: pare e comece a testar a instância usando o AWS FIS](#)
- [Tutorial: Execute o estresse da CPU em uma instância usando o AWS FIS](#)
- [Tutorial: Teste as interrupções da instância Spot usando o AWS FIS](#)
- [Tutorial: simular um evento de conectividade](#)
- [Tutorial: programar um experimento recorrente](#)

Tutorial: pare e comece a testar a instância usando o AWS FIS

Você pode usar o AWS AWS Fault Injection Service (FIS) para testar como seus aplicativos lidam com a parada e a inicialização da instância. Use este tutorial para criar um modelo de experimento que usa a `aws:ec2:stop-instances` ação AWS FIS para interromper uma instância e depois uma segunda instância.

Pré-requisitos

Para concluir este tutorial, certifique-se de fazer o seguinte:

- Inicie duas EC2 instâncias de teste na sua conta. Depois de iniciar suas instâncias, observe IDs as duas instâncias.
- Crie uma função do IAM que permita que o serviço AWS FIS execute a `aws:ec2:stop-instances` ação em seu nome. Para obter mais informações, consulte [Funções do IAM para experimentos AWS do FIS](#).
- Certifique-se de ter acesso ao AWS FIS. Para obter mais informações, consulte [Exemplos de políticas do AWS FIS](#).

Etapa 1: criar um modelo de experimento

Crie o modelo de experimento usando o console AWS FIS. No modelo, você especifica duas ações que serão executadas sequencialmente por três minutos cada. A primeira ação interrompe uma das

instâncias de teste, que o AWS FIS escolhe aleatoriamente. A segunda ação interrompe as duas instâncias de teste.

Criar um modelo de experimento

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Modelos de experimento.
3. Escolha Criar modelo de experimento.
4. Para a Etapa 1, Especificar detalhes do modelo, faça o seguinte:
 - a. Em Descrição e nome, insira uma descrição para o modelo, como Amazon S3 Network Disrupt Connectivity.
 - b. Escolha Avançar e vá para a Etapa 2, Especifique ações e alvos.
5. Em Ações, faça o seguinte:
 - a. Selecione Adicionar ação.
 - b. Insira um nome para a ação. Por exemplo, digite **stopOneInstance**.
 - c. Em Tipo de ação, escolha aws:ec2:stop-instances.
 - d. Para Target, mantenha o alvo que o AWS FIS cria para você.
 - e. Em Parâmetros de ação, Iniciar instâncias após a duração, especifique 3 minutos (PT3M).
 - f. Escolha Salvar.
6. Em Destinos, faça o seguinte:
 - a. Escolha Editar para o destino que o AWS FIS criou automaticamente para você na etapa anterior.
 - b. Substitua o nome padrão por um nome mais descritivo. Por exemplo, digite **oneRandomInstance**.
 - c. Verifique se o Tipo de recurso é aws:ec2:instance.
 - d. Em Método de destino, escolha Recurso e IDs, em seguida, escolha IDs uma das duas instâncias de teste.
 - e. Em Modo de seleção, escolha Contagem. Em Número de recursos, insira **1**.
 - f. Escolha Salvar.
7. Selecione Adicionar destino e faça o seguinte:
 - a. Insira um nome para o destino. Por exemplo, digite **bothInstances**.

- b. Em Tipo de recurso, escolha `aws:ec2:instance`.
 - c. Em Método de destino, escolha Recurso e IDs, em seguida, escolha IDs uma das duas instâncias de teste.
 - d. Em Modo de seleção, escolha Todos.
 - e. Escolha Salvar.
8. Na seção Ações, escolha Adicionar ação. Faça o seguinte:
 - a. Em Nome, insira um nome para a ação. Por exemplo, digite **stopBothInstances**.
 - b. Em Tipo de ação, escolha `aws:ec2:stop-instances`.
 - c. Em Começar depois, escolha a primeira ação que você adicionou (**stopOneInstance**).
 - d. Em Destino, escolha o segundo destino que você adicionou (**bothInstances**).
 - e. Em Parâmetros de ação, Iniciar instâncias após a duração, especifique 3 minutos (PT3M).
 - f. Escolha Salvar.
9. Escolha Avançar para ir para a Etapa 3, Configurar o acesso ao serviço.
10. Em Acesso ao serviço, escolha Usar um perfil do IAM existente e, em seguida, escolha o perfil do IAM que você criou conforme descrito nos pré-requisitos deste tutorial. Se sua função não for exibida, verifique se ela tem a relação de confiança necessária. Para obter mais informações, consulte [the section called “Perfil do experimento”](#).
11. Escolha Avançar para ir para a Etapa 4, Definir configurações opcionais.
12. (Opcional) Em Tags, escolha Adicionar nova tag e especifique uma chave de tag e um valor de tag. As tags que você adiciona são aplicadas ao seu modelo de experimento, não aos experimentos que são executados usando o modelo.
13. Escolha Avançar para ir para a Etapa 5, Revisar e criar.
14. Revise o modelo e escolha Criar modelo de experimento. Quando a confirmação for solicitada, insira `create`, em seguida, escolha Criar modelo de experimento.

(Opcional) Para visualizar o modelo de experimento JSON

Escolha a guia Exportar. Este é um exemplo do JSON criado pelo procedimento de console anterior.

```
{
  "description": "Test instance stop and start",
  "targets": {
    "bothInstances": {
      "resourceType": "aws:ec2:instance",
```

```
    "resourceArns": [
      "arn:aws:ec2:region:123456789012:instance/instance_id_1",
      "arn:aws:ec2:region:123456789012:instance/instance_id_2"
    ],
    "selectionMode": "ALL"
  },
  "oneRandomInstance": {
    "resourceType": "aws:ec2:instance",
    "resourceArns": [
      "arn:aws:ec2:region:123456789012:instance/instance_id_1",
      "arn:aws:ec2:region:123456789012:instance/instance_id_2"
    ],
    "selectionMode": "COUNT(1)"
  }
},
"actions": {
  "stopBothInstances": {
    "actionId": "aws:ec2:stop-instances",
    "parameters": {
      "startInstancesAfterDuration": "PT3M"
    },
    "targets": {
      "Instances": "bothInstances"
    },
    "startAfter": [
      "stopOneInstance"
    ]
  },
  "stopOneInstance": {
    "actionId": "aws:ec2:stop-instances",
    "parameters": {
      "startInstancesAfterDuration": "PT3M"
    },
    "targets": {
      "Instances": "oneRandomInstance"
    }
  }
},
"stopConditions": [
  {
    "source": "none"
  }
],
"roleArn": "arn:aws:iam::123456789012:role/AllowFISEC2Actions",
```

```
"tags": {}  
}
```

Etapa 2: iniciar o experimento

Quando terminar de criar seu modelo de experimento, você poderá usá-lo para iniciar um experimento.

Para iniciar um experimento

1. Você deve estar na página de detalhes do modelo de experimento que acabou de criar. Caso contrário, escolha Modelos de experimento e selecione o ID do modelo de experimento para abrir a página de detalhes.
2. Escolha Start experiment (Iniciar experimento).
3. (Opcional) Para adicionar uma tag ao experimento, escolha Adicionar nova tag e insira uma chave de tag e um valor de tag.
4. Escolha Start experiment (Iniciar experimento). Quando a confirmação for solicitada, insira **start** e escolha Iniciar experimento.

Etapa 3: acompanhar o progresso do experimento

Você pode acompanhar o progresso de um experimento em andamento até que ele seja concluído, interrompido ou falhe.

Para acompanhar o progresso de um experimento

1. Você deve estar na página de detalhes do modelo de experimento que acabou de iniciar. Caso contrário, escolha Experimentos e selecione o ID do experimento para abrir a página de detalhes.
2. Para ver o estado do experimento, marque Estado no painel Detalhes. Para obter mais informações, consulte [Estados do experimento](#).
3. Quando o estado do experimento for Em execução, vá para a próxima etapa.

Etapa 4: verificar o resultado do experimento

Você pode verificar se as instâncias foram interrompidas e iniciadas pelo experimento conforme o esperado.

Para verificar o resultado do experimento

1. Abra o EC2 console da Amazon <https://console.aws.amazon.com/ec2/> em uma nova guia ou janela do navegador. Isso permite que você continue acompanhando o progresso do experimento no console AWS FIS enquanto visualiza o resultado do experimento no EC2 console da Amazon.
2. No painel de navegação, escolha Instances (Instâncias).
3. Quando o estado da primeira ação muda de Pendente para Executando (console AWS FIS), o estado de uma das instâncias de destino muda de Executando para Parada (EC2 console da Amazon).
4. Depois de três minutos, o estado da primeira ação muda para Concluído, o estado da segunda ação muda para Em execução e o estado da outra instância de destino muda para Interrompido.
5. Depois de três minutos, o estado da segunda ação muda para Concluído, o estado das instâncias de destino muda para Em execução e o estado do experimento muda para Concluído.

Etapa 5: limpar

Se você não precisar mais das EC2 instâncias de teste que criou para esse experimento, poderá encerrá-las.

Para encerrar as instâncias

1. Abra o EC2 console da Amazon em <https://console.aws.amazon.com/ec2/>.
2. No painel de navegação, escolha Instances (Instâncias).
3. Selecione ambas as instâncias de teste e escolha Instance state (Estado da instância) e Terminate instance (Encerrar instância).
4. Quando a confirmação for solicitada, escolha Terminate (Encerrar).

Se você não precisar mais do modelo de experimento, poderá excluí-lo.

Para excluir um modelo de experimento usando o console AWS FIS

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Modelos de experimento.
3. Selecione o modelo do experimento e escolha Ações, Excluir modelo de experimento.
4. Quando a confirmação for solicitada, insira **delete** e escolha Excluir modelo de experimento.

Tutorial: Execute o estresse da CPU em uma instância usando o AWS FIS

Você pode usar o AWS AWS Fault Injection Service (FIS) para testar como seus aplicativos lidam com o estresse da CPU. Use este tutorial para criar um modelo de experimento que usa o AWS FIS para executar um documento SSM pré-configurado que executa o estresse da CPU em uma instância. O tutorial usa uma condição de parada para interromper o experimento quando a utilização da CPU da instância excede um limite configurado.

Para obter mais informações, consulte [the section called “Documentos AWS FIS SSM pré-configurados”](#).

Pré-requisitos

Antes de usar o AWS FIS para executar o estresse da CPU, preencha os seguintes pré-requisitos.

Criar um perfil do IAM

Crie uma função e anexe uma política que permita que o AWS FIS use a `aws:ssm:send-command` ação em seu nome. Para obter mais informações, consulte [Funções do IAM para experimentos AWS do FIS](#).

Verifique o acesso ao AWS FIS

Certifique-se de ter acesso ao AWS FIS. Para obter mais informações, consulte [Exemplos de políticas do AWS FIS](#).

Preparar uma instância do EC2 de teste

- Execute uma instância do EC2 usando o Amazon Linux 2 ou o Ubuntu, conforme exigido pelos documentos do SSM pré-configurados.
- A instância deve ser gerenciada pelo SSM. Para verificar se a instância é gerenciada pelo SSM, abra o [console do Fleet Manager](#). Se a instância não for gerenciada pelo SSM, verifique se o agente SSM está instalado e se a instância tem uma função do IAM vinculada à política da Amazon SSMManaged InstanceCore. Para verificar o Atendente SSM instalado, conecte-se à sua instância e execute o comando a seguir.

Amazon Linux 2

```
yum info amazon-ssm-agent
```

Ubuntu

```
apt list amazon-ssm-agent
```

- Habilita o monitoramento detalhado da instância. Isso fornece dados em períodos de 1 minuto, mediante pagamento adicional. Selecione a instância e escolha Ações, Monitorar e solucionar problemas, Gerenciar monitoramento detalhado.

Etapa 1: criar um CloudWatch alarme para uma condição de parada

Configure um CloudWatch alarme para que você possa interromper o experimento se a utilização da CPU exceder o limite especificado. O procedimento a seguir define o limite de 50% de utilização da CPU para a instância de destino. Para obter mais informações, consulte [Condições de parada](#).

Para criar um alarme que indique quando a utilização da CPU excede um limite

1. Abra o console do Amazon EC2 em <https://console.aws.amazon.com/ec2/>.
2. No painel de navegação, escolha Instâncias.
3. Selecione a instância de destino e escolha Ações, Monitorar e solucionar problemas, Gerenciar CloudWatch alarmes.
4. Em Notificação de alarme, use a opção para desativar as notificações do Amazon SNS.
5. Em Limites de alarme, use as seguintes configurações:
 - Agrupar amostras por: Máximo
 - Tipo de dados a serem amostrados: utilização da CPU
 - Porcentagem: **50**
 - Período: **1 Minute**
6. Quando terminar de configurar o alarme, escolha Criar.

Etapa 2: criar um modelo de experimento

Crie o modelo de experimento usando o console AWS FIS. No modelo, você especifica a seguinte ação a ser executada: [AWSFIS-Runaws:ssm:send-command/](#) -CPU-stress.

Criar um modelo de experimento

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Modelos de experimento.
3. Escolha Criar modelo de experimento.
4. Para a Etapa 1, Especificar detalhes do modelo, faça o seguinte:
 - a. Em Descrição e nome, insira uma descrição para o modelo.
 - b. Escolha Avançar e vá para a Etapa 2, Especifique ações e alvos.
5. Em Ações, faça o seguinte:
 - a. Selecione Adicionar ação.
 - b. Insira um nome para a ação. Por exemplo, digite **runCpuStress**.
 - c. Em Tipo de ação, escolha aws:ssm:send-command/ -CPU-stress AWSFIS-Run. Isso adiciona automaticamente o ARN do documento do SSM ao ARN do documento.
 - d. Para Target, mantenha o alvo que o AWS FIS cria para você.
 - e. Em Parâmetros de ação, Parâmetros do documento, insira o seguinte:

```
 {"DurationSeconds": "120"} 
```
 - f. Em Parâmetros de ação, Duração, especifique 5 minutos (PT5M).
 - g. Escolha Salvar.
6. Em Destinos, faça o seguinte:
 - a. Escolha Editar para o destino que o AWS FIS criou automaticamente para você na etapa anterior.
 - b. Substitua o nome padrão por um nome mais descritivo. Por exemplo, digite **testInstance**.
 - c. Verifique se o Tipo de recurso é aws:ec2:instance.
 - d. Em Método de destino, escolha Recurso e IDs, em seguida, escolha o ID da instância de teste.
 - e. Em Modo de seleção, escolha Todos.
 - f. Escolha Salvar.
7. Escolha Avançar para ir para a Etapa 3, Configurar o acesso ao serviço.

8. Em Acesso ao serviço, escolha Usar um perfil do IAM existente e, em seguida, escolha o perfil do IAM que você criou conforme descrito nos pré-requisitos deste tutorial. Se sua função não for exibida, verifique se ela tem a relação de confiança necessária. Para obter mais informações, consulte [the section called “Perfil do experimento”](#).
9. Escolha Avançar para ir para a Etapa 4, Definir configurações opcionais.
10. Em Condições de parada, selecione o CloudWatch alarme que você criou na Etapa 1.
11. (Opcional) Em Tags, escolha Adicionar nova tag e especifique uma chave de tag e um valor de tag. As tags que você adiciona são aplicadas ao seu modelo de experimento, não aos experimentos que são executados usando o modelo.
12. Escolha Avançar para ir para a Etapa 5, Revisar e criar.
13. Revise o modelo e escolha Criar modelo de experimento. Quando a confirmação for solicitada, insira ecreate, em seguida, escolha Criar modelo de experimento.

(Opcional) Para visualizar o modelo de experimento JSON

Escolha a guia Exportar. Este é um exemplo do JSON criado pelo procedimento de console anterior.

```
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": [
        "arn:aws:ec2:region:123456789012:instance/instance_id"
      ],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:region::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\": \"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  }
}
```

```
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "arn:aws:cloudwatch:region:123456789012:alarm:awsec2-instance_id-
GreaterThanOrEqualToThreshold-CPUUtilization"
    }
  ],
  "roleArn": "arn:aws:iam::123456789012:role/AllowFISSMActions",
  "tags": {}
}
```

Etapa 3: iniciar o experimento

Quando terminar de criar seu modelo de experimento, você poderá usá-lo para iniciar um experimento.

Para iniciar um experimento

1. Você deve estar na página de detalhes do modelo de experimento que acabou de criar. Caso contrário, escolha Modelos de experimento e selecione o ID do modelo de experimento para abrir a página de detalhes.
2. Escolha Start experiment (Iniciar experimento).
3. (Opcional) Para adicionar uma tag ao experimento, escolha Adicionar nova tag e insira uma chave de tag e um valor de tag.
4. Escolha Start experiment (Iniciar experimento). Quando a confirmação for solicitada, insira **start**. Escolha Start experiment (Iniciar experimento).

Etapa 4: acompanhar o progresso do experimento

Você pode acompanhar o progresso de um experimento em andamento até que ele seja concluído, interrompido ou falhe.

Para acompanhar o progresso de um experimento

1. Você deve estar na página de detalhes do modelo de experimento que acabou de iniciar. Caso contrário, escolha Experimentos e selecione o ID do modelo de experimento para abrir a página de detalhes do experimento.

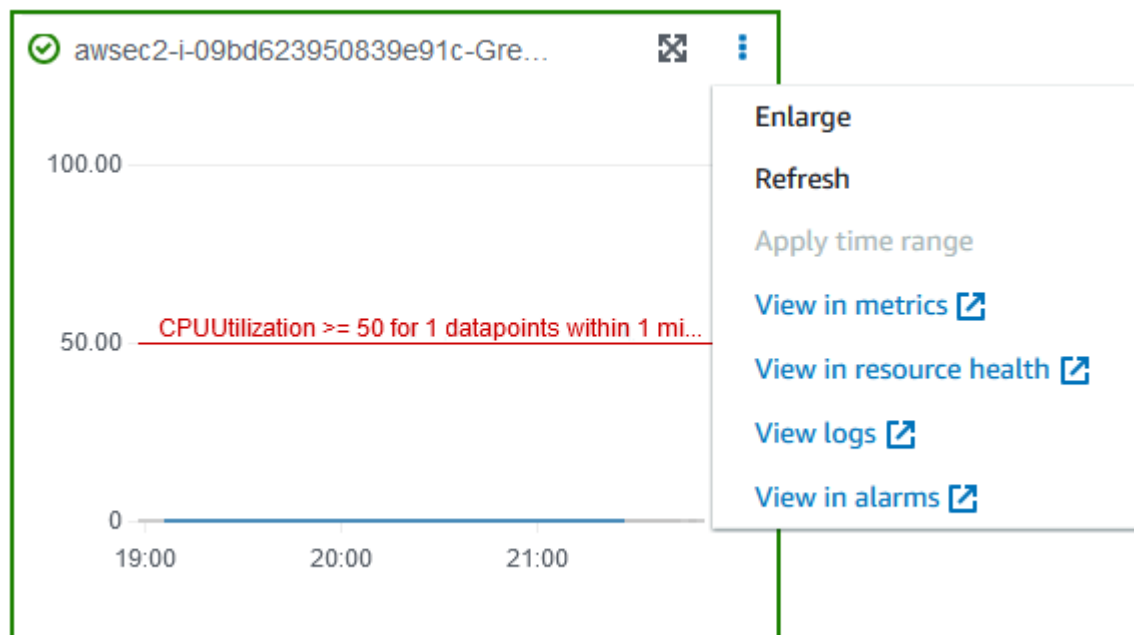
2. Para ver o estado do experimento, marque Estado no painel Detalhes. Para obter mais informações, consulte [Estados do experimento](#).
3. Quando o estado do experimento for Em execução, vá para a próxima etapa.

Etapa 5: verificar os resultados do experimento

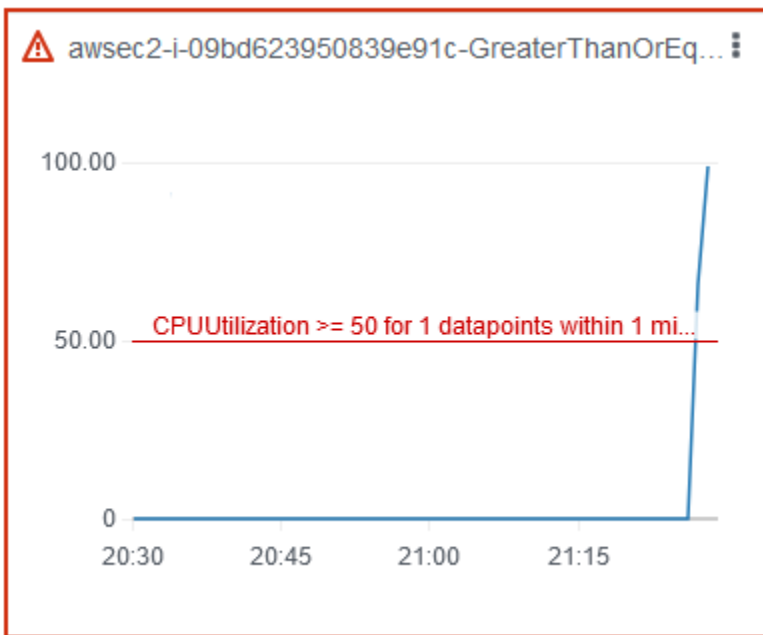
Você pode monitorar a utilização da CPU da sua instância enquanto o experimento está sendo executado. Quando a utilização da CPU atinge o limite, o alarme é acionado e o experimento é interrompido pela condição de parada.

Para verificar os resultados do experimento

1. Escolha a guia Condições de parada. A borda verde e o ícone de marca de seleção verde indicam que o estado inicial do alarme é OK. A linha vermelha indica o limite do alarme. Se você preferir um gráfico mais detalhado, escolha Ampliar no menu do widget.



2. Quando a utilização da CPU excede o limite, a borda vermelha e o ícone de ponto de exclamação vermelho na guia Condições de parada indicam que o estado do alarme foi alterado para ALARM. No painel Detalhes, o estado do experimento é Interrompido. Se você selecionar o estado, a mensagem exibida será “Experimento interrompido pela condição de parada”.



3. Quando a utilização da CPU diminui abaixo do limite, a borda verde e o ícone de marca de seleção verde indicam que o estado do alarme mudou para OK.
4. (Opcional) Escolha Exibir em alarmes no menu do widget. Isso abre a página de detalhes do alarme no CloudWatch console, onde você pode obter mais detalhes sobre o alarme ou editar as configurações do alarme.

Etapa 6: limpar

Se a instância de teste do EC2 criada durante o experimento não for mais necessária, você poderá encerrá-la.

Para terminar as instâncias

1. Abra o console do Amazon EC2 em <https://console.aws.amazon.com/ec2/>.
2. No painel de navegação, escolha Instâncias.
3. Selecione as instâncias de teste e escolha Estado da instância e Encerrar instância.
4. Quando a confirmação for solicitada, escolha Terminate (Encerrar).

Se você não precisar mais do modelo de experimento, poderá excluí-lo.

Para excluir um modelo de experimento usando o console AWS FIS

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Modelos de experimento.
3. Selecione o modelo do experimento e escolha Ações, Excluir modelo de experimento.
4. Quando a confirmação for solicitada, insira **delete** e escolha Excluir modelo de experimento.

Tutorial: Teste as interrupções da instância Spot usando o AWS FIS

As instâncias spot usam a EC2 capacidade ociosa que está disponível, com um desconto de até 90% em comparação com os preços sob demanda. No entanto, a Amazon EC2 pode interromper suas Instâncias Spot quando precisar da capacidade de volta. Ao usar instâncias spot, é preciso estar preparado para possíveis interrupções. Para obter mais informações, consulte [Interrupções de instâncias spot](#) no Guia do EC2 usuário da Amazon.

Você pode usar o AWS AWS Fault Injection Service (FIS) para testar como seus aplicativos lidam com uma interrupção de instância spot. Use este tutorial para criar um modelo de experimento que usa a `aws:ec2:send-spot-instance-interruptions` ação AWS FIS para interromper uma de suas instâncias spot.

Como alternativa, para iniciar o experimento usando o EC2 console da Amazon, consulte [Iniciar uma interrupção de instância spot no](#) Guia do usuário da Amazon EC2 .

Pré-requisitos

Antes de usar o AWS FIS para interromper uma instância spot, preencha os seguintes pré-requisitos.

1. Criar um perfil do IAM

Crie uma função e anexe uma política que permita que o AWS FIS execute a `aws:ec2:send-spot-instance-interruptions` ação em seu nome. Para obter mais informações, consulte [Funções do IAM para experimentos AWS do FIS](#).

2. Verifique o acesso ao AWS FIS

Certifique-se de ter acesso ao AWS FIS. Para obter mais informações, consulte [Exemplos de políticas do AWS FIS](#).

3. (Opcional) Criar uma solicitação de instância spot

Se você quiser usar uma nova instância spot para esse experimento, use o comando [run-instances](#) para solicitar uma instância spot. O padrão é encerrar as Instâncias spot que são interrompidas. Se você definir o comportamento de interrupção como `stop`, também deverá definir o tipo como `persistent`. Para este tutorial, não defina o comportamento de interrupção como `hibernate`, pois o processo de hibernação começa imediatamente.

```
aws ec2 run-instances \  
  --image-id ami-0ab193018fEXAMPLE \  
  --instance-type "t2.micro" \  
  --count 1 \  
  --subnet-id subnet-1234567890abcdef0 \  
  --security-group-ids sg-111222333444aaab \  
  --instance-market-options file://spot-options.json \  
  --query Instances[*].InstanceId
```

Este é um exemplo do arquivo `spot-options.json`.

```
{  
  "MarketType": "spot",  
  "SpotOptions": {  
    "SpotInstanceType": "persistent",  
    "InstanceInterruptionBehavior": "stop"  
  }  
}
```

A opção `--query` no comando de exemplo faz com que o comando retorne somente o ID da instância spot. O seguinte é um exemplo de saída.

```
[  
  "i-0abcdef1234567890"  
]
```

4. Adicione uma tag para que o AWS FIS possa identificar a Instância Spot de destino

Use o comando [create-tags](#) para adicionar a tag `Name=interruptMe` para sua Instância Spot de destino.

```
aws ec2 create-tags \  
  --tags
```

```
--resources i-0abcdef1234567890 \  
--tags Key=Name,Value=interruptMe
```

Etapa 1: criar um modelo de experimento

Crie o modelo de experimento usando o console AWS FIS. No modelo, você especifica a ação que será executada. A ação interrompe a instância spot com a tag especificada. Se houver mais de uma instância spot com a tag, o AWS FIS escolherá uma delas aleatoriamente.

Para criar um modelo de experimento

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Modelos de experimento.
3. Escolha Criar modelo de experimento.
4. Para a Etapa 1, Especificar detalhes do modelo, faça o seguinte:
 - a. Em Descrição e nome, insira uma descrição e um nome para o modelo.
 - b. Escolha Avançar e vá para a Etapa 2, Especifique ações e alvos.
5. Em Ações, faça o seguinte:
 - a. Selecione Adicionar ação.
 - b. Insira um nome para a ação. Por exemplo, digite **interruptSpotInstance**.
 - c. Em Tipo de ação, escolha aws:ec2:. send-spot-instance-interruptions
 - d. Para Target, mantenha o alvo que o AWS FIS cria para você.
 - e. Para Parâmetros de ação, Duração antes da interrupção, especifique 2 minutos (PT2M).
 - f. Escolha Salvar.
6. Em Destinos, faça o seguinte:
 - a. Escolha Editar para o destino que o AWS FIS criou automaticamente para você na etapa anterior.
 - b. Substitua o nome padrão por um nome mais descritivo. Por exemplo, digite **oneSpotInstance**.
 - c. Verifique se o Tipo de recurso é aws:ec2:spot-instance.
 - d. Em Método de destino, escolha Tags, filtros e parâmetros de recursos.

- e. Em Tags de recursos, escolha Adicionar nova tag e insira da chave da tag e o valor da tag. Use a tag que você adicionou à instância spot para interromper, conforme descrito nos Pré-requisitos deste tutorial.
 - f. Em Filtros de recursos, escolha Adicionar novo filtro e insira **State.Name** como o caminho e **running** como o valor.
 - g. Em Modo de seleção, escolha Contagem. Em Número de recursos, insira **1**.
 - h. Escolha Salvar.
7. Escolha Avançar para ir para a Etapa 3, Configurar o acesso ao serviço.
 8. Em Acesso ao serviço, escolha Usar um perfil do IAM existente e, em seguida, escolha o perfil do IAM que você criou conforme descrito nos pré-requisitos deste tutorial. Se sua função não for exibida, verifique se ela tem a relação de confiança necessária. Para obter mais informações, consulte [the section called “Perfil do experimento”](#).
 9. Escolha Avançar para ir para a Etapa 4, Definir configurações opcionais.
 10. (Opcional) Em Tags, escolha Adicionar nova tag e especifique uma chave de tag e um valor de tag. As tags que você adiciona são aplicadas ao seu modelo de experimento, não aos experimentos que são executados usando o modelo.
 11. Escolha Avançar para ir para a Etapa 5, Revisar e criar.
 12. Revise o modelo e escolha Criar modelo de experimento. Quando a confirmação for solicitada, insira ecreate, em seguida, escolha Criar modelo de experimento.

(Opcional) Para visualizar o modelo de experimento JSON

Escolha a guia Exportar. Este é um exemplo do JSON criado pelo procedimento de console anterior.

```
{
  "description": "Test Spot Instance interruptions",
  "targets": {
    "oneSpotInstance": {
      "resourceType": "aws:ec2:spot-instance",
      "resourceTags": {
        "Name": "interruptMe"
      },
    },
    "filters": [
      {
        "path": "State.Name",
        "values": [
```

```

        "running"
      ]
    }
  ],
  "selectionMode": "COUNT(1)"
}
},
"actions": {
  "interruptSpotInstance": {
    "actionId": "aws:ec2:send-spot-instance-interruptions",
    "parameters": {
      "durationBeforeInterruption": "PT2M"
    },
    "targets": {
      "SpotInstances": "oneSpotInstance"
    }
  }
},
"stopConditions": [
  {
    "source": "none"
  }
],
"roleArn": "arn:aws:iam::123456789012:role/AllowFISSpotInterruptionActions",
"tags": {
  "Name": "my-template"
}
}

```

Etapa 2: iniciar o experimento

Quando terminar de criar seu modelo de experimento, você poderá usá-lo para iniciar um experimento.

Para iniciar um experimento

1. Você deve estar na página de detalhes do modelo de experimento que acabou de criar. Caso contrário, escolha Modelos de experimento e selecione o ID do modelo de experimento para abrir a página de detalhes.
2. Escolha Start experiment (Iniciar experimento).
3. (Opcional) Para adicionar uma tag ao experimento, escolha Adicionar nova tag e insira uma chave de tag e um valor de tag.

4. Escolha Start experiment (Iniciar experimento). Quando a confirmação for solicitada, insira **start** e escolha Iniciar experimento.

Etapa 3: acompanhar o progresso do experimento

Você pode acompanhar o progresso de um experimento em andamento até que ele seja concluído, interrompido ou falhe.

Para acompanhar o progresso de um experimento

1. Você deve estar na página de detalhes do modelo de experimento que acabou de iniciar. Caso contrário, escolha Experimentos e selecione o ID do experimento para abrir a página de detalhes.
2. Para ver o estado do experimento, marque Estado no painel Detalhes. Para obter mais informações, consulte [Estados do experimento](#).
3. Quando o estado do experimento for Em execução, vá para a próxima etapa.

Etapa 4: verificar o resultado do experimento

Quando a ação desse experimento for concluída, ocorrerá o seguinte:

- A instância spot de destino recebe uma [recomendação de rebalanceamento da instância](#).
- Um [aviso de interrupção da Instância Spot](#) é emitido dois minutos antes de a Amazon EC2 encerrar ou interromper sua instância.
- Passados os dois minutos, a instância spot é encerrada ou interrompida.
- Uma instância spot que foi interrompida pelo AWS FIS permanece parada até que você a reinicie.

Para verificar se a instância foi interrompida pelo experimento

1. Abra o EC2 console da Amazon em <https://console.aws.amazon.com/ec2/>.
2. No painel de navegação, abra Spot Requests (Solicitações de spot) e Instances (Instâncias) em guias ou janelas separadas do navegador.
3. Em Spot Requests (Solicitações de spot), selecione a solicitação de instância spot. O status inicial é fulfilled. Após a conclusão do experimento, o status muda da seguinte forma:
 - `terminate`: o status se altera para `instance-terminated-by-experiment`.

- `stop`: o status se altera para `marked-for-stop-by-experiment` e depois `instance-stopped-by-experiment`.
4. Em Instances (Instâncias), selecione a instância spot. O status inicial é `Running`. Dois minutos depois que você recebe o aviso de interrupção da instância spot, o status se altera como se segue:
 - `stop`: o status se altera para `Stopping` e depois `Stopped`.
 - `terminate`: o status se altera para `Shutting-down` e depois `Terminated`.

Etapa 5: limpar

Se você criou a instância spot de teste para esse experimento com um comportamento de interrupção de `stop` e não precisa mais dela, é possível cancelar a solicitação da instância spot e encerrar a instância spot.

Para cancelar a solicitação e encerrar a instância usando o AWS CLI

1. Use o [cancel-spot-instance-requests](#) comando para cancelar a solicitação da Instância Spot.

```
aws ec2 cancel-spot-instance-requests --spot-instance-request-ids sir-ksie869j
```

2. Use o comando [terminate-instances](#) para encerrar a instância.

```
aws ec2 terminate-instances --instance-ids i-0abcdef1234567890
```

Se você não precisar mais do modelo de experimento, poderá excluí-lo.

Para excluir um modelo de experimento usando o console AWS FIS

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Modelos de experimento.
3. Selecione o modelo do experimento e escolha Ações, Excluir modelo de experimento.
4. Quando a confirmação for solicitada, insira **delete** e escolha Excluir modelo de experimento.

Tutorial: simular um evento de conectividade

Você pode usar o AWS AWS Fault Injection Service (FIS) para simular uma variedade de eventos de conectividade. O AWS FIS simula eventos de conectividade bloqueando as conexões de rede de uma das seguintes formas:

- `all` – Nega todo o tráfego que entra e sai da sub-rede. Observe que essa opção permite tráfego dentro da sub-rede, incluindo tráfego de e para interfaces de rede na sub-rede.
- `availability-zone` – Nega tráfego intra-VPC de e para sub-redes em outras zonas de disponibilidade.
- `dynamodb` – Nega tráfego de e para o endpoint regional do DynamoDB na região atual.
- `prefix-list` – Nega tráfego de e para a lista de prefixos especificada.
- `s3` – Nega tráfego de e para o endpoint regional do Amazon S3 na região atual.
- `s3express`— Nega tráfego de e para o endpoint zonal do Amazon S3 Express One Zone na AZ das sub-redes de destino. As sub-redes de destino devem residir em AZs onde o S3 Express One Zone está disponível atualmente. Para obter mais informações, consulte [Zonas e regiões de disponibilidade do S3 Express One Zone](#).
- `vpc` – Nega todo o tráfego que entra e sai da VPC.

Use este tutorial para criar um modelo de experimento que usa a `aws:network:disrupt-connectivity` ação AWS FIS para introduzir a perda de conectividade com o Amazon S3 em uma sub-rede de destino.

Tópicos

- [Pré-requisitos](#)
- [Etapa 1: criar um modelo de experimento AWS FIS](#)
- [Etapa 2: fazer ping em um endpoint do Amazon S3](#)
- [Etapa 3: inicie seu experimento AWS FIS](#)
- [Etapa 4: Acompanhe o progresso do seu experimento AWS FIS](#)
- [Etapa 5: verificar interrupção na rede do Amazon S3](#)
- [Etapa 5: limpar](#)

Pré-requisitos

Antes de começar este tutorial, você precisa de uma função com as permissões apropriadas e de uma instância de teste do Amazon EC2: Conta da AWS

Uma função com permissões em seu Conta da AWS

Crie uma função e anexe uma política que permita ao AWS FIS realizar a `aws:network:disrupt-connectivity` ação em seu nome.

Seu perfil do IAM requer a seguinte política:

- [AWSFaultInjectionSimulatorNetworkAccess](#)— Concede permissão de serviço AWS FIS na rede Amazon EC2 e em outros serviços necessários para AWS realizar ações FIS relacionadas à infraestrutura de rede.

Note

Para simplificar, este tutorial usa uma política AWS gerenciada. Para uso em produção, recomendamos que você conceda apenas as permissões mínimas necessárias para seu caso de uso.

Para obter mais informações sobre como criar uma função do IAM, consulte [Funções do IAM para experimentos do AWS FIS \(AWS CLI\)](#) ou Como [criar uma função do IAM \(console\)](#) no Guia do usuário do IAM.

Uma instância de teste do Amazon EC2

Inicie e conecte-se a uma instância de teste do Amazon EC2. Você pode usar o seguinte tutorial para inicializar e se conectar a uma instância do Amazon EC2: [Tutorial: Comece a usar instâncias do Linux do Amazon EC2](#) no Guia do usuário do Amazon EC2.

Etapa 1: criar um modelo de experimento AWS FIS

Crie o modelo de experimento usando o AWS FIS Console de gerenciamento da AWS. Um modelo AWS FIS é composto por ações, metas, condições de parada e uma função experimental. Para obter mais informações sobre como os modelos funcionam, consulte [Modelos de experimento para o AWS FIS](#).

Antes de começar, certifique-se de ter o seguinte pronto:

- Um perfil do IAM com as permissões corretas.
- Uma instância do Amazon EC2.
- O ID da sub-rede da instância do Amazon EC2.

Criar um modelo de experimento

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação à esquerda, escolha Modelos de experimento.
3. Escolha Criar modelo de experimento.
4. Para a Etapa 1, Especificar detalhes do modelo, faça o seguinte:
 - a. Em Descrição e nome, insira uma descrição para o modelo, como Amazon S3 Network Disrupt Connectivity.
 - b. Escolha Avançar e vá para a Etapa 2, Especifique ações e alvos.
5. Em Ação de erro, escolha Adicionar ação.
 - a. Em Nome, insira `disruptConnectivity`.
 - b. Em Tipo de ação, selecione `aws:network:disrupt-connectivity`.
 - c. Em Parâmetros de ação, defina a Duração como 2 minutos.
 - d. Em Escopo, selecione `s3`.
 - e. Na parte superior, escolha Salvar.
6. Em Destinos, você deve ver o destino que foi criado automaticamente. Escolha Editar.
 - a. Verifique se o Tipo de recurso é `aws:ec2:subnet`.
 - b. Em Método de destino, selecione Recurso e IDs, em seguida, escolha a sub-rede que você usou ao criar sua instância do Amazon EC2 [nas](#) etapas de pré-requisitos.
 - c. Verifique se o Modo de seleção é Todos.
 - d. Escolha Salvar.
7. Escolha Avançar para ir para a Etapa 3, Configurar o acesso ao serviço.
8. Em Acesso ao serviço, selecione o perfil do IAM que você criou conforme descrito nos [Pré-requisitos](#) deste tutorial. Se sua função não for exibida, verifique se ela tem a relação de confiança necessária. Para obter mais informações, consulte [the section called “Perfil do experimento”](#).
9. Escolha Avançar para ir para a Etapa 4, Definir configurações opcionais.

10. (Opcional) Em Condições de parada, você pode selecionar um CloudWatch alarme para interromper o experimento se a condição ocorrer. Para obter mais informações, consulte [Condições de parada para o AWS FIS](#).
11. (Opcional) Em Registros, você pode selecionar um bucket do Amazon S3 ou enviar registros CloudWatch para seu experimento.
12. Escolha Avançar para ir para a Etapa 5, Revisar e criar.
13. Revise o modelo e escolha Criar modelo de experimento. Quando a confirmação for solicitada, insira `create`, em seguida, escolha Criar modelo de experimento.

Etapa 2: fazer ping em um endpoint do Amazon S3

Verifique se sua instância do Amazon EC2 é capaz de alcançar um endpoint do Amazon S3.

1. Conecte-se à instância do Amazon EC2 que você criou nas etapas [Pré-requisitos](#).

Para solução de problemas, consulte [Solucionar problemas de conexão com a instância](#) no Guia do usuário do Amazon EC2.

2. Verifique Região da AWS onde sua instância está localizada. Você pode fazer isso no console do Amazon EC2 ou executando o seguinte comando.

```
hostname
```

Por exemplo, se você lançou uma instância do Amazon EC2 em `us-west-2`, verá o seguinte resultado.

```
[ec2-user@ip-172.16.0.0 ~]$ hostname  
ip-172.16.0.0.us-west-2.compute.internal
```

3. Faça ping em um endpoint do Amazon S3 em seu. Região da AWS Substitua *Região da AWS* pela sua região.

```
ping -c 1 s3.Região da AWS.amazonaws.com
```

Para a saída, você deve ver um ping bem-sucedido com 0% de perda de pacotes, conforme mostrado no exemplo a seguir.

```
PING s3.us-west-2.amazonaws.com (x.x.x.x) 56(84) bytes of data.
```

```
64 bytes from s3-us-west-2.amazonaws.com (x.x.x.x: icmp_seq=1 ttl=249 time=1.30 ms)

--- s3.us-west-2.amazonaws.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.306/1.306/1.306/0.000 ms
```

Etapa 3: inicie seu experimento AWS FIS

Comece um experimento com o modelo de experimento que você acabou de criar.

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação à esquerda, escolha Modelos de experimento.
3. Selecione o ID do modelo de experimento que você criou para abrir sua página de detalhes.
4. Escolha Start experiment (Iniciar experimento).
5. (Opcional) Na página de confirmação, adicione tags ao experimento.
6. Na página de confirmação, escolha Iniciar experimento.

Etapa 4: Acompanhe o progresso do seu experimento AWS FIS

Você pode acompanhar o progresso de um experimento em andamento até que ele seja concluído, interrompido ou falhe.

1. Você deve estar na página de detalhes do modelo de experimento que acabou de iniciar. Se não estiver, escolha Experimentos e selecione o ID do experimento para abrir sua página de detalhes.
2. Para ver o estado do experimento, marque Estado no painel Detalhes. Para obter mais informações, consulte [Estados do experimento](#).
3. Quando o estado do experimento for Em execução, vá para a próxima etapa.

Etapa 5: verificar interrupção na rede do Amazon S3

Você pode validar o progresso do experimento fazendo ping no endpoint do Amazon S3.

- Na sua instância do Amazon EC2, faça ping no endpoint do Amazon S3 em sua Região da AWS. Substitua *Região da AWS* pela sua região.

```
ping -c 1 s3.Região da AWS.amazonaws.com
```

Para a saída, você deve ver um ping malsucedido com 100% de perda de pacotes, conforme mostrado no exemplo a seguir.

```
ping -c 1 s3.us-west-2.amazonaws.com
PING s3.us-west-2.amazonaws.com (x.x.x.x) 56(84) bytes of data.

--- s3.us-west-2.amazonaws.com ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Etapa 5: limpar

Se você não precisa mais da instância do Amazon EC2 que você criou para este experimento ou do modelo do AWS FIS, pode removê-los.

Para remover a instância do Amazon EC2.

1. Abra o console do Amazon EC2 em <https://console.aws.amazon.com/ec2/>.
2. No painel de navegação, escolha Instâncias.
3. Selecione a instância e escolha Estado da instância e, em seguida, escolha Encerrar instância.
4. Quando a confirmação for solicitada, escolha Terminate (Encerrar).

Para excluir o modelo de experimento usando o console AWS FIS

1. Abra o console AWS FIS em <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Modelos de experimento.
3. Selecione o modelo do experimento e escolha Ações, Excluir modelo de experimento.
4. Quando a confirmação for solicitada, insira delete e escolha Excluir modelo de experimento.

Tutorial: programar um experimento recorrente

Com o AWS AWS Fault Injection Service (FIS), você pode realizar experimentos de injeção de falhas em suas AWS cargas de trabalho. Esses experimentos são executados em modelos que contêm

uma ou mais ações a serem executadas em alvos específicos. Quando você também usa Amazon EventBridge, você pode programar seus experimentos como uma tarefa única ou tarefas recorrentes.

Use este tutorial para criar um EventBridge cronograma que executa um modelo de experimento AWS FIS a cada 5 minutos.

Tarefas

- [Pré-requisitos](#)
- [Etapa 1: criar uma política e um perfil do IAM](#)
- [Etapa 2: criar um Amazon EventBridge agendador](#)
- [Etapa 3: verificar seu experimento](#)
- [Etapa 4: limpar](#)

Pré-requisitos

Antes de começar este tutorial, AWS é necessário ter um modelo de experimento FIS que você deseja executar em um cronograma. Se você já tem um modelo de experimento de trabalho, anote o ID do modelo e a Região da AWS. Caso contrário, você pode criar um modelo seguindo as instruções em [the section called “Testar início e interrupção da instância”](#), e, em seguida, retornar a este tutorial.

Etapa 1: criar uma política e um perfil do IAM

Para criar uma política e um perfil do IAM.

1. Abra o console do IAM em <https://console.aws.amazon.com/iam/>.
2. No painel de navegação à esquerda, escolha Funções e, em seguida, Criar função.
3. Escolha Política de confiança personalizada e, em seguida, insira o seguinte trecho para permitir que o Amazon EventBridge Agendador assuma a função em seu nome.

JSON

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

```

        "Effect": "Allow",
        "Principal": {
            "Service": "scheduler.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }
]
}

```

Escolha Próximo.

4. Em Adicionar permissões, escolha Criar política.
5. Escolha JSON e, em seguida, insira a política a seguir. Substitua o *your-experiment-template-id* valor pelo ID do modelo do seu experimento nas etapas de pré-requisitos.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "fis:StartExperiment",
      "Resource": [
        "arn:aws:fis:*:*:experiment-template/your-experiment-template-id",
        "arn:aws:fis:*:*:experiment/*"
      ]
    }
  ]
}

```

Você pode restringir o agendador para executar somente modelos de experimentos AWS FIS que tenham um valor de tag específico. Por exemplo, a política a seguir concede a StartExperiment permissão para todos os experimentos do AWS FIS, mas restringe o agendador a executar somente modelos de experimentos marcados. Purpose=Schedule

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "fis:StartExperiment",
      "Resource": "arn:aws:fis:*:*:experiment/*"
    },
    {
      "Effect": "Allow",
      "Action": "fis:StartExperiment",
      "Resource": "arn:aws:fis:*:*:experiment-template/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Purpose": "Schedule"
        }
      }
    }
  ]
}
```

Escolha Próximo: tags.

6. Selecione Próximo: revisar.
7. Em Revisar política, nomeie sua política FIS_RecurringExperiment e escolha Criar política.
8. Em Adicionar permissões, adicione a nova FIS_RecurringExperiment política à sua função e escolha Avançar.
9. Em Nomear, revisar e criar, nomeie a função FIS_RecurringExperiment_role e escolha Criar função.

Etapa 2: criar um Amazon EventBridge agendador

Para criar um Amazon EventBridge Agendador

1. Abra o EventBridge console da Amazon em <https://console.aws.amazon.com/events/>.

2. No painel de navegação esquerdo, escolha Cronogramas.
3. Verifique se você está usando o mesmo modelo Região da AWS de experimento AWS do FIS.
4. Escolha Criar cronograma e preencha o seguinte:
 - Em Nome do cronograma, insira `FIS_recurring_experiment_tutorial`.
 - Em Padrão de programação, selecione Programação recorrente.
 - Em Tipo de programação, selecione Programação baseada em taxa.
 - Em Expressão rate, escolha 5 minutos.
 - Em Janela de horário flexível, selecione Desativado.
 - (Opcional) Em Período de tempo, selecione o seu fuso horário.
 - Escolha Próximo.
5. Em Selecionar destino, escolha Tudo e APIs, em seguida, pesquise AWS FIS.
6. Escolha AWS FIS e, em seguida, selecione StartExperiment.
7. Em Entrada, insira a seguinte carga JSON. Substitua o *your-experiment-template-id* valor pelo ID do modelo do seu experimento. ClientToken é um identificador exclusivo para o programador. Neste tutorial, estamos usando uma palavra-chave de contexto permitida pelo Amazon EventBridge Scheduler. Para obter mais informações, consulte [Adicionar atributos de contexto](#) no Guia EventBridge do usuário da Amazon.

```
{
  "ClientToken": "<aws.scheduler.execution-id>",
  "ExperimentTemplateId": "your-experiment-template-id"
}
```

Escolha Próximo.

8. (Opcional) Em Configurações, você pode definir Política de repetição, Fila de mensagens não entregues (DLQ) e as configurações de Criptografia. Como alternativa, é possível manter os valores padrão.
9. Em Permissões, selecione Usar função existente, e, em seguida, pesquise por `FIS_RecurringExperiment_role`.
10. Escolha Avançar.
11. Em Revisar e criar cronograma, revise os detalhes do programador e escolha Criar cronograma.

Etapa 3: verificar seu experimento

Para verificar se seu experimento AWS FIS foi executado dentro do cronograma

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação esquerdo, escolha Experimentos.
3. Cinco minutos depois de criar sua programação, você deve ver seu experimento sendo executado.

Etapa 4: limpar

Para desativar seu Amazon EventBridge Agendador

1. Abra o EventBridge console da Amazon em <https://console.aws.amazon.com/events/>.
2. No painel de navegação esquerdo, escolha Cronogramas.
3. Selecione seu programador recém-criado e escolha Desativar.

Trabalhando com a biblioteca de AWS FIS cenários

Os cenários definem eventos ou condições que os clientes podem aplicar para testar a resiliência de seus aplicativos, como a interrupção dos recursos computacionais nos quais o aplicativo está sendo executado. Os cenários são criados e de propriedade da AWS e minimizam o trabalho pesado indiferenciado ao fornecer a você um grupo de metas e ações de falha predefinidas (por exemplo, interromper 30% das instâncias em um grupo de escalonamento automático) para deficiências comuns de aplicativos.

Os cenários são fornecidos por meio de uma biblioteca de cenários somente para console e executados usando um modelo de experimento do AWS FIS . Para realizar um experimento usando um cenário, você selecionará o cenário na biblioteca, especificará parâmetros que correspondam aos detalhes do seu workload e o salvará como um modelo de experimento em sua conta.

Tópicos

- [Visualizar um cenário](#)
- [Usar um cenário](#)
- [Exportação de um cenário](#)
- [Referência de cenários](#)

Visualizar um cenário

Para visualizar um cenário usando o console:

1. Abra o AWS FIS console em <https://console.aws.amazon.com/fis/>.
2. No painel de navegação, selecione Biblioteca de cenários.
3. Para ver informações sobre um cenário específico, selecione o cartão de cenário para abrir um painel dividido.
 - Na guia Descrição no painel dividido na parte inferior da página, você pode ver uma breve descrição do cenário. Você também pode encontrar um breve resumo dos pré-requisitos contendo um resumo dos recursos de destino necessários e todas as ações necessárias para preparar os recursos para uso com o cenário. Por fim, você também pode ver informações adicionais sobre os destinos e ações no cenário, bem como a duração prevista quando o experimento for executado com sucesso com as configurações padrão.

- Na guia Conteúdo no painel dividido na parte inferior da página, você pode visualizar uma versão parcialmente preenchida do modelo de experimento que será criado a partir do cenário.
- Na guia Detalhes no painel dividido na parte inferior da página, você pode encontrar uma explicação detalhada de como o cenário é implementado. Isso pode conter informações detalhadas sobre como os aspectos individuais do cenário são aproximados. Onde aplicável, você também pode ler sobre quais métricas usar como condições de parada e fornecer observabilidade para aprender com o experimento. Por fim, você encontrará recomendações sobre como expandir o modelo de experimento resultante.

Usar um cenário

Para usar um cenário usando o console:

1. Abra o AWS FIS console em <https://console.aws.amazon.com/fis/>.
2. No painel de navegação, selecione Biblioteca de cenários.
3. Para ver informações sobre um cenário específico, selecione o cartão de cenário para abrir um painel dividido
4. Para usar o cenário, selecione o cartão de cenário e escolha Criar modelo com cenário.
5. Na exibição Criar modelo de experimento, preencha todos os itens que faltam.
 - a. Alguns cenários permitem que você edite parâmetros que são compartilhados entre várias ações ou alvos. Essa funcionalidade será desativada quando você fizer qualquer alteração no cenário, incluindo alterações pela edição de parâmetros compartilhados. Para usar esse recurso, selecione o botão Editar parâmetros compartilhados. Edite os parâmetros no modal e selecione o botão Salvar.
 - b. Alguns modelos de experimento podem ter parâmetros de ação ou destino ausentes, destacados em cada carta de ação e destino. Selecione o botão Editar para cada cartão, adicione as informações que faltam e selecione o botão Salvar no cartão.
 - c. Todos os modelos exigem uma função de execução de acesso ao serviço. É possível escolher uma função existente ou criar uma função nova para este modelo de experimento.
 - d. Recomendamos definir uma ou mais condições de parada opcionais selecionando um CloudWatch alarme existente da AWS. Saiba mais sobre [Condições de parada para o AWS FIS](#). Se você ainda não tiver um alarme configurado, siga as instruções em [Usando CloudWatch alarmes da Amazon](#) e atualize o modelo do experimento posteriormente.

- e. Recomendamos habilitar registros experimentais opcionais nos CloudWatch registros da Amazon ou em um bucket do Amazon S3. Saiba mais sobre [Registro de experimentos para AWS FIS](#). Se você ainda não tiver os recursos apropriados configurados, poderá atualizar o modelo do experimento posteriormente.
6. Em Criar modelo de experimento, selecione Criar modelo de experimento.
7. Na visualização de modelos de experimento do AWS FIS console, selecione Iniciar experimento. Saiba mais sobre [Gerenciando modelos AWS de experimentos do FIS](#).

Exportação de um cenário

Os cenários são uma experiência somente para console. Embora sejam semelhantes aos modelos de experimentos, os cenários não são modelos de experimentos completos e não podem ser importados diretamente para o AWS FIS. Se você quiser usar cenários como parte de sua própria automação, você pode usar um dos dois caminhos:

1. Siga as etapas [Usar um cenário](#) para criar um modelo de AWS FIS experimento válido e exportar esse modelo.
2. Siga as etapas em [Visualizar um cenário](#) e na etapa 3, na guia Conteúdo, copie e salve o conteúdo do cenário e, em seguida, adicione os parâmetros ausentes manualmente para criar um modelo de experimento válido.

Referência de cenários

Os cenários incluídos na biblioteca de cenários foram projetados para usar [tags](#) sempre que possível e cada cenário descreve as tags necessárias nas seções Pré-requisitos e Como funciona da descrição do cenário. Você pode marcar seus recursos com essas tags predefinidas ou definir suas próprias tags usando a experiência compartilhada de edição de parâmetros (consulte [Usar um cenário](#)).

Esta referência descreve os cenários comuns na biblioteca de cenários do AWS FIS. Você também pode listar os cenários compatíveis usando o console do AWS FIS.

Para obter mais informações, consulte [Trabalhando com a biblioteca de AWS FIS cenários](#).

O AWS FIS é compatível com os seguintes cenários do Amazon EC2. Esses cenários são voltados para instâncias que usam [tags](#). Você pode usar suas próprias tags ou as tags-padrão incluídas no cenário. Alguns desses cenários [usam documentos do SSM](#).

- Estresse do EC2: falha na instância - Explore o efeito de falha de instância ao parar uma ou mais instâncias do EC2.

Instâncias de destino na região atual que tenham uma tag específica anexada. Nesse cenário, interromperemos essas instâncias e as reiniciaremos no final da duração da ação, por padrão, 5 minutos.

- Estresse do EC2: disco - Explore o impacto do aumento da utilização do disco em seu aplicativo baseado em EC2.

Neste cenário, vamos direcionar instâncias do EC2 na região atual que tenham uma tag específica anexada. Nesse cenário, você pode personalizar uma quantidade crescente de utilização do disco injetada nas instâncias do EC2 de destino para a duração da ação, por padrão 5 minutos para cada ação de estresse do disco.

- Estresse do EC2: CPU - Explore o impacto do aumento da CPU no seu aplicativo baseado em EC2.

Neste cenário, vamos direcionar instâncias do EC2 na região atual que tenham uma tag específica anexada. Neste cenário, você pode personalizar uma quantidade crescente de estresse na CPU injetado nas instâncias do EC2 de destino durante a duração da ação, por padrão, 5 minutos para cada ação de estresse na CPU.

- Estresse do EC2: memória - Explore o impacto do aumento da utilização de memória no seu aplicativo baseado em EC2.

Neste cenário, vamos direcionar instâncias do EC2 na região atual que tenham uma tag específica anexada. Neste cenário, você pode personalizar uma quantidade crescente de estresse de memória injetado em instâncias do EC2 específicas durante a duração da ação, por padrão, 5 minutos para cada ação de estresse de memória.

- Estresse do EC2: latência de rede - Explore o impacto do aumento da latência de rede no seu aplicativo baseado em EC2.

Neste cenário, vamos direcionar instâncias do EC2 na região atual que tenham uma tag específica anexada. Neste cenário, você pode personalizar uma quantidade crescente de latência de rede injetada em instâncias do EC2 específicas durante a duração da ação, por padrão, 5 minutos para cada ação de latência.

O AWS FIS é compatível com os seguintes cenários do Amazon EKS. Esses cenários são voltados para pods do EKS que usam rótulos de aplicações do Kubernetes. Você pode usar seus próprios

rótulos ou os rótulos-padrão incluídos no cenário. Consulte mais informações sobre o EKS com FIS em [Ações do EKS Pod](#).

- Estresse do EKS: exclusão de pod - Explore o efeito da falha do pod do EKS excluindo um ou mais pods.

Nesse cenário, teremos como destino pods na região atual que estão associados a um rótulo do aplicativo. Nesse cenário, encerraremos todos os pods correspondentes. A recriação de pods será controlada pela configuração do kubernetes.

- Estresse do EKS: CPU - Explore o impacto do aumento da CPU no seu aplicativo baseado em EKS.

Nesse cenário, teremos como destino pods na região atual que estão associados a um rótulo do aplicativo. Neste cenário, você pode personalizar uma quantidade crescente de estresse na CPU injetado nos pods do EKS de destino durante a duração da ação, por padrão, 5 minutos para cada ação de estresse na CPU.

- Estresse do EKS: disco - Explore o impacto do aumento da utilização do disco em seu aplicativo baseado em EKS.

Nesse cenário, teremos como destino pods na região atual que estão associados a um rótulo do aplicativo. Neste cenário, você pode personalizar uma quantidade crescente de estresse no disco injetado nos pods do EKS de destino durante a duração da ação, por padrão, 5 minutos para cada ação de estresse na CPU.

- Estresse do EKS: memória - Explore o impacto do aumento da utilização de memória no seu aplicativo baseado em EKS.

Nesse cenário, teremos como destino pods na região atual que estão associados a um rótulo do aplicativo. Neste cenário, você pode personalizar uma quantidade crescente de estresse de memória injetado em pods do EKS específicas durante a duração da ação, por padrão, 5 minutos para cada ação de estresse de memória.

- Estresse do EKS: latência de rede - Explore o impacto do aumento da latência de rede no seu aplicativo baseado em EKS.

Nesse cenário, teremos como destino pods na região atual que estão associados a um rótulo do aplicativo. Neste cenário, você pode personalizar uma quantidade crescente de latência de rede injetada em pods do EKS específicas durante a duração da ação, por padrão, 5 minutos para cada ação de latência.

O AWS FIS oferece suporte aos seguintes cenários para aplicativos Single-AZ, Multi-AZ e multirregionais. Esses cenários são voltados para vários tipos de recursos.

- **AZ Availability: Power Interruption:** injeção dos sintomas esperados de uma interrupção completa de energia em uma zona de disponibilidade (AZ). Saiba mais sobre [AZ Availability: Power Interruption](#).
- **AZ: Application Slowdown-** Adicione latência entre recursos em uma única zona de disponibilidade (AZ) para desacelerar um aplicativo. Saiba mais sobre [AZ: Application Slowdown](#).
- **Cross-AZ: Traffic Slowdown-** Injeção da perda de pacotes para interromper e diminuir o tráfego entre as zonas de disponibilidade (). AZs Saiba mais sobre [Cross-AZ: Traffic Slowdown](#).
- **Cross-Region: Connectivity:** bloqueie o tráfego de rede da aplicação da região do experimento para a região de destino e pause a replicação de dados entre regiões. Saiba mais sobre como usar [Cross-Region: Connectivity](#).

AWS FIS suporta os seguintes cenários para volumes do Amazon EBS. Esses cenários têm como alvo volumes usando tags. Você pode usar suas próprias tags ou as tags-padrão incluídas no cenário. Os volumes de destino devem estar na mesma zona de disponibilidade. Para obter mais informações, [teste de falhas no Amazon EBS](#).

- **EBS: Sustained Latency—** Explore o impacto da I/O latência persistente em seu aplicativo.

Nesse cenário, teremos como alvo volumes na zona de disponibilidade atual que tenham uma tag específica anexada. Esse cenário injeta uma latência constante de 500 ms em 50% das operações de leitura e 100% das operações de gravação de um volume, usando uma única ação de latência em um período de 15 minutos. Nesse cenário, você pode personalizar a quantidade de latência injetada, a porcentagem de I/O injetada e a duração da ação.

- **EBS: Increasing Latency—** Explore o impacto do aumento da I/O latência em seu aplicativo.

Nesse cenário, teremos como alvo volumes na zona de disponibilidade atual que tenham uma tag específica anexada. Esse cenário injeta uma latência crescente de 50 ms, 200 ms, 700 ms, 1 segundo e 15 segundos em 10% das operações de leitura e 25% das operações de gravação de um volume usando cinco ações de latência em um período de 15 minutos. Nesse cenário, você pode personalizar a quantidade de latência injetada, a porcentagem de I/O injetada e a duração da ação para cada ação de latência.

- **EBS: Intermittent Latency—** Explore o impacto dos picos intermitentes de I/O latência em seu aplicativo.

Nesse cenário, teremos como alvo volumes na zona de disponibilidade atual que tenham uma tag específica anexada. Esse cenário injeta três picos nítidos de latência intermitente de 30 segundos, 10 segundos e 20 segundos em 0,1% das I/O operações de leitura e gravação de um volume, usando três ações de latência, com intervalos de recuperação entre cada pico em um período de 15 minutos. Nesse cenário, você pode personalizar a quantidade de latência injetada, a porcentagem de I/O injetada e a duração da ação para cada ação de latência.

- EBS: Decreasing Latency— Explore o impacto da diminuição da I/O latência em seu aplicativo.

Nesse cenário, teremos como alvo volumes na zona de disponibilidade atual que tenham uma tag específica anexada. Esse cenário injeta uma latência decrescente de 20 segundos, 5 segundos, 900 ms, 300 ms e 40 ms em 10% das operações de leitura e gravação de um volume, usando cinco ações de latência em um período de 15 minutos. Nesse cenário, você pode personalizar a quantidade de latência injetada, a porcentagem de I/O injetada e a duração da ação para cada ação de latência.

AZ Availability: Power Interruption

Você pode usar o cenário AZ Availability: Power Interruption para induzir os sintomas esperados de uma interrupção completa de energia em uma zona de disponibilidade (AZ).

Esse cenário pode ser usado para demonstrar que as aplicações Multi-AZ operam conforme o esperado durante uma interrupção única e completa de energia da AZ. Isso inclui perda de computação zonal (Amazon EC2, EKS e ECS), nenhum redimensionamento da computação no AZ, perda de conectividade de sub-rede, failover de RDS, failover, acesso prejudicado aos buckets de diretório S3 Express One Zone e ElastiCache volumes do EBS que não respondem. Por padrão, as ações para as quais nenhum destino foi encontrado serão ignoradas.

Ações

Juntas, as ações a seguir criam muitos dos sintomas esperados de uma interrupção completa da energia em uma única AZ. Disponibilidade da AZ: a interrupção de energia afeta apenas os serviços que devem sofrer impacto durante uma única interrupção de energia da AZ. Por padrão, o cenário injeta sintomas de interrupção de energia por 30 minutos e, depois, injeta sintomas que podem ocorrer durante a recuperação por mais 30 minutos.

Stop-Instances

Durante uma interrupção de energia da AZ, as instâncias do EC2 na AZ afetada serão encerradas. Depois que a energia for restaurada, as instâncias serão reinicializadas. AZ Availability: Power Interruption inclui [aws:ec2:stop-instances](#) para interromper todas as instâncias na AZ afetada durante a interrupção. Após a duração, as instâncias são reiniciadas. A interrupção de instâncias do EC2 gerenciadas pelo Amazon EKS faz com que os pods do EKS dependentes sejam excluídos. A interrupção de instâncias do EC2 gerenciadas pelo Amazon ECS faz com que as tarefas do ECS dependentes sejam interrompidas.

Essa ação é voltada para instâncias do EC2 em execução na AZ afetada. Por padrão, ela é voltada para instâncias com uma tag chamada `AzImpairmentPower` com um valor de `StopInstances`. Você pode adicionar essa tag às instâncias ou substituir a tag padrão pela sua própria tag no modelo de experimento. Por padrão, se nenhuma instância válida for encontrada, essa ação será ignorada.

Stop-ASG-Instances

Durante uma interrupção de energia do AZ, as instâncias do EC2 gerenciadas por um grupo do Auto Scaling na AZ afetada serão encerradas. Depois que a energia for restaurada, as instâncias serão reinicializadas. AZ Availability: Power Interruption inclui [aws:ec2:stop-instances](#) para interromper todas as instâncias, inclusive aquelas gerenciadas pelo Auto Scaling, na AZ afetada durante a interrupção. Após a duração, as instâncias são reiniciadas.

Essa ação é voltada para instâncias do EC2 em execução na AZ afetada. Por padrão, ela é voltada para instâncias com uma tag chamada `AzImpairmentPower` com um valor de `IceAsg`. Você pode adicionar essa tag às instâncias ou substituir a tag padrão pela sua própria tag no modelo de experimento. Por padrão, se nenhuma instância válida for encontrada, essa ação será ignorada.

Pausar inicialização de instâncias

Durante uma interrupção de energia da AZ, as chamadas de API do EC2 para provisionar capacidade na AZ falharão. Em particular, o seguinte APIs será afetado: `ec2:StartInstances` e `ec2:CreateFleet`, e `ec2:RunInstances`. AZ Availability: Power Interruption inclui [aws:ec2:api-insufficient-instance-capacity -error](#) para evitar que novas instâncias sejam provisionadas na AZ afetada.

Essa ação é voltada para perfis do IAM usados para provisionar instâncias. Eles devem ser segmentados usando um ARN. Por padrão, se nenhum perfil do IAM válido for encontrado, essa ação será ignorada.

Pausar escalabilidade do ASG

Durante uma interrupção de energia da AZ, as chamadas de API do EC2 feitas pelo ambiente de gerenciamento do Auto Scaling para recuperar a capacidade perdida na AZ falharão. Em particular, o seguinte APIs será afetado: `ec2:StartInstances` e `ec2:CreateFleet`, e `ec2:RunInstances`. AZ Availability: Power Interruption inclui [aws:ec2: asg-insufficient-instance-capacity -error](#) para evitar que novas instâncias sejam provisionadas na AZ afetada. Isso também impede que o Amazon EKS e o Amazon ECS escalem na AZ afetada.

Essa ação é voltada para grupos do Auto Scaling. Por padrão, ela é voltada para grupos do Auto Scaling com uma tag chamada `AzImpairmentPower` com um valor de `IceAsg`. Você pode adicionar essa tag aos grupos do Auto Scaling ou substituir a tag padrão pela sua própria tag no modelo de experimento. Por padrão, se nenhum grupo do Auto Scaling válido for encontrado, essa ação será ignorada.

Pausar conectividade de rede

Durante uma interrupção de energia na AZ, a rede na AZ ficará indisponível. Quando isso acontece, certos serviços da AWS podem levar alguns minutos para atualizar o DNS a fim de refletir que os endpoints privados na AZ afetada não estão disponíveis. Durante esse período, as pesquisas de DNS podem retornar endereços IP inacessíveis. AZ Availability: Power Interruption inclui [aws:network:disrupt-connectivity](#) para bloquear toda a conectividade de rede de todas as sub-redes na AZ afetada por dois minutos. Isso forçará tempos limite e atualizações de DNS para a maioria das aplicações. O encerramento da ação após dois minutos permite a recuperação subsequente do DNS do serviço regional enquanto a AZ continua indisponível.

Essa ação é voltada para sub-redes. Por padrão, ela é voltada para clusters com uma tag chamada `AzImpairmentPower` com um valor de `DisruptSubnet`. Você pode adicionar essa tag às sub-redes ou substituir a tag padrão pela sua própria tag no modelo de experimento. Por padrão, se nenhuma sub-rede válida for encontrada, essa ação será ignorada.

Failover do RDS

Durante uma interrupção de energia da AZ, os nós do RDS na AZ afetada serão encerrados. Os nós do RDS de AZ única na AZ afetada ficarão totalmente indisponíveis. Para clusters Multi-AZ, será feito failover do nó do gravador para uma AZ não afetada e os nós de leitura na AZ afetada ficarão indisponíveis. Para clusters Multi-AZ, AZ Availability: Power Interruption inclui [aws:rds: failover-db-cluster](#) para realizar o failover se o gravador estiver na AZ afetada.

Essa ação é voltada para clusters do RDS. Por padrão, ela é voltada para clusters com uma tag chamada `AzImpairmentPower` com um valor de `DisruptRds`. Você pode adicionar essa tag aos clusters ou substituir a tag padrão pela sua própria tag no modelo do experimento. Por padrão, se nenhum cluster válido for encontrado, essa ação será ignorada.

Pausar o grupo de ElastiCache replicação

Durante uma interrupção de alimentação do AZ, ElastiCache os nós no AZ não estão disponíveis. `AZ Availability: Power Interruption` inclui [aws:elasticache: replicationgroup-interrupt-az-power](#) para encerrar ElastiCache nós na AZ afetada. Durante a interrupção, novas instâncias não serão provisionadas na AZ afetada, portanto, o grupo de replicação permanecerá com capacidade reduzida.

Essa ação tem como alvo grupos ElastiCache de replicação. Por padrão, ele tem como alvo grupos de replicação com uma tag nomeada `AzImpairmentPower` com um valor de `ElasticacheImpact`. Você pode adicionar essa tag aos seus grupos de replicação ou substituir a tag padrão pela sua própria tag no modelo de experimento. Por padrão, se nenhum grupo de replicação válido for encontrado, essa ação será ignorada. Observe que somente grupos de replicação com nós na AZ afetada serão considerados alvos válidos.

Inicie o ARC Zonal Autoshift

Cinco minutos após o início da interrupção de alimentação do AZ, a ação de recuperação desvia `aws:arc:start-zonal-autoshift` automaticamente o tráfego de recursos do AZ especificado pelos 25 minutos restantes da interrupção de energia. Após esse período, o tráfego volta para o AZ original. Observe que, durante uma interrupção de energia do AZ no mundo real, AWS detectará a deficiência e mudará o tráfego de recursos se a mudança automática estiver ativada. Embora o tempo dessa mudança varie, estima-se que ocorra cinco minutos após o início da deficiência.

Essa ação tem como alvo os recursos habilitados para o deslocamento automático do Amazon Application Recovery Controller (ARC). Por padrão, ele segmenta recursos com a chave `AzImpairmentPower` e o valor da tag `RecoverAutoshiftResources`. Você pode adicionar essa tag aos seus recursos ou substituir a tag padrão pela sua própria tag no modelo do experimento. Por exemplo, talvez você queira usar uma tag específica do aplicativo. Por padrão, se nenhum recurso válido for encontrado, essa ação será ignorada.

Pausar E/S do EBS

Quando a energia da AZ é restaurada após uma interrupção, uma porcentagem muito pequena de instâncias pode apresentar volumes do EBS sem resposta. AZ Availability: Power Interruption inclui [aws:efs:pause-io](#) para deixar um volume do EBS em um estado sem resposta.

Por padrão, somente os volumes definidos para persistir após o encerramento da instância são direcionados. Essa ação se destina a volumes com uma tag chamada `AzImpairmentPower` com um valor de `APIPauseVolume`. Você pode adicionar essa tag aos volumes ou substituir a tag padrão pela sua própria tag no modelo do experimento. Por padrão, se nenhum volume válido for encontrado, essa ação será ignorada.

Interrompa a conectividade com buckets de diretório S3 Express One Zone

Durante uma interrupção de alimentação do AZ, os dados armazenados nos buckets do diretório S3 Express One Zone no AZ não estão acessíveis. Disponibilidade do AZ: o Power Interruption inclui [aws:network:disrupt-connectivity para interromper a conectividade](#) entre sub-redes e buckets de diretório de uma zona na AZ afetada durante o experimento, resultando em tempos limite para as operações da API do plano de dados do endpoint zonal. Use essa ação para testar a interrupção quando a computação está co-localizada com o armazenamento em uma AZ.

Essa ação é voltada para sub-redes. Por padrão, ele tem como alvo sub-redes com uma tag nomeada `AzImpairmentPower` com um valor de `DisruptSubnet`. Você pode adicionar essa tag às sub-redes ou substituir a tag padrão pela sua própria tag no modelo de experimento. Por padrão, se nenhuma sub-rede válida for encontrada, essa ação será ignorada.

Limitações

- Esse cenário não inclui [condições de interrupção](#). As condições de interrupção corretas para a aplicação devem ser adicionadas ao modelo de experimento.
- Na AZ de destino, os pods do Amazon EKS em execução no EC2 serão encerrados com nós de processamento do EC2 e o início de novos nós do EC2 será bloqueado. No entanto, os pods do Amazon EKS executados no AWS Fargate não são compatíveis.
- Na AZ de destino, as tarefas do Amazon ECS em execução no EC2 serão encerradas com nós de processamento do EC2 e o início de novos nós do EC2 será bloqueado. No entanto, as tarefas do Amazon ECS executadas no AWS Fargate não são compatíveis.
- [Multi-AZ do Amazon RDS](#) com duas instâncias de banco de dados em espera legíveis não é compatível. Nesse caso, as instâncias serão encerradas, o RDS fará um failover e a capacidade

será imediatamente provisionada de volta na AZ afetada. O modo de espera legível na AZ afetada permanecerá disponível.

Requisitos

- Adicione a permissão necessária à [função do AWS FIS experimento](#).
- As tags de recursos devem ser aplicadas aos recursos que serão alvo do experimento. Eles podem usar sua própria convenção de marcação ou as tags-padrão definidas no cenário.

Permissões

O deslocamento automático zonal do ARC usa uma função vinculada ao serviço IAM `AWSServiceRoleForZonalAutoshiftPracticeRun` para realizar o deslocamento zonal em seu nome. Essa função usa a política gerenciada do IAM [AWSZonalAutoshiftPracticeRunSLRPolicy](#). Você não precisa criar a função manualmente. Quando você cria um modelo de experimento a partir do cenário de interrupção de energia AZ no Console de gerenciamento da AWS, no ou em um AWS SDK AWS CLI, o ARC cria a função vinculada ao serviço para você. Para obter mais informações, consulte [Usando a função vinculada ao serviço para o deslocamento automático zonal](#) no ARC.

A política a seguir concede ao AWS FIS as permissões necessárias para executar um experimento com o cenário AZ Availability: Power Interruption. Essa política deve ser anexada ao [perfil do experimento](#).

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowFISExperimentLoggingActionsCloudwatch",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:PutResourcePolicy",
        "logs:DescribeResourcePolicies",
        "logs:DescribeLogGroups"
      ]
    }
  ],
}
```

```

    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:*:*:network-acl/*",
    "Condition": {
      "StringEquals": {
        "ec2:CreateAction": "CreateNetworkAcl",
        "aws:RequestTag/managedByFIS": "true"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": "ec2:CreateNetworkAcl",
    "Resource": "arn:aws:ec2:*:*:network-acl/*",
    "Condition": {
      "StringEquals": {
        "aws:RequestTag/managedByFIS": "true"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:CreateNetworkAclEntry",
      "ec2>DeleteNetworkAcl"
    ],
    "Resource": [
      "arn:aws:ec2:*:*:network-acl/*",
      "arn:aws:ec2:*:*:vpc/*"
    ],
    "Condition": {
      "StringEquals": {
        "ec2:ResourceTag/managedByFIS": "true"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": "ec2:CreateNetworkAcl",
    "Resource": "arn:aws:ec2:*:*:vpc/*"
  },

```

```
{
  "Effect": "Allow",
  "Action": [
    "ec2:DescribeVpcs",
    "ec2:DescribeManagedPrefixLists",
    "ec2:DescribeSubnets",
    "ec2:DescribeNetworkAcls"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": "ec2:ReplaceNetworkAclAssociation",
  "Resource": [
    "arn:aws:ec2:*:*:subnet/*",
    "arn:aws:ec2:*:*:network-acl/*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "rds:FailoverDBCluster"
  ],
  "Resource": [
    "arn:aws:rds:*:*:cluster:*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "rds:RebootDBInstance"
  ],
  "Resource": [
    "arn:aws:rds:*:*:db:*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "elasticache:DescribeReplicationGroups",
    "elasticache:InterruptClusterAzPower"
  ],
  "Resource": [
    "arn:aws:elasticache:*:*:replicationgroup:*"
  ]
}
```

```

    ]
  },
  {
    "Sid": "TargetResolutionByTags",
    "Effect": "Allow",
    "Action": [
      "tag:GetResources"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:StartInstances",
      "ec2:StopInstances"
    ],
    "Resource": "arn:aws:ec2:*:*:instance/*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeInstances"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "kms:CreateGrant"
    ],
    "Resource": [
      "arn:aws:kms:*:*:key/*"
    ],
    "Condition": {
      "StringLike": {
        "kms:ViaService": "ec2.*.amazonaws.com"
      },
      "Bool": {
        "kms:GrantIsForAWSResource": "true"
      }
    }
  },
  {
    "Effect": "Allow",

```

```

    "Action": [
      "ec2:DescribeVolumes"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:PauseVolumeIO"
    ],
    "Resource": "arn:aws:ec2:*:*:volume/*"
  },
  {
    "Sid": "AllowInjectAPI",
    "Effect": "Allow",
    "Action": [
      "ec2:InjectApiError"
    ],
    "Resource": [
      "*"
    ],
    "Condition": {
      "ForAnyValue:StringEquals": {
        "ec2:FisActionId": [
          "aws:ec2:api-insufficient-instance-capacity-error",
          "aws:ec2:asg-insufficient-instance-capacity-error"
        ]
      }
    }
  },
  {
    "Sid": "DescribeAsg",
    "Effect": "Allow",
    "Action": [
      "autoscaling:DescribeAutoScalingGroups"
    ],
    "Resource": [
      "*"
    ]
  }
]
}

```

Conteúdo do cenário

O conteúdo a seguir define o cenário. Esse JSON pode ser salvo e usado para criar um [modelo de experimento](#) usando o [create-experiment-template](#) comando da AWS Command Line Interface (AWS CLI). A versão mais recente do cenário está disponível na biblioteca de cenários no console do FIS.

```
{
  "targets": {
    "IAM-role": {
      "resourceType": "aws:iam:role",
      "resourceArns": [],
      "selectionMode": "ALL"
    },
    "EBS-Volumes": {
      "resourceType": "aws:ec2:ebs-volume",
      "resourceTags": {
        "AzImpairmentPower": "ApiPauseVolume"
      },
      "selectionMode": "COUNT(1)",
      "parameters": {
        "availabilityZoneIdentifier": "us-east-1a"
      },
      "filters": [
        {
          "path": "Attachments.DeleteOnTermination",
          "values": [
            "false"
          ]
        }
      ]
    },
    "EC2-Instances": {
      "resourceType": "aws:ec2:instance",
      "resourceTags": {
        "AzImpairmentPower": "StopInstances"
      },
      "filters": [
        {
          "path": "State.Name",
          "values": [
            "running"
          ]
        }
      ]
    }
  }
}
```

```
        {
            "path": "Placement.AvailabilityZone",
            "values": [
                "us-east-1a"
            ]
        }
    ],
    "selectionMode": "ALL"
},
"ASG": {
    "resourceType": "aws:ec2:autoscaling-group",
    "resourceTags": {
        "AzImpairmentPower": "IceAsg"
    },
    "selectionMode": "ALL"
},
"ASG-EC2-Instances": {
    "resourceType": "aws:ec2:instance",
    "resourceTags": {
        "AzImpairmentPower": "IceAsg"
    },
    "filters": [
        {
            "path": "State.Name",
            "values": [
                "running"
            ]
        },
        {
            "path": "Placement.AvailabilityZone",
            "values": [
                "us-east-1a"
            ]
        }
    ],
    "selectionMode": "ALL"
},
"Subnet": {
    "resourceType": "aws:ec2:subnet",
    "resourceTags": {
        "AzImpairmentPower": "DisruptSubnet"
    },
    "filters": [
        {
```

```

        "path": "AvailabilityZone",
        "values": [
            "us-east-1a"
        ]
    },
    ],
    "selectionMode": "ALL",
    "parameters": {}
},
"RDS-Cluster": {
    "resourceType": "aws:rds:cluster",
    "resourceTags": {
        "AzImpairmentPower": "DisruptRds"
    },
    "selectionMode": "ALL",
    "parameters": {
        "writerAvailabilityZoneIdentifiers": "us-east-1a"
    }
},
"ElastiCache-Cluster": {
    "resourceType": "aws:elasticache:replicationgroup",
    "resourceTags": {
        "AzImpairmentPower": "DisruptElasticache"
    },
    "selectionMode": "ALL",
    "parameters": {
        "availabilityZoneIdentifier": "us-east-1a"
    }
}
},
"actions": {
    "Pause-Instance-Launches": {
        "actionId": "aws:ec2:api-insufficient-instance-capacity-error",
        "parameters": {
            "availabilityZoneIdentifiers": "us-east-1a",
            "duration": "PT30M",
            "percentage": "100"
        },
        "targets": {
            "Roles": "IAM-role"
        }
    },
    "Pause-EBS-IO": {
        "actionId": "aws:ebs:pause-volume-io",

```

```
    "parameters": {
      "duration": "PT30M"
    },
    "targets": {
      "Volumes": "EBS-Volumes"
    },
    "startAfter": [
      "Stop-Instances",
      "Stop-ASG-Instances"
    ]
  },
  "Stop-Instances": {
    "actionId": "aws:ec2:stop-instances",
    "parameters": {
      "completeIfInstancesTerminated": "true",
      "startInstancesAfterDuration": "PT30M"
    },
    "targets": {
      "Instances": "EC2-Instances"
    }
  },
  "Pause-ASG-Scaling": {
    "actionId": "aws:ec2:asg-insufficient-instance-capacity-error",
    "parameters": {
      "availabilityZoneIdentifiers": "us-east-1a",
      "duration": "PT30M",
      "percentage": "100"
    },
    "targets": {
      "AutoScalingGroups": "ASG"
    }
  },
  "Stop-ASG-Instances": {
    "actionId": "aws:ec2:stop-instances",
    "parameters": {
      "completeIfInstancesTerminated": "true",
      "startInstancesAfterDuration": "PT30M"
    },
    "targets": {
      "Instances": "ASG-EC2-Instances"
    }
  },
  "Pause-network-connectivity": {
    "actionId": "aws:network:disrupt-connectivity",
```

```

    "parameters": {
      "duration": "PT2M",
      "scope": "all"
    },
    "targets": {
      "Subnets": "Subnet"
    }
  },
  "Failover-RDS": {
    "actionId": "aws:rds:failover-db-cluster",
    "parameters": {},
    "targets": {
      "Clusters": "RDS-Cluster"
    }
  },
  "Pause-ElastiCache": {
    "actionId": "aws:elasticache:replicationgroup-interrupt-az-power",
    "parameters": {
      "duration": "PT30M"
    },
    "targets": {
      "ReplicationGroups": "ElastiCache-Cluster"
    }
  }
},
"stopConditions": [
  {
    "source": "aws:cloudwatch:alarm",
    "value": ""
  }
],
"roleArn": "",
"tags": {
  "Name": "AZ Impairment: Power Interruption"
},
"logConfiguration": {
  "logSchemaVersion": 2
},
"experimentOptions": {
  "accountTargeting": "single-account",
  "emptyTargetResolutionMode": "skip"
},
"description": "Affect multiple resource types in a single AZ, targeting by tags
and explicit ARNs, to approximate power interruption in one AZ."

```

```
}
```

AZ: Application Slowdown

Você pode usar o cenário AZ: Desaceleração do aplicativo para introduzir latência adicional entre os recursos em uma única zona de disponibilidade (AZ). Essa latência cria muitos dos sintomas de lentidão do aplicativo, uma interrupção parcial, às vezes conhecida como falha cinza. Ele adiciona latência aos fluxos de rede entre os recursos de destino. Os fluxos de rede representam o tráfego entre os recursos de computação — os pacotes de dados que transportam solicitações, respostas e outras comunicações entre seus servidores, contêineres e serviços. O cenário pode ajudar a validar as configurações de observabilidade, ajustar os limites de alarme, descobrir a sensibilidade do aplicativo à lentidão e praticar decisões operacionais críticas, como a evacuação de AZ.

Por padrão, o cenário adiciona 200 ms de latência a 100% dos fluxos de rede entre os recursos de destino na AZ selecionada por um período de 30 minutos. Você pode usar a caixa de diálogo Editar parâmetros compartilhados no console do AWS FIS para ajustar os seguintes parâmetros no nível do cenário, que então se aplicam às ações subjacentes:

- Zona de disponibilidade - você pode selecionar a AZ a ser prejudicada no cenário.
- Milissegundos (ms) de latência - ajuste isso com base na sensibilidade e nas necessidades do seu aplicativo. Você pode definir uma latência mais baixa para aplicativos mais sensíveis ou maior para testar o tratamento do tempo limite, por exemplo. Considere usar múltiplos da latência atual do seu aplicativo como base.
- Porcentagem de fluxos - reduza para prejudicar um subconjunto do tráfego. Por exemplo, você pode adicionar uma latência de 200 ms afetando 25% dos fluxos de rede para testes ainda mais sutis.
- Duração - defina por quanto tempo o experimento será executado. Você pode encurtar para testes mais rápidos ou executar testes prolongados mais longos. Por exemplo, defina a duração para 2 horas para testar seus mecanismos de recuperação em condições precárias.
- Direcionamento de recursos — você pode definir recursos de destino para o cenário geral usando tags (para instâncias do EC2 ou tarefas do ECS no EC2 ou Fargate) ou rótulos (para pods do EKS no EC2). Você pode especificar suas próprias tags e rótulos ou usar os padrões fornecidos no cenário. Se você não quiser usar tags ou rótulos, poderá editar a ação para direcionar recursos especificando outros parâmetros.
- Personalização - Se você não quiser direcionar os recursos do EC2 ou do ECS, você pode deixar as ações com as tags padrão. O experimento não encontrará nenhum recurso para direcionar

e a ação será ignorada. No entanto, se você não quiser direcionar os recursos do EKS, remova completamente a ação e o destino do EKS do cenário, pois isso exige que um identificador de cluster do EKS seja fornecido. Para uma personalização ainda mais granular, você pode modificar ações individuais diretamente no modelo do experimento.

Ações

Juntas, as ações a seguir ajudam a criar muitos dos sintomas de lentidão de um aplicativo em uma única AZ, introduzindo latência adicional nos fluxos de rede, que então se propagam pelo aplicativo. Essas ações são executadas paralelamente, cada uma adicionando latência de 200 ms por 30 minutos por padrão. Após esse período, a latência retorna aos níveis normais. O cenário precisa de pelo menos um dos seguintes tipos de recursos para ser executado: instância EC2, tarefa ECS ou pod EKS.

Latência da rede ECS

AZ: A desaceleração do aplicativo inclui [aws:ecs: task-network-latency](#) para introduzir latência nas tarefas do ECS. A ação tem como alvo tarefas na AZ selecionada. Por padrão, ele direciona tarefas com uma [tag](#) nomeada `AZApplicationSlowdown` com um valor de `LatencyForECS`. Você pode substituir a tag padrão pela sua própria ou adicionar a tag de cenário às suas tarefas. Se nenhuma tarefa válida for encontrada, essa ação será ignorada. Antes de executar um experimento no ECS, você deve seguir as [etapas de configuração das ações de tarefas do ECS](#).

Latência da rede EKS

AZ: A desaceleração do aplicativo inclui [aws:eks: pod-network-latency](#) para introduzir latência nos pods EKS. A ação tem como alvo pods na AZ selecionada. Por padrão, ele tem como alvo pods dentro de um cluster que têm rótulos com o formato `key=value`. O rótulo padrão fornecido é `AZApplicationSlowdown=LatencyForEKS`. Você pode substituir o rótulo padrão pelo seu próprio ou adicionar esse rótulo aos seus pods. Se nenhum pod válido for encontrado, essa ação será ignorada. Antes de realizar um experimento no EKS, você deve seguir as [etapas de configuração das ações do pod EKS](#).

Latência de rede EC2

[AZ: O Application Slowdown usa a ação aws:ssm:send-command para executar o documento - Network-Latency-Sources para introduzir a latência nas instâncias do EC2AWSFIS-Run](#). A ação tem como alvo instâncias na AZ selecionada. Por padrão, ele tem como alvo instâncias com uma [tag](#) nomeada `AZApplicationSlowdown` com um valor de `LatencyForEC2`. Você pode substituir a tag

padrão pela sua própria ou adicionar essa tag às suas instâncias. Se nenhuma instância válida for encontrada, essa ação será ignorada. Antes de executar um experimento no EC2 usando SSM, você deve [configurar o agente do AWS Systems Manager](#).

Limitações

- Esse cenário não inclui [condições de interrupção](#). As condições de interrupção corretas para a aplicação devem ser adicionadas ao modelo de experimento.

Requisitos

- Adicione as permissões necessárias à [função experimental](#) do AWS FIS.
- Você precisa direcionar um ou mais recursos de qualquer um dos três tipos a seguir na AZ selecionada: instâncias do EC2, tarefas do ECS ou pods do EKS.
- Todos os alvos do cenário devem estar na mesma VPC.

Permissões

Para executar esse cenário, você precisa de uma função do IAM com uma política de confiança que permita ao FIS assumir a função e as políticas gerenciadas para os tipos de recursos que você almeja no experimento: EC2, ECS e EKS. Quando você cria um modelo de experimento a partir do cenário AZ: Desaceleração do aplicativo, o FIS cria a função para você com a política de confiança e as seguintes políticas gerenciadas pela AWS:

- [AWSFaultInjectionSimulatorEC2Acesso](#)
- [AWSFaultInjectionSimulatorECSAccess](#)
- [AWSFaultInjectionSimulatorEKSAcesso](#)

Se você estiver usando uma [função do IAM](#) existente para executar o cenário AZ: Desaceleração do aplicativo, você pode anexar a seguinte política para conceder ao AWS FIS as permissões necessárias:

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

```

    "Effect": "Allow",
    "Action": "ecs:DescribeTasks",
    "Resource": "*"
  },
  {
    "Sid": "DescribeContainerInstances",
    "Effect": "Allow",
    "Action": "ecs:DescribeContainerInstances",
    "Resource": "arn:aws:ecs:*:*:container-instance/*/*"
  },
  {
    "Sid": "DescribeInstances",
    "Effect": "Allow",
    "Action": "ec2:DescribeInstances",
    "Resource": "*"
  },
  {
    "Sid": "DescribeSubnets",
    "Effect": "Allow",
    "Action": "ec2:DescribeSubnets",
    "Resource": "*"
  },
  {
    "Sid": "DescribeCluster",
    "Effect": "Allow",
    "Action": "eks:DescribeCluster",
    "Resource": "arn:aws:eks:*:*:cluster/*"
  },
  {
    "Sid": "TargetResolutionByTags",
    "Effect": "Allow",
    "Action": "tag:GetResources",
    "Resource": "*"
  },
  {
    "Sid": "SendCommand",
    "Effect": "Allow",
    "Action": [
      "ssm:SendCommand"
    ],
    "Resource": [
      "arn:aws:ec2:*:*:instance/*",
      "arn:aws:ssm:*:*:managed-instance/*",
      "arn:aws:ssm:*:*:document/*"
    ]
  }
}

```

```

    ],
    {
      "Sid": "ListCommands",
      "Effect": "Allow",
      "Action": [
        "ssm:ListCommands"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CancelCommand",
      "Effect": "Allow",
      "Action": [
        "ssm:CancelCommand"
      ],
      "Resource": "*"
    }
  ]
}

```

Conteúdo do cenário

O conteúdo a seguir define o cenário. Esse JSON pode ser salvo e usado para criar um [modelo de experimento](#) usando o [create-experiment-template](#) comando da AWS Command Line Interface (AWS CLI). Para obter a versão mais recente do cenário, visite a biblioteca de cenários no console FIS e acesse a guia Conteúdo.

```

{
  "tags": {
    "Name": "AZ: Application Slowdown"
  },
  "description": "Add latency between resources within a single AZ.",
  "actions": {
    "LatencyForEKS": {
      "actionId": "aws:eks:pod-network-latency",
      "parameters": {
        "delayMilliseconds": "200",
        "duration": "PT30M",
        "flowsPercent": "100",
        "interface": "DEFAULT",
        "kubernetesServiceAccount": "fis-service-account",
        "sources": "us-east-1a"
      }
    }
  }
}

```

```

    },
    "targets": {
      "Pods": "TargetsForEKS"
    }
  },
  "LatencyForEC2": {
    "actionId": "aws:ssm:send-command",
    "parameters": {
      "duration": "PT30M",
      "documentArn": "arn:aws:ssm:us-east-1::document/AWSFIS-Run-Network-
Latency-Sources",
      "documentParameters": "{\"DelayMilliseconds\": \"200\", \"Sources\": \"us-
east-1a\", \"Interface\": \"DEFAULT\", \"TrafficType\": \"egress\", \"DurationSeconds\":
\"1800\", \"FlowsPercent\": \"100\", \"InstallDependencies\": \"True\"}"
    },
    "targets": {
      "Instances": "TargetsForEC2"
    }
  },
  "LatencyForECS": {
    "actionId": "aws:ecs:task-network-latency",
    "parameters": {
      "delayMilliseconds": "200",
      "duration": "PT30M",
      "flowsPercent": "100",
      "installDependencies": "true",
      "sources": "us-east-1a",
      "useEcsFaultInjectionEndpoints": "true"
    },
    "targets": {
      "Tasks": "TargetsForECS"
    },
    "startAfter": []
  }
},
"targets": {
  "TargetsForEKS": {
    "parameters": {
      "availabilityZoneIdentifier": "us-east-1a",
      "clusterIdentifier": "",
      "namespace": "default",
      "selectorType": "labelSelector",
      "selectorValue": "AZApplicationSlowdown=LatencyForEKS"
    }
  },

```

```
    "resourceType": "aws:eks:pod",
    "selectionMode": "ALL"
  },
  "TargetsForEC2": {
    "filters": [
      {
        "path": "Placement.AvailabilityZone",
        "values": [
          "us-east-1a"
        ]
      }
    ],
    "resourceTags": {
      "AZApplicationSlowdown": "LatencyForEC2"
    },
    "resourceType": "aws:ec2:instance",
    "selectionMode": "ALL"
  },
  "TargetsForECS": {
    "filters": [
      {
        "path": "AvailabilityZone",
        "values": [
          "us-east-1a"
        ]
      }
    ],
    "resourceTags": {
      "AZApplicationSlowdown": "LatencyForECS"
    },
    "resourceType": "aws:ecs:task",
    "selectionMode": "ALL"
  }
},
"experimentOptions": {
  "accountTargeting": "single-account",
  "emptyTargetResolutionMode": "skip"
},
"stopConditions": [
  {
    "source": "none"
  }
]
```

}

Cross-AZ: Traffic Slowdown

Você pode usar o cenário Cross-AZ: Traffic Slowdown para injetar perda de pacotes para interromper e diminuir a velocidade do tráfego entre as zonas de disponibilidade (). AZs A perda de pacotes prejudica a comunicação entre AZ, uma interrupção parcial, às vezes conhecida como falha cinza. Ele injeta perda de pacotes nos fluxos de rede entre os recursos de destino. Os fluxos de rede representam o tráfego entre os recursos de computação — os pacotes de dados que transportam solicitações, respostas e outras comunicações entre seus servidores, contêineres e serviços. O cenário pode ajudar a validar as configurações de observabilidade, ajustar os limites de alarme, descobrir a sensibilidade e as dependências do aplicativo na comunicação entre AZ e praticar decisões operacionais críticas, como a evacuação de AZ.

Por padrão, o cenário injeta 15% de perda de pacotes em 100% dos fluxos de rede de saída para os recursos de destino da AZ selecionada por um período de 30 minutos. Você pode usar a caixa de diálogo Editar parâmetros compartilhados no console do AWS FIS para ajustar os seguintes parâmetros no nível do cenário, que então se aplicam às ações subjacentes:

- Zona de disponibilidade - você pode selecionar a AZ a ser prejudicada, e a perda de pacotes será injetada dessa AZ para outras AZs dentro da região.
- Perda de pacotes - ajuste a perda de pacotes para testes sutis de interrupção, como 5%, ou mais, para testar mecanismos graves de degradação e recuperação da comunicação, como 50% ou até 100%, para verificar o impacto total da conectividade.
- Porcentagem de fluxos - reduza para prejudicar um subconjunto do tráfego. Por exemplo, você pode injetar 15% de perda de pacotes afetando 25% dos fluxos de rede para testes ainda mais sutis.
- Duração - defina por quanto tempo o experimento será executado. Você pode encurtar para testes mais rápidos ou executar testes prolongados mais longos. Por exemplo, defina a duração para 2 horas para ajudar a testar os mecanismos de recuperação em condições precárias.
- Direcionamento de recursos — você pode definir recursos de destino para o cenário geral usando tags (para instâncias do EC2 ou tarefas do ECS no EC2 ou Fargate) ou rótulos (para pods do EKS no EC2). Você pode especificar suas próprias tags e rótulos ou usar os padrões fornecidos no cenário. Se você não quiser usar tags ou rótulos, poderá editar a ação para direcionar recursos especificando outros parâmetros.
- Personalização - Se você não quiser direcionar os recursos do EC2 ou do ECS, você pode deixar as ações com as tags padrão. O experimento não encontrará nenhum recurso para direcionar

e a ação será ignorada. No entanto, se você não quiser direcionar os recursos do EKS, remova completamente a ação e o destino do EKS do cenário, pois isso exige que um identificador de cluster do EKS seja fornecido. Para uma personalização ainda mais granular, você pode modificar ações individuais diretamente no modelo do experimento.

Ações

Juntas, as ações a seguir ajudam a criar os sintomas de uma desaceleração do tráfego entre as AZs, introduzindo a perda de pacotes na comunicação de saída da AZ de destino para outras AZs na região na camada de rede. Essas ações são executadas paralelamente, cada uma injetando 15% de perda de pacotes por 30 minutos por padrão. Após esse período, a comunicação volta ao normal. O cenário precisa de pelo menos um dos seguintes tipos de recursos na AZ selecionada para ser executado: instância EC2, tarefa ECS ou pod EKS.

Perda de pacotes de rede ECS

Cross-AZ: A desaceleração do tráfego inclui [aws:ecs: task-network-packet-loss para injetar perda de pacotes](#) para tarefas do ECS. A ação visa tarefas na AZ selecionada e prejudica sua comunicação externa com todas as outras AZs na região. Você pode personalizar ainda mais o escopo do impacto editando a ação e adicionando ou removendo AZs do Sources campo. Por padrão, ele direciona tarefas com uma [tag](#) nomeada CrossAZTrafficSlowdown com um valor dePacketLossForECS. Você pode substituir a tag padrão pela sua própria ou adicionar a tag de cenário às suas tarefas. Se nenhuma tarefa válida for encontrada, essa ação será ignorada. Antes de executar um experimento no ECS, você deve seguir as [etapas de configuração das ações de tarefas do ECS](#).

Perda de pacotes de rede EKS

Cross-AZ: A desaceleração do tráfego inclui [aws:eks: pod-network-packet-loss](#) para injetar perda de pacotes em pods EKS. A ação tem como alvo os pods na AZ selecionada e prejudica sua comunicação de saída com todas as outras AZs na região. Você pode personalizar ainda mais o escopo do impacto editando a ação e adicionando ou removendo AZs do Sources campo. Por padrão, ele tem como alvo pods dentro de um cluster que têm rótulos com o formato key=value. O rótulo padrão fornecido éCrossAZTraffic=PacketLossForEKS. Você pode substituir o rótulo padrão pelo seu próprio ou adicionar esse rótulo aos seus pods. Se nenhum pod válido for encontrado, essa ação será ignorada. Antes de realizar um experimento no EKS, você deve seguir as [etapas de configuração das ações do pod EKS](#).

Perda de pacotes de rede EC2

Cross-AZ: Traffic Slowdown usa a ação [aws:ssm:send-command para executar o documento AWSFIS-Run -Network-Packet-Loss-Sources para injetar perda de pacotes](#) para instâncias do EC2 e prejudica sua comunicação de saída com todas as outras na região. AZs Você pode personalizar ainda mais o escopo do impacto editando a ação e adicionando ou removendo AZs do Sources campo. A ação tem como alvo instâncias na AZ selecionada. Por padrão, ele tem como alvo instâncias com uma [tag](#) nomeada CrossAZTrafficSlowdown com um valor dePacketLossForEC2. Você pode substituir a tag padrão pela sua própria ou adicionar essa tag às suas instâncias. Se nenhuma instância válida for encontrada, essa ação será ignorada. Antes de executar um experimento no EC2 usando SSM, você deve [configurar o agente do AWS Systems Manager](#).

Limitações

- Esse cenário não inclui [condições de interrupção](#). As condições de interrupção corretas para a aplicação devem ser adicionadas ao modelo de experimento.

Requisitos

- Adicione as permissões necessárias à [função experimental](#) do AWS FIS.
- Você precisa direcionar um ou mais recursos de qualquer um dos três tipos a seguir na AZ selecionada: instâncias do EC2, tarefas do ECS ou pods do EKS.
- Todos os alvos do cenário devem estar na mesma VPC.

Permissões

Para executar esse cenário, você precisa de uma função do IAM com uma política de confiança que permita ao FIS assumir a função e as políticas gerenciadas para os tipos de recursos que você almeja no experimento: EC2, ECS e EKS. Quando você cria um modelo de experimento a partir do cenário Cross-AZ: Traffic Slowdown, a FIS cria a função para você com a política de confiança e as seguintes políticas gerenciadas pela AWS:

- [AWSFaultInjectionSimulatorEC2Acesso](#)
- [AWSFaultInjectionSimulatorECSAccess](#)
- [AWSFaultInjectionSimulatorEKSAcesso](#)

Se você estiver usando uma [função do IAM](#) existente para executar o cenário Cross-AZ: Traffic Slowdown, você pode anexar a seguinte política para conceder ao AWS FIS as permissões necessárias:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DescribeTasks",
      "Effect": "Allow",
      "Action": "ecs:DescribeTasks",
      "Resource": "*"
    },
    {
      "Sid": "DescribeContainerInstances",
      "Effect": "Allow",
      "Action": "ecs:DescribeContainerInstances",
      "Resource": "arn:aws:ecs:*:*:container-instance/*/*"
    },
    {
      "Sid": "DescribeInstances",
      "Effect": "Allow",
      "Action": "ec2:DescribeInstances",
      "Resource": "*"
    },
    {
      "Sid": "DescribeSubnets",
      "Effect": "Allow",
      "Action": "ec2:DescribeSubnets",
      "Resource": "*"
    },
    {
      "Sid": "DescribeCluster",
      "Effect": "Allow",
      "Action": "eks:DescribeCluster",
      "Resource": "arn:aws:eks:*:*:cluster/*"
    },
    {
      "Sid": "TargetResolutionByTags",
      "Effect": "Allow",
      "Action": "tag:GetResources",
      "Resource": "*"
    }
  ],
}
```

```

    {
      "Sid": "SendCommand",
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:instance/*",
        "arn:aws:ssm:*:*:managed-instance/*",
        "arn:aws:ssm:*:*:document/*"
      ]
    },
    {
      "Sid": "ListCommands",
      "Effect": "Allow",
      "Action": [
        "ssm:ListCommands"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CancelCommand",
      "Effect": "Allow",
      "Action": [
        "ssm:CancelCommand"
      ],
      "Resource": "*"
    }
  ]
}

```

Conteúdo do cenário

O conteúdo a seguir define o cenário. Esse JSON pode ser salvo e usado para criar um [modelo de experimento](#) usando o [create-experiment-template](#) comando da AWS Command Line Interface (AWS CLI). Para obter a versão mais recente do cenário, visite a biblioteca de cenários no console FIS e acesse a guia Conteúdo.

```

{
  "tags": {
    "Name": "Cross-AZ: Traffic Slowdown"
  },
  "description": "Inject packet loss to disrupt and slow down traffic between AZs.",

```

```

"actions": {
  "PacketLossForEC2": {
    "actionId": "aws:ssm:send-command",
    "parameters": {
      "duration": "PT30M",
      "documentArn": "arn:aws:ssm:us-east-1::document/AWSFIS-Run-Network-
Packet-Loss-Sources",
      "documentParameters": "{\"Sources\": \"us-east-1b,us-east-1c,us-
east-1d,us-east-1e,us-east-1f\", \"LossPercent\": \"15\", \"Interface\": \"DEFAULT\",
\"TrafficType\": \"egress\", \"DurationSeconds\": \"1800\", \"FlowsPercent\": \"100\",
\"InstallDependencies\": \"True\"}"
    },
    "targets": {
      "Instances": "TargetsForEC2"
    }
  },
  "PacketLossForECS": {
    "actionId": "aws:ecs:task-network-packet-loss",
    "parameters": {
      "sources": "us-east-1b,us-east-1c,us-east-1d,us-east-1e,us-east-1f",
      "lossPercent": "15",
      "duration": "PT30M",
      "flowsPercent": "100",
      "installDependencies": "true",
      "useEcsFaultInjectionEndpoints": "true"
    },
    "targets": {
      "Tasks": "TargetsForECS"
    }
  },
  "PacketLossForEKS": {
    "actionId": "aws:eks:pod-network-packet-loss",
    "parameters": {
      "sources": "us-east-1b,us-east-1c,us-east-1d,us-east-1e,us-east-1f",
      "lossPercent": "15",
      "duration": "PT30M",
      "flowsPercent": "100",
      "interface": "DEFAULT",
      "kubernetesServiceAccount": "fis-service-account"
    },
    "targets": {
      "Pods": "TargetsForEKS"
    }
  }
}

```

```
},
"targets": {
  "TargetsForEC2": {
    "filters": [
      {
        "path": "Placement.AvailabilityZone",
        "values": [
          "us-east-1a"
        ]
      }
    ],
    "resourceTags": {
      "CrossAZTrafficSlowdown": "PacketLossForEC2"
    },
    "resourceType": "aws:ec2:instance",
    "selectionMode": "ALL"
  },
  "TargetsForECS": {
    "filters": [
      {
        "path": "AvailabilityZone",
        "values": [
          "us-east-1a"
        ]
      }
    ],
    "resourceTags": {
      "CrossAZTrafficSlowdown": "PacketLossForECS"
    },
    "resourceType": "aws:ecs:task",
    "selectionMode": "ALL"
  },
  "TargetsForEKS": {
    "parameters": {
      "availabilityZoneIdentifier": "us-east-1a",
      "clusterIdentifier": "",
      "namespace": "default",
      "selectorType": "labelSelector",
      "selectorValue": "CrossAZTrafficSlowdown=PacketLossForEKS"
    },
    "resourceType": "aws:eks:pod",
    "selectionMode": "ALL"
  }
},
}
```

```
"experimentOptions": {
  "accountTargeting": "single-account",
  "emptyTargetResolutionMode": "skip"
},
"stopConditions": [
  {
    "source": "none"
  }
]
}
```

Cross-Region: Connectivity

Você pode usar o Cross-Region: Connectivity cenário para bloquear o tráfego de rede de aplicativos da região do experimento para a região de destino e pausar a replicação entre regiões para tabelas globais multirregionais do Amazon S3 e do Amazon DynamoDB. Região cruzada: a conectividade afeta o tráfego de saída da aplicação da região na qual você executa o experimento (região do experimento). O tráfego de entrada sem estado da região que você deseja isolar da região do experimento (região de destino) pode não estar bloqueado. O tráfego dos serviços gerenciados da AWS pode não estar bloqueado.

Esse cenário pode ser usado para demonstrar que as aplicações multirregionais operam conforme o esperado quando os recursos na região de destino não estão acessíveis pela região do experimento. Isso inclui bloquear o tráfego de rede da região do experimento para a região de destino, visando gateways de trânsito e tabelas de rotas. Ele também pausa a replicação entre regiões para tabelas globais do S3 e do DynamoDB. Por padrão, as ações para as quais nenhum destino foi encontrado serão ignoradas.

Ações

Juntas, as ações a seguir bloqueiam a conectividade entre regiões para os serviços da AWS incluídos. As ações são executadas em paralelo. Por padrão, o cenário bloqueia o tráfego por três horas, o que você pode aumentar até uma duração máxima de 12 horas.

Interrupção da conectividade do gateway de trânsito

Cross Region: Connectivity inclui [aws:network: transit-gateway-disrupt-cross -region-connectivity](#) para bloquear o tráfego de rede entre regiões da região do experimento até VPCs a VPCs região de destino conectada por um gateway de trânsito. Isso não afeta o acesso aos endpoints da VPC na

região do experimento, mas bloqueará o tráfego da região do experimento destinado a um endpoint da VPC na região de destino.

Essa ação é voltada aos gateways de trânsito que conectam a região do experimento e a região de destino. Por padrão, ela é voltada para gateways de trânsito com uma [tag](#) chamada `DisruptTransitGateway` com um valor de `Allowed`. Você pode adicionar essa tag aos gateways de trânsito ou substituir a tag padrão pela sua própria tag no modelo do experimento. Por padrão, se nenhum gateway de trânsito válido for encontrado, essa ação será ignorada.

Interromper a conectividade da sub-rede

Cross Region: Connectivity inclui [aws:network: route-table-disrupt-cross -region-connectivity](#) para bloquear o tráfego de rede entre regiões da região do experimento para blocos IP públicos da AWS VPCs na região de destino. Esses blocos de IP público incluem endpoints de serviço da AWS na região de destino, por exemplo, o endpoint regional do S3, e blocos de IP da AWS para serviços gerenciados, por exemplo, os endereços IP usados para balanceadores de carga e o Amazon API Gateway. Essa ação também bloqueia a conectividade de rede em conexões de emparelhamento de VPC entre regiões da região do experimento até a região de destino. Isso não afeta o acesso aos endpoints da VPC na região do experimento, mas bloqueará o tráfego da região do experimento destinado a um endpoint da VPC na região de destino.

Essa ação é voltada para sub-redes na região do experimento. Por padrão, ela é voltada para sub-redes com uma [tag](#) chamada `DisruptSubnet` com um valor de `Allowed`. Você pode adicionar essa tag às sub-redes ou substituir a tag padrão pela sua própria tag no modelo de experimento. Por padrão, se nenhuma sub-rede válida for encontrada, essa ação será ignorada.

Interrompa a conectividade do VPC Endpoint

Cross Region: Connectivity inclui [aws:network: disrupt-vpc-endpoint](#) interrompa a conectividade com um serviço associado aos endpoints VPC de destino. Por exemplo, se um VPC endpoint criar um link privado para `com.amazonaws.us-east-1.ec2`, a conectividade com esse serviço será interrompida.

Essa ação tem como alvo os endpoints do VPC na região do experimento. Por padrão, ele tem como alvo os endpoints VPC da interface com uma [tag](#) nomeada `DisruptVpcEndpoint` com um valor. `Allowed` Você pode adicionar essa tag aos seus VPC endpoints ou substituir a tag padrão pela sua própria tag no modelo de experimento. Por padrão, se nenhum VPC endpoint válido for encontrado, essa ação será ignorada.

Pausar a replicação do S3

Cross Region: Connectivity inclui [aws:s3: bucket-pause-replication](#) para pausar a replicação do S3 da região do experimento para a região de destino dos buckets de destino. A replicação da região de destino para a região do experimento não será afetada. Depois que o cenário terminar, a replicação do bucket será retomada a partir do ponto em que foi pausada. Observe que o tempo necessário para a replicação manter todos os objetos sincronizados variará com base na duração do experimento e na taxa de upload do objeto para o bucket.

Essa ação é voltada para os buckets do S3 na região do experimento com a [replicação entre regiões](#) (CRR) habilitada para um bucket do S3 na região de destino. Por padrão, ela é voltada para buckets com uma [tag](#) chamada `DisruptS3` com um valor de `Allowed`. Você pode adicionar essa tag aos buckets ou substituir a tag padrão pela sua própria tag no modelo de experimento. Por padrão, se nenhum bucket válido for encontrado, essa ação será ignorada.

Pausar a replicação do DynamoDB

Cross-Region: Connectivity inclui [aws:dynamodb: global-table-pause-replication](#) para pausar a replicação entre a região do experimento e todas as outras regiões, incluindo a região de destino. Isso evita a replicação para dentro e para fora da região do experimento, mas não afeta a replicação entre outras regiões. Depois que o cenário terminar, a replicação da tabela será retomada a partir do ponto em que foi pausada. Observe que o tempo necessário para a replicação manter todos os dados sincronizados variará com base na duração do experimento e na taxa de alterações na tabela.

Essa ação tem como alvo tanto o DynamoDB Multi-region quanto, eventualmente, tabelas globais consistentes na região do experimento. Por padrão, ela é voltada para tabelas com uma [tag](#) chamada `DisruptDynamoDb` com um valor de `Allowed`. Você pode adicionar essa tag às tabelas ou substituir a tag padrão pela sua própria tag no modelo de experimento. Por padrão, se nenhuma tabela global válida for encontrada, essa ação será ignorada.

Pausar a replicação multirregional do MemoryDB

Cross-Region: Connectivity inclui [aws:memorydb: multi-region-cluster-pause -replication para pausar a replicação](#) do cluster membro regional na região do experimento para o resto dos clusters no cluster multirregional de destino. A replicação entre outros clusters de membros regionais não será afetada. Depois que o cenário terminar, a replicação será retomada a partir do ponto em que foi pausada. Observe que o tempo de replicação para sincronizar dados entre os clusters membros variará com base na duração do experimento e na taxa de dados gravados nos clusters.

Essa ação tem como alvo clusters multirregionais do MemoryDB com um membro regional na região do experimento. Por padrão, ele tem como alvo clusters multirregionais com uma [tag](#) nomeada `DisruptMemoryDB` com um valor de `Allowed`. Você pode adicionar essa tag aos seus clusters multirregionais ou substituir a tag padrão pela sua própria tag no modelo de experimento. Por padrão, se nenhum cluster válido for encontrado, essa ação será ignorada.

Limitações

- Esse cenário não inclui [condições de interrupção](#). As condições de interrupção corretas para a aplicação devem ser adicionadas ao modelo de experimento.

Requisitos

- Adicione a permissão necessária ao [perfil do experimento](#) do AWS FIS.
- As tags de recursos devem ser aplicadas aos recursos que serão alvo do experimento. Eles podem usar sua própria convenção de marcação ou as tags-padrão definidas no cenário.

Permissões

A política a seguir concede ao AWS FIS as permissões necessárias para executar um experimento com o cenário Cross-Region: Connectivity. Essa política deve ser anexada ao [perfil do experimento](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RouteTableDisruptConnectivity1",
      "Effect": "Allow",
      "Action": "ec2:CreateRouteTable",
      "Resource": "arn:aws:ec2:*:*:route-table/*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/managedByFIS": "true"
        }
      }
    },
    {
      "Sid": "RouteTableDisruptConnectivity2",
      "Effect": "Allow",
      "Action": "ec2:CreateRouteTable",
```

```

    "Resource": "arn:aws:ec2:*:*:vpc/*"
  },
  {
    "Sid": "RouteTableDisruptConnectivity21",
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:*:*:route-table/*",
    "Condition": {
      "StringEquals": {
        "ec2:CreateAction": "CreateRouteTable",
        "aws:RequestTag/managedByFIS": "true"
      }
    }
  },
  {
    "Sid": "RouteTableDisruptConnectivity3",
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:*:*:network-interface/*",
    "Condition": {
      "StringEquals": {
        "ec2:CreateAction": "CreateNetworkInterface",
        "aws:RequestTag/managedByFIS": "true"
      }
    }
  },
  {
    "Sid": "RouteTableDisruptConnectivity4",
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:*:*:prefix-list/*",
    "Condition": {
      "StringEquals": {
        "ec2:CreateAction": "CreateManagedPrefixList",
        "aws:RequestTag/managedByFIS": "true"
      }
    }
  },
  {
    "Sid": "RouteTableDisruptConnectivity5",
    "Effect": "Allow",
    "Action": "ec2>DeleteRouteTable",
    "Resource": [
      "arn:aws:ec2:*:*:route-table/*",

```

```

        "arn:aws:ec2:*:*:vpc/*"
    ],
    "Condition": {
        "StringEquals": {
            "ec2:ResourceTag/managedByFIS": "true"
        }
    }
},
{
    "Sid": "RouteTableDisruptConnectivity6",
    "Effect": "Allow",
    "Action": "ec2:CreateRoute",
    "Resource": "arn:aws:ec2:*:*:route-table/*",
    "Condition": {
        "StringEquals": {
            "ec2:ResourceTag/managedByFIS": "true"
        }
    }
},
{
    "Sid": "RouteTableDisruptConnectivity7",
    "Effect": "Allow",
    "Action": "ec2:CreateNetworkInterface",
    "Resource": "arn:aws:ec2:*:*:network-interface/*",
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/managedByFIS": "true"
        }
    }
},
{
    "Sid": "RouteTableDisruptConnectivity8",
    "Effect": "Allow",
    "Action": "ec2:CreateNetworkInterface",
    "Resource": [
        "arn:aws:ec2:*:*:subnet/*",
        "arn:aws:ec2:*:*:security-group*"
    ]
},
{
    "Sid": "RouteTableDisruptConnectivity9",
    "Effect": "Allow",
    "Action": "ec2>DeleteNetworkInterface",
    "Resource": "arn:aws:ec2:*:*:network-interface/*",

```

```

    "Condition": {
      "StringEquals": {
        "ec2:ResourceTag/managedByFIS": "true"
      }
    },
    {
      "Sid": "RouteTableDisruptConnectivity10",
      "Effect": "Allow",
      "Action": "ec2:CreateManagedPrefixList",
      "Resource": "arn:aws:ec2:*:*:prefix-list/*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/managedByFIS": "true"
        }
      }
    },
    {
      "Sid": "RouteTableDisruptConnectivity11",
      "Effect": "Allow",
      "Action": [
        "ec2>DeleteManagedPrefixList",
        "ec2:ModifyManagedPrefixList"
      ],
      "Resource": "arn:aws:ec2:*:*:prefix-list/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/managedByFIS": "true"
        }
      }
    },
    {
      "Sid": "EC2DescribeResources",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeVpcPeeringConnections",
        "ec2:DescribeManagedPrefixLists",
        "ec2:DescribeSubnets",
        "ec2:DescribeRouteTables",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeTransitGatewayPeeringAttachments",
        "ec2:DescribeTransitGatewayAttachments",

```

```

        "ec2:DescribeTransitGateways",
        "ec2:DescribeSecurityGroups"
    ],
    "Resource": "*"
},
{
    "Sid": "RouteTableDisruptConnectivity14",
    "Effect": "Allow",
    "Action": "ec2:ReplaceRouteTableAssociation",
    "Resource": [
        "arn:aws:ec2:*:*:subnet/*",
        "arn:aws:ec2:*:*:route-table/*"
    ]
},
{
    "Sid": "RouteTableDisruptConnectivity15",
    "Effect": "Allow",
    "Action": "ec2:GetManagedPrefixListEntries",
    "Resource": "arn:aws:ec2:*:*:prefix-list/*"
},
{
    "Sid": "RouteTableDisruptConnectivity16",
    "Effect": "Allow",
    "Action": "ec2:AssociateRouteTable",
    "Resource": [
        "arn:aws:ec2:*:*:subnet/*",
        "arn:aws:ec2:*:*:route-table/*"
    ]
},
{
    "Sid": "RouteTableDisruptConnectivity17",
    "Effect": "Allow",
    "Action": "ec2:DisassociateRouteTable",
    "Resource": "arn:aws:ec2:*:*:route-table/*",
    "Condition": {
        "StringEquals": {
            "ec2:ResourceTag/managedByFIS": "true"
        }
    }
},
{
    "Sid": "RouteTableDisruptConnectivity18",
    "Effect": "Allow",
    "Action": "ec2:DisassociateRouteTable",

```

```

    "Resource": "arn:aws:ec2:*:*:subnet/*"
  },
  {
    "Sid": "RouteTableDisruptConnectivity19",
    "Effect": "Allow",
    "Action": "ec2:ModifyVpcEndpoint",
    "Resource": "arn:aws:ec2:*:*:route-table/*",
    "Condition": {
      "StringEquals": {
        "ec2:ResourceTag/managedByFIS": "true"
      }
    }
  },
  {
    "Sid": "TransitGatewayDisruptConnectivity1",
    "Effect": "Allow",
    "Action": [
      "ec2:DisassociateTransitGatewayRouteTable",
      "ec2:AssociateTransitGatewayRouteTable"
    ],
    "Resource": [
      "arn:aws:ec2:*:*:transit-gateway-route-table/*",
      "arn:aws:ec2:*:*:transit-gateway-attachment/*"
    ]
  },
  {
    "Sid": "S3CrossRegion1",
    "Effect": "Allow",
    "Action": "s3:ListAllMyBuckets",
    "Resource": "*"
  },
  {
    "Sid": "S3CrossRegion3",
    "Effect": "Allow",
    "Action": "s3:PauseReplication",
    "Resource": "arn:aws:s3::*:*",
    "Condition": {
      "StringLike": {
        "s3:DestinationRegion": "*"
      }
    }
  },
  {
    "Sid": "S3CrossRegion4",

```

```

    "Effect": "Allow",
    "Action": [
      "s3:GetReplicationConfiguration",
      "s3:PutReplicationConfiguration"
    ],
    "Resource": "arn:aws:s3:::*",
    "Condition": {
      "BoolIfExists": {
        "s3:isReplicationPauseRequest": "true"
      }
    }
  },
  {
    "Sid": "DynamoDbPauseReplication",
    "Effect": "Allow",
    "Action": [
      "dynamodb:DescribeTable",
      "dynamodb:PutResourcePolicy",
      "dynamodb:GetResourcePolicy",
      "dynamodb>DeleteResourcePolicy"
    ],
    "Resource": [
      "arn:aws:dynamodb:*:*:table/*"
    ]
  },
  {
    "Sid": "DynamoDbMrscPauseReplication",
    "Effect": "Allow",
    "Action": [
      "dynamodb:InjectError"
    ],
    "Resource": ["*"]
  },
  {
    "Sid": "ResolveResourcesViaTags",
    "Effect": "Allow",
    "Action": "tag:GetResources",
    "Resource": "*"
  },
  {
    "Sid": "MemDbCrossRegion",
    "Effect": "Allow",
    "Action": [
      "memorydb:DescribeMultiRegionClusters",

```

```

        "memorydb:PauseMultiRegionClusterReplication"
    ],
    "Resource": [
        "arn:aws:memorydb:*:*:multiregioncluster/*"
    ]
},
{
    "Sid": "DisruptVPCE1",
    "Effect": "Allow",
    "Action": "ec2:CreateSecurityGroup",
    "Resource": [
        "arn:aws:ec2:*:*:vpc/*",
        "arn:aws:ec2:*:*:security-group/*"
    ]
},
{
    "Sid": "DisruptVPCE2",
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:*:*:security-group/*",
    "Condition": {
        "StringEquals": {
            "ec2:CreateAction": "CreateSecurityGroup",
            "aws:RequestTag/managedByFIS": "true"
        }
    }
},
{
    "Sid": "DisruptVPCE3",
    "Effect": "Allow",
    "Action": [
        "ec2>DeleteSecurityGroup",
        "ec2:RevokeSecurityGroupEgress"
    ],
    "Resource": "arn:aws:ec2:*:*:security-group/*",
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/managedByFIS": "true"
        }
    }
},
{
    "Sid": "DisruptVPCE4",
    "Effect": "Allow",

```

```

    "Action": "vpce:AllowMultiRegion",
    "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*"
  },
  {
    "Sid": "ModifyVPCE",
    "Effect": "Allow",
    "Action": "ec2:ModifyVpcEndpoint",
    "Resource": [
      "arn:aws:ec2:*:*:vpc-endpoint/*",
      "arn:aws:ec2:*:*:security-group/*"
    ]
  }
]
}

```

Conteúdo do cenário

O conteúdo a seguir define o cenário. Esse JSON pode ser salvo e usado para criar um [modelo de experimento](#) usando o [create-experiment-template](#) comando da AWS Command Line Interface (AWS CLI). A versão mais recente do cenário está disponível na biblioteca de cenários no console do FIS.

```

{
  "targets": {
    "Transit-Gateway": {
      "resourceType": "aws:ec2:transit-gateway",
      "resourceTags": {
        "TgwTag": "TgwValue"
      },
      "selectionMode": "ALL"
    },
    "Subnet": {
      "resourceType": "aws:ec2:subnet",
      "resourceTags": {
        "SubnetKey": "SubnetValue"
      },
      "selectionMode": "ALL",
      "parameters": {}
    },
    "VPC-Endpoint": {
      "resourceType": "aws:ec2:vpc-endpoint",
      "resourceTags": {
        "DisruptPrivateLink": "Allowed"
      },
    },
  }
}

```

```

        "selectionMode": "ALL"
    },
    "S3-Bucket": {
        "resourceType": "aws:s3:bucket",
        "resourceTags": {
            "S3Impact": "Allowed"
        },
        "selectionMode": "ALL"
    },
    "DynamoDB-Global-Table": {
        "resourceType": "aws:dynamodb:global-table",
        "resourceTags": {
            "DisruptDynamoDb": "Allowed"
        },
        "selectionMode": "ALL"
    },
    "MemoryDB-Multi-Region-Cluster": {
        "resourceType": "aws:memorydb:multi-region-cluster",
        "resourceTags": {
            "DisruptMemoryDb": "Allowed"
        },
        "selectionMode": "ALL"
    }
},
"actions": {
    "Disrupt-Transit-Gateway-Connectivity": {
        "actionId": "aws:network:transit-gateway-disrupt-cross-region-
connectivity",
        "parameters": {
            "duration": "PT3H",
            "region": "eu-west-1"
        },
        "targets": {
            "TransitGateways": "Transit-Gateway"
        }
    },
    "Disrupt-Subnet-Connectivity": {
        "actionId": "aws:network:route-table-disrupt-cross-region-
connectivity",
        "parameters": {
            "duration": "PT3H",
            "region": "eu-west-1"
        },
        "targets": {

```

```

        "Subnets": "Subnet"
    }
},
"Disrupt-Vpc-Endpoint": {
    "actionId": "aws:network:disrupt-vpc-endpoint",
    "parameters": {
        "duration": "PT3H"
    },
    "targets": {
        "VPCEndpoints": "VPC-Endpoint"
    }
},
"Pause-S3-Replication": {
    "actionId": "aws:s3:bucket-pause-replication",
    "parameters": {
        "duration": "PT3H",
        "region": "eu-west-1"
    },
    "targets": {
        "Buckets": "S3-Bucket"
    }
},
"Pause-DynamoDB-Replication": {
    "actionId": "aws:dynamodb:global-table-pause-replication",
    "parameters": {
        "duration": "PT3H"
    },
    "targets": {
        "Tables": "DynamoDB-Global-Table"
    }
},
"Pause-MemoryDB-Multi-Region-Cluster-Replication": {
    "actionId": "aws:memorydb:multi-region-cluster-pause-replication",
    "parameters": {
        "duration": "PT3H",
        "region": "eu-west-1"
    },
    "targets": {
        "MultiRegionClusters": "MemoryDB-Multi-Region-Cluster"
    }
}
},
"stopConditions": [
    {

```

```
        "source": "none"
      }
    ],
    "roleArn": "",
    "logConfiguration": {
      "logSchemaVersion": 2
    },
    "tags": {
      "Name": "Cross-Region: Connectivity"
    },
    "experimentOptions": {
      "accountTargeting": "single-account",
      "emptyTargetResolutionMode": "skip"
    },
    "description": "Block application network traffic from experiment Region to
target Region and pause cross-Region replication"
  }
}
```

Trabalhando com experimentos com várias contas para AWS FIS

Você pode criar e gerenciar modelos de experimentos com várias contas usando o AWS FIS console ou a linha de comando. Você cria um experimento com várias contas especificando a opção de experimento de segmentação de conta como "multi-account" e adicionando configurações de conta de destino. Depois de criar um modelo de experimento de várias contas, você pode usá-lo para executar um experimento.

Com um experimento com várias contas, você pode configurar e executar cenários de falha do mundo real em um aplicativo que abrange várias AWS contas em uma região. Você executa experimentos com várias contas por uma conta de orquestrador que afeta os recursos em várias contas de destino.

Quando você executa um experimento com várias contas, as contas de destino com os recursos afetados serão notificadas por meio dos painéis do AWS Health, informando os usuários nas contas de destino. Com experimentos com várias contas, você pode:

- Execute cenários de falha do mundo real em aplicativos que abrangem várias contas com os controles centrais e as grades de proteção fornecidas. AWS FIS
- Controlar os efeitos de um experimento com várias contas usando perfis do IAM com permissões e tags refinadas para definir o escopo de cada destino.
- Visualize centralmente as ações realizadas AWS FIS em cada conta a partir dos AWS FIS registros Console de gerenciamento da AWS e por meio deles.
- Monitore e audite as chamadas AWS FIS de API feitas em cada conta com a AWS CloudTrail.

Esta seção ajuda você nos conceitos básicos dos experimentos com várias contas.

Tópicos

- [Conceitos dos experimentos com várias contas](#)
- [Práticas recomendadas para experimentos com várias contas](#)
- [Pré-requisitos para experimentos com várias contas](#)
- [Criar um modelo de experimento com várias contas](#)
- [Atualizar uma configuração de conta de destino](#)
- [Excluir uma configuração da conta de destino](#)

Conceitos dos experimentos com várias contas

Veja a seguir os principais conceitos de experimentos de várias contas:

- **Conta do orquestrador** - A conta do orquestrador atua como uma conta central para configurar e gerenciar o experimento no AWS FIS console, bem como para centralizar o registro. A conta do orquestrador possui o modelo e o AWS FIS experimento do experimento.
- **Contas de destino** - Uma conta de destino é uma conta individual da AWS com recursos que podem ser afetados por um experimento com AWS FIS várias contas.
- **Configurações das contas de destino**: você define as contas de destino que fazem parte de um experimento adicionando configurações de conta de destino ao modelo de experimento. A configuração da conta de destino é um elemento do modelo de experimento que é necessário para experimentos com várias contas. Você define uma para cada conta de destino definindo um ID AWS da conta, uma função do IAM e uma descrição opcional.

Práticas recomendadas para experimentos com várias contas

Veja a seguir as práticas recomendadas para usar experimentos com várias contas:

- Ao configurar destinos para experimentos com várias contas, recomendamos segmentar com tags de recursos consistentes em todas as contas de destino. Um AWS FIS experimento resolverá recursos com tags consistentes em cada conta de destino. Uma ação deve resolver pelo menos um recurso de destino em qualquer conta de destino ou falhará, exceto nos experimentos com `emptyTargetResolutionMode` definido como `skip`. As cotas de ação são aplicáveis por conta. Se você quiser segmentar recursos por recurso ARNs, aplica-se o mesmo limite de conta única por ação.
- Ao direcionar recursos em uma ou mais zonas de disponibilidade usando parâmetros ou filtros, você deve especificar uma ID de AZ, não um nome de AZ. A ID da AZ é um identificador exclusivo e consistente de uma zona de disponibilidade em todas as contas. Para saber como encontrar o ID AZ para as zonas de disponibilidade em sua conta, consulte [Zona de disponibilidade IDs para seus recursos da AWS](#).

Pré-requisitos para experimentos com várias contas

Para usar condições de interrupção em um experimento com várias contas, você deve primeiro configurar alarmes entre contas. Os perfis do IAM são definidos quando você cria um modelo de

experimento com várias contas. Você pode criar os perfis do IAM necessários antes de criar o modelo.

Conteúdo

- [Permissões para experimentos com várias contas](#)
- [Condições de interrupção para experimentos com várias contas \(opcional\)](#)
- [Medidas de segurança para experimentos em várias contas \(opcional\)](#)

Permissões para experimentos com várias contas

Experimentos com várias contas usam o encadeamento de perfis do IAM para conceder permissões ao AWS FIS para realizar ações em recursos nas contas de destino. Para experimentos com várias contas, você configura perfis do IAM em cada conta de destino e na conta de orquestrador. Esses perfis do IAM exigem uma relação de confiança entre as contas de destino e a conta de orquestrador, e entre a conta de orquestrador e o AWS FIS.

Os perfis do IAM para as contas de destino contêm as permissões necessárias para agir sobre os recursos e são criados para um modelo de experimento ao adicionar configurações da conta de destino. Você criará um perfil do IAM para a conta de orquestrador com permissão para assumir os perfis das contas de destino e estabelecer uma relação de confiança com o AWS FIS. Esse perfil do IAM é usado como o `roleArn` do modelo de experimento.

Para saber mais sobre o encadeamento de perfis, consulte [Termos e conceitos das funções](#) no Guia do usuário do IAM.

No exemplo a seguir, você configurará permissões para uma conta de orquestrador A para realizar um experimento com `aws:ebs:pause-volume-io` na conta de destino B.

1. Na conta B, crie um perfil do IAM com as permissões necessárias para executar a ação. Consulte as permissões necessárias para cada ação em [Referência das ações](#). O exemplo a seguir mostra as permissões que uma conta de destino concede para executar a ação de E/S de pausar volume do EBS [the section called “aws:ebs:pause-volume-io”](#).

JSON

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

```

    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeVolumes"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:PauseVolumeIO"
      ],
      "Resource": "arn:aws:ec2:us-east-1:123456789012:volume/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "tag:GetResources"
      ],
      "Resource": "*"
    }
  ]
}

```

- Depois, adicione uma política de confiança na conta B que crie uma relação de confiança com a conta A. Escolha um nome para o perfil do IAM na conta A, que você criará na etapa 3.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "AccountIdA"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringLike": {
          "sts:ExternalId":
            "arn:aws:fis:region:accountIdA:experiment/*"
        }
      },
      "ArnEquals": {

```

```

        "aws:PrincipalArn":
        "arn:aws:iam::111122223333:role/role_name"
    }
}
]
}

```

3. Na conta A, crie um perfil do IAM. O nome desse perfil deve corresponder ao perfil especificado na política de confiança na etapa 2. Para atingir várias contas, você concede ao orquestrador permissões para assumir cada perfil. O exemplo a seguir mostra as permissões da conta A para assumir a conta B. Se você tiver contas de destino adicionais, adicionará uma função adicional ARNs a essa política. Você só pode ter um ARN de perfil por conta de destino.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": [
        "arn:aws:iam::111122223333:role/role_name"
      ]
    }
  ]
}

```

4. Esse perfil do IAM da conta A é usado como o `roleArn` do modelo de experimento. O exemplo a seguir mostra a política de confiança exigida na função do IAM que concede AWS FIS permissões para assumir a conta A, a conta do orquestrador.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "fis.amazonaws.com"
        ]
      }
    }
  ]
}

```

```
    ],
    },
    "Action": "sts:AssumeRole"
  }
]
```

Você também pode usar o StackSets para provisionar vários perfis do IAM ao mesmo tempo. Para usar CloudFormation StackSets, você precisará configurar as StackSet permissões necessárias em suas AWS contas. Para saber mais, consulte Como [trabalhar com a AWS CloudFormation StackSets](#).

Condições de interrupção para experimentos com várias contas (opcional)

Uma condição de parada é um mecanismo para interromper um experimento se ele atingir um limite que você define como um alarme. Para configurar uma condição de interrupção para o experimento com várias contas, você pode usar alarmes entre contas. É necessário habilitar o compartilhamento em cada conta de destino a fim de disponibilizar o alarme para a conta de orquestrador usando permissões somente leitura. Depois de compartilhadas, você pode combinar métricas de diferentes contas de destino usando a matemática de métricas. A seguir, você pode adicionar esse alarme como condição de interrupção para o experimento.

Para saber mais sobre painéis de várias contas, consulte [Habilitando a funcionalidade de várias contas em](#) CloudWatch

Medidas de segurança para experimentos em várias contas (opcional)

As medidas de segurança são usadas para interromper todos os experimentos em execução e impedir o início de novos experimentos. Talvez você queira usar uma medida de segurança para evitar experimentos do FIS durante determinados períodos ou em resposta a alarmes de integridade da aplicação. Cada AWS conta tem uma alavanca de segurança por Região da AWS. Quando uma medida de segurança é acionada, ela afeta todos os experimentos em execução na mesma conta e região da medida de segurança. Para interromper e impedir experimentos em várias contas, a medida de segurança deve ser acionada na mesma conta e região em que os experimentos estão sendo executados.

Criar um modelo de experimento com várias contas

Para saber como criar um modelo de experimento por meio do Console de gerenciamento da AWS

Consulte [Criar um modelo de experimento](#).

Para criar um modelo de experimento usando a CLI

1. Abra o AWS Command Line Interface
2. Para criar um experimento a partir de um arquivo JSON salvo com a opção de experimento de segmentação de conta definida como "multi-account" (por exemplo, `my-template.json`), substitua os valores do espaço reservado por seus próprios valores e execute o comando a seguir. *italics* [create-experiment-template](#)

```
aws fis create-experiment-template --cli-input-json file://my-template.json
```

Isso retornará o modelo de experimento na resposta. Copie a id da resposta, que é a ID do modelo de experimento.

3. Execute o [create-target-account-configuration](#) comando para adicionar uma configuração de conta de destino ao modelo de experimento. Substitua os valores do espaço reservado por seus próprios valores, usando id a etapa 2 como valor para o `--experiment-template-id` parâmetro e, em seguida, execute o seguinte. *italics* O parâmetro `--description` é opcional. Repita essa etapa para cada conta de destino.

```
aws fis create-target-account-configuration --experiment-template-id EXTxxxxxxxxx  
--account-id 111122223333 --role-arn arn:aws:iam::111122223333:role/role-name --  
description "my description"
```

4. Execute o [get-target-account-configuration](#) comando para recuperar os detalhes de uma configuração específica da conta de destino.

```
aws fis get-target-account-configuration --experiment-template-id EXTxxxxxxxxx --  
account-id 111122223333
```

5. Depois de adicionar todas as configurações da conta de destino, você pode executar o [list-target-account-configurations](#) comando de comando para ver se as configurações da conta de destino foram criadas.

```
aws fis list-target-account-configurations --experiment-template-id EXTxxxxxxxxx
```

Você também pode verificar se adicionou as configurações da conta de destino executando o [get-experiment-template](#) comando. O modelo retornará um campo somente leitura

`targetAccountConfigurationsCount` que é uma contagem de todas as configurações da conta de destino no modelo de experimento.

- Quando estiver tudo pronto, você poderá executar o modelo de experimento usando o comando [start-experiment](#).

```
aws fis start-experiment --experiment-template-id EXTxxxxxxxxx
```

Atualizar uma configuração de conta de destino

Você poderá atualizar uma configuração de conta de destino existente se quiser alterar o ARN do perfil ou a descrição da conta. Quando você atualiza uma configuração de conta de destino, as alterações não afetam nenhum experimento em execução que use o modelo.

Para atualizar a configuração de uma conta de destino usando o Console de gerenciamento da AWS

- Abra o AWS FIS console em <https://console.aws.amazon.com/fis/>.
- No painel de navegação, selecione Modelos de experimento.
- Selecione o modelo do experimento e escolha Ações, Atualizar modelo de experimento.
- No painel lateral, escolha Etapa 3, Configurar o acesso ao serviço.
- Modifique as configurações da conta de destino e escolha Atualizar modelo de experimento.
- Selecione a Etapa 5, Revisar e criar.

Para atualizar a configuração de uma conta de destino usando a CLI

Execute o [update-target-account-configuration](#) comando a comando, substituindo os valores do espaço reservado por seus próprios valores. *italics* Os parâmetros `--role-arn` e `--description` são opcionais e não serão atualizados se não forem incluídos.

```
aws fis update-target-account-configuration --experiment-template-id EXTxxxxxxxxx  
--account-id 111122223333 --role-arn arn:aws:iam::111122223333:role/role-name --  
description "my description"
```

Excluir uma configuração da conta de destino

Se não precisar mais de uma configuração de conta de destino, você poderá excluí-la. Quando você exclui uma configuração da conta de destino, qualquer experimento em execução que use o modelo não é afetado. O experimento continua em andamento até ser concluído ou interrompido.

Para excluir uma configuração de conta de destino usando o Console de gerenciamento da AWS

1. Abra o AWS FIS console em <https://console.aws.amazon.com/fis/>.
2. No painel de navegação, selecione Modelos de experimento.
3. Selecione o modelo do experimento e escolha Ações, Atualizar.
4. No painel lateral, escolha Etapa 3, Configurar o acesso ao serviço.
5. Em Configurações da conta de destino, selecione Remover para o ARN do perfil da conta de destino que você deseja excluir.
6. Selecione a Etapa 5, Revisar e criar.
7. Revise o modelo e escolha Atualizar modelo de experimento. Quando a confirmação for solicitada, insira `update` e escolha Atualizar modelo de experimento.

Para excluir uma configuração da conta de destino usando a CLI

Execute o [delete-target-account-configuration](#) comando, substituindo os valores do espaço reservado por seus próprios valores. *italics*

```
aws fis update-target-account-configuration --experiment-template-id EXTxxxxxxxxx --  
account-id 111122223333
```

Agendar experimentos

Com o AWS Fault Injection Service (FIS), você pode realizar experimentos de injeção de falhas em suas workloads da AWS. Esses experimentos são executados em modelos que contêm uma ou mais ações a serem executadas em alvos específicos. Agora você pode agendar seus experimentos como uma tarefa única ou tarefas recorrentes diretamente do console do FIS. Além das [regras programadas](#), o FIS agora oferece um novo recurso de agendamento. O FIS agora se integra ao EventBridge Scheduler e cria regras em seu nome. EventBridge O Scheduler é um agendador sem servidor que permite criar, executar e gerenciar tarefas a partir de um serviço gerenciado central.

Important

O Experiment Scheduler with não AWS Fault Injection Service está disponível na AWS GovCloud (Leste dos EUA) e na AWS GovCloud (Oeste dos EUA).

Tópicos

- [Criar um perfil de agendador](#)
- [Criar um agendamento de experimento](#)
- [Atualizar um agendamento de experimento](#)
- [Desabilitar ou excluir um agendamento de experimento](#)

Criar um perfil de agendador

Uma função de execução é uma função do IAM que é AWS FIS assumida para interagir com o EventBridge agendador e para o programador do Event Bridge iniciar o experimento FIS. Você anexa políticas de permissão a essa função para conceder ao EventBridge Scheduler acesso para invocar o FIS Experiment. As etapas a seguir descrevem como criar uma nova função de execução e uma política EventBridge para permitir iniciar um experimento.

Criar uma função do programador usando a CLI da AWS

Esse é o perfil do IAM necessário para que o Event Bridge possa agendar experimentos em nome do cliente.

1. Copie a seguinte política JSON de assumir a função e salve-a localmente como `fis-execution-role.json`. Essa política de confiança permite que o EventBridge Scheduler assumira a função em seu nome.

JSON

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

2. No AWS Command Line Interface (AWS CLI), insira o seguinte comando para criar uma nova função. Substitua `FisSchedulerExecutionRole` pelo nome que você deseja atribuir a essa função.

```
aws iam create-role --role-name FisSchedulerExecutionRole --assume-role-policy-document file://fis-execution-role.json
```

Se o teste for bem-sucedido, você verá o seguinte resultado:

```
{
  "Role": {
    "Path": "/",
    "RoleName": "FisSchedulerExecutionRole",
    "RoleId": "AROAZL22PDN5A6WKRBNUN",
    "Arn": "arn:aws:iam::123456789012:role/FisSchedulerExecutionRole",
    "CreateDate": "2023-08-24T17:23:05+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
```

```

        "Service": "scheduler.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
}
}
}

```

- Para criar uma nova política que permita ao EventBridge Scheduler invocar o experimento, copie o seguinte JSON e salve-o localmente como `fis-start-experiment-permissions.json`. A política a seguir permite que o EventBridge Scheduler execute a `fis:StartExperiment` ação em todos os modelos de experimentos em sua conta. Substitua o `*` no final do `"arn:aws:fis:*:*:experiment-template/*"` pelo ID do seu modelo de experimento se quiser limitar a função a um único modelo de experimento.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "fis:StartExperiment",
      "Resource": [
        "arn:aws:fis:*:*:experiment-template/*",
        "arn:aws:fis:*:*:experiment/*"
      ]
    }
  ]
}

```

- Execute o seguinte comando para criar a nova política de permissão. Substitua `FisSchedulerPolicy` pelo nome que você deseja atribuir a essa política.

```
aws iam create-policy --policy-name FisSchedulerPolicy --policy-document file://fis-start-experiment-permissions.json
```

Se for bem-sucedido, você verá o seguinte resultado: Observe o ARN da política. Você usa esse ARN na próxima etapa para anexar a política à nossa função de execução.

```
{
  "Policy": {
    "PolicyName": "FisSchedulerPolicy",
    "PolicyId": "ANPAZL22PDN5ESVUWXLBD",
    "Arn": "arn:aws:iam::123456789012:policy/FisSchedulerPolicy",
    "Path": "/",
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
    "PermissionsBoundaryUsageCount": 0,
    "IsAttachable": true,
    "CreateDate": "2023-08-24T17:34:45+00:00",
    "UpdateDate": "2023-08-24T17:34:45+00:00"
  }
}
```

- Para associar a política à sua função de execução, execute o comando a seguir. Substitua `your-policy-arn` pelo ARN da política que você criou na etapa anterior. Substitua `FisSchedulerExecutionRole` pelo nome da sua função de execução.

```
aws iam attach-role-policy --policy-arn your-policy-arn --role-name
FisSchedulerExecutionRole
```

A operação `attach-role-policy` não retorna uma resposta na linha de comando.

- Você pode restringir o agendador a executar somente modelos de AWS FIS experimentos que tenham um valor de tag específico. Por exemplo, a política a seguir concede a `fis:StartExperiment` permissão para todos os AWS FIS experimentos, mas restringe o agendador a executar somente modelos de experimentos marcados. `Purpose=Schedule`
JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "fis:StartExperiment",
      "Resource": "arn:aws:fis:*:*:experiment/*"
    },
    {
      "Effect": "Allow",
```

```
"Action": "fis:StartExperiment",
"Resource": "arn:aws:fis:*:*:experiment-template/*",
"Condition": {
  "StringEquals": {
    "aws:ResourceTag/Purpose": "Schedule"
  }
}
]
```

Criar um agendamento de experimento

Antes de programar um experimento, você precisa de um ou mais [Componentes do modelo de experimento](#) para sua programação invocar. Você pode usar um recurso existente da AWS ou criar um novo.

Depois que o modelo de experimento for criado, clique em Ações e selecione Programar experimento. Você será redirecionado para a página do cronograma. O nome do cronograma será preenchido para você.

Siga até a seção de padrões de programação e escolha entre programação única ou recorrente. Preencha os campos de entrada obrigatórios e navegue até as permissões.

The screenshot shows the AWS FIS console interface for configuring a schedule pattern. The page is titled "Schedule pattern" and includes the following sections:

- Occurrence:** A section with an "Info" link and the text "You can define an one-time or recurrent schedule." It contains two radio buttons: "One-time schedule" (which is selected) and "Recurring schedule".
- Date and time:** A section with the text "The date and time to invoke the target." It contains three input fields: a date field with a calendar icon (placeholder "YYYY/MM/DD"), a time field (placeholder "hh:mm"), and a timezone dropdown menu (placeholder "(UTC -04:00) America/New...").
- Flexible time window:** A section with the text "If you choose a flexible time window, Scheduler invokes your schedule within the time window you specify. For example, if you choose 15 minutes, your schedule runs within 15 minutes after the schedule start time." It contains a dropdown menu with the placeholder "Select".
- Schedule state:** A section with the text "Enable schedule" and "You can choose not to enable the schedule now. You will be able to enable the schedule after it has been created." It contains a radio button labeled "Enable" which is selected.

O estado do cronograma será habilitado por padrão. Observação: se você desativar o estado do cronograma, o experimento não será programado, mesmo que você crie um cronograma.

AWS FIS O Experiment Scheduler é construído sobre o [EventBridge Scheduler](#). Você pode consultar a documentação dos vários [tipos de programação compatíveis](#).

Para atualizar uma programação usando o console

1. Abra o [console do AWS FIS](#).
2. No painel de navegação à esquerda, escolha Modelos de experimento.
3. Escolha o modelo de experimento para o qual você deseja criar o cronograma.
4. Clique em Ações e selecione Programar experimento no menu suspenso.
 - a. Em Nome do cronograma, o nome é preenchido automaticamente.
 - b. Em Padrão de programação, selecione Programação recorrente.
 - c. Em Tipo de programação, você pode selecionar uma programação baseada em taxa, consulte os [tipos de programação](#).
 - d. Em Expressão rate, escolha uma taxa que seja mais lenta do que o runtime do seu experimento, por exemplo, 5 minutos.
 - e. Em Cronograma, selecione seu Fuso horário.
 - f. Em Data e hora de início, especifique uma data e hora de início.
 - g. Em Data e hora de término, especifique uma data e hora de término.
 - h. Em Estado da programação, ative a opção Habilitar programação.
 - i. Em Permissões, selecione Usar função existente, e, em seguida, pesquise por `FisSchedulerExecutionRole`.
 - j. Escolha Avançar.
5. Selecione Revisar e criar cronograma, revise os detalhes do programador e escolha Criar cronograma.

Atualizar um agendamento de experimento

É possível atualizar um cronograma de evento para que ele ocorra em uma data e hora específicas que forem convenientes.

Para atualizar a execução de um experimento usando o console

1. Abra o [console do Amazon FIS](#).
2. No painel de navegação, selecione Modelos de experimento.
3. Escolha o tipo de recurso: modelo de experimento para o qual um cronograma já foi criado.
4. Clique no ID do experimento para ver o modelo. Em seguida, navegue até a guia de programações.
5. Verifique se há um cronograma associado ao experimento. Selecione a programação associada e clique no botão Atualizar cronograma.

Desabilitar ou excluir um agendamento de experimento

Para impedir que um experimento seja executado ou executado de acordo com um cronograma, você pode excluir ou desativar a regra. As etapas a seguir explicam como excluir ou desativar a execução de um experimento usando o AWS console.

Para excluir uma desativar uma regra

1. Abra o [console do Amazon FIS](#).
2. No painel de navegação, selecione Modelos de experimento.
3. Escolha o tipo de recurso: modelo de experimento para o qual um cronograma já foi criado.
4. Clique no ID do experimento para ver o modelo. Em seguida, navegue até a guia de programações.
5. Verifique se há um cronograma associado ao experimento. Selecione a programação associada e clique no botão Atualizar cronograma.
6. Execute um destes procedimentos:
 - a. Para excluir o cronograma, selecione o botão ao lado da regra Excluir cronograma. Digite delete e clique no botão Excluir cronograma.
 - b. Para desativar o cronograma, selecione o botão ao lado da regra Desativar cronograma. Digite disable e clique no botão Desativar cronograma.

Alavancas de segurança para AWS FIS

As medidas de segurança são usadas para interromper todos os experimentos em execução e impedir o início de novos experimentos. Talvez você queira usar uma medida de segurança para evitar experimentos do FIS durante determinados períodos ou em resposta a alarmes de integridade da aplicação. Cada AWS conta tem uma alavanca de segurança por Região da AWS

Para experimentos em andamento que são interrompidos pela medida de segurança, você paga apenas pela duração da ação que foi executada até a interrupção do experimento. Os experimentos que são impedidos de começar não incorrerão em nenhum custo. As seções a seguir fornecem informações sobre como começar a usar as medidas de segurança.

Tópicos

- [Conceitos para medidas de segurança](#)
- [Trabalhar com medidas de segurança](#)

Conceitos para medidas de segurança

Uma medida de segurança pode ser acionada ou desacionada.

- Se for desacionada, os experimentos do FIS serão permitidos. Por padrão, as medidas de segurança ficam desacionadas.
- Se for acionada, os experimentos em andamento serão interrompidos e nenhum novo experimento poderá ser iniciado.

Um experimento afetado por uma medida de segurança terminará em um dos seguintes status:

- Interrompido, se o experimento estava em andamento quando a medida de segurança foi acionada.
- Cancelado, se o experimento foi iniciado quando a medida de segurança já estava acionada.

Não é possível retomar ou executar novamente um experimento que tenha sido interrompido ou cancelado. No entanto, você pode iniciar um novo experimento usando o mesmo modelo de experimento quando a medida de segurança for desacionada.

Recurso das medidas de segurança

As medidas de segurança são um recurso definido pelo nome do recurso da Amazon (ARN). As medidas de segurança incluem os seguintes parâmetros:

- Status, que é Acionada ou Desacionada.
- Motivo, que é uma string inserida pelo usuário para registrar por que o status da medida de segurança foi alterado.

Trabalhar com medidas de segurança

Esta seção detalhará como visualizar, acionar e desengatar as alavancas de segurança usando o AWS FIS console ou a linha de comando.

Visualizar uma medida de segurança

Você pode ver o estado de uma medida de segurança para sua conta e região seguindo as etapas abaixo.

Como visualizar uma medida de segurança usando o console

1. [Abra o AWS FIS console](#)
2. No painel de navegação, selecione Experimentos.
3. Se a medida de segurança estiver acionada, você verá um banner de alerta na parte superior da página. Se não houver nenhum banner de alerta, a medida de segurança estará desacionada.

Como visualizar uma medida de segurança usando a CLI

- Use o seguinte comando:

```
aws fis get-safety-lever --id "default"
```

Uma medida de segurança pode estar em um dos seguintes estados:

- Desacionada: a medida de segurança não está afetando nenhum experimento. Os experimentos podem ser executados livremente. As medidas de segurança ficam desacionadas por padrão.

- **Acionando:** a medida de segurança está mudando de desacionada para acionada. Pode haver experimentos que ainda não foram interrompidos. A medida de segurança não poderá ser trocada enquanto estiver nesse estado.
- **Acionada:** a medida de segurança está ativa e nenhum experimento está em andamento. Todo novo experimento que tentar começar com a medida de segurança acionada será cancelado.

Acionar uma medida de segurança

Como acionar uma medida de segurança usando o console

1. [Abra o AWS FIS console](#)
2. No painel de navegação, selecione Experimentos.
3. Escolha o botão Interromper todos os experimentos.
4. Insira o motivo para acionar a medida de segurança.
5. Escolha Confirmar.

Como acionar uma medida de segurança usando a CLI

- Use o seguinte comando. Preencha o campo do motivo com sua resposta.

```
aws fis update-safety-lever-state --id "default" --state  
"status=engaged,reason=xxxxx"
```

Desacionar uma medida de segurança

Como desacionar uma medida de segurança usando o console

1. [Abra o AWS FIS console](#)
2. No painel de navegação, selecione Experimentos.
3. Escolha o botão Desacionar medida de segurança.
4. Insira o motivo para desacionar a medida de segurança.
5. Escolha Confirmar.

Como desacionar uma medida de segurança usando a CLI

- Use o seguinte comando:

```
aws fis update-safety-lever-state --id "default" --state  
"status=disengaged,reason=recovered"
```

Monitorando AWS experimentos FIS

Você pode usar as seguintes ferramentas para monitorar o progresso e o impacto de seus experimentos do AWS AWS Fault Injection Service (FIS).

AWS Console FIS e AWS CLI

Use o console AWS FIS ou o AWS CLI para monitorar o progresso de um experimento em execução. Você pode ver o status de cada ação no experimento e os resultados de cada ação. Para obter mais informações, consulte [the section called “Visualizar seus experimentos”](#).

CloudWatch métricas de uso e alarmes

Use métricas CloudWatch de uso para fornecer visibilidade sobre o uso dos recursos da sua conta. AWS As métricas de uso do FIS correspondem às cotas AWS de serviço. Também é possível configurar alarmes que alertem você quando o uso se aproximar de uma cota de serviço. Para obter mais informações, consulte [Monitore usando CloudWatch](#).

Você também pode criar condições de parada para seus experimentos AWS FIS criando CloudWatch alarmes que definem quando um experimento ultrapassa os limites. Quando o alarme é acionado, o experimento é interrompido. Para obter mais informações, consulte [Condições de parada](#). Para obter mais informações sobre a criação de CloudWatch alarmes, consulte [Criar um CloudWatch alarme com base em um limite estático](#) e [Criar um CloudWatch alarme com base na detecção de anomalias no](#) Guia do usuário da Amazon CloudWatch .

AWS Registro de experimentos do FIS

Habilite o registro de experimentos para capturar informações detalhadas sobre seu experimento à medida que ele é executado. Para ter mais informações, consulte [Registro em log do experimento](#).

Eventos de alteração no estado da experiência

A Amazon EventBridge permite que você responda automaticamente a eventos do sistema ou alterações de recursos. AWS O FIS emite uma notificação quando o estado de um experimento muda. É possível criar regras para os eventos de seu interesse para especificar a ação automatizada a ser executada quando um evento corresponder a uma regra. Por exemplo, enviar uma notificação para um tópico do Amazon SNS ou invocar uma função do Lambda. Para obter mais informações, consulte [Monitore usando EventBridge](#).

CloudTrail troncos

Use AWS CloudTrail para capturar informações detalhadas sobre as chamadas feitas para a API AWS FIS e armazená-las como arquivos de log no Amazon S3. CloudTrail também registra as chamadas feitas ao serviço APIs para os recursos nos quais você está executando experimentos. Você pode usar esses CloudTrail registros para determinar quais chamadas foram feitas, o endereço IP de origem da chamada, quem fez a chamada, quando a chamada foi feita e assim por diante.

AWS Notificações do Health Dashboard

O AWS Health fornece visibilidade contínua sobre o desempenho de seus recursos e a disponibilidade de seus AWS serviços e contas. Quando você inicia um experimento, o AWS FIS emite uma notificação para o seu AWS Health Dashboard. A notificação fica presente durante o experimento em cada conta que possui recursos-alvo de um experimento, incluindo experimentos com várias contas. Experimentos com várias contas com apenas ações que não incluem destinos, como `aws:ssm:start-automation-execution` e `aws:fis:wait`, não emitirão uma notificação. As informações sobre o perfil usado para permitir o experimento serão listadas em Recursos afetados. Para saber mais sobre o AWS Health Dashboard, consulte [AWS Health Dashboard](#) no Guia do usuário do AWS Health.

Note

O AWS Health entrega eventos em uma base de melhor esforço.

Monitore AWS as métricas de uso do FIS usando a Amazon CloudWatch

Você pode usar CloudWatch a Amazon para monitorar o impacto dos experimentos do AWS FIS nos alvos. Você também pode monitorar seu uso AWS do FIS.

Para obter mais informações sobre como visualizar o estado de um experimento, consulte [Visualizar seus experimentos](#).

Monitore os AWS experimentos do FIS

Ao planejar seus experimentos do AWS FIS, identifique as CloudWatch métricas que você pode usar para identificar a linha de base ou o “estado estacionário” dos tipos de recursos de destino do

experimento. Depois de iniciar um experimento, você pode monitorar essas CloudWatch métricas para os alvos selecionados por meio do modelo de experimento.

Para obter mais informações sobre as CloudWatch métricas disponíveis para um tipo de recurso de destino suportado pelo AWS FIS, consulte o seguinte:

- [Monitore suas instâncias usando CloudWatch](#)
- [Métricas do Amazon ECS CloudWatch](#)
- [Monitoramento de métricas do Amazon RDS usando CloudWatch](#)
- [Monitorando métricas do Run Command usando CloudWatch](#)

AWS Métricas de uso do FIS

Você pode usar métricas de CloudWatch uso para dar visibilidade ao uso dos recursos da sua conta. Use essas métricas para visualizar seu uso atual do serviço em CloudWatch gráficos e painéis.

AWS As métricas de uso do FIS correspondem às cotas AWS de serviço. Também é possível configurar alarmes que alertem você quando o uso se aproximar de uma cota de serviço. Para obter mais informações sobre CloudWatch alarmes, consulte o [Guia do CloudWatch usuário da Amazon](#).

AWS O FIS publica a seguinte métrica no namespace AWS/Usage.

Métrica	Descrição
ResourceCount	O número total dos recursos especificados em execução na sua conta. Os recursos são definidos pelas dimensões associadas à métrica.

As dimensões a seguir são usadas para refinar as métricas de uso publicadas pela AWS FIS.

Dimensão	Descrição
Service	O nome do AWS serviço que contém o recurso. Para métricas de uso do AWS FIS, o valor dessa dimensão é FIS.

Dimensão	Descrição
Type	O tipo de entidade que está sendo relatado. Atualmente, o único valor válido para as métricas de uso do AWS FIS é <code>Resource</code> .
Resource	O tipo de recurso que está em execução. Os valores possíveis são <code>ExperimentTemplates</code> para modelos de experimentos e <code>ActiveExperiments</code> para experimentos ativos.
Class	Essa dimensão está reservada para uso futuro.

Monitore experimentos AWS do FIS usando a Amazon EventBridge

Quando o estado de um experimento muda, o AWS FIS emite uma notificação. Essas notificações são disponibilizadas como eventos pela Amazon EventBridge (anteriormente chamada de CloudWatch Eventos). O AWS FIS emite esses eventos com base no melhor esforço. Os eventos são entregues quase EventBridge em tempo real.

Com EventBridge, você pode criar regras que acionam ações programáticas em resposta a um evento. Por exemplo, é possível configurar uma regra que invoque um tópico do SNS para enviar uma notificação por e-mail ou que invoque uma função do Lambda para realizar alguma ação.

Para obter mais informações sobre EventBridge, consulte [Introdução à Amazon EventBridge](#) no Guia do EventBridge usuário da Amazon.

A seguir está a sintaxe de um evento de mudança de estado do experimento:

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "FIS Experiment State Change",
  "source": "aws.fis",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "region",
  "resources": [
```

```
    "arn:aws:fis:region:account_id:experiment/experiment-id"
  ],
  "detail": {
    "experiment-id": "EXPabcd1efg2HIJKL3",
    "experiment-template-id": "EXTa1b2c3de5f6g7h",
    "new-state": {
      "status": "new_value",
      "reason": "reason_string"
    },
    "old-state": {
      "status": "old_value",
      "reason": "reason_string"
    }
  }
}
```

experiment-id

O ID do experimento cujo estado mudou.

experiment-template-id

A ID do modelo de experimento usado pelo experimento.

new_value

O novo estado do experimento. Os valores possíveis são:

- completed
- failed
- initiating
- running
- stopped
- stopping

old_value

O estado anterior do experimento. Os valores possíveis são:

- initiating
- pending
- running

- stopping

Registro de experimentos para AWS FIS

Você pode usar o registro de experimentos para capturar informações detalhadas sobre seu experimento à medida que ele é executado.

Você é cobrado pelo registro do experimento com base nos custos associados a cada tipo de destino do registro. Para obter mais informações, consulte [Amazon CloudWatch Pricing](#) (em Paid Tier, Logs, Vended Logs) e [Amazon S3 Pricing](#).

Permissões

Você deve conceder permissões AWS do FIS para enviar registros para cada destino de log que você configurar. Para obter mais informações, consulte o seguinte no Guia do usuário do Amazon CloudWatch Logs:

- [Registros enviados para CloudWatch Logs](#)
- [Logs enviados ao Amazon S3](#)

Esquema de logs

A seguir está o esquema usado no registro de logs de experimentos. A versão atual do esquema é a 2. Os campos para `details` dependem do valor de `log_type`. Os campos para `resolved_targets` dependem do valor de `target_type`. Para obter mais informações, consulte [the section called “Exemplo de registros de log”](#).

```
{
  "id": "EXP123abc456def789",
  "log_type": "experiment-start | target-resolution-start | target-resolution-detail
| target-resolution-end | action-start | action-error | action-end | experiment-end",
  "event_timestamp": "yyyy-mm-ddThh:mm:ssZ",
  "version": "2",
  "details": {
    "account_id": "123456789012",
    "action_end_time": "yyyy-mm-ddThh:mm:ssZ",
    "action_id": "String",
    "action_name": "String",
    "action_start_time": "yyyy-mm-ddThh:mm:ssZ",
```

```

    "action_state": {
      "status": "pending | initiating | running | completed | cancelled |
stopping | stopped | failed",
      "reason": "String"
    },
    "action_targets": "String to string map",
    "error_information": "String",
    "experiment_end_time": "yyyy-mm-ddThh:mm:ssZ",
    "experiment_state": {
      "status": "pending | initiating | running | completed | stopping | stopped
| failed",
      "reason": "String"
    },
    "experiment_start_time": "yyyy-mm-ddThh:mm:ssZ",
    "experiment_template_id": "String",
    "page": Number,
    "parameters": "String to string map",
    "resolved_targets": [
      {
        "field": "value"
      }
    ],
    "resolved_targets_count": Number,
    "status": "failed | completed",
    "target_name": "String",
    "target_resolution_end_time": "yyyy-mm-ddThh:mm:ssZ",
    "target_resolution_start_time": "yyyy-mm-ddThh:mm:ssZ",
    "target_type": "String",
    "total_pages": Number,
    "total_resolved_targets_count": Number
  }
}

```

Notas de lançamento

- A versão 2 inclui:
 - O `target_type` campo e altera o `resolved_targets` campo de uma lista de ARNs para uma lista de objetos. Os campos válidos para o objeto `resolved_targets` dependem do valor de `target_type`, que é o [tipo de recurso](#) dos destinos.
 - Os tipos de eventos `action-error` e `target-resolution-detail` que adicionam o campo `account_id`.

- A versão 1 é a versão inicial.

Destinos de logs

AWS O FIS suporta a entrega de registros para os seguintes destinos:

- Um bucket do Amazon S3
- Um grupo de CloudWatch registros do Amazon Logs

Entrega de logs do S3

Os logs são entregues no local a seguir.

```
bucket-and-optional-prefix/AWSLogs/account-id/fis/region/experiment-id/YYYY/MM/DD/account-id_awsfislogs_region_experiment-id_YYYYMMDDHHMMZ_hash.log
```

Pode levar alguns minutos para que os logs sejam entregues ao bucket.

CloudWatch Entrega de registros e registros

Os registros são enviados para um fluxo de registros named `/aws/fis/experiment-id`.

Os logs são entregues ao grupo de logs em menos de um minuto.

Exemplo de registros de log

Veja a seguir exemplos de registros de log de um experimento que executa a `aws:ec2:reboot-instances` ação em uma EC2 instância selecionada aleatoriamente.

Registros

- [experiment-start](#)
- [target-resolution-start](#)
- [target-resolution-detail](#)
- [target-resolution-end](#)
- [action-start](#)
- [action-end](#)

- [action-error](#)
- [experiment-end](#)

experiment-start

O seguinte é um exemplo de registro para o evento `experiment-start`.

```
{
  "id": "EXPhjAXCGY78HV2a4A",
  "log_type": "experiment-start",
  "event_timestamp": "2023-05-31T18:50:45Z",
  "version": "2",
  "details": {
    "experiment_template_id": "EXTCDh1M8HHkhxoaQ",
    "experiment_start_time": "2023-05-31T18:50:43Z"
  }
}
```

target-resolution-start

O seguinte é um exemplo de registro para o evento `target-resolution-start`.

```
{
  "id": "EXPhjAXCGY78HV2a4A",
  "log_type": "target-resolution-start",
  "event_timestamp": "2023-05-31T18:50:45Z",
  "version": "2",
  "details": {
    "target_resolution_start_time": "2023-05-31T18:50:45Z",
    "target_name": "EC2InstancesToReboot"
  }
}
```

target-resolution-detail

O seguinte é um exemplo de registro para o evento `target-resolution-detail`. Se a resolução do destino falhar, o registro também incluirá o campo `error_information`.

```
{
```

```
"id": "EXPhjAXCGY78HV2a4A",
"log_type": "target-resolution-detail",
"event_timestamp": "2023-05-31T18:50:45Z",
"version": "2",
"details": {
  "target_resolution_end_time": "2023-05-31T18:50:45Z",
  "target_name": "EC2InstancesToReboot",
  "target_type": "aws:ec2:instance",
  "account_id": "123456789012",
  "resolved_targets_count": 2,
  "status": "completed"
}
}
```

target-resolution-end

Se a resolução do destino falhar, o registro também incluirá o campo `error_information`. Se `total_pages` for maior que 1, o número de destinos resolvidos excedeu o limite de tamanho para um registro. Há `target-resolution-end` registros adicionais que contêm os destinos resolvidos restantes.

Veja a seguir um exemplo de registro do `target-resolution-end` evento para uma EC2 ação.

```
{
  "id": "EXPhjAXCGY78HV2a4A",
  "log_type": "target-resolution-end",
  "event_timestamp": "2023-05-31T18:50:45Z",
  "version": "2",
  "details": {
    "target_resolution_end_time": "2023-05-31T18:50:46Z",
    "target_name": "EC2InstanceToReboot",
    "target_type": "aws:ec2:instance",
    "resolved_targets": [
      {
        "arn": "arn:aws:ec2:us-east-1:123456789012:instance/i-0f7ee2abffc330de5"
      }
    ],
    "page": 1,
    "total_pages": 1
  }
}
```

```
}
```

O seguinte é um exemplo de registro para o evento `target-resolution-end` para uma ação do EKS.

```
{
  "id": "EXP24YfiucfyVPJpEJn",
  "log_type": "target-resolution-end",
  "event_timestamp": "2023-05-31T18:50:45Z",
  "version": "2",
  "details": {
    "target_resolution_end_time": "2023-05-31T18:50:46Z",
    "target_name": "myPods",
    "target_type": "aws:eks:pod",
    "resolved_targets": [
      {
        "pod_name": "example-696fb6498b-sxhw5",
        "namespace": "default",
        "cluster_arn": "arn:aws:eks:us-east-1:123456789012:cluster/fis-demo-
cluster",
        "target_container_name": "example"
      }
    ],
    "page": 1,
    "total_pages": 1
  }
}
```

action-start

O seguinte é um exemplo de registro para o evento `action-start`. Se o modelo do experimento especificar parâmetros para a ação, o registro também incluirá o campo `parameters`.

```
{
  "id": "EXPhjAXCGY78HV2a4A",
  "log_type": "action-start",
  "event_timestamp": "2023-05-31T18:50:56Z",
  "version": "2",
  "details": {
    "action_name": "Reboot",
    "action_id": "aws:ec2:reboot-instances",
    "action_start_time": "2023-05-31T18:50:56Z",
```

```
    "action_targets": {"Instances":"EC2InstancesToReboot"}
  }
}
```

action-error

O seguinte é um exemplo de registro para o evento `action-error`. Esse evento só é retornado quando uma ação falha. Ele é retornado para cada conta em que a ação falha.

```
{
  "id": "EXPhjAXCGY78HV2a4A",
  "log_type": "action-error",
  "event_timestamp": "2023-05-31T18:50:56Z",
  "version": "2",
  "details": {
    "action_name": "pause-io",
    "action_id": "aws:ebs:pause-volume-io",
    "account_id": "123456789012",
    "action_state": {
      "status": "failed",
      "reason": "Unable to start Pause Volume IO. Target volumes must be attached
to an instance type based on the Nitro system. VolumeId(s): [vol-1234567890abcdef0]:"
    }
  }
}
```

action-end

O seguinte é um exemplo de registro para o evento `action-end`.

```
{
  "id": "EXPhjAXCGY78HV2a4A",
  "log_type": "action-end",
  "event_timestamp": "2023-05-31T18:50:56Z",
  "version": "2",
  "details": {
    "action_name": "Reboot",
    "action_id": "aws:ec2:reboot-instances",
    "action_end_time": "2023-05-31T18:50:56Z",
    "action_state": {
      "status": "completed",
      "reason": "Action was completed."
    }
  }
}
```

```
    }  
  }  
}
```

experiment-end

O seguinte é um exemplo de registro para o evento `experiment-end`.

```
{  
  "id": "EXPhjAXCGY78HV2a4A",  
  "log_type": "experiment-end",  
  "event_timestamp": "2023-05-31T18:50:57Z",  
  "version": "2",  
  "details": {  
    "experiment_end_time": "2023-05-31T18:50:57Z",  
    "experiment_state": {  
      "status": "completed",  
      "reason": "Experiment completed"  
    }  
  }  
}
```

Habilitar registro em log de experimento

O registro em log de experimento está desativado por padrão. Para receber logs de um experimento, você deve criar o experimento a partir de um modelo de experimento com o registro em log ativado. Na primeira vez que você executa um experimento configurado para usar um destino que não tenha sido usado anteriormente para log, atrasamos o experimento para configurar a entrega de logs para esse destino, o que leva cerca de 15 segundos.

Para habilitar o registro em log usando o console

1. Abra o console AWS FIS em <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Modelos de experimento.
3. Selecione o modelo do experimento e escolha Ações, Atualizar modelo de experimento.
4. Para Logs, configure as opções de destino. Para enviar logs para um bucket do S3, escolha Enviar para um bucket do Amazon S3 e insira o nome e o prefixo do bucket. Para enviar registros para CloudWatch registros, escolha Enviar para CloudWatch registros e entre no grupo de registros.

5. Escolha Atualizar modelo de experimento.

Para habilitar o registro de experimentos usando o AWS CLI

Use o [update-experiment-template](#) comando e especifique uma configuração de log.

Desabilitar registro em log

Se não desejar mais receber logs para seus experimentos, desabilite o registro em log de experimento.

Para desabilitar o registro em log usando o console

1. Abra o console AWS FIS em. <https://console.aws.amazon.com/fis/>
2. No painel de navegação, selecione Modelos de experimento.
3. Selecione o modelo do experimento e escolha Ações, Atualizar modelo de experimento.
4. Para Registros, desmarque Enviar para um bucket do Amazon S3 e Enviar para CloudWatch registros.
5. Escolha Atualizar modelo de experimento.

Para desativar o registro de experimentos usando o AWS CLI

Use o [update-experiment-template](#) comando e especifique uma configuração de log vazia.

Registre chamadas de API com AWS CloudTrail

AWS O AWS Fault Injection Service (FIS) é integrado AWS CloudTrail a um serviço que fornece um registro das ações realizadas por um usuário, uma função ou um AWS serviço no AWS FIS. CloudTrail captura todas as chamadas de API para AWS FIS como eventos. As chamadas capturadas incluem chamadas do console do AWS FIS e chamadas de código para as operações da API do AWS FIS. Se você criar uma trilha, poderá habilitar a entrega contínua de CloudTrail eventos para um bucket do Amazon S3, incluindo eventos para AWS o FIS. 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 à AWS FIS, 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](#).

Use CloudTrail

CloudTrail é ativado no seu Conta da AWS quando você cria a conta. Quando a atividade ocorre no AWS FIS, essa atividade é registrada em um CloudTrail evento junto com outros eventos de AWS serviço no histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes no seu Conta da AWS. Para obter mais informações, consulte [Visualização de eventos com histórico de CloudTrail eventos](#).

Para um registro contínuo dos eventos em seu Conta da AWS, incluindo eventos do AWS FIS, 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:

- [Crie uma trilha para sua AWS conta](#)
- [CloudTrail Serviços e integrações compatíveis](#)
- [Configurando notificações do Amazon SNS para CloudTrail](#)
- [Recebendo arquivos de CloudTrail log de várias regiões](#) e [recebendo arquivos de CloudTrail log de várias contas](#)

Todas as ações do AWS FIS são registradas CloudTrail e documentadas na Referência da [API do AWS Fault Injection Service](#). Para as ações experimentais que são realizadas em um recurso de destino, consulte a documentação de referência da API para o serviço proprietário do recurso. Por exemplo, para ações que são realizadas em uma EC2 instância da Amazon, consulte a [Referência de EC2 API da Amazon](#).

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar o seguinte:

- Se a solicitação foi feita com credenciais de usuário da raiz ou do .
- Se a solicitação foi feita com credenciais de segurança temporárias de uma função ou de um usuário federado.
- Se a solicitação foi feita por outro AWS serviço.

Para obter mais informações, consulte [Elemento userIdentity do CloudTrail](#) .

Entenda as AWS entradas do arquivo de log do FIS

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log para um bucket do Amazon S3 que você especificar. CloudTrail os arquivos de log contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros da solicitação e assim por diante. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das chamadas públicas de API, portanto, eles não aparecem em nenhuma ordem específica.

A seguir está um exemplo de entrada de CloudTrail registro para uma chamada para a StopExperiment ação AWS FIS.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAIOSFODNN7EXAMPLE:jdoe",
    "arn": "arn:aws:sts::111122223333:assumed-role/example/jdoe",
    "accountId": "111122223333",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/example",
        "accountId": "111122223333",
        "userName": "example"
      },
      "webIdFederationData": {},
      "attributes": {
        "creationDate": "2020-12-03T09:40:42Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2020-12-03T09:44:20Z",
  "eventSource": "fis.amazonaws.com",
  "eventName": "StopExperiment",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.51.100.25",
  "userAgent": "Boto3/1.22.9 Python/3.8.13 Linux/5.4.186-113.361.amzn2int.x86_64
  Botocore/1.25.9",
```

```
"requestParameters": {
  "clientToken": "1234abc5-6def-789g-012h-ijklm34no56p",
  "experimentTemplateId": "ABCDE1fgHIJkLmNop",
  "tags": {}
},
"responseElements": {
  "experiment": {
    "actions": {
      "exampleAction1": {
        "actionId": "aws:ec2:stop-instances",
        "duration": "PT10M",
        "state": {
          "reason": "Initial state",
          "status": "pending"
        },
        "targets": {
          "Instances": "exampleTag1"
        }
      },
      "exampleAction2": {
        "actionId": "aws:ec2:stop-instances",
        "duration": "PT10M",
        "state": {
          "reason": "Initial state",
          "status": "pending"
        },
        "targets": {
          "Instances": "exampleTag2"
        }
      }
    },
    "creationTime": 1605788649.95,
    "endTime": 1606988660.846,
    "experimentTemplateId": "ABCDE1fgHIJkLmNop",
    "id": "ABCDE1fgHIJkLmNop",
    "roleArn": "arn:aws:iam::111122223333:role/AllowFISActions",
    "startTime": 1605788650.109,
    "state": {
      "reason": "Experiment stopped",
      "status": "stopping"
    },
    "stopConditions": [
      {
        "source": "aws:cloudwatch:alarm",
```

```

        "value": "arn:aws:cloudwatch:us-east-1:111122223333:alarm:example"
    }
  ],
  "tags": {},
  "targets": {
    "ExampleTag1": {
      "resourceTags": {
        "Example": "tag1"
      },
      "resourceType": "aws:ec2:instance",
      "selectionMode": "RANDOM(1)"
    },
    "ExampleTag2": {
      "resourceTags": {
        "Example": "tag2"
      },
      "resourceType": "aws:ec2:instance",
      "selectionMode": "RANDOM(1)"
    }
  }
}
},
"requestID": "1abcd23e-f4gh-567j-klm8-9np01q234r56",
"eventID": "1234a56b-c78d-9e0f-g1h2-34jk56m7n890",
"readOnly": false,
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "111122223333",
"eventCategory": "Management"
}

```

Veja a seguir um exemplo de entrada de CloudTrail registro para uma ação de API que o AWS FIS invocou como parte de um experimento que inclui a ação `aws:ssm:send-command` AWS FIS. O elemento `userIdentity` reflete uma solicitação feita com credenciais temporárias obtidas ao assumir um perfil. O nome do perfil assumido aparece em `userName`. O ID do experimento, `EXP21nT17WMzA6Dnugz`, aparece no `principalId` como parte do ARN da função assumida.

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROATZZZ4JPIXUEXAMPLE:EXP21nT17WMzA6dnUgz",

```

```
    "arn": "arn:aws:sts::111122223333:assumed-role/AllowActions/
EXP21nT17WMzA6dnUgz",
    "accountId": "111122223333",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROATZZZ4JPIXUEXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/AllowActions",
        "accountId": "111122223333",
        "userName": "AllowActions"
      },
      "webIdFederationData": {},
      "attributes": {
        "creationDate": "2022-05-30T13:23:19Z",
        "mfaAuthenticated": "false"
      }
    },
    "invokedBy": "fis.amazonaws.com"
  },
  "eventTime": "2022-05-30T13:23:19Z",
  "eventSource": "ssm.amazonaws.com",
  "eventName": "ListCommands",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "fis.amazonaws.com",
  "userAgent": "fis.amazonaws.com",
  "requestParameters": {
    "commandId": "51dab97f-489b-41a8-a8a9-c9854955dc65"
  },
  "responseElements": null,
  "requestID": "23709ced-c19e-471a-9d95-cf1a06b50ee6",
  "eventID": "145fe5a6-e9d5-45cc-be25-b7923b950c83",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "eventCategory": "Management"
}
```

Solução de problemas AWS FIS

Para solucionar erros, AWS FIS retorna erros detalhados dos registros de experimentos da `GetExperiment` API e do FIS. Os erros são retornados como parte do estado do experimento quando o status do experimento é Com falha. Quando várias ações falham, a primeira ação com falha é retornada como o erro do experimento. Você pode revisar os logs de experimentos do FIS em busca de outros erros. Para saber como registrar e monitorar AWS FIS experimentos, consulte [Monitorando AWS experimentos FIS](#).

Dependendo do tipo de falha, é possível receber um dos seguintes erros:

- **Motivo:** descrição detalhada da falha específica. Os valores de motivo não devem ser usados para automação, pois estão sujeitos a alterações.
- **Código:** o tipo de falha. Os valores de código não devem ser usados para automação, pois estão sujeitos a alterações, a menos que especificado de outra forma na tabela abaixo.
- **Local:** contexto da seção do modelo de experimento que falhou, como a ação ou o destino.
- **ID da conta:** a AWS conta em que a falha ocorreu.

Códigos de erro

Código de erro	Descrição do código
<code>ConfigurationFailure</code>	A ação, o destino, o experimento ou o log não estão configurados corretamente. Verifique o erro <code>location</code> e garanta que os parâmetros e as configurações estejam corretos.
<code>DependentServiceFailure</code>	Houve uma falha em outro AWS serviço. Tente executar o experimento novamente.
<code>InternalFailure</code>	Ocorreu um erro interno durante a execução do experimento. Você pode automatizar com base nesse código de erro.
<code>InvalidTarget</code>	Não foi possível resolver um destino durante a resolução do destino ou no início de uma ação.

Código de erro	Descrição do código
	<p>Isso pode ocorrer devido a um dos motivos a seguir:</p> <ul style="list-style-type: none">• O destino não existe, por exemplo, se tiver sido excluído ou se o ARN estiver incorreto.• Há uma etiqueta para o destino que não resolve nenhum recurso.• Há uma ação que não está vinculada a um destino. <p>Para solucionar problemas, revise os logs a fim de identificar quais destinos não foram resolvidos. Verifique se todas as ações estão vinculadas a destinos e se o ID do recurso ou as etiquetas existem e não foram digitados incorretamente.</p>
AuthorizationFailure	<p>Há duas causas principais de falha no experimento devido a erros de permissão:</p> <ul style="list-style-type: none">• O perfil do IAM de destino não tem as permissões adequadas para resolver destinos ou agir nos recursos. Para corrigir esse erro, revise as permissões necessárias para as ações em FIS Actions Reference e adicione-as ao perfil do IAM do experimento.• A criação do perfil vinculado ao serviço da AWS (SLR) para o FIS foi negada por uma política de controle de serviços (SCP) em sua organização. O FIS usa o SLR para gerenciar o monitoramento e a seleção de recursos para experimentos. Para obter mais informações, consulte Permissões de função vinculadas ao serviço para FIS AWS.

Código de erro	Descrição do código
QuotaExceededFailure	A cota para o tipo de recurso foi excedida. Para determinar se a cota pode ser aumentada, consulte Cotas e limitações para o serviço de injeção de AWS falhas .

Segurança no serviço de injeção de AWS falhas

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de data centers e arquiteturas de rede criados 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 isso como segurança da nuvem e segurança na nuvem:

- **Segurança da nuvem** — AWS é responsável por proteger a infraestrutura que executa AWS os serviços na AWS nuvem. AWS também fornece serviços que você pode usar com segurança. Auditores terceirizados testam e verificam regularmente a eficácia de nossa segurança como parte dos Programas de Conformidade Programas de [AWS](#) de . Para saber mais sobre os programas de conformidade que se aplicam ao Serviço de Injeção de AWS Falhas, consulte [AWS Serviços no Escopo por Programa de Conformidade AWS](#) .
- **Segurança na nuvem** — Sua responsabilidade é determinada pelo AWS serviço que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da empresa e as leis e regulamentos aplicáveis.

Esta documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usar o AWS FIS. Os tópicos a seguir mostram como configurar o AWS FIS para atender aos seus objetivos de segurança e conformidade. Você também aprenderá a usar outros AWS serviços que ajudam a monitorar e proteger seus recursos do AWS FIS.

Conteúdo

- [Proteção de dados no AWS Fault Injection Service](#)
- [Gerenciamento de identidade e acesso para o AWS Fault Injection Service](#)
- [Segurança da infraestrutura no serviço de injeção de AWS falhas](#)
- [Acesse AWS o FIS usando uma interface VPC endpoint \(\)AWS PrivateLink](#)

Proteção de dados no AWS Fault Injection Service

O modelo de [responsabilidade AWS compartilhada modelo](#) de de se aplica à proteção de dados no AWS Fault Injection Service. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle

sobre o conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para saber mais sobre a privacidade de dados, consulte as [Data Privacy FAQ](#). Para saber mais sobre a proteção de dados na Europa, consulte a postagem do blog [AWS Shared Responsibility Model and RGPD](#) no Blog de segurança da AWS .

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com Centro de Identidade do AWS IAM ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com AWS os recursos. Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail. Para obter informações sobre o uso de CloudTrail trilhas para capturar AWS atividades, consulte Como [trabalhar com CloudTrail trilhas](#) no Guia AWS CloudTrail do usuário.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sensíveis armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-3 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para saber mais sobre os endpoints FIPS disponíveis, consulte [Federal Information Processing Standard \(FIPS\) 140-3](#).

É altamente recomendável que nunca sejam colocadas informações confidenciais ou sensíveis, como endereços de e-mail de clientes, em tags ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com o AWS FIS ou outro Serviços da AWS usando o console, a API ou AWS SDKs. AWS CLI Quaisquer dados inseridos em tags ou em campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é fortemente recomendável que não sejam incluídas informações de credenciais no URL para validar a solicitação nesse servidor.

Criptografia em repouso

AWS O FIS sempre criptografa seus dados em repouso. Os dados no AWS FIS são criptografados em repouso usando criptografia transparente do lado do servidor. Isso ajuda a reduzir a carga e a complexidade operacionais necessárias para proteger dados confidenciais. Com a criptografia de dados em repouso, você pode criar aplicativos confidenciais que atendem a requisitos de conformidade e regulamentação de criptografia.

Criptografia em trânsito

AWS O FIS criptografa os dados em trânsito entre o serviço e outros serviços integrados AWS . Todos os dados que passam entre o AWS FIS e os serviços integrados são criptografados usando o Transport Layer Security (TLS). Para obter mais informações sobre outros AWS serviços integrados, consulte [Suportado Serviços da AWS](#).

Gerenciamento de identidade e acesso para o AWS Fault Injection Service

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) para usar os recursos do AWS FIS. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Conteúdo

- [Público](#)
- [Autenticação com identidades](#)
- [Gerenciar o acesso usando políticas](#)
- [Como o AWS Fault Injection Service funciona com o IAM](#)
- [AWS Exemplos de políticas de serviço de injeção de falhas](#)
- [Use funções vinculadas ao serviço para o AWS Fault Injection Service](#)
- [AWS políticas gerenciadas para o AWS Fault Injection Service](#)

Público

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz no AWS FIS.

Usuário do serviço — Se você usar o serviço AWS FIS para fazer seu trabalho, seu administrador fornecerá as credenciais e as permissões de que você precisa. À medida que você usa mais recursos do AWS FIS para fazer seu trabalho, talvez precise de permissões adicionais. Entender como o acesso é gerenciado pode ajudar a solicitar as permissões corretas ao administrador.

Administrador de serviços — Se você AWS é responsável pelos recursos do FIS em sua empresa, provavelmente tem acesso total ao AWS FIS. É seu trabalho determinar quais recursos e recursos do AWS FIS seus usuários do serviço devem acessar. Envie as solicitações ao administrador do IAM para alterar as permissões dos usuários de serviço. Revise as informações nesta página para compreender os conceitos básicos do IAM.

Administrador do IAM — Se você for administrador do IAM, talvez queira saber detalhes sobre como criar políticas para gerenciar o acesso ao AWS FIS.

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.

Identidade federada

Como prática recomendada, exija que os usuários humanos usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório corporativo, provedor de identidade da web ou Directory Service que acessa Serviços da AWS usando credenciais de uma fonte de identidade. As identidades federadas assumem funções que oferecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos Centro de Identidade do AWS IAM. Para saber mais, consulte [O que é o IAM Identity Center?](#) no Guia do usuário do Centro de Identidade do AWS IAM .

Usuários e grupos do IAM

Um [usuário do IAM](#) é uma identidade com permissões específicas para uma única pessoa ou aplicação. É recomendável usar credenciais temporárias, em vez de usuários do IAM com credenciais de longo prazo. Para obter mais informações, consulte [Exigir que usuários humanos usem a federação com um provedor de identidade para acessar AWS usando credenciais temporárias](#) no Guia do usuário do IAM.

Um [grupo do IAM](#) especifica um conjunto de usuários do IAM e facilita o gerenciamento de permissões para grandes conjuntos de usuários. Para ter mais informações, consulte [Casos de uso de usuários do IAM](#) no Guia do usuário do IAM.

Perfis do IAM

Uma [perfil do IAM](#) é uma identidade com permissões específicas que oferece credenciais temporárias. Você pode assumir uma função [mudando de um usuário para uma função do IAM \(console\)](#) ou chamando uma operação de AWS API AWS CLI ou. Para saber mais, consulte [Métodos para assumir um perfil](#) no Manual do usuário do IAM.

Os perfis do IAM são úteis para acesso de usuário federado, permissões de usuário do IAM temporárias, acesso entre contas, acesso entre serviços e aplicações em execução no Amazon EC2. Consulte mais informações em [Acesso a recursos entre contas no IAM](#) no Guia do usuário do IAM.

Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política define permissões quando associada a uma identidade ou recurso. AWS avalia essas

políticas quando um diretor faz uma solicitação. A maioria das políticas é armazenada AWS como documentos JSON. Para ter mais informações sobre documentos de política JSON, consulte [Visão geral das políticas JSON](#) no Guia do usuário do IAM.

Por meio de políticas, os administradores especificam quem tem acesso a que, definindo qual entidade principal pode realizar ações em quais recursos e sob quais condições.

Por padrão, usuários e perfis não têm permissões. Um administrador do IAM cria políticas do IAM e as adiciona aos perfis, os quais os usuários podem então assumir. As políticas do IAM definem permissões, independentemente do método usado para realizar a operação.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissão JSON que você anexa a uma identidade (usuário, grupo ou perfil). Essas políticas controlam quais ações as identidades podem realizar, em quais recursos e sob quais condições. Para saber como criar uma política baseada em identidade, consulte [Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente](#) no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser políticas em linha (incorporadas diretamente em uma única identidade) ou políticas gerenciadas (políticas autônomas anexadas a várias identidades). Para saber como escolher entre uma política gerenciada e políticas em linha, consulte [Escolher entre políticas gerenciadas e políticas em linha](#) no Guia do usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. Entre os exemplos estão políticas de confiança de perfil do IAM e políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. É necessário [especificar uma entidade principal](#) em uma política baseada em recursos.

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais que podem definir o máximo de permissões concedidas por tipos de políticas mais comuns:

- Limites de permissões: definem o número máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM. Para saber mais sobre limites de permissões, consulte [Limites de permissões para identidades do IAM](#) no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs) — Especifique as permissões máximas para uma organização ou unidade organizacional em AWS Organizations. Para saber mais, consulte [Políticas de controle de serviço](#) no Guia do usuário do AWS Organizations .
- Políticas de controle de recursos (RCPs) — Defina o máximo de permissões disponíveis para recursos em suas contas. Para obter mais informações, consulte [Políticas de controle de recursos \(RCPs\)](#) no Guia AWS Organizations do usuário.
- Políticas de sessão: políticas avançadas transmitidas como um parâmetro durante a criação de uma sessão temporária para um perfil ou um usuário federado. Para saber mais, consulte [Políticas de sessão](#) no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte [Lógica de avaliação de políticas](#) no Guia do usuário do IAM.

Como o AWS Fault Injection Service funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao AWS FIS, saiba quais recursos do IAM estão disponíveis para uso com o AWS FIS.

Recursos do IAM que você pode usar com o AWS Fault Injection Service

Recurso do IAM	AWS Suporte FIS
Políticas baseadas em identidade	Sim
Políticas baseadas em recurso	Não
Ações de políticas	Sim
Recursos de políticas	Sim

Recurso do IAM	AWS Suporte FIS
Chaves de condição de política (específicas do serviço)	Sim
ACLs	Não
ABAC (tags em políticas)	Sim
Credenciais temporárias	Sim
Permissões de entidade principal	Sim
Perfis de serviço	Sim
Perfis vinculados a serviço	Sim

Para ter uma visão de alto nível de como o AWS FIS e outros AWS serviços funcionam com a maioria dos recursos do IAM, consulte [AWS os serviços que funcionam com o IAM no Guia do usuário do IAM](#).

Políticas baseadas em identidade para FIS AWS

Compatível com políticas baseadas em identidade: sim

As políticas baseadas em identidade são documentos de políticas de permissões JSON que podem ser anexados a uma identidade, como usuário do IAM, grupo de usuários ou perfil. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que 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.

Com as políticas baseadas em identidade do IAM, é possível especificar ações e recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. Para saber mais sobre todos os elementos que podem ser usados em uma política JSON, consulte [Referência de elemento de política JSON do IAM](#) no Guia do usuário do IAM.

Exemplos de políticas baseadas em identidade para FIS AWS

Para ver exemplos de políticas baseadas em identidade do AWS FIS, consulte. [AWS Exemplos de políticas de serviço de injeção de falhas](#)

Políticas baseadas em recursos no FIS AWS

Compatibilidade com políticas baseadas em recursos: não

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse atributo e em que condições. É necessário [especificar uma entidade principal](#) em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Para permitir o acesso entre contas, é possível especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em recursos. Consulte mais informações em [Acesso a recursos entre contas no IAM](#) no Guia do usuário do IAM.

Ações políticas para o AWS FIS

Compatível com ações de políticas: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que podem ser usadas para permitir ou negar acesso em uma política. Incluem ações em uma política para conceder permissões para executar a operação associada.

Para ver uma lista de ações do AWS FIS, consulte [Ações definidas pelo AWS Fault Injection Service](#) na Referência de Autorização de Serviço.

As ações de política no AWS FIS usam o seguinte prefixo antes da ação:

```
fis
```

Para especificar várias ações em uma única instrução, separe-as com vírgulas.

```
"Action": [  
  "fis:action1",  
  "fis:action2"  
]
```

Você também pode especificar várias ações usando caracteres-curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `List`, inclua a seguinte ação:

```
"Action": "fis:List*"
```

Recursos políticos para o AWS FIS

Compatível com recursos de políticas: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. Como prática recomendada, especifique um recurso usando seu [nome do recurso da Amazon \(ARN\)](#). Para ações que não oferecem compatibilidade com permissões em nível de recurso, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*" 
```

Algumas ações da API AWS FIS oferecem suporte a vários recursos. Para especificar vários recursos em uma única instrução, separe-os ARNs com vírgulas.

```
"Resource": [
  "resource1",
  "resource2"
]
```

Para ver uma lista dos tipos de recursos do AWS FIS e seus ARNs, consulte Tipos de [recursos definidos pelo AWS Fault Injection Service](#) na Referência de Autorização de Serviço. Para saber com quais ações você pode especificar o ARN de cada recurso, consulte [Ações definidas pelo AWS Fault Injection Service](#).

Chaves de condição de política para AWS FIS

Compatível com chaves de condição de política específicas de serviço: sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Condition` especifica quando as instruções são executadas com base em critérios definidos. É possível criar expressões condicionais que usem [agentes de condição](#), como “igual a”

ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação. Para ver todas as chaves de condição AWS globais, consulte as [chaves de contexto de condição AWS global](#) no Guia do usuário do IAM.

Para ver uma lista das chaves de condição do AWS FIS, consulte [Chaves de condição para o serviço de injeção de AWS falhas](#) na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar uma chave de condição, consulte [Ações definidas pelo AWS Fault Injection Service](#).

Para ver exemplos de políticas baseadas em identidade do AWS FIS, consulte [AWS Exemplos de políticas de serviço de injeção de falhas](#)

ACLs na AWS FIS

Suportes ACLs: Não

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

ABAC com AWS FIS

Compatível com ABAC (tags em políticas): sim

O controle de acesso por atributo (ABAC) é uma estratégia de autorização que define permissões com base em atributos chamados de tags. Você pode anexar tags a entidades e AWS recursos do IAM e, em seguida, criar políticas ABAC para permitir operações quando a tag do diretor corresponder à tag no recurso.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no [elemento de condição](#) de uma política usando as `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou chaves de condição `aws:TagKeys`.

Se um serviço for compatível com as três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço for compatível com as três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para saber mais sobre o ABAC, consulte [Definir permissões com autorização do ABAC](#) no Guia do usuário do IAM. Para visualizar um tutorial com etapas para configurar o ABAC, consulte [Usar controle de acesso por atributo \(ABAC\)](#) no Guia do usuário do IAM.

Para visualizar um exemplo de política baseada em identidade para limitar o acesso a um recurso baseado em tags para esse recurso, consulte [Exemplo: use tags para controlar o uso de recursos](#).

Usando credenciais temporárias com AWS o FIS

Compatível com credenciais temporárias: sim

As credenciais temporárias fornecem acesso de curto prazo aos AWS recursos e são criadas automaticamente quando você usa a federação ou troca de funções. AWS recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para ter mais informações, consulte [Credenciais de segurança temporárias no IAM](#) e [Serviços da Serviços da AWS que funcionam com o IAM](#) no Guia do usuário do IAM.

Permissões principais entre serviços para AWS FIS

Compatibilidade com o recurso de encaminhamento de sessões de acesso (FAS): sim

As sessões de acesso direto (FAS) usam as permissões do principal chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) de fazer solicitações aos serviços posteriores. Para obter detalhes da política ao fazer solicitações de FAS, consulte [Sessões de acesso direto](#).

Funções de serviço para AWS FIS

Compatível com perfis de serviço: sim

O perfil de serviço é um [perfil do IAM](#) que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para saber mais, consulte [Criar um perfil para delegar permissões a um AWS service \(Serviço da AWS\)](#) no Guia do Usuário do IAM.

Funções vinculadas a serviços para FIS AWS

Compatibilidade com perfis vinculados a serviços: sim

Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode assumir o perfil de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para perfis vinculados ao serviço.

Para obter detalhes sobre como criar ou gerenciar funções vinculadas ao serviço AWS FIS, consulte [Use funções vinculadas ao serviço para o AWS Fault Injection Service](#)

AWS Exemplos de políticas de serviço de injeção de falhas

Por padrão, usuários e funções não têm permissão para criar ou modificar recursos do AWS FIS. Para conceder permissão aos usuários para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM.

Para aprender a criar uma política baseada em identidade do IAM ao usar esses documentos de política em JSON de exemplo, consulte [Criar políticas do IAM \(console\)](#) no Guia do usuário do IAM.

Para obter detalhes sobre ações e tipos de recursos definidos pelo AWS FIS, incluindo o formato de cada um dos ARNs tipos de recursos, consulte [Ações, recursos e chaves de condição para o AWS Fault Injection Service](#) na Referência de Autorização de Serviço.

Conteúdo

- [Práticas recomendadas de política](#)
- [Exemplo: usar o console AWS FIS](#)
- [Exemplo: Listar as ações disponíveis AWS do FIS](#)
- [Exemplo: criar um modelo de experimento para uma ação específica](#)
- [Exemplo: iniciar um experimento](#)
- [Exemplo: use tags para controlar o uso de recursos](#)
- [Exemplo: excluir um modelo de experimento com uma tag específica](#)
- [Exemplo: permitir que os usuários visualizem suas próprias permissões](#)
- [Exemplo: use chaves de condição para ec2:InjectApiError](#)
- [Exemplo: use chaves de condição para aws:s3:bucket-pause-replication](#)

Práticas recomendadas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do AWS FIS em sua conta. Essas ações podem incorrer em custos para sua Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas

AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para saber mais, consulte [Políticas gerenciadas pela AWS](#) ou [Políticas gerenciadas pela AWS para funções de trabalho](#) no Guia do usuário do IAM.

- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para saber mais sobre como usar o IAM para aplicar permissões, consulte [Políticas e permissões no IAM](#) no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: é possível adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, é possível escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como CloudFormation. Para saber mais, consulte [Elementos da política JSON do IAM: condição](#) no Guia do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações práticas para ajudar a criar políticas seguras e funcionais. Para saber mais, consulte [Validação de políticas do IAM Access Analyzer](#) no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para saber mais, consulte [Configuração de acesso à API protegido por MFA](#) no Guia do Usuário do IAM.

Para saber mais sobre as práticas recomendadas do IAM, consulte [Práticas recomendadas de segurança no IAM](#) no Guia do usuário do IAM.

Exemplo: usar o console AWS FIS

Para acessar o console do AWS Fault Injection Service, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os recursos do AWS FIS 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.

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 a ações que correspondam à operação de API que estiverem tentando executar.

O exemplo de política a seguir concede permissão para listar e visualizar todos os recursos do AWS FIS usando o console do AWS FIS, mas não para criá-los, atualizá-los ou excluí-los. Também concede permissões para visualizar os recursos disponíveis usados por todas as ações do AWS FIS que você pode especificar em um modelo de experimento.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FISReadOnlyActions",
      "Effect": "Allow",
      "Action": [
        "fis:List*",
        "fis:Get*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AdditionalReadOnlyActions",
      "Effect": "Allow",
      "Action": [
        "ssm:Describe*",
        "ssm:Get*",
        "ssm:List*",
        "ec2:DescribeInstances",
        "rds:DescribeDBClusters",
        "ecs:DescribeClusters",
        "ecs:ListContainerInstances",
        "eks:DescribeNodegroup",
        "cloudwatch:DescribeAlarms",
        "iam:ListRoles"
      ],
      "Resource": "*"
    }
  ]
}
```

```
    },
    {
      "Sid": "PermissionsToCreateServiceLinkedRole",
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:AWSServiceName": "fis.amazonaws.com"
        }
      }
    }
  ]
}
```

Exemplo: Listar as ações disponíveis AWS do FIS

A política a seguir concede permissão para listar as ações disponíveis do AWS FIS.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "fis:ListActions"
      ],
      "Resource": "arn:aws:fis:*:*:action/*"
    }
  ]
}
```

Exemplo: criar um modelo de experimento para uma ação específica

A política a seguir concede permissão para criar um modelo de experimento para a ação `aws:ec2:stop-instances`.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PolicyExample",
      "Effect": "Allow",
      "Action": [
        "fis:CreateExperimentTemplate"
      ],
      "Resource": [
        "arn:aws:fis:*:*:action/aws:ec2:stop-instances",
        "arn:aws:fis:*:*:experiment-template/*"
      ]
    },
    {
      "Sid": "PolicyPassRoleExample",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::111122223333:role/role-name"
      ]
    }
  ]
}
```

Exemplo: iniciar um experimento

A política a seguir concede permissão para iniciar um experimento usando a função e o modelo de experimento especificados do IAM. Também permite que o AWS FIS crie uma função vinculada ao serviço em nome do usuário. Para obter mais informações, consulte [Use funções vinculadas ao serviço para o AWS Fault Injection Service](#).

JSON

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

```

"Statement": [
  {
    "Sid": "PolicyExample",
    "Effect": "Allow",
    "Action": [
      "fis:StartExperiment"
    ],
    "Resource": [
      "arn:aws:fis:*:*:experiment-template/experiment-template-id",
      "arn:aws:fis:*:*:experiment/*"
    ]
  },
  {
    "Sid": "PolicyExampleforServiceLinkedRole",
    "Effect": "Allow",
    "Action": "iam:CreateServiceLinkedRole",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "iam:AWSServiceName": "fis.amazonaws.com"
      }
    }
  }
]
}

```

Exemplo: use tags para controlar o uso de recursos

A política a seguir concede permissão para realizar experimentos a partir de modelos de experimentos que tenham a tag `Purpose=Test`. Ele não concede permissão para criar ou modificar modelos de experimentos ou executar experimentos usando modelos que não tenham a tag especificada.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "fis:StartExperiment",

```

```

    "Resource": "arn:aws:fis:*:*:experiment-template/*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/Purpose": "Test"
      }
    }
  }
]
}

```

Exemplo: excluir um modelo de experimento com uma tag específica

A política a seguir concede permissão para excluir um modelo de experimento com a tag Purpose=Test.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "fis:DeleteExperimentTemplate"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Purpose": "Test"
        }
      }
    }
  ]
}

```

Exemplo: permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui

permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou AWS .

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

Exemplo: use chaves de condição para **ec2:InjectApiError**

O exemplo de política a seguir usa a chave de condição `ec2:FisTargetArns` para definir o escopo dos recursos de destino. Essa política permite as ações do AWS FIS `aws:ec2:api-insufficient-instance-capacity-error` e `aws:ec2:asg-insufficient-instance-capacity-error`

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:InjectApiError",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "ec2:FisActionId": [
            "aws:ec2:api-insufficient-instance-capacity-error",
            "aws:ec2:asg-insufficient-instance-capacity-error"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:InjectApiError",
      "Resource": "*",
      "Condition": {
        "ForAllValues:ArnLike": {
          "ec2:FisTargetArns": [
            "arn:aws:autoscaling:*:*:autoScalingGroup:uuid:autoScalingGroupName/asg-name"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "autoscaling:DescribeAutoScalingGroups",
      "Resource": "*"
    }
  ]
}
```

Exemplo: use chaves de condição para `aws:s3:bucket-pause-replication`

O exemplo de política a seguir usa a chave de S3:IsReplicationPauseRequest condição para permitir `PutReplicationConfiguration` e `GetReplicationConfiguration` somente quando usada pelo AWS FIS no contexto da ação do AWS FIS. `aws:s3:bucket-pause-replication`

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "S3:PauseReplication"
      ],
      "Resource": "arn:aws:s3:::mybucket",
      "Condition": {
        "StringEquals": {
          "s3:DestinationRegion": "region"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "S3:PutReplicationConfiguration",
        "S3:GetReplicationConfiguration"
      ],
      "Resource": "arn:aws:s3:::mybucket",
      "Condition": {
        "BoolIfExists": {
          "s3:IsReplicationPauseRequest": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "S3:ListBucket"
      ],
      "Resource": "arn:aws:s3:::*"
    }
  ]
}
```

```
    },  
    {  
      "Effect": "Allow",  
      "Action": [  
        "tag:GetResources"  
      ],  
      "Resource": "*"   
    }  
  ]  
}
```

Use funções vinculadas ao serviço para o AWS Fault Injection Service

AWS O Fault Injection Service usa AWS Identity and Access Management funções [vinculadas ao serviço](#) (IAM). Uma função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao AWS FIS. As funções vinculadas a serviços são predefinidas pelo AWS FIS e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Uma função vinculada ao serviço facilita a configuração do AWS FIS porque você não precisa adicionar manualmente as permissões necessárias para gerenciar o monitoramento e a seleção de recursos para experimentos. AWS O FIS define as permissões de suas funções vinculadas ao serviço e, a menos que seja definido de outra forma, somente o AWS FIS pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, que não pode ser anexada a nenhuma outra entidade do IAM.

Além da função vinculada ao serviço, você também deve especificar um perfil do IAM que conceda permissão para modificar os recursos que você especifica como alvos em um modelo de experimento. Para obter mais informações, consulte [Funções do IAM para experimentos AWS do FIS](#).

Você pode excluir um perfil vinculado ao serviço somente depois de excluir os atributos relacionados. Isso protege seus recursos AWS do FIS porque você não pode remover inadvertidamente a permissão para acessar os recursos.

Permissões de função vinculadas ao serviço para FIS AWS

AWS O FIS usa a função vinculada ao serviço chamada `AWSServiceRoleForFIS` para permitir que ele gerencie o monitoramento e a seleção de recursos para experimentos.

A função vinculada ao serviço AWSServiceRoleForFIS confia nos seguintes serviços para assumir a função:

- `fis.amazonaws.com`

A função AWSServiceRoleFor vinculada ao serviço FIS usa a política gerenciada Amazon. FISService RolePolicy Essa política permite que o AWS FIS gerencie o monitoramento e a seleção de recursos para experimentos. Para obter mais informações, consulte [Amazon FISService RolePolicy](#) na Referência de políticas AWS gerenciadas.

É necessário configurar as permissões para permitir que uma entidade do IAM (como um usuário, grupo ou perfil) crie, edite ou exclua uma função vinculada ao serviço. Para que a função vinculada ao serviço AWSServiceRoleForFIS seja criada com sucesso, a identidade do IAM com a qual você usa o AWS FIS deve ter as permissões necessárias. Para conceder as permissões necessárias, anexe a política a seguir à identidade do IAM.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "fis.amazonaws.com"
        }
      }
    }
  ]
}
```

Para obter mais informações, consulte [Permissões do perfil vinculado a serviço](#) no Guia do usuário do IAM.

Crie uma função vinculada ao serviço para o FIS AWS

Não é necessário criar manualmente um perfil vinculado ao serviço. Quando você inicia um experimento do AWS FIS na Console de gerenciamento da AWS, na ou na AWS API AWS CLI, o AWS FIS cria a função vinculada ao serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, será possível usar esse mesmo processo para recriar o perfil em sua conta. Quando você inicia um experimento do AWS FIS, o AWS FIS cria a função vinculada ao serviço para você novamente.

Editar uma função vinculada ao serviço para FIS AWS

AWS O FIS não permite que você edite a função vinculada ao serviço `AWSServiceRoleForFIS`. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, será possível editar a descrição do perfil usando o IAM. Para saber mais, consulte [Editar um perfil vinculado ao serviço](#) no Guia do usuário do IAM.

Excluir uma função vinculada ao serviço para FIS AWS

Se você não precisar mais usar um recurso ou serviço que requer um perfil vinculado ao serviço, é recomendável excluí-lo. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. No entanto, você deve limpar os recursos de seu perfil vinculado ao serviço antes de excluí-lo manualmente.

Note

Se o serviço AWS FIS estiver usando a função quando você tentar limpar os recursos, a limpeza poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

Para limpar os recursos AWS do FIS usados pelo `AWSServiceRoleForFIS`

Verifique se nenhum de seus experimentos está sendo executado no momento. Se necessário, interrompa seus experimentos. Para obter mais informações, consulte [Interromper um experimento](#).

Como excluir manualmente o perfil vinculado a serviço usando o IAM

Use o console do IAM AWS CLI, o ou a AWS API para excluir a função vinculada ao serviço `AWSServiceRoleForFIS`. Para saber mais, consulte [Excluir um perfil vinculado ao serviço](#) no Guia do usuário do IAM.

Regiões suportadas para funções AWS vinculadas ao serviço FIS

AWS O FIS oferece suporte ao uso de funções vinculadas a serviços em todas as regiões em que o serviço está disponível. Para obter mais informações, consulte [Endpoints e cotas do Fault Injection Service AWS](#).

AWS políticas gerenciadas para o AWS Fault Injection Service

Uma política AWS gerenciada é uma política autônoma criada e administrada por AWS. AWS as políticas gerenciadas são projetadas para fornecer permissões para muitos casos de uso comuns, para que você possa começar a atribuir permissões a usuários, grupos e funções.

Lembre-se de que as políticas AWS gerenciadas podem não conceder permissões de privilégio mínimo para seus casos de uso específicos porque elas estão disponíveis para uso de todos os AWS clientes. Recomendamos que você reduza ainda mais as permissões definindo as [políticas gerenciadas pelo cliente](#) que são específicas para seus casos de uso.

Você não pode alterar as permissões definidas nas políticas AWS gerenciadas. Se AWS atualizar as permissões definidas em uma política AWS gerenciada, a atualização afetará todas as identidades principais (usuários, grupos e funções) às quais a política está anexada. AWS é mais provável que atualize uma política AWS gerenciada quando uma nova AWS service (Serviço da AWS) é lançada ou novas operações de API são disponibilizadas para serviços existentes.

Para saber mais, consulte [AWS Políticas gerenciadas pela](#) no Guia do usuário do IAM.

AWS política gerenciada: Amazon FISService RolePolicy

Essa política é anexada à função vinculada ao serviço chamada AWSServiceRoleForFIS para permitir que AWS o FIS gereencie o monitoramento e a seleção de recursos para experimentos. Para obter mais informações, consulte [Use funções vinculadas ao serviço para o AWS Fault Injection Service](#).

AWS política gerenciada: AWSFault Injection Simulator EC2 Acesso

Use essa política em uma função experimental para conceder permissão ao AWS FIS para realizar experimentos que usem as [ações do AWS FIS para o Amazon EC2](#). Para obter mais informações, consulte [the section called “Perfil do experimento”](#).

Para ver as permissões dessa política, consulte [AWSFaultInjectionSimulatorEC2Acesso](#) na Referência de Política AWS Gerenciada.

AWS política gerenciada: AWSFault InjectionSimulator ECSAccess

Use essa política em uma função experimental para conceder permissão ao AWS FIS para realizar experimentos que usem as [ações do AWS FIS para o Amazon ECS](#). Para obter mais informações, consulte [the section called “Perfil do experimento”](#).

Para visualizar as permissões para esta política, consulte [AWSFaultInjectionSimulatorECSAccess](#) na Referência de políticas gerenciadas pela AWS .

AWS política gerenciada: AWSFault InjectionSimulator EKSAccess

Use essa política em uma função experimental para conceder permissão ao AWS FIS para realizar experimentos que usem as [ações do AWS FIS para o Amazon EKS](#). Para obter mais informações, consulte [the section called “Perfil do experimento”](#).

Para visualizar as permissões para esta política, consulte [AWSFaultInjectionSimulatorEKSAccess](#) na Referência de políticas gerenciadas pela AWS .

AWS política gerenciada: AWSFault InjectionSimulatorNetworkAccess

Use essa política em uma função experimental para conceder permissão ao AWS FIS para realizar experimentos que usem as ações de [rede do AWS FIS](#). Para obter mais informações, consulte [the section called “Perfil do experimento”](#).

Para visualizar as permissões para esta política, consulte [AWSFaultInjectionSimulatorNetworkAccess](#) na Referência de políticas gerenciadas pela AWS .

AWS política gerenciada: AWSFault InjectionSimulator RDSAccess

Use essa política em uma função experimental para conceder permissão ao AWS FIS para realizar experimentos que usem as [ações do AWS FIS para o Amazon RDS](#). Para obter mais informações, consulte [the section called “Perfil do experimento”](#).

Para visualizar as permissões para esta política, consulte [AWSFaultInjectionSimulatorRDSAccess](#) na Referência de políticas gerenciadas pela AWS .

AWS política gerenciada: AWSFault InjectionSimulator SSMAccess

Use essa política em uma função de experimento para conceder permissão ao AWS FIS para executar experimentos que usem as [ações do AWS FIS para Systems Manager](#). Para obter mais informações, consulte [the section called “Perfil do experimento”](#).

Para visualizar as permissões para esta política, consulte [AWSFaultInjectionSimulatorSSMAccess](#) na Referência de políticas gerenciadas pela AWS .

AWS Atualizações do FIS nas políticas AWS gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas do AWS FIS desde que esse serviço começou a rastrear essas alterações.

Alteração	Descrição	Data
AWSFaultInjectionSimulatorEC2Acesso — Atualização de uma política existente	Foi adicionada a permissão necessária para os cenários “AZ: Desaceleração do aplicativo” e “Cross-AZ: desaceleração do tráfego”. As permissões são: ec2: DescribeSubnets	12 de novembro de 2025
AWSFaultInjectionSimulatorECSAcesso – atualização para uma política existente	Foram adicionadas as permissões necessárias para os cenários “AZ: Desaceleração do aplicativo” e “Cross-AZ: desaceleração do tráfego”. As permissões são: ecs: DescribeContainerInstances, ec2: DescribeSubnets e ec2: DescribeInstances	12 de novembro de 2025
AWSFaultInjectionSimulatorECSAcesso – atualização para uma política existente	Permissões adicionadas para permitir que o AWS FIS resolva os alvos do ECS.	25 de janeiro de 2024
AWSFaultInjectionSimulatorNetworkAccess – atualização para uma política existente	Permissões adicionadas para permitir que o AWS FIS execute experimentos usando as aws:network:transit-gateway-disrupt-cross-region-connectivity ações aws:network:route-table-disrupt-cross-region-connectivity e.	25 de janeiro de 2024

Alteração	Descrição	Data
AWSFaultInjectionSimulatorE C2Acesso — Atualização de uma política existente	Permissões adicionadas para permitir que o AWS FIS resolva instâncias do EC2.	13 de novembro de 2023
AWSFaultInjectionSimulatorE KSAccess – atualização para uma política existente	Permissões adicionadas para permitir que o AWS FIS resolva os alvos do EKS.	13 de novembro de 2023
AWSFaultInjectionSimulatorR DSAccess – atualização para uma política existente	Permissões adicionadas para permitir que o AWS FIS resolva alvos do RDS.	13 de novembro de 2023
AWSFaultInjectionSimulatorE C2Acesso — Atualização de uma política existente	Permissões adicionadas para permitir que o AWS FIS execute documentos SSM em instâncias do EC2 e encerre instâncias do EC2.	2 de junho de 2023
AWSFaultInjectionSimulatorS SMAccess – atualização para uma política existente	Permissões adicionadas para permitir que o AWS FIS execute documentos SSM em instâncias do EC2.	2 de junho de 2023
AWSFaultInjectionSimulatorE CSAccess – atualização para uma política existente	Permissões adicionadas para permitir que o AWS FIS execute experimentos usando as novas <code>aws:ecs:task</code> ações.	1º de junho de 2023
AWSFaultInjectionSimulatorE KSAccess – atualização para uma política existente	Permissões adicionadas para permitir que o AWS FIS execute experimentos usando as novas <code>aws:eks:pod</code> ações.	1º de junho de 2023
AWSFaultInjectionSimulatorE C2Acesso — Nova política	Foi adicionada uma política para permitir que o AWS FIS execute um experimento que usa ações do AWS FIS para o Amazon EC2.	26 de outubro de 2022

Alteração	Descrição	Data
AWSFaultInjectionSimulatorECSAccess – Nova política	Foi adicionada uma política para permitir que o AWS FIS execute um experimento que usa ações do AWS FIS para o Amazon ECS.	26 de outubro de 2022
AWSFaultInjectionSimulatorEKSAccess – Nova política	Foi adicionada uma política para permitir que o AWS FIS execute um experimento que usa ações do AWS FIS para o Amazon EKS.	26 de outubro de 2022
AWSFaultInjectionSimulatorNetworkAccess – Nova política	Foi adicionada uma política para permitir que o AWS FIS execute um experimento que usa ações de rede AWS do FIS.	26 de outubro de 2022
AWSFaultInjectionSimulatorRDSAccess – Nova política	Foi adicionada uma política para permitir que o AWS FIS execute um experimento que usa ações do AWS FIS para o Amazon RDS.	26 de outubro de 2022
AWSFaultInjectionSimulatorSMSAccess – Nova política	Foi adicionada uma política para permitir que o AWS FIS execute um experimento que usa ações do AWS FIS para Systems Manager.	26 de outubro de 2022
Amazon FISService RolePolicy — Atualização de uma política existente	Permissões adicionadas para permitir que o AWS FIS descreva sub-redes.	26 de outubro de 2022
Amazon FISService RolePolicy — Atualização de uma política existente	Permissões adicionadas para permitir que o AWS FIS descreva clusters EKS.	7 de julho de 2022

Alteração	Descrição	Data
Amazon FISService RolePolicy — Atualização de uma política existente	Permissões adicionadas para permitir que AWS o FIS liste e descreva as tarefas em seus clusters.	7 de fevereiro de 2022
Amazon FISService RolePolicy — Atualização de uma política existente	Condição <code>events:ManagedBy</code> removida da ação <code>events:DescribeRule</code> .	6 de janeiro de 2022
Amazon FISService RolePolicy — Atualização de uma política existente	Permissões adicionadas para permitir que AWS o FIS recupere o histórico dos CloudWatch alarmes usados em condições de parada.	30 de junho de 2021
AWS A FIS começou a rastrear as mudanças	AWS A FIS começou a monitorar as mudanças em suas políticas AWS gerenciadas	1.º de março de 2021

Segurança da infraestrutura no serviço de injeção de AWS falhas

Como serviço gerenciado, o AWS Fault Injection Service 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 o AWS FIS 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.

Acesse AWS o FIS usando uma interface VPC endpoint ()AWS PrivateLink

Você pode estabelecer uma conexão privada entre sua VPC e o AWS Fault Injection Service criando uma interface VPC endpoint. Os VPC endpoints são alimentados por [AWS PrivateLink](#), uma tecnologia que permite acessar o AWS FIS de forma privada APIs sem um gateway de internet, dispositivo NAT, conexão VPN ou conexão Direct Connect. AWS As instâncias na sua VPC não precisam de endereços IP públicos para se comunicar com AWS o FIS. APIs

Cada endpoint de interface é representado por uma ou mais [interfaces de rede elástica](#) nas sub-redes.

Para obter mais informações, consulte [Acesso Serviços da AWS por meio AWS PrivateLink](#) do AWS PrivateLink Guia.

Considerações sobre endpoints AWS FIS VPC

Antes de configurar uma interface VPC endpoint para AWS FIS, consulte [Access and using AWS service \(Serviço da AWS\) an interface VPC](#) endpoint no Guia.AWS PrivateLink

AWS O FIS oferece suporte para fazer chamadas para todas as suas ações de API a partir da sua VPC.

Crie uma interface VPC endpoint para FIS AWS

Você pode criar um VPC endpoint para o serviço AWS FIS usando o console Amazon VPC ou o (). AWS Command Line Interface AWS CLI Para obter mais informações, consulte [Create a VPC endpoint](#) (Criar um endpoint da VPC) no Guia do AWS PrivateLink .

Crie um VPC endpoint para AWS FIS usando o seguinte nome de serviço:.

com. amazonaws . *region* . fis

Se você habilitar o DNS privado para o endpoint, poderá fazer solicitações de API ao AWS FIS usando seu nome DNS padrão para a região, por exemplo, . fis . us-east-1 . amazonaws . com

Crie uma política de VPC endpoint para FIS AWS

Você pode anexar uma política de endpoint ao seu VPC endpoint que controla o acesso ao FIS. AWS Essa política especifica as seguintes informações:

- A entidade principal que pode realizar ações.
- As ações que podem ser realizadas.
- Os recursos aos quais as ações podem ser aplicadas.

Para obter mais informações, consulte [Controlar o acesso aos endpoints da VPC usando políticas de endpoint](#) no Guia AWS PrivateLink .

Exemplo: política de VPC endpoint para ações específicas do FIS AWS

A política de VPC endpoint a seguir concede acesso às ações do AWS FIS listadas em todos os recursos a todos os diretores.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "fis:ListExperimentTemplates",
        "fis:StartExperiment",
        "fis:StopExperiment",
        "fis:GetExperiment"
      ],
      "Resource": "*",
      "Principal": "*"
    }
  ]
}
```

Exemplo: política de VPC endpoint que nega o acesso de um determinado endpoint Conta da AWS

A política de VPC endpoint a seguir nega o Conta da AWS acesso especificado a todas as ações e recursos, mas concede a todos os outros Contas da AWS acessos a todas as ações e recursos.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*",
      "Principal": "*"
    },
  ],
}
```

```
{
  "Effect": "Deny",
  "Action": "*",
  "Resource": "*",
  "Principal": {
    "AWS": [ "123456789012" ]
  }
}
```

Marcando seus recursos do AWS FIS

Uma tag é um rótulo de metadados que você ou AWS atribui a um AWS recurso. Cada tag consiste em uma chave e um valor. Para tags atribuídas por você, é possível definir a chave e o valor. Por exemplo, você pode definir a chave como `purpose` e o valor como `test` para um recurso.

As tags ajudam a:

- Identifique e organize seus AWS recursos. Muitos AWS serviços oferecem suporte à marcação, então você pode atribuir a mesma tag a recursos de serviços diferentes para indicar que os recursos estão relacionados.
- Controle o acesso aos seus AWS recursos. Para obter mais informações, consulte [Controlling Access Using Tags](#) no Guia do usuário do IAM da .

Restrições de marcação

As seguintes restrições básicas se aplicam às tags nos recursos AWS do FIS:

- O número máximo de tags que você pode atribuir a um recurso: 50
- Comprimento máximo da chave: 128 caracteres Unicode
- Comprimento máximo de valor: 256 caracteres Unicode
- Caracteres válidos para chaves e valores: a-z, A-Z, 0-9, espaço e os seguintes caracteres: `_ . : / = + - e @`
- As chaves e os valores diferenciam letras maiúsculas de minúsculas
- Não é possível usar `aws:` como prefixo para chaves, pois ele é reservado para uso da AWS

Trabalhar com tags

Os seguintes recursos do AWS AWS Fault Injection Service (FIS) oferecem suporte à marcação:

- Ações
- Experimentos
- Modelos de experimentos

Você pode usar o console para trabalhar com tags para experimentos e modelos de experimentos. Para obter mais informações, consulte:

- [Marcar um experimento](#)
- [Marcar modelos de experimentos](#)

Você pode usar os AWS CLI comandos a seguir para trabalhar com tags para ações, experimentos e modelos de experimentos:

- [tag-resource](#) – Adiciona tags a um recurso.
- [untag-resource](#) – Remove tags de um recurso.
- [list-tags-for-resource](#)— Liste as tags de um recurso específico.

Cotas e limitações para o serviço de injeção de AWS falhas

Você Conta da AWS tem cotas padrão, anteriormente chamadas de limites, para cada AWS serviço. A menos que especificado de outra forma, cada cota é específica da região. Você pode solicitar aumentos para as cotas marcadas como ajustáveis na tabela abaixo.

Para ver as cotas do AWS FIS em sua conta, abra o console [Service Quotas](#). No painel de navegação, escolha Serviços da AWS e selecione AWS Fault Injection Service. Valores até e incluindo as cotas aprovadas automaticamente são aplicados instantaneamente. As cotas aprovadas automaticamente estão descritas na coluna de descrição na tabela abaixo. Se você precisar de cotas que excedam os limites aprovados automaticamente, envie uma solicitação. Valores acima dos limites aprovados automaticamente são analisados pelo suporte ao cliente e aprovados sempre que possível.

Para solicitar um aumento da cota, consulte [Requesting a quota increase](#) no Guia do usuário do Service Quotas.

Você Conta da AWS tem as seguintes cotas relacionadas ao AWS FIS.

Nome	Padrão	Ajusté	Description
Duração da ação em horas	Cada região compatível: 12	Não	O número máximo de horas permitidas para executar uma ação nesta conta na região atual.
Tabela global de minutos de ação por multirregião fortemente consistente (MRSC) para aws:dynamodb: action global-table-pause-replication	Cada região suportada: 5.040	Sim	O máximo de minutos cumulativos de ação por tabela global do MRSC que o aws:dynamodb: global-table-pause-replication pode atingir em uma janela contínua de 7 dias.
Ações por modelo de experimento	Cada região compatível: 20	Não	O número máximo de ações que você pode criar em um modelo de

Nome	Padrão	Ajuste	Description
			experimento nesta conta na região atual.
Experimentos ativos	Cada região compatível: 5	Não	O número máximo de experimentos ativos que você pode executar simultaneamente nesta conta na região atual.
Retenção de dados do experimento concluída em dias	Cada região compatível: 120	Não	O número máximo de dias permitido para o AWS FIS reter dados sobre experimentos concluídos nessa conta na região atual.
Duração do experimento em horas	Cada região compatível: 12	Não	O número máximo de horas permitidas para executar um experimento nesta conta na região atual.
Modelos de experimentos	Cada região com suporte: 500	Não	O número máximo de modelos de experimento que você pode criar nesta conta na região atual.
Número máximo de listas de prefixos gerenciados em aws:network: -region-connectivity route-table-disrupt-cross	Cada região compatível: 15	Não	O número máximo de listas de prefixos gerenciados que aws:network: route-table-disrupt-cross -region-connectivity permitirá, por ação.

Nome	Padrão	Ajuste	Description
Número máximo de tabelas de rotas em <code>aws:network: -region-connectivity route-table-disrupt-cross</code>	Cada região com suporte: 10	Não	O número máximo de tabelas de rotas que <code>aws:network: route-table-disrupt-cross -region-connectivity</code> permitirá, por ação.
Número máximo de rotas em <code>aws:network: -region-connectivity route-table-disrupt-cross</code>	Cada região compatível: 200	Não	O número máximo de rotas que o <code>aws:network: route-table-disrupt-cross -region-connectivity</code> permitirá, por ação.
Ações paralelas por experimento	Cada região com suporte: 10	Não	O número máximo de ações que você pode executar em paralelo em um experimento nessa conta, na região atual.
Condições de parada por modelo de experimento	Cada região compatível: 5	Não	O número máximo de condições de parada que você pode adicionar a um modelo de experimento nessa conta, na região atual.
Escolha clusters do Aurora DSQL para <code>aws:dsql: action. cluster-connection-failure</code>	Cada região compatível: 100	Sim	O número máximo de clusters do Aurora DSQL que <code>aws:dsql: cluster-connection-failure</code> pode atingir quando você identifica alvos usando tags, por experimento.

Nome	Padrão	Ajusté	Description
Grupos de Auto Scaling de destino para aws:ec2: -error asg-insufficient-instance-capacity	Cada região com suporte: 500	Sim	O número máximo de grupos de Auto Scaling que o aws:ec2: asg-insufficient-instance-capacity -error pode atingir quando você identifica alvos usando tags, por experimento.
Buckets de destino para aws:s3: bucket-pause-replication	Cada região com suporte: 25	Sim	O número máximo de buckets S3 que aws:s3: bucket-pause-replication pode atingir quando você identifica alvos usando tags, por experimento.
Clusters de destino para aws:ecs: drain-container-instances	Cada região compatível: 100	Sim	O número máximo de clusters que aws:ecs: drain-container-instances pode atingir quando você identifica alvos usando tags, por experimento.
Clusters de destino para aws:rds: failover-db-cluster	Cada região compatível: 160	Sim	O número máximo de clusters que aws:rds: failover-db-cluster pode atingir quando você identifica alvos usando tags, por experimento.

Nome	Padrão	Ajuste	Description
Destino DBInstances para aws:rds:reboot-db-instances	Cada região suportada: 130	Sim	O número máximo de DBInstances que aws:rds:reboot-db-instances pode atingir quando você identifica alvos usando tags, por experimento.
Instâncias de destino para aws:ec2:reboot-instances	Cada região compatível: 600	Sim	O número máximo de instâncias que aws:ec2:reboot-instances pode atingir quando você identifica alvos usando tags, por experimento.
Instâncias de destino para aws:ec2:stop-instances	Cada região compatível: 400	Sim	O número máximo de instâncias que aws:ec2:stop-instances pode atingir quando você identifica alvos usando tags, por experimento.
Instâncias de destino para aws:ec2:terminate-instances	Cada região com suporte: 300	Sim	O número máximo de instâncias que aws:ec2:terminate-instances pode atingir quando você identifica alvos usando tags, por experimento.
Instâncias de destino para aws:ssm:send-command	Cada região compatível: 200	Sim	O número máximo de instâncias que aws:ssm:send-command pode atingir quando você identifica alvos usando tags, por experimento.

Nome	Padrão	Ajusté	Description
Direcione os streams de dados do Kinesis para aws:kinesis: action. stream-expired-iterator-exception	Cada região suportada: 280	Sim	O número máximo de streams de dados do Kinesis que aws:kinesis: stream-expired-iterator-exception pode atingir quando você identifica alvos usando tags, por experimento.
Direcione os streams de dados do Kinesis para aws:kinesis: action. stream-provisioned-throughput-exception	Cada região suportada: 280	Sim	O número máximo de streams de dados do Kinesis que aws:kinesis: stream-provisioned-throughput-exception pode atingir quando você identifica alvos usando tags, por experimento.
Alvo ManagedResources para aws:arc: start-zonal-autoshift action.	Cada região compatível: 200	Sim	O número máximo de recursos gerenciados que aws:arc: start-zonal-autoshift pode atingir quando você identifica alvos usando tags, por experimento.
Grupos de nós de destino para aws:eks: terminate-nodegroup-instances	Cada região compatível: 100	Sim	O número máximo de grupos de nós que aws:eks: terminate-nodegroup-instances pode segmentar quando você identifica alvos usando tags, por experimento.

Nome	Padrão	Ajusté	Description
Pods de destino para aws:eks: pod-cpu-stress	Cada região com suporte: 1.000	Sim	O número máximo de pods que aws:eks: pod-cpu-stress pode atingir quando você identifica alvos usando parâmetros, por experimento.
Pods de destino para aws:eks: pod-delete	Cada região com suporte: 1.000	Sim	O número máximo de pods que aws:pod-delete pode atingir quando você identifica alvos usando parâmetros, por experimento.
Pods de destino para aws:eks: pod-io-stress	Cada região com suporte: 1.000	Sim	O número máximo de pods que aws:eks: pod-io-stress pode atingir quando você identifica alvos usando parâmetros, por experimento.
Pods de destino para aws:eks: pod-memory-stress	Cada região com suporte: 1.000	Sim	O número máximo de pods que aws:eks: pod-memory-stress pode atingir quando você identifica alvos usando parâmetros, por experimento.

Nome	Padrão	Ajusté	Description
Pods de destino para aws:eks: pod-network-blackhole-port	Cada região com suporte: 1.000	Sim	O número máximo de pods que aws:eks: pod-network-blackhole-port pode atingir quando você identifica alvos usando parâmetros, por experimento.
Pods de destino para aws:eks: pod-network-latency	Cada região com suporte: 1.000	Sim	O número máximo de pods que aws:eks: pod-network-latency pode atingir quando você identifica alvos usando parâmetros, por experimento.
Pods de destino para aws:eks: pod-network-packet-loss	Cada região com suporte: 1.000	Sim	O número máximo de pods que aws:eks: pod-network-packet-loss pode atingir quando você identifica alvos usando parâmetros, por experimento.
Alvo ReplicationGroups para aws:elasticache: - Depreciação planejada interrupt-cluster-az-power	Cada região compatível: 5	Sim	O número máximo ReplicationGroups que aws:elasticache: interrupt-cluster-az-power pode atingir quando você identifica alvos usando tags/parâmetros, por experimento.

Nome	Padrão	Ajuste	Description
Destino ReplicationGroups para aws:elasticache: replicationgroup-interrupt-az-power	Cada região compatível: 20	Sim	O número máximo de ReplicationGroups que aws:elasticache: replicationgroup-interrupt-az-power pode atingir por experimento. Um limite diário se aplica à segmentação. Para obter mais informações, visite: https://docs.aws.amazon.com/fis/latest/userguide/fis-quotas.html .
Destino SpotInstances para aws:ec2: send-spot-instance-interruptions	Cada região compatível: 100	Sim	O número máximo de SpotInstances alvos aws:ec2: send-spot-instance-interruptions pode ser definido quando você identifica alvos usando tags, por experimento.

Nome	Padrão	Ajuste	Description
Sub-redes de destino para aws:network:disrupt-connectivity	Cada região compatível: 100	Sim	O número máximo de sub-redes que aws:network:disrupt-connectivity pode atingir quando você identifica alvos usando tags, por experimento. As cotas acima de 5 se aplicam somente ao parâmetro scope:all. Se for necessário ter uma cota maior para outro tipo de escopo, entre em contato com o suporte ao cliente.
Sub-redes de destino para aws:network:-region-connectivity route-table-disrupt-cross	Cada região compatível: 50	Sim	O número máximo de sub-redes que aws:network: route-table-disrupt-cross -region-connectivity pode atingir quando você identifica alvos usando tags, por experimento.
Tarefas de destino para aws:ecs:stop-task	Cada região com suporte: 500	Sim	O número máximo de tarefas que aws:stop-task pode atingir quando você identifica alvos usando tags, por experimento.

Nome	Padrão	Ajusté	Description
Tarefas de destino para aws:ecs: task-cpu-stress	Cada região compatível: 200	Sim	O número máximo de tarefas que aws:ecs: task-cpu-stress pode segmentar quando você identifica alvos usando tags/parâmetros, por experimento.
Tarefas de destino para aws:ecs: task-io-stress	Cada região compatível: 200	Sim	O número máximo de tarefas que aws:ecs: task-io-stress pode segmentar quando você identifica alvos usando tags/parâmetros, por experimento.
Tarefas de destino para aws:ecs: task-kill-process	Cada região compatível: 200	Sim	O número máximo de tarefas que aws:ecs: task-kill-process pode segmentar quando você identifica alvos usando tags/parâmetros, por experimento.
Tarefas de destino para aws:ecs: task-network-blackhole-port	Cada região compatível: 200	Sim	O número máximo de tarefas que aws:ecs: task-network-blackhole-port pode segmentar quando você identifica alvos usando tags/parâmetros, por experimento.

Nome	Padrão	Ajuste	Description
Tarefas de destino para aws:ecs: task-network-latency	Cada região compatível: 200	Sim	O número máximo de tarefas que aws:ecs: task-network-latency pode segmentar quando você identifica alvos usando tags/parâmetros, por experimento.
Tarefas de destino para aws:ecs: task-network-packet-loss	Cada região compatível: 200	Sim	O número máximo de tarefas que aws:ecs: task-network-packet-loss pode segmentar quando você identifica alvos usando tags/parâmetros, por experimento.
Destino TransitGateways para aws:network: -region-connectivity transit-gateway-disrupt-cross	Cada região compatível: 50	Sim	O número máximo de Transit Gateways que aws:network: transit-gateway-disrupt-cross -region-connectivity pode atingir quando você identifica alvos usando tags, por experimento.
Interfaces virtuais de destino para aws:directconnect:virtual-interface	Cada região compatível: 5	Sim	O número máximo de interfaces virtuais que aws:directconnect:virtual-interface pode atingir quando você identifica alvos usando tags, por experimento.

Nome	Padrão	Ajusté	Description
Volumes de destino para aws:eks:pause-volume-io	Cada região compatível: 160	Sim	O número máximo de volumes que aws:eks:pause-volume-io pode segmentar quando você identifica alvos usando tags, por experimento.
Configurações de conta de destino por modelo de experimento	Cada região compatível: 40	Sim	O número máximo de configurações da conta de destino que você pode criar para um modelo de experimento nessa conta na região atual.
Funções de destino para aws:lambda:action. invocation-add-delay	Cada região compatível: 140	Sim	O número máximo de funções do Lambda que aws:lambda: invocation-add-delay pode segmentar quando você identifica alvos usando tags, por experimento.
Funções de destino para a ação aws:lambda:invocation-error.	Cada região compatível: 140	Sim	Número máximo de funções do Lambda que o aws:lambda:invocation-error pode direcionar quando você identifica os destinos usando etiquetas, por experimento.

Nome	Padrão	Ajuste	Description
Funções de destino para <code>aws:lambda: action. invocation-http-integration-response</code>	Cada região compatível: 140	Sim	O número máximo de funções do Lambda que <code>aws:lambda: invocation-http-integration-response</code> pode segmentar quando você identifica alvos usando tags, por experimento.
Clusters multirregionais de destino para a ação <code>aws:memorydb: -replication. multi-region-cluster-pause</code>	Cada região compatível: 50	Sim	O número máximo de clusters multirregionais do MemoryDB que o <code>aws:memorydb: multi-region-cluster-pause -replication</code> pode atingir quando você identifica destinos usando tags, por experimento. Se for necessário ter uma cota maior, entre em contato com o suporte ao cliente.
Tabelas de destino para <code>aws:dynamodb: action global-table-pause-replication</code>	Cada região compatível: 60	Sim	O número máximo de tabelas globais que o <code>aws:dynamodb: global-table-pause-replication</code> pode segmentar, por experimento.

Seu uso do AWS FIS está sujeito às seguintes limitações adicionais:

Nome	Limitação
Limite diário de metas ReplicationGroups para aws:elasticache: replicationgroup-interrupt-az-power	O limite é de 20 ReplicationGroups segmentados por conta, por região, por dia. Você pode solicitar um aumento criando um caso de suporte no console do AWS Support Center .

Histórico do documento

A tabela a seguir descreve atualizações importantes na documentação do Guia do usuário do AWS Fault Injection Service.

Alteração	Descrição	Data
Nova ação para AWS Direct Connect	Você pode usar a ação <code>aws:directconnect:virtual-interface</code> para testar a resiliência da AWS Direct Connect conexão interrompendo temporariamente as sessões do Border Gateway Protocol entre as redes locais e os pares associados às interfaces virtuais de destino.	18 de dezembro de 2025
Novos cenários	Agora você pode usar os novos cenários “AZ: Desaceleração do aplicativo” e “Cross-AZ: desaceleração do tráfego”.	12 de novembro de 2025
Novo parâmetro para documentos SSM	Os documentos SSM <code>AWSFIS-Run-Network-Latency-Sources</code> e <code>-Network-Packet-Loss-Sources</code> agora oferecem suporte ao parâmetro <code>AWSFIS-RunFlowsPercent</code> .	12 de novembro de 2025
Novo parâmetro para ações ECS e EKS	As ações <code>aws:ecs:task-network-latency</code> , <code>aws:ecs:</code> , <code>aws:eks: pod-network-latency</code> e <code>aws:eks: task-network-packet-loss</code> agora oferecem suporte	12 de novembro de 2025

	ao parâmetro FlowsPercent. pod-network-packet-loss	
AWS atualizações de políticas gerenciadas	Política gerenciada atualizada AWSFaultInjectionSimulatorEC2Acesso : DescribeSubnets permissão ec2: adicionada.	12 de novembro de 2025
AWS atualizações de políticas gerenciadas	Política gerenciada atualizada AWSFaultInjectionSimulatorECSAccess : foram adicionadas as permissões ecs: DescribeContainerInstances, ec2: DescribeSubnets e ec2: DescribeInstances	12 de novembro de 2025
Novas ações para o Amazon Kinesis Data Streams	Você pode usar as ações aws:kinesis:stream.	15 de outubro de 2025
Novo experimento para o Amazon EBS	Você pode usar a volume-io-latency ação aws:ebs: para simular I/O latência elevada em seus volumes do Amazon EBS.	16 de setembro de 2025
Novo escopo de ação AWS FIS	Você pode usar a ação aws:network:disrupt-connectivity para interromper a conectividade com seus buckets de diretório do S3 Express One Zone. Agora, esse escopo também está incluído no cenário de Disponibilidade de AZ: Interrupção de Energia.	18 de agosto de 2025

[Suporte ao MemoryDB no FIS AWS](#)

Você pode usar o AWS FIS para testar como seu aplicativo com clusters multirregionais do Amazon MemoryDB responde à interrupção da replicação de dados durante uma interrupção na rede entre regiões.

15 de julho de 2025

[Suporte ARC no AWS FIS](#)

Você pode usar o AWS FIS para testar como o deslocamento automático zonal ARC recupera automaticamente seu aplicativo durante uma interrupção de alimentação do AZ.

26 de março de 2025

[Nova configuração do relatório do experimento](#)

Agora você pode ativar o AWS FIS para gerar relatórios para experimentos que resumem as ações e respostas do experimento a partir de CloudWatch painéis.

12 de novembro de 2024

[Novas ações do Lambda](#)

Agora você pode usar as ações `aws:lambda:function` para injetar falhas nas invocações de suas funções do Lambda.

31 de outubro de 2024

[Novo recurso de medidas de segurança](#)

AWSO FIS agora suporta alavancas de segurança que permitem interromper rapidamente todos os experimentos em execução e impedir que novos experimentos comecem.

3 de setembro de 2024

Novo capítulo de solução de problemas	AWSA FIS adicionou um guia de solução de problemas que inclui códigos de erro e contexto para experimentos com falha.	13 de agosto de 2024
Nova ação	Agora você pode usar a ação <code>aws:dynamodb:global-table-pause-replication</code> para pausar a replicação de dados entre a tabela global de destino e as respectivas tabelas de réplica. Não haverá mais suporte à ação <code>aws:dynamodb:encrypted-global-table-pause-replication</code> .	24 de abril de 2024
Nova opção de experimento do modo de ações	Você pode definir o modo de ações para <code>skip-all</code> a fim de gerar uma pré-visualização de destinos antes de realizar um experimento.	13 de março de 2024
AWS atualizações de políticas gerenciadas	AWSO FIS atualizou as políticas gerenciadas existentes.	25 de janeiro de 2024
Novos cenários e ações	Agora você pode usar cenários AWS FIS entre regiões: conectividade e disponibilidade de AZ: interrupção de energia.	30 de novembro de 2023

Nova ação	Agora, você pode usar a ação <code>aws:ec2:asg-insufficient-in-stance-capacity-error</code> .	30 de novembro de 2023
Nova ação	Agora, você pode usar a ação <code>aws:ec2:api-insufficient-in-stance-capacity-error</code> .	30 de novembro de 2023
Nova ação	Agora, você pode usar a ação <code>aws:network:route-table-disrupt-cross-region-connectivity</code> .	30 de novembro de 2023
Nova ação	Agora, você pode usar a ação <code>aws:network:transit-gateway-disrupt-cross-region-connectivity</code> .	30 de novembro de 2023
Nova ação	Agora, você pode usar a ação <code>aws:dynamodb:encrypted-global-table-pause-replication</code> .	30 de novembro de 2023
Nova ação	Agora, você pode usar a ação <code>aws:s3:bucket-pause-replication</code> .	30 de novembro de 2023
Nova ação	Agora, você pode usar a ação <code>aws:elasticache:interrupt-cluster-az-power</code> .	30 de novembro de 2023
Novas opções do experimento	Agora você pode usar as opções do experimento AWS FIS para segmentação de contas e resolução de metas vazias.	27 de novembro de 2023
Mudança de nome do AWS FIS	Nome do serviço atualizado para AWS Fault Injection Service.	15 de novembro de 2023

AWSatualizações de políticas gerenciadas	AWSO FIS atualizou as políticas gerenciadas existentes.	13 de novembro de 2023
Nova biblioteca de cenários	Agora você pode usar o recurso de biblioteca de cenários do AWS FIS.	7 de novembro de 2023
Novo agendador de experimentos	Agora você pode usar o recurso de agendador de experimentos do AWS FIS.	7 de novembro de 2023
AWSatualizações de políticas gerenciadas	AWSO FIS atualizou as políticas gerenciadas existentes.	2 de junho de 2023
Novas ações	Você pode usar as novas ações <code>aws:ecs:task</code> e <code>aws:eks:pod</code> .	1º de junho de 2023
AWSatualizações de políticas gerenciadas	AWSO FIS atualizou as políticas gerenciadas existentes.	1º de junho de 2023
Novo documento do SSM pré-configurado	Você pode usar o seguinte documento SSM pré-configurado: <code>AWSFIS-Run -Disk-File</code> .	28 de abril de 2023
Nova ação	Você pode usar a <code>aws:ebs:pause-volume-io</code> ação para pausar I/O entre os volumes de destino e as instâncias às quais eles estão conectados.	27 de janeiro de 2023

Nova ação	Você pode usar a ação <code>aws:network:disrupt-connectivity</code> para negar tipos específicos de tráfego para as sub-redes de destino.	26 de outubro de 2022
Nova ação	Você pode usar a ação <code>aws:eks:inject-kubernetes-custom-resource</code> para executar um experimento ChaosMesh ou Litmus em um único cluster de destino.	7 de julho de 2022
Registro em log do experimento	Você pode configurar seus modelos de experimento para enviar registros de atividades do experimento para o CloudWatch Logs ou para um bucket do S3.	28 de fevereiro de 2022
Novas notificações	Quando o estado de um experimento muda, o AWS FIS emite uma notificação. Essas notificações são disponibilizadas como eventos pela Amazon EventBridge.	24 de fevereiro de 2022
Nova ação	Você pode usar a ação <code>aws:ecs:stop-task</code> para interromper a tarefa especificada.	9 de fevereiro de 2022

Nova ação	Você pode usar a ação <code>aws:cloudwatch:assert-alarm-state</code> para verificar se os alarmes especificados estão em um dos estados de alarme especificado.	5 de novembro de 2021
Novos documentos do SSM pré-configurados	Você pode usar os seguintes documentos SSM pré-configurados: <code>AWSFIS-Run -IO-Stress</code> , <code>-Network-Blackhold-Port</code> , <code>-Network-Latency-Sources</code> , <code>-Network-Packet-Loss</code> e <code>AWSFIS-Run -Network-Packet-Loss-Sources</code> . <code>AWSFIS-Run AWSFIS-Run AWSFIS-Run</code>	4 de novembro de 2021
Nova ação	Você pode usar a ação <code>aws:ec2:send-spot-instance-interruptions</code> para enviar um aviso de interrupção da instância spot às instâncias spot de destino e depois interromper as instâncias spot de destino.	20 de outubro de 2021
Nova ação	Você pode usar a ação <code>aws:ssm:start-automation-execution</code> para iniciar a execução de um runbook de automação.	17 de setembro de 2021
Lançamento inicial	A versão inicial do Guia do Usuário do AWS Fault Injection Service.	15 de março de 2021

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.