User Guide

AWS Fault Injection Service

AWS Fault Injection Service: User Guide

Table of Contents

What is AWS Fault Injection Service?

AWS Fault Injection Service (AWS FIS) is a managed service that enables you to perform fault injection experiments on your AWS workloads. Fault injection is based on the principles of chaos engineering. These experiments stress an application by creating disruptive events so that you can observe how your application responds. You can then use this information to improve the performance and resiliency of your applications so that they behave as expected.

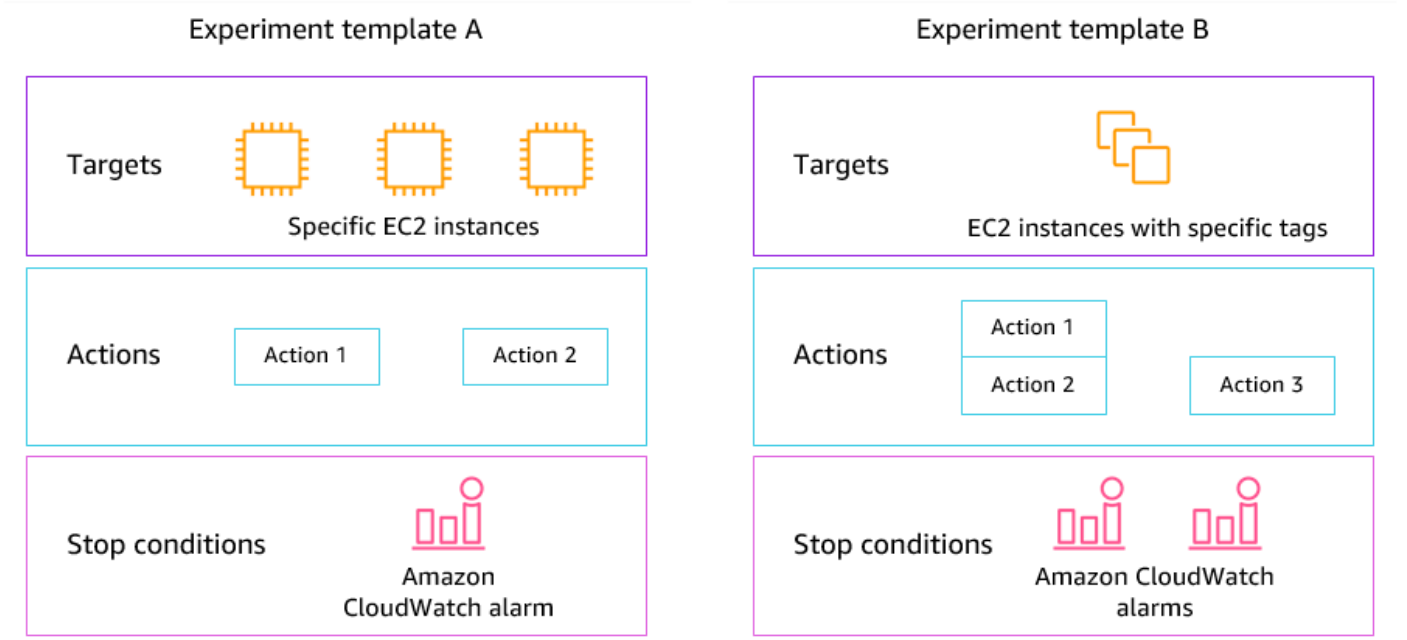
To use AWS FIS, you set up and run experiments that help you create the real-world conditions needed to uncover application issues that can be difficult to find otherwise. AWS FIS provides templates that generate disruptions, and the controls and guardrails that you need to run experiments in production, such as automatically rolling back or stopping the experiment if specific conditions are met.

> **Important**
>
> AWS FIS carries out real actions on real AWS resources in your system. Therefore, before you use AWS FIS to run experiments in production, we strongly recommend that you complete a planning phase and run the experiments in a pre-production environment.

For more information about planning your experiment, see Test Reliability and Plan your AWS FIS experiments. For more information about AWS FIS, see AWS Fault Injection Service.

AWS FIS concepts

To use AWS FIS, you run *experiments* on your AWS resources to test your theory of how an application or system will perform under fault conditions. To run experiments, you first create an *experiment template*. An experiment template is the blueprint of your experiment. It contains the *actions*, *targets*, and *stop conditions* for the experiment. After you create an experiment template, you can use it to run an experiment. While your experiment is running, you can track its progress and view its status. An experiment is complete when all of the actions in the experiment have run.



Actions

An *action* is an activity that AWS FIS performs on an AWS resource during an experiment. AWS FIS provides a set of preconfigured actions based on the type of AWS resource. Each action runs for a specified duration during an experiment, or until you stop the experiment. Actions can run sequentially or simultaneously (in parallel).

Targets

A *target* is one or more AWS resources on which AWS FIS performs an action during an experiment. You can choose specific resources, or you can select a group of resources based on specific criteria, such as tags or state.

Stop conditions

AWS FIS provides the controls and guardrails that you need to run experiments safely on your AWS workloads. A *stop condition* is a mechanism to stop an experiment if it reaches a threshold that you define as an Amazon CloudWatch alarm. If a stop condition is triggered while the experiment is running, AWS FIS stops the experiment.

Supported AWS services

AWS FIS provides preconfigured actions for specific types of targets across AWS services. AWS FIS supports actions for target resources for the following AWS services:

- Amazon CloudWatch
- Amazon DynamoDB
- Amazon EBS
- Amazon EC2
- Amazon ECS
- Amazon EKS
- Amazon ElastiCache
- Amazon RDS
- Amazon S3
- AWS Systems Manager
- Amazon VPC

For single-account experiments, the target resources must be in the same AWS account as the experiment. You can run AWS FIS experiments that target resources in a different AWS account account using AWS FIS multi-account experiments.

For more information, see Actions for AWS FIS.

Access AWS FIS

You can work with AWS FIS in any of the following ways:

- **AWS Management Console** — Provides a web interface that you can use to access AWS FIS. For more information, see Working with the AWS Management Console.
- **AWS Command Line Interface (AWS CLI)** — Provides commands for a broad set of AWS services, including AWS FIS, and is supported on Windows, macOS, and Linux. For more information, see AWS Command Line Interface. For more information about the commands for AWS FIS, see fis in the *AWS CLI Command Reference*.

- **AWS CloudFormation** — Create templates that describe your AWS resources. You use the templates to provision and manage these resources as a single unit. For more information, see the AWS Fault Injection Service resource type reference.
- **AWS SDKs** — Provides language-specific APIs and takes care of many of the connection details, such as calculating signatures, handling request retries, and handling errors. For more information, see AWS SDKs.
- **HTTPS API** — Provides low-level API actions that you can call using HTTPS requests. For more information, see the AWS Fault Injection Service API Reference.

Pricing for AWS FIS

You are charged per minute that an action runs, from start to finish, based on the number of target accounts for your experiment. For more information, see AWS FIS Pricing.

Plan your AWS FIS experiments

Fault injection is the process of stressing an application in testing or production environments by creating disruptive events, such as server outages or API throttling. From observing how the system responds, you can then implement improvements. When you run experiments on your system, it can help you to identify systemic weaknesses in a controlled manner, before those weaknesses affect the customers who depend on your system. Then you can proactively address the issues to help prevent unpredictable outcomes.

Before you get started running fault injection experiments using AWS FIS, we recommend that you familiarize yourself with the following principles and guidelines.

> **Important**
>
> AWS FIS carries out real actions on real AWS resources in your system. Therefore, before you get started using AWS FIS to run experiments, we strongly recommend that you first complete a planning phase and a test in a pre-production or test environment.

Contents

Basic principles and guidelines

Before starting experiments with AWS FIS, take the following steps:

1. **Identify the target deployment for the experiment** — Start by identifying the target deployment. If this is your first experiment, we recommend starting in a pre-production or test environment.
2. **Review the application architecture** — You must ensure that you have identified all of the application components, dependencies, and recovery procedures for each component. Begin with reviewing the application architecture. Depending on the application, refer to the AWS Well-Architected Framework.

3. **Define steady-state behavior** — Define the steady-state behavior of your system in terms of important technical and business metrics, such as latency, CPU load, failed sign-ins per minute, number of retries, or page load speed.
4. **Form a hypothesis** — Form a hypothesis of how you expect the system behavior to change during the experiment. A hypothesis definition follows this format:

   If *`fault injection action`* is performed, the *`business or technical metric impact`* should not exceed *`value`*.

   For example, a hypothesis for an authentication service might read as follows: "If network latency increases by 10%, there is less than a 1% increase in sign-in failures." After the experiment is completed, you evaluate whether the application resiliency aligns with your business and technical expectations.

We also recommend following these guidelines when working with AWS FIS:

- Always start experimenting with AWS FIS in a test environment. Never start with a production environment. As you progress in your fault injection experiments, you can experiment in other controlled environments beyond the test environment.
- Build your team's confidence in your application resilience by starting with small, simple experiments, such as running the **aws:ec2:stop-instances** action on one target.
- Fault injection can cause real issues. Proceed with caution and make sure that your first fault injections are on test instances so that no customers are affected.
- Test, test, and test some more. Fault injection is meant to be implemented in a controlled environment with well-planned experiments. This allows you to build confidence in the abilities of your application and your tools to withstand turbulent conditions.
- We strongly recommend that you have an excellent monitoring and alerting program in place before you begin. Without it, you won't be able to understand or measure the impact of your experiments, which is critical to sustainable fault injection practices.

Experiment planning guidelines

With AWS FIS, you run experiments on your AWS resources to test your theory of how an application or system will perform under fault conditions.

The following are recommended guidelines for planning your AWS FIS experiments.

- **Review outage history** — Review the previous outages and events for your system. This can help you to build up a picture of the overall health and resiliency of your system. Before you start running experiments on your system, you should address known issues and weaknesses in your system.
- **Identify services with the largest impact** — Review your services and identify the ones that have the biggest impact on your end users or customers if they go down or do not function correctly.
- **Identify the target system** — The target system is the system on which you will run experiments. If you are new to AWS FIS or you have never run fault injection experiments before, we recommend that you start by running experiments on a pre-production or test system.
- **Consult with your team** — Ask what they are worried about. You can form a hypothesis to prove or disprove their concerns. You can also ask your team what they are not worried about. This question can reveal two common fallacies: the sunk cost fallacy and the confirmation bias fallacy. Forming a hypothesis based on your team's answers can help provide more information about the reality of your system's state.
- **Review your application architecture** — Conduct a review of your system or application and ensure that you have identified all of the application components, dependencies, and recovery procedures for every component.

  We recommend that you review the AWS Well-Architected Framework. The framework can help you build secure, high-performing, resilient, and efficient infrastructure for your applications and workloads. For more information, see AWS Well-Architected.
- **Identify the applicable metrics** — You can monitor the impact of an experiment on your AWS resources using Amazon CloudWatch metrics. You can use these metrics to determine the baseline or "steady state" when your application is performing optimally. Then, you can monitor these metrics during or after the experiment to determine the impact. For more information, see Monitor AWS FIS usage metrics using Amazon CloudWatch.
- **Define an acceptable performance threshold for your system** — Identify the metric that represents an acceptable, steady state for your system. You will use this metric to create one or more CloudWatch alarms that represent a stop condition for your experiment. If the alarm is triggered, the experiment is automatically stopped. For more information, see Stop conditions for AWS FIS.

Tutorials for AWS Fault Injection Service

The following tutorials show you how to create and run experiments using AWS Fault Injection Service (AWS FIS).

Tutorials

- Tutorial: Test instance stop and start using AWS FIS
- Tutorial: Run CPU stress on an instance using AWS FIS
- Tutorial: Test Spot Instance interruptions using AWS FIS
- Tutorial: Simulate a connectivity event
- Tutorial: Schedule a recurring experiment

Tutorial: Test instance stop and start using AWS FIS

You can use AWS Fault Injection Service (AWS FIS) to test how your applications handle instance stop and start. Use this tutorial to create an experiment template that uses the AWS FIS `aws:ec2:stop-instances` action to stop one instance and then a second instance.

Prerequisites

To complete this tutorial, ensure that you do the following:

- Launch two test EC2 instances in your account. After you launch your instances, note the IDs of both instances.
- Create an IAM role that enables the AWS FIS service to perform the `aws:ec2:stop-instances` action on your behalf. For more information, see IAM roles for AWS FIS experiments.
- Ensure that you have access to AWS FIS. For more information, see AWS FIS policy examples.

Step 1: Create an experiment template

Create the experiment template using the AWS FIS console. In the template, you specify two actions that will run sequentially for three minutes each. The first action stops one of the test instances, which AWS FIS chooses randomly. The second action stops both test instances.

To create an experiment template

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. Choose **Create experiment template**.
4. For **Description and name**, enter a description and a name for the template.
5. For **Actions**, do the following:
   a. Choose **Add action**.
   b. Enter a name for the action. For example, enter **`stopOneInstance`**.
   c. For **Action type**, choose **aws:ec2:stop-instances**.
   d. For **Target** keep the target that AWS FIS creates for you.
   e. For **Action parameters**, **Start instances after duration**, specify 3 minutes (PT3M).
   f. Choose **Save**.
6. For **Targets**, do the following:
   a. Choose **Edit** for the target that AWS FIS automatically created for you in the previous step.
   b. Replace the default name with a more descriptive name. For example, enter **`oneRandomInstance`**.
   c. Verify that **Resource type** is **aws:ec2:instance**.
   d. For **Target method**, choose **Resource IDs**, and then choose the IDs of the two test instances.
   e. For **Selection mode**, choose **Count**. For **Number of resources**, enter **1**.
   f. Choose **Save**.
7. Choose **Add target** and do the following:
   a. Enter a name for the target. For example, enter **`bothInstances`**.
   b. For **Resource type**, choose **aws:ec2:instance**.
   c. For **Target method**, choose **Resource IDs**, and then choose the IDs of the two test instances.
   d. For **Selection mode**, choose **All**.
   e. Choose **Save**.
8. From the **Actions** section, choose **Add action**. Do the following:

a. For **Name**, enter a name for the action. For example, enter **`stopBothInstances`**.

b. For **Action type**, choose **aws:ec2:stop-instances**.

c. For **Start after**, choose the first action that you added (**`stopOneInstance`**).

d. For **Target**, choose the second target that you added (**`bothInstances`**).

e. For **Action parameters**, **Start instances after duration**, specify 3 minutes (PT3M).

f. Choose **Save**.

9. For **Service Access**, choose **Use an existing IAM role**, and then choose the IAM role that you created as described in the prerequisites for this tutorial. If your role is not displayed, verify that it has the required trust relationship. For more information, see the section called "Experiment role".

10. (Optional) For **Tags**, choose **Add new tag** and specify a tag key and tag value. The tags that you add are applied to your experiment template, not the experiments that are run using the template.

11. Choose **Create experiment template**. When prompted for confirmation, enter **`create`** and then choose **Create experiment template**.

(Optional) To view the experiment template JSON

Choose the **Export** tab. The following is an example of the JSON created by the preceding console procedure.

```
{
    "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": {}
}
```

Step 2: Start the experiment

When you have finished creating your experiment template, you can use it to start an experiment.

To start an experiment

1. You should be on the details page for the experiment template that you just created. Otherwise, choose **Experiment templates** and then select the ID of the experiment template to open the details page.
2. Choose **Start experiment**.
3. (Optional) To add a tag to your experiment, choose **Add new tag** and enter a tag key and a tag value.
4. Choose **Start experiment**. When prompted for confirmation, enter **`start`** and choose **Start experiment**.

Step 3: Track the experiment progress

You can track the progress of a running experiment until the experiment is completed, stopped, or failed.

To track the progress of an experiment

1. You should be on the details page for the experiment that you just started. Otherwise, choose **Experiments** and then select the ID of the experiment to open the details page.
2. To view the state of the experiment, check **State** in the **Details** pane. For more information, see experiment states.
3. When the state of the experiment is **Running**, go to the next step.

Step 4: Verify the experiment result

You can verify that the instances were stopped and started by the experiment as expected.

To verify the result of the experiment

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/ in a new browser tab or window. This allows you to continue to track the progress of the experiment in the AWS FIS console while viewing the result of the experiment in the Amazon EC2 console.
2. In the navigation pane, choose **Instances**.
3. When the state of the first action changes from **Pending** to **Running** (AWS FIS console), the state of one of the target instances changes from **Running** to **Stopped** (Amazon EC2 console).

4. After three minutes, the state of the first action changes to **Completed**, the state of the second action changes to **Running**, and the state of the other target instance changes to **Stopped**.
5. After three minutes, the state of the second action changes to **Completed**, the state of the target instances changes to **Running**, and the state of the experiment changes to **Completed**.

Step 5: Clean up

If you no longer need the test EC2 instances that you created for this experiment, you can terminate them.

To terminate the instances

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Instances**.
3. Select both test instances and choose **Instance state**, **Terminate instance**.
4. When prompted for confirmation, choose **Terminate**.

If you no longer need the experiment template, you can delete it.

To delete an experiment template using the AWS FIS console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. Select the experiment template, and choose **Actions**, **Delete experiment template**.
4. When prompted for confirmation, enter **`delete`** and then choose **Delete experiment template**.

Tutorial: Run CPU stress on an instance using AWS FIS

You can use AWS Fault Injection Service (AWS FIS) to test how your applications handle CPU stress. Use this tutorial to create an experiment template that uses AWS FIS to run a pre-configured SSM document that runs CPU stress on an instance. The tutorial uses a stop condition to halt the experiment when the CPU utilization of the instance exceeds a configured threshold.

For more information, see the section called “Pre-configured AWS FIS SSM documents”.

Prerequisites

Before you can use AWS FIS to run CPU stress, complete the following prerequisites.

Create an IAM role

Create a role and attach a policy that enables AWS FIS to use the `aws:ssm:send-command` action on your behalf. For more information, see IAM roles for AWS FIS experiments.

Verify access to AWS FIS

Ensure that you have access to AWS FIS. For more information, see AWS FIS policy examples.

Prepare a test EC2 instance

- Launch an EC2 instance using Amazon Linux 2 or Ubuntu, as required by the pre-configured SSM documents.
- The instance must be managed by SSM. To verify that the instance is managed by SSM, open the Fleet Manager console. If the instance is not managed by SSM, verify that the SSM Agent is installed and that the instance has an attached IAM role with the **AmazonSSMManagedInstanceCore** policy. To verify the installed SSM Agent, connect to your instance and run the following command.

  Amazon Linux 2

  ```
  yum info amazon-ssm-agent
  ```

  Ubuntu

  ```
  apt list amazon-ssm-agent
  ```

- Enable detailed monitoring for the instance. This provides data in 1-minute periods, for an additional charge. Select the instance and choose **Actions**, **Monitor and troubleshoot**, **Manage detailed monitoring**.

Step 1: Create a CloudWatch alarm for a stop condition

Configure a CloudWatch alarm so that you can stop the experiment if CPU utilization exceeds the threshold that you specify. The following procedure sets the threshold to 50% CPU utilization for the target instance. For more information, see Stop conditions.

To create an alarm that indicates when CPU utilization exceeds a threshold

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Instances**.
3. Select the target instance and choose **Actions**, **Monitor and troubleshoot**, **Manage CloudWatch alarms**.
4. For **Alarm notification**, use the toggle to turn off Amazon SNS notifications.
5. For **Alarm thresholds**, use the following settings:
   - **Group samples by**: **Maximum**
   - **Type of data to sample**: **CPU utilization**
   - **Percent**: **`50`**
   - **Period**: **`1 Minute`**
6. When you're done configuring the alarm, choose **Create**.

Step 2: Create an experiment template

Create the experiment template using the AWS FIS console. In the template, you specify the following action to run: aws:ssm:send-command/AWSFIS-Run-CPU-Stress.

To create an experiment template

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. Choose **Create experiment template**.
4. For **Description and name**, enter a description and a name for the template.
5. For **Actions**, do the following:
   a. Choose **Add action**.
   b. Enter a name for the action. For example, enter **`runCpuStress`**.
   c. For **Action type**, choose **aws:ssm:send-command/AWSFIS-Run-CPU-Stress**. This automatically adds the ARN of the SSM document to **Document ARN**.
   d. For **Target** keep the target that AWS FIS creates for you.
   e. For **Action parameters**, **Document parameters**, enter the following:

```
{"DurationSeconds":"120"}
```

f. For **Action parameters**, **Duration**, specify 5 minutes (PT5M).

g. Choose **Save**.

6. For **Targets**, do the following:

a. Choose **Edit** for the target that AWS FIS automatically created for you in the previous step.

b. Replace the default name with a more descriptive name. For example, enter **`testInstance`**.

c. Verify that **Resource type** is **aws:ec2:instance**.

d. For **Target method**, choose **Resource IDs**, and then choose the ID of the test instance.

e. For **Selection mode**, choose **All**.

f. Choose **Save**.

7. For **Service Access**, choose **Use an existing IAM role**, and then choose the IAM role that you created as described in the prerequisites for this tutorial. If your role is not displayed, verify that it has the required trust relationship. For more information, see the section called "Experiment role".

8. For **Stop conditions**, select the CloudWatch alarm that you created in Step 1.

9. (Optional) For **Tags**, choose **Add new tag** and specify a tag key and tag value. The tags that you add are applied to your experiment template, not the experiments that are run using the template.

10. Choose **Create experiment template**.

(Optional) To view the experiment template JSON

Choose the **Export** tab. The following is an example of the JSON created by the preceding console procedure.

```
{
    "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/AllowFISSSMActions",
    "tags": {}
}
```

Step 3: Start the experiment

When you have finished creating your experiment template, you can use it to start an experiment.

To start an experiment

1. You should be on the details page for the experiment template that you just created. Otherwise, choose **Experiment templates** and then select the ID of the experiment template to open the details page.
2. Choose **Start experiment**.
3. (Optional) To add a tag to your experiment, choose **Add new tag** and enter a tag key and a tag value.
4. Choose **Start experiment**. When prompted for confirmation, enter **`start`**. Choose **Start experiment**.

Step 4: Track the experiment progress

You can track the progress of a running experiment until the experiment completes, stops, or fails.

To track the progress of an experiment

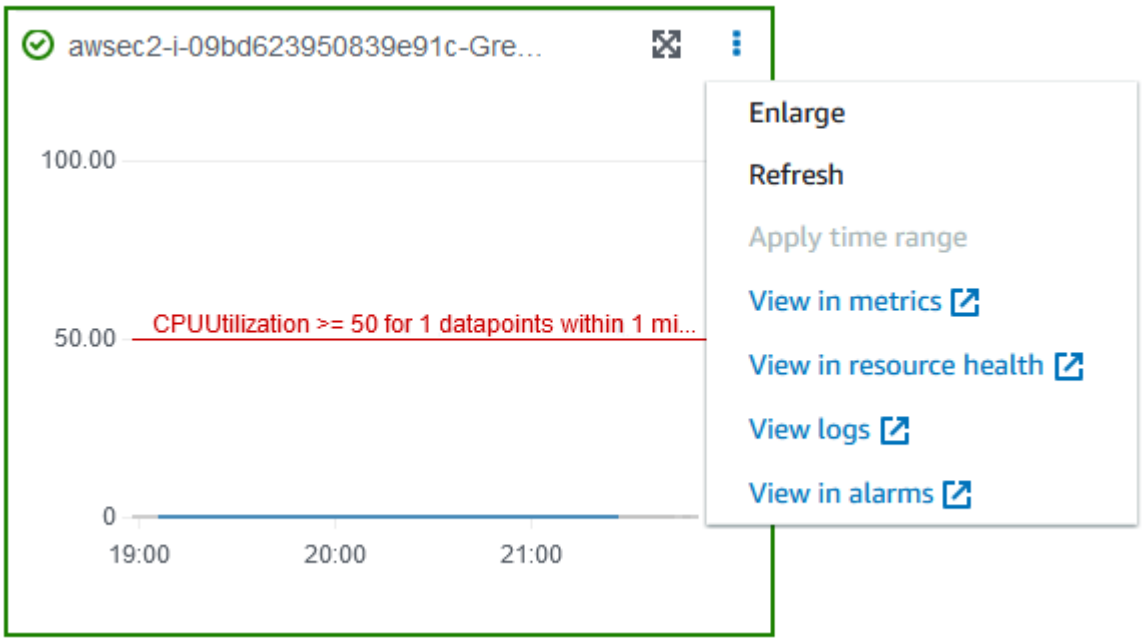
1. You should be on the details page for the experiment that you just started. Otherwise, choose **Experiments** and then select the ID of the experiment to open the details page for the experiment.
2. To view the state of the experiment, check **State** in the **Details** pane. For more information, see experiment states.
3. When the experiment state is **Running**, go to the next step.

Step 5: Verify the experiment results

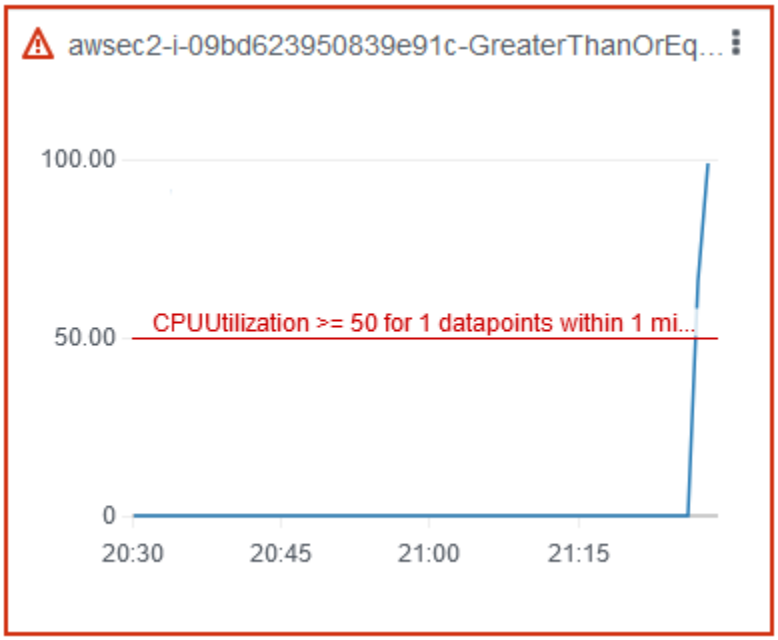
You can monitor the CPU utilization of your instance while the experiment is running. When the CPU utilization reaches the threshold, the alarm is triggered and the experiment is halted by the stop condition.

To verify the results of the experiment

1. Choose the **Stop conditions** tab. The green border and green checkmark icon indicate that the initial state of the alarm is 0K. The red line indicates the alarm threshold. If you prefer a more detailed graph, choose **Enlarge** from the widget menu.



2. When CPU utilization exceeds the threshold, the red border and red exclamation point icon in the **Stop conditions** tab indicate that the alarm state changed to ALARM. In the **Details** pane, the state of the experiment is **Stopped**. If you select the state, the message displayed is "Experiment halted by stop condition".



3. When CPU utilization decreases below the threshold, the green border and green checkmark icon indicate that the alarm state changed to OK.

4. (Optional) Choose **View in alarms** from the widget menu. This opens the alarm details page in the CloudWatch console, where you can get more detail about the alarm or edit the alarm settings.

Step 6: Clean up

If you no longer need the test EC2 instance that you created for this experiment, you can terminate it.

To terminate the instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Instances**.
3. Select the test instances and choose **Instance state**, **Terminate instance**.
4. When prompted for confirmation, choose **Terminate**.

If you no longer need the experiment template, you can delete it.

To delete an experiment template using the AWS FIS console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. Select the experiment template, and choose **Actions**, **Delete experiment template**.
4. When prompted for confirmation, enter **`delete`** and then choose **Delete experiment template**.

Tutorial: Test Spot Instance interruptions using AWS FIS

Spot Instances use spare EC2 capacity that is available, for up to a 90% discount compared to On-Demand pricing. However, Amazon EC2 can interrupt your Spot Instances when it needs the capacity back. When using Spot Instances, you must be prepared for potential interruptions. For more information, see Spot Instance interruptions in the *Amazon EC2 User Guide*.

You can use AWS Fault Injection Service (AWS FIS) to test how your applications handle a Spot Instance interruption. Use this tutorial to create an experiment template that uses the AWS

FIS `aws:ec2:send-spot-instance-interruptions` action to interrupt one of your Spot Instances.

Alternatively, to initiate the experiment using the Amazon EC2 console, see Initiate a Spot Instance interruption in the *Amazon EC2 User Guide*.

Prerequisites

Before you can use AWS FIS to interrupt a Spot Instance, complete the following prerequisites.

1. Create an IAM role

Create a role and attach a policy that enables AWS FIS to perform the `aws:ec2:send-spot-instance-interruptions` action on your behalf. For more information, see IAM roles for AWS FIS experiments.

2. Verify access to AWS FIS

Ensure that you have access to AWS FIS. For more information, see AWS FIS policy examples.

3. (Optional) Create a Spot Instance request

If you'd like a new Spot Instance to use for this experiment, use the run-instances command to request a Spot Instance. The default is to terminate Spot Instances that are interrupted. If you set the interruption behavior to `stop`, you must also set the type to `persistent`. For this tutorial, do not set the interruption behavior to `hibernate`, as the hibernation process begins immediately.

```
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
```

The following is an example of the `spot-options.json` file.

```
{
    "MarketType": "spot",
    "SpotOptions": {
        "SpotInstanceType": "persistent",
        "InstanceInterruptionBehavior": "stop"
```

```
    }
}
```

The `--query` option in the example command makes it so that the command returns only the instance ID of the Spot Instance. The following is example output.

```
[
    "i-0abcdef1234567890"
]
```

4. Add a tag so that AWS FIS can identify the target Spot Instance

Use the create-tags command to add the tag Name=interruptMe to your target Spot Instance.

```
aws ec2 create-tags \
    --resources i-0abcdef1234567890 \
    --tags Key=Name,Value=interruptMe
```

Step 1: Create an experiment template

Create the experiment template using the AWS FIS console. In the template, you specify the action that will run. The action interrupts the Spot Instance with the specified tag. If there is more than one Spot Instance with the tag, AWS FIS chooses one of them at random.

To create an experiment template

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. Choose **Create experiment template**.
4. For **Description and name**, enter a description and a name for the template.
5. For **Actions**, do the following:
    a. Choose **Add action**.
    b. Enter a name for the action. For example, enter **`interruptSpotInstance`**.
    c. For **Action type**, choose **aws:ec2:send-spot-instance-interruptions**.
    d. For **Target** keep the target that AWS FIS creates for you.
    e. For **Action parameters**, **Duration before interruption**, specify 2 Minutes (PT2M).
    f. Choose **Save**.

6. For **Targets**, do the following:

   a. Choose **Edit** for the target that AWS FIS automatically created for you in the previous step.

   b. Replace the default name with a more descriptive name. For example, enter **`oneSpotInstance`**.

   c. Verify that **Resource type** is **aws:ec2:spot-instance**.

   d. For **Target method**, choose **Resource tags, filters, and parameters**.

   e. For **Resource tags**, choose **Add new tag**, and enter the tag key and tag value. Use the tag that you added to the Spot Instance to interrupt, as described in the Prerequisites for this tutorial.

   f. For **Resource filters** choose **Add new filter** and enter **`State.Name`** as the path and **`running`** as the value.

   g. For **Selection mode**, choose **Count**. For **Number of resources**, enter **1**.

   h. Choose **Save**.

7. For **Service Access**, choose **Use an existing IAM role**, and then choose the IAM role that you created as described in the prerequisites for this tutorial. If your role is not displayed, verify that it has the required trust relationship. For more information, see the section called "Experiment role".

8. (Optional) For **Tags**, choose **Add new tag** and specify a tag key and tag value. The tags that you add are applied to your experiment template, not the experiments that are run using the template.

9. Choose **Create experiment template**. When prompted for confirmation, enter **`create`** and then choose **Create experiment template**.

(Optional) To view the experiment template JSON

Choose the **Export** tab. The following is an example of the JSON created by the preceding console procedure.

```
{
    "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"
    }
}
```

Step 2: Start the experiment

When you have finished creating your experiment template, you can use it to start an experiment.

To start an experiment

1. You should be on the details page for the experiment template that you just created. Otherwise, choose **Experiment templates** and then select the ID of the experiment template to open the details page.

2. Choose **Start experiment**.
3. (Optional) To add a tag to your experiment, choose **Add new tag** and enter a tag key and a tag value.
4. Choose **Start experiment**. When prompted for confirmation, enter **`start`** and choose **Start experiment**.

Step 3: Track the experiment progress

You can track the progress of a running experiment until the experiment is completed, stopped, or failed.

To track the progress of an experiment

1. You should be on the details page for the experiment that you just started. Otherwise, choose **Experiments** and then select the ID of the experiment to open the details page.
2. To view the state of the experiment, check **State** in the **Details** pane. For more information, see experiment states.
3. When the state of the experiment is **Running**, go to the next step.

Step 4: Verify the experiment result

When the action for this experiment is completed, the following occurs:

- The target Spot Instance receives an instance rebalance recommendation.
- A Spot Instance interruption notice is issued two minutes before Amazon EC2 terminates or stops your instance.
- After two minutes, the Spot Instance is terminated or stopped.
- A Spot Instance that was stopped by AWS FIS remains stopped until you restart it.

To verify that the instance was interrupted by the experiment

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. From the navigation pane, open **Spot Requests** and **Instances** in separate browser tabs or windows.

3. For **Spot Requests**, select the Spot Instance request. The initial status is `fulfilled`. After the experiment completes, the status changes as follows:
   - `terminate` - The status changes to `instance-terminated-by-experiment`.
   - `stop` - The status changes to `marked-for-stop-by-experiment` and then `instance-stopped-by-experiment`.
4. For **Instances**, select the Spot Instance. The initial status is `Running`. Two minutes after you receive the Spot Instance interruption notice, the status changes as follows:
   - `stop` - The status changes to `Stopping` and then `Stopped`.
   - `terminate` - The status changes to `Shutting-down` and then `Terminated`.

Step 5: Clean up

If you created the test Spot Instance for this experiment with an interruption behavior of `stop` and you no longer need it, you can cancel the Spot Instance request and terminate the Spot Instance.

To cancel the request and terminate the instance using the AWS CLI

1. Use the cancel-spot-instance-requests command to cancel the Spot Instance request.

```
aws ec2 cancel-spot-instance-requests --spot-instance-request-ids sir-ksie869j
```

2. Use the terminate-instances command to terminate the instance.

```
aws ec2 terminate-instances --instance-ids i-0abcdef1234567890
```

If you no longer need the experiment template, you can delete it.

To delete an experiment template using the AWS FIS console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. Select the experiment template, and choose **Actions**, **Delete experiment template**.
4. When prompted for confirmation, enter **`delete`** and then choose **Delete experiment template**.

Tutorial: Simulate a connectivity event

You can use AWS Fault Injection Service (AWS FIS) to simulate a variety of connectivity events. AWS FIS simulates connectivity events by blocking network connections in one of the following ways:

- `all` – Denies all traffic entering and leaving the subnet. Note that this option allows intra-subnet traffic, including traffic to and from network interfaces in the subnet.
- `availability-zone` – Denies intra-VPC traffic to and from subnets in other Availability Zones.
- `dynamodb` – Denies traffic to and from the Regional endpoint for DynamoDB in the current Region.
- `prefix-list` – Denies traffic to and from the specified prefix list.
- `s3` – Denies traffic to and from the Regional endpoint for Amazon S3 in the current Region.
- `vpc` – Denies traffic entering and leaving the VPC.

Use this tutorial to create an experiment template that uses the AWS FIS `aws:network:disrupt-connectivity` action to introduce connectivity loss with Amazon S3 in a target subnet.

Topics

Prerequisites

Before beginning this tutorial, you need a role with the appropriate permissions in your AWS account, and a test Amazon EC2 instance:

A role with permissions in your AWS account

Create a role and attach a policy that enables AWS FIS to perform the `aws:network:disrupt-connectivity` action on your behalf.

Your IAM role requires the following policy:

- AWSFaultInjectionSimulatorNetworkAccess – Grants AWS FIS service permission in Amazon EC2 networking and other required services to perform AWS FIS actions related to network infrastructure.

> **Note**
>
> For simplicity, this tutorial uses an AWS managed policy. For production use, we recommend that you instead grant only the minimum permissions necessary for your use case.
> For more information about how to create an IAM role, see IAM roles for AWS FIS experiments (AWS CLI) or Creating an IAM role (console) in the *IAM User Guide*.

A test Amazon EC2 instance

Launch and connect to a test Amazon EC2 instance. You can use the following tutorial to launch and connect to an Amazon EC2 instance: Tutorial: Get started with Amazon EC2 Linux instances in the *Amazon EC2 User Guide*.

Step 1: Create an AWS FIS experiment template

Create the experiment template by using the AWS FIS AWS Management Console. An AWS FIS template is made up of actions, targets, stop conditions, and an experiment role. For more information about how the templates work, see Experiment templates for AWS FIS.

Before you begin, make sure you have the following ready:

- An IAM role with the correct permissions.
- An Amazon EC2 instance.
- The subnet ID of your Amazon EC2 instance.

To create an experiment template

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.

2. In the left navigation pane, choose **Experiment templates**.
3. Choose **Create experiment template**.
4. Enter a description for the template, such as `Amazon S3 Network Disrupt Connectivity`.
5. Under **Actions**, choose **Add action**.
   a. For the **Name**, enter `disruptConnectivity`.
   b. For **Action type**, select **aws:network:disrupt-connectivity**.
   c. Under **Action parameters**, set the **Duration** to `2 minutes`.
   d. Under **Scope**, select **s3**.
   e. At the top, choose **Save**.
6. Under **Targets**, you should see the target that has been created automatically. Choose **Edit**.
   a. Verify that **Resource type** is `aws:ec2:subnet`.
   b. Under **Target method**, select **Resource IDs**, and then choose the subnet that you used when creating your Amazon EC2 instance in the Prerequisites steps.
   c. Verify that **Selection mode** is **All**.
   d. Choose **Save**.
7. Under **Service Access**, select the IAM role that you created as described in the Prerequisites for this tutorial. If your role is not displayed, verify that it has the required trust relationship. For more information, see the section called “Experiment role”.
8. (Optional) Under **Stop conditions**, you can select a CloudWatch alarm to stop the experiment if the condition occurs. For more information, see Stop conditions for AWS FIS.
9. (Optional) Under **Logs**, you can select an Amazon S3 bucket, or send logs to CloudWatch for your experiment.
10. Choose **Create experiment template**, and when prompted for confirmation, enter `create`. Then choose **Create experiment template**.

Step 2: Ping an Amazon S3 endpoint

Verify that your Amazon EC2 instance is able to reach an Amazon S3 endpoint.

1. Connect to the Amazon EC2 instance that you created in the Prerequisites steps.

For troubleshooting, see Troubleshoot connecting to your instance in the *Amazon EC2 User Guide*.

2. Check to see the AWS Region where your instance is located. You can do this in the Amazon EC2 console or by running the following command.

```
hostname
```

For example, if you launched an Amazon EC2 instance in `us-west-2`, you'll see the following output.

```
[ec2-user@ip-172.16.0.0 ~]$ hostname
ip-172.16.0.0.us-west-2.compute.internal
```

3. Ping an Amazon S3 endpoint in your AWS Region. Replace *AWS Region* with your Region.

```
ping -c 1 s3.AWS Region.amazonaws.com
```

For the output, you should see a successful ping with 0% packet loss, as shown in the following example.

```
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
```

Step 3: Start your AWS FIS experiment

Start an experiment with the experiment template that you just created.

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the left navigation pane, choose **Experiment templates**.
3. Select the ID of the experiment template that you created to open its details page.
4. Choose **Start experiment**.
5. (Optional) In the confirmation page, add tags for your experiment.

6. In the confirmation page, choose **Start experiment**.

Step 4: Track your AWS FIS experiment progress

You can track the progress of a running experiment until the experiment is completed, stopped, or has failed.

1. You should be on the details page for the experiment that you just started. If you're not, choose **Experiments**, and then select the ID of the experiment to open its details page.
2. To view the state of the experiment, check the **State** in the details pane. For more information, see Experiment states.
3. When the state of the experiment is **Running**, move to the next step.

Step 5: Verify Amazon S3 network disruption

You can validate the experiment progress by by pinging the Amazon S3 endpoint.

- From your Amazon EC2 instance, ping the Amazon S3 endpoint in your AWS Region. Replace *AWS Region* with your Region.

  ```
  ping -c 1 s3.AWS Region.amazonaws.com
  ```

  For the output, you should see an unsuccessful ping with 100% packet loss, as shown in the following example.

  ```
  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
  ```

Step 5: Clean up

If you no longer need the Amazon EC2 instance that you created for this experiment or the AWS FIS template, you can remove them.

To remove the Amazon EC2 instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Instances**.
3. Select the test instance, choose **Instance state**, and then choose **Terminate instance**.
4. When prompted for confirmation, choose **Terminate**.

To delete the experiment template using the AWS FIS console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. Select the experiment template, and then choose **Actions**, **Delete experiment template**.
4. When prompted for confirmation, enter `delete`, and then choose **Delete experiment template**.

Tutorial: Schedule a recurring experiment

With AWS Fault Injection Service (AWS FIS), you can perform fault injection experiments on your AWS workloads. These experiments run on templates that contain one or more actions to run on specified targets. When you also use Amazon EventBridge, you can schedule your experiments as a one-time task or recurring tasks.

Use this tutorial to create an EventBridge schedule that runs an AWS FIS experiment template every 5 minutes.

Tasks

Prerequisites

Before beginning this tutorial, must have an AWS FIS experiment template that you want to run on a schedule. If you already have a working experiment template, make note of the template ID and AWS Region. Otherwise, you can create a template by following the instructions in the section called "Test instance stop and start", and then return to this tutorial.

Step 1: Create an IAM role and policy

To create an IAM role and policy

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Roles**, and then **Create Role**.
3. Choose **Custom trust policy**, and then insert the following snippet to allow Amazon EventBridge Scheduler to assume the role on your behalf.

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

   Choose **Next**.
4. Under **Add permissions**, choose **Create policy**.
5. Choose **JSON**, and then insert the following policy. Replace the *your-experiment-template-id* value with the template ID of your experiment from the Prerequisites steps.

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

```
            "Effect": "Allow",
            "Action": "fis:StartExperiment",
            "Resource": [
                "arn:aws:fis:*:*:experiment-template/your-experiment-template-id",
                "arn:aws:fis:*:*:experiment/*"
            ]
        }
    ]
}
```

You can restrict the scheduler to only run AWS FIS experiments that have a specific tag value. For example, the following policy grants the `StartExperiment` permission for all AWS FIS experiment templates, but restricts the scheduler to only run experiments that are tagged `Purpose=Schedule`.

```
{
    "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"
                }
            }
        }
    ]
}
```

Choose **Next: Tags**.

6. Choose **Next: Review**.
7. Under **Review policy**, name your policy `FIS_RecurringExperiment`, and then choose **Create policy**.

8. Under **Add permissions**, add the new `FIS_RecurringExperiment` policy to your role, and then choose **Next**.
9. Under **Name, review, and create**, name the role `FIS_RecurringExperiment_role`, and then choose **Create role**.

Step 2: Create an Amazon EventBridge Scheduler

To create an Amazon EventBridge Scheduler

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the left navigation pane, choose **Schedules**.
3. Verify that you are in the same AWS Region as your AWS FIS experiment template.
4. Choose **Create schedule**, and fill in the following:
   - Under **Schedule name**, insert `FIS_recurring_experiment_tutorial`.
   - Under **Schedule pattern**, select **Recurring schedule**.
   - Under **Schedule type**, select **Rate-based schedule**.
   - Under **Rate expression**, choose **5 minutes**.
   - Under **Flexible time window**, select **Off**.
   - (Optional) Under **Timeframe**, select your time zone.
   - Choose **Next**.
5. Under **Select target**, choose **All APIs**, and then search for **AWS FIS**.
6. Choose **AWS FIS**, and then select **StartExperiment**.
7. Under **Input**, insert the following JSON payload. Replace the *`your-experiment-template-id`* value with the template ID of your experiment. The `ClientToken` is a unique identifier for the scheduler. In this tutorial, we are using a context keyword allowed by Amazon EventBridge Scheduler. For more information, see Adding context attributes in the *Amazon EventBridge User Guide*.

```
{
    "ClientToken": "<aws.scheduler.execution-id>",
    "ExperimentTemplateId": "your-experiment-template-id"
}
```

Choose **Next**.

8. (Optional) Under **Settings**, you can set the **Retry policy**, **Dead-letter queue (DLQ)**, and **Encryption** settings. Alternatively, you can keep the default values.
9. Under **Permissions**, select **Use existing role**, and then search for `FIS_RecurringExperiment_role`.
10. Choose **Next**.
11. Under **Review and create schedule**, review your scheduler details, and then choose **Create schedule**.

Step 3: Verify your experiment

To verify that your AWS FIS experiment ran on schedule

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the left navigation pane, choose **Experiments**.
3. Five minutes after you create your schedule, you should see your experiment running.

Step 4: Clean up

To disable your Amazon EventBridge Scheduler

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the left navigation pane, choose **Schedules**.
3. Select your newly created scheduler, and then choose **Disable**.

Actions for AWS FIS

An action is the fault injection activity that you run on a target using AWS Fault Injection Service (AWS FIS). AWS FIS provides preconfigured actions for specific types of targets across AWS services. You add actions to an experiment template, which you then use to run experiments.

Contents

Action identifiers

Each AWS FIS action has an identifier with the following format:

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

For example, the following action stops the target Amazon EC2 instances:

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

For a complete list of actions, see the AWS FIS actions reference. To get the list using the AWS CLI, see List the actions.

Action parameters

Some AWS FIS actions have additional parameters that are specific to the action. These parameters are used to pass information to AWS FIS when the action is run.

AWS FIS supports custom fault types using the `aws:ssm:send-command` action, which uses the SSM Agent and an SSM command document to create the fault condition on the targeted instances. The `aws:ssm:send-command` action includes a `documentArn` parameter that takes the Amazon Resource Name (ARN) of an SSM document as a value. You specify values for parameters when you add the action to your experiment template.

For more information about specifying parameters for the `aws:ssm:send-command` action, see Use the aws:ssm:send-command action.

Where possible, you can input a rollback configuration (also referred to as a *post action*) within the action parameters. A post action returns the target to the state that it was in before the action was run. The post action runs after the time specified in the action duration. Not all actions can support post actions. For example, if the action terminates an Amazon EC2 instance, you cannot recover the instance after it has been terminated.

Action targets

An action runs on the target resources that you specify. After you define a target, you can specify its name when you define an action.

```
"targets": {
    "resource_type": "resource_name"
}
```

AWS FIS actions support the following resource types for action targets:

- **Auto Scaling groups** – Amazon EC2 Auto Scaling groups
- **Buckets** – Amazon S3 buckets
- **Cluster** – Amazon EKS clusters
- **Clusters** – Amazon ECS clusters or Amazon Aurora DB clusters
- **DBInstances** – Amazon RDS DB instances
- **Encrypted global tables** – Amazon DynamoDB; global tables encrypted with a customer managed key
- **Global tables** – Amazon DynamoDB; global tables
- **Instances** – Amazon EC2 instances

- **Nodegroups** – Amazon EKS node groups
- **Pods** – Kubernetes pods on Amazon EKS
- **ReplicationGroups** – ElastiCache Redis Replication Groups
- **Roles** – IAM roles
- **SpotInstances** – Amazon EC2 Spot Instances
- **Subnets** – VPC subnets
- **Tasks** – Amazon ECS tasks
- **TransitGateways** – Transit gateways
- **Volumes** – Amazon EBS volumes

For examples, see the section called “Example actions”.

AWS FIS actions reference

This reference describes the common actions in AWS FIS, including information about the action parameters and the required IAM permissions. You can also list the supported AWS FIS actions using the AWS FIS console or the list-actions command from the AWS Command Line Interface (AWS CLI).

For more information, see Actions for AWS FIS and How AWS Fault Injection Service works with IAM.

Actions

- Fault injection actions
- Wait action
- Amazon CloudWatch actions
- Amazon DynamoDB actions
- Amazon EBS actions
- Amazon EC2 actions
- Amazon ECS actions
- Amazon EKS actions
- Amazon ElastiCache actions

- Network actions
- Amazon RDS actions
- Amazon S3 actions
- Systems Manager actions

Fault injection actions

AWS FIS supports the following fault injection actions.

Actions

- aws:fis:inject-api-internal-error
- aws:fis:inject-api-throttle-error
- aws:fis:inject-api-unavailable-error

aws:fis:inject-api-internal-error

Injects Internal Errors into requests made by the the target IAM role.

Resource type

- **aws:iam:role**

Parameters

- **duration** – The duration, from one minute to 12 hours. In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.
- **service** – The target AWS API namespace. The supported value is `ec2`.
- **percentage** – The percentage (1-100) of calls to inject the fault into.
- **operations** – The operations to inject the fault into, separated using commas. For a list of the API actions for the `ec2` namespace, see Actions in the *Amazon EC2 API Reference*.

Permissions

- `fis:InjectApiInternalError`

aws:fis:inject-api-throttle-error

Injects throttling errors into requests made by the target IAM role.

Resource type

- **aws:iam:role**

Parameters

- **duration** – The duration, from one minute to 12 hours. In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.
- **service** – The target AWS API namespace. The supported value is `ec2`.
- **percentage** – The percentage (1-100) of calls to inject the fault into.
- **operations** – The operations to inject the fault into, separated using commas. For a list of the API actions for the `ec2` namespace, see Actions in the *Amazon EC2 API Reference*.

Permissions

- `fis:InjectApiThrottleError`

aws:fis:inject-api-unavailable-error

Injects Unavailable errors into requests made by the target IAM role.

Resource type

- **aws:iam:role**

Parameters

- **duration** – The duration, from one minute to 12 hours. In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.
- **service** – The target AWS API namespace. The supported value is `ec2`.
- **percentage** – The percentage (1-100) of calls to inject the fault into.

- **operations** – The operations to inject the fault into, separated using commas. For a list of the API actions for the `ec2` namespace, see Actions in the *Amazon EC2 API Reference*.

Permissions

- `fis:InjectApiUnavailableError`

Wait action

AWS FIS supports the following wait action.

aws:fis:wait

Runs the AWS FIS wait action.

Parameters

- **duration** – The duration, from one minute to 12 hours. In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.

Permissions

- None

Amazon CloudWatch actions

AWS FIS supports the following Amazon CloudWatch action.

aws:cloudwatch:assert-alarm-state

Verifies that the specified alarms are in one of the specified alarm states.

Resource type

- None

Parameters

- **alarmArns** – The ARNs of the alarms, separated by commas. You can specify up to five alarms.
- **alarmStates** – The alarm states, separated by commas. The possible alarm states are `OK`, `ALARM`, and `INSUFFICIENT_DATA`.

Permissions

- `cloudwatch:DescribeAlarms`

Amazon DynamoDB actions

AWS FIS supports the following Amazon DynamoDB action.

aws:dynamodb:global-table-pause-replication

Pauses Amazon DynamoDB global table replication to any replica table. Tables may continue to be replicated for up to 5 minutes after action begins.

The following statement will be dynamically appended to the policy for the target DynamoDB global table:

```
{
   "Statement":[
      {
         "Sid": "DoNotModifyFisDynamoDbPauseReplicationEXPxxxxxxxxxxxxxx"
         "Effect":"Deny",
         "Principal":{
            "AWS":"arn:aws:iam::123456789012:role/aws-service-role/
replication.dynamodb.amazonaws.com/AWSServiceRoleForDynamoDBReplication"
         },
         "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"
         }
       }
      }
   ]
}
```

The following statement will be dynamically appended to the policy for stream for the target DynamoDB global table:

```
{
   "Statement":[
      {
         "Sid": "DoNotModifyFisDynamoDbPauseReplicationEXPxxxxxxxxxxxxxxx"
         "Effect":"Deny",
         "Principal":{
            "AWS":"arn:aws:iam::123456789012:role/aws-service-role/
replication.dynamodb.amazonaws.com/AWSServiceRoleForDynamoDBReplication"
         },
         "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"
         }
      }
   ]
}
```

If a target table or stream does not have any attached resource polices, a resource policy is created for the duration of the experiment, and automatically deleted when the experiment ends. Otherwise, the fault statement is inserted into an existing policy, without any additional

modifications to the existing policy statements. The fault statement is then removed from the policy at the end of the experiment.

Resource type

- **aws:dynamodb:global-table**

Parameters

- **duration** – In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.

Permissions

- `dynamodb:PutResourcePolicy`
- `dynamodb:DeleteResourcePolicy`
- `dynamodb:GetResourcePolicy`
- `dynamodb:DescribeTable`
- `tag:GetResources`

Amazon EBS actions

AWS FIS supports the following Amazon EBS action.

aws:ebs:pause-volume-io

Pauses I/O operations on target EBS volumes. The target volumes must be in the same Availability Zone and must be attached to instances built on the Nitro System. The volumes can't be attached to instances on an Outpost.

To initiate the experiment using the Amazon EC2 console, see Fault testing on Amazon EBS in the *Amazon EC2 User Guide*.

Resource type

- **aws:ec2:ebs-volume**

Parameters

- **duration** – The duration, from one second to 12 hours. In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute, PT5S represents five seconds, and PT6H represents six hours. In the AWS FIS console, you enter the number of seconds, minutes, or hours. If the duration is small, such as PT5S, the I/O is paused for the specified duration, but it might take longer for the experiment to complete due to the time it takes to initialize the experiment.

Permissions

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

Amazon EC2 actions

AWS FIS supports the following Amazon EC2 actions.

Actions

- 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 also supports fault injection actions through the AWS Systems Manager SSM Agent. Systems Manager uses an SSM document that defines actions to perform on EC2 instances. You can use your own document to inject custom faults, or you can use pre-configured SSM documents. For more information, see the section called “Use SSM documents”.

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

Injects `InsufficientInstanceCapacity` error responses on requests made by the target IAM roles. Supported operations are RunInstances, CreateCapacityReservation, StartInstances,

CreateFleet calls. Requests that include capacity asks in multiple Availability Zones are not supported. This action doesn't support defining targets using resource tags, filters, or parameters.

Resource type

- **aws:iam:role**

Parameters

- **duration** – In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.
- **availabilityzoneIdentifiers** – The comma separated list of Availability Zones. Supports Zone IDs (e.g. `"use1-az1, use1-az2"`) and Zone names (e.g. `"us-east-1a"`).
- **percentage** – The percentage (1-100) of calls to inject the fault into.

Permissions

- `ec2:InjectApiError`with condition key `ec2:FisActionId` value set to `aws:ec2:api-insufficient-instance-capacity-error` and `ec2:FisTargetArns` condition key set to target IAM roles.

For an example policy, see Example: Use condition keys for `ec2:InjectApiError`.

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

Injects `InsufficientInstanceCapacity` error responses on requests made by the target Auto Scaling groups. This action only supports Auto Scaling groups using launch templates. To learn more about insufficient instance capacity errors, see the Amazon EC2 user guide.

Resource type

- **aws:ec2:autoscaling-group**

Parameters

- **duration** – In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.
- **availabilityzoneIdentifiers** – The comma separated list of Availability Zones. Supports Zone IDs (e.g. "`use1-az1, use1-az2`") and Zone names (e.g. "`us-east-1a`").
- **percentage** – Optional. The percentage (1-100) of the target Auto Scaling group's launch requests to inject the fault. The default is 100.

Permissions

- `ec2:InjectApiError`with condition key ec2:FisActionId value set to `aws:ec2:asg-insufficient-instance-capacity-error` and `ec2:FisTargetArns` condition key set to target Auto Scaling groups.
- `autoscaling:DescribeAutoScalingGroups`

For an example policy, see Example: Use condition keys for `ec2:InjectApiError`.

aws:ec2:reboot-instances

Runs the Amazon EC2 API action RebootInstances on the target EC2 instances.

Resource type

- **aws:ec2:instance**

Parameters

- None

Permissions

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

AWS managed policy

- AWSFaultInjectionSimulatorEC2Access

aws:ec2:send-spot-instance-interruptions

Interrupts the target Spot Instances. Sends a Spot Instance interruption notice to target Spot Instances two minutes before interrupting them. The interruption time is determined by the specified **durationBeforeInterruption** parameter. Two minutes after the interruption time, the Spot Instances are terminated or stopped, depending on their interruption behavior. A Spot Instance that was stopped by AWS FIS remains stopped until you restart it.

Immediately after the action is initiated, the target instance receives an EC2 instance rebalance recommendation. If you specified **durationBeforeInterruption**, there could be a delay between the rebalance recommendation and the interruption notice.

For more information, see the section called "Test Spot Instance interruptions". Alternatively, to initiate the experiment by using the Amazon EC2 console, see Initiate a Spot Instance interruption in the *Amazon EC2 User Guide*.

Resource type

- **aws:ec2:spot-instance**

Parameters

- **durationBeforeInterruption** – The time to wait before interrupting the instance, from 2 to 15 minutes. In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT2M represents two minutes. In the AWS FIS console, you enter the number of minutes.

Permissions

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

AWS managed policy

- AWSFaultInjectionSimulatorEC2Access

aws:ec2:stop-instances

Runs the Amazon EC2 API action StopInstances on the target EC2 instances.

Resource type

- **aws:ec2:instance**

Parameters

- **startInstancesAfterDuration** – Optional. The time to wait before starting the instance, from one minute to 12 hours. In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours. If the instance has an encrypted EBS volume, you must grant AWS FIS permission to the KMS key used to encrypt the volume, or add the experiment role to the KMS key policy.
- **completeIfInstancesTerminated** – Optional. If true, and if `startInstancesAfterDuration` is also true, this action will not fail when targeted EC2 instances have been terminated by a separate request outside of FIS and cannot be restarted. For example, Auto Scaling groups may terminate stopped EC2 instances under their control before this action completes. The default is false.

Permissions

- `ec2:StopInstances`
- `ec2:StartInstances`
- `ec2:DescribeInstances` – Optional. Required with **completeIfInstancesTerminated** to validate instance state at end of action.
- `kms:CreateGrant` – Optional. Required with **startInstancesAfterDuration** to restart instances with encrypted volumes.

AWS managed policy

- AWSFaultInjectionSimulatorEC2Access

aws:ec2:terminate-instances

Runs the Amazon EC2 API action TerminateInstances on the target EC2 instances.

Resource type

- **aws:ec2:instance**

Parameters

- None

Permissions

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

AWS managed policy

- AWSFaultInjectionSimulatorEC2Access

Amazon ECS actions

AWS FIS supports the following Amazon ECS actions.

Actions

- 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

Runs the Amazon ECS API action UpdateContainerInstancesState to drain the specified percentage of underlying Amazon EC2 instances on the target clusters.

Resource type

- **aws:ecs:cluster**

Parameters

- **drainagePercentage** – The percentage (1-100).
- **duration** – The duration, from one minute to 12 hours. In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.

Permissions

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

AWS managed policy

- AWSFaultInjectionSimulatorECSAccess

aws:ecs:stop-task

Runs the Amazon ECS API action StopTask to stop the target task.

Resource type

- **aws:ecs:task**

Parameters

- None

Permissions

- `ecs:DescribeTasks`

- `ecs:ListTasks`
- `ecs:StopTask`
- `tag:GetResources`

AWS managed policy

- AWSFaultInjectionSimulatorECSAccess

aws:ecs:task-cpu-stress

Runs CPU stress on the target tasks. Uses the AWSFIS-Run-CPU-Stress SSM document. The tasks must be managed by AWS Systems Manager. For more information, see Use the ECS task actions.

Resource type

- **aws:ecs:task**

Parameters

- **duration** – The duration of the stress test, in ISO 8601 format.
- **percent** – Optional. The target load percentage, from 0 (no load) to 100 (full load). The default is 100.
- **workers** – Optional. The number of stressors to use. The default is 0, which uses all stressors.
- **installDependencies** – Optional. If this value is `True`, Systems Manager installs the required dependencies on the sidecar container for the SSM agent, if they are not already installed. The default is `True`. The dependency is **stress-ng**.

Permissions

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

aws:ecs:task-io-stress

Runs I/O stress on the target tasks. Uses the AWSFIS-Run-IO-Stress SSM document. The tasks must be managed by AWS Systems Manager. For more information, see Use the ECS task actions.

Resource type

- **aws:ecs:task**

Parameters

- **duration** – The duration of the stress test, in ISO 8601 format.
- **percent** – Optional. The percentage of free space on the file system to use during the stress test. The default is 80%.
- **workers** – Optional. The number of workers. Workers perform a mix of sequential, random, and memory-mapped read/write operations, forced synchronizing, and cache dropping. Multiple child processes perform different I/O operations on the same file. The default is 1.
- **installDependencies** – Optional. If this value is `True`, Systems Manager installs the required dependencies on the sidecar container for the SSM agent, if they are not already installed. The default is `True`. The dependency is **stress-ng**.

Permissions

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

aws:ecs:task-kill-process

Stops the specified process in the tasks, using the **killall** command. Uses the AWSFIS-Run-Kill-Process SSM document. The task definition must have `pidMode` set to `task`. The tasks must be managed by AWS Systems Manager. For more information, see Use the ECS task actions.

Resource type

- **aws:ecs:task**

Parameters

- **processName** – The name of the process to stop.
- **signal** – Optional. The signal to send along with the command. The possible values are `SIGTERM` (which the receiver can choose to ignore) and `SIGKILL` (which cannot be ignored). The default is `SIGTERM`.
- **installDependencies** – Optional. If this value is `True`, Systems Manager installs the required dependencies on the sidecar container for the SSM agent, if they are not already installed. The default is `True`. The dependency is **killall**.

Permissions

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

aws:ecs:task-network-blackhole-port

Drops inbound or outbound traffic for the specified protocol and port. Uses the AWSFIS-Run-Network-Blackhole-Port SSM document. The task definition must have `pidMode` set to `task`. The tasks must be managed by AWS Systems Manager. You can't set `networkMode` to `bridge` in the task definition. For more information, see Use the ECS task actions.

Resource type

- **aws:ecs:task**

Parameters

- **duration** – The duration of the test, in ISO 8601 format.
- **port** – The port number.
- **trafficType** – The type of traffic. The possible values are `ingress` and `egress`.
- **protocol** – Optional. The protocol. The possible values are `tcp` and udp. The default is `tcp`.
- **installDependencies** – Optional. If this value is `True`, Systems Manager installs the required dependencies on the sidecar container for the SSM agent, if they are not already installed. The default is `True`. The dependencies are **atd**, **dig**, and **iptables**.

Permissions

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

aws:ecs:task-network-latency

Adds latency and jitter to the network interface using the **tc** tool for traffic to or from specific sources. Uses the AWSFIS-Run-Network-Latency-Sources SSM document. The task definition must have `pidMode` set to `task`. The tasks must be managed by AWS Systems Manager. You can't set `networkMode` to `bridge` in the task definition. For more information, see Use the ECS task actions.

Resource type

- **aws:ecs:task**

Parameters

- **duration** – The duration of the test, in ISO 8601 format.
- **interface** – Optional. The network interface. The default is `eth0`.
- **delayMilliseconds** – Optional. The delay, in milliseconds. The default is 200.
- **jitterMilliseconds** – Optional. The jitter, in milliseconds. The default is 10.
- **sources** – Optional. The sources, separated by commas. The possible values are: an IPv4 address, an IPv4 CIDR block, a domain name, DYNAMODB, and S3. If you specify DYNAMODB or S3, this applies only to the Regional endpoint in the current Region. The default is 0.0.0.0/0, which matches all IPv4 traffic.
- **installDependencies** – Optional. If this value is `True`, Systems Manager installs the required dependencies on the sidecar container for the SSM agent, if they are not already installed. The default is `True`. The dependencies are **atd**, **dig**, **jq**, and **tc**.

Permissions

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

- `ssm:CancelCommand`

aws:ecs:task-network-packet-loss

Adds packet loss to the network interface using the **tc** tool. Uses the AWSFIS-Run-Network-Packet-Loss-Sources SSM document. The task definition must have `pidMode` set to `task`. The tasks must be managed by AWS Systems Manager. You can't set `networkMode` to `bridge` in the task definition. For more information, see Use the ECS task actions.

Resource type

- **aws:ecs:task**

Parameters

- **duration** – The duration of the test, in ISO 8601 format.
- **interface** – Optional. The network interface. The default is `eth0`.
- **lossPercent** – Optional. The percentage of packet loss. The default is 7%.
- **sources** – Optional. The sources, separated by commas. The possible values are: an IPv4 address, an IPv4 CIDR block, a domain name, `DYNAMODB`, and `S3`. If you specify `DYNAMODB` or `S3`, this applies only to the Regional endpoint in the current Region. The default is 0.0.0.0/0, which matches all IPv4 traffic.
- **installDependencies** – Optional. If this value is `True`, Systems Manager installs the required dependencies on the sidecar container for the SSM agent, if they are not already installed. The default is `True`. The dependencies are **atd**, **dig**, **jq**, and **tc**.

Permissions

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

Amazon EKS actions

AWS FIS supports the following Amazon EKS actions.

Actions

- 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

Runs a ChaosMesh or Litmus experiment on a single target cluster. You must install ChaosMesh or Litmus on the target cluster.

When you create an experiment template and define a target of type `aws:eks:cluster`, you must target this action to a single Amazon Resource Name (ARN). This action doesn't support defining targets using resource tags, filters, or parameters.

When you install ChaosMesh, you must specify the appropriate container runtime. Starting with Amazon EKS version 1.23, the default runtime changed from Docker to **containerd**. Starting with version 1.24, Docker was removed.

Resource type

- **aws:eks:cluster**

Parameters

- **kubernetesApiVersion** – The API version of the Kubernetes custom resource. The possible values are `chaos-mesh.org/v1alpha1` | `litmuschaos.io/v1alpha1`.
- **kubernetesKind** – The Kubernetes custom resource kind. The value depends on the API version.
  - `chaos-mesh.org/v1alpha1` – The possible values are `AWSChaos` | `DNSChaos` | `GCPChaos` | `HTTPChaos` | `IOChaos` | `JVMChaos` | `KernelChaos` | `NetworkChaos` |

`PhysicalMachineChaos` | `PodChaos` | `PodHttpChaos` | `PodIOChaos` | `PodNetworkChaos` | `Schedule` | `StressChaos` | `TimeChaos` |

- `litmuschaos.io/v1alpha1` – The possible value is `ChaosEngine`.

- **kubernetesNamespace** – The Kubernetes namespace.
- **kubernetesSpec** – The `spec` section of the Kubernetes custom resource, in JSON format.
- **maxDuration** – The maximum time allowed for the automation execution to complete, from one minute to 12 hours. In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.

Permissions

No AWS Identity and Access Management (IAM) permissions are required for this action. The permissions required to use this action are controlled by Kubernetes using RBAC authorization. For more information, see Using RBAC Authorization in the official Kubernetes documentation. For more information about Chaos Mesh, see the official Chaos Mesh documentation. For more information about Litmus, see the official Litmus documentation.

aws:eks:pod-cpu-stress

Runs CPU stress on the target pods. For more information, see Use the EKS pod actions.

Resource type

- **aws:eks:pod**

Parameters

- **duration** – The duration of the stress test, in ISO 8601 format.
- **percent** – Optional. The target load percentage, from 0 (no load) to 100 (full load). The default is 100.
- **workers** – Optional. The number of stressors to use. The default is 0, which uses all stressors.
- **kubernetesServiceAccount** – The Kubernetes service account. For information about the required permissions, see the section called “Configure the Kubernetes service account”.
- **fisPodContainerImage** – Optional. The container image used to create the fault injector pod. The default is to use the images provided by AWS FIS. For more information, see the section called “Pod container images”.

- **maxErrorsPercent** – Optional. The percentage of targets that can fail before the fault injection fails. The default is 0.
- **fisPodLabels** – Optional. The Kubernetes labels that are attached to the fault orchestration pod created by FIS.
- **fisPodAnnotations** – Optional. The Kubernetes annotations that are attached to the fault orchestration pod created by FIS.
- **fisPodSecurityPolicy** – Optional. The Kubernetes Security Standards policy to use for the fault orchestration pod created by FIS and the ephemeral containers. Possible values are `privileged`, `baseline` and `restricted`. This action is compatible with all policy levels.

Permissions

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

AWS managed policy

- AWSFaultInjectionSimulatorEKSAccess

aws:eks:pod-delete

Deletes the target pods. For more information, see Use the EKS pod actions.

Resource type

- **aws:eks:pod**

Parameters

- **gracePeriodSeconds** – Optional. The duration, in seconds, to wait for the pod to terminate gracefully. If the value is 0, we perform the action immediately. If the value is nil, we use the default grace period for the pod.
- **kubernetesServiceAccount** – The Kubernetes service account. For information about the required permissions, see the section called “Configure the Kubernetes service account”.

- **fisPodContainerImage** – Optional. The container image used to create the fault injector pod. The default is to use the images provided by AWS FIS. For more information, see the section called "Pod container images".
- **maxErrorsPercent** – Optional. The percentage of targets that can fail before the fault injection fails. The default is 0.
- **fisPodLabels** – Optional. The Kubernetes labels that are attached to the fault orchestration pod created by FIS.
- **fisPodAnnotations** – Optional. The Kubernetes annotations that are attached to the fault orchestration pod created by FIS.
- **fisPodSecurityPolicy** – Optional. The Kubernetes Security Standards policy to use for the fault orchestration pod created by FIS and the ephemeral containers. Possible values are `privileged`, `baseline` and `restricted`. This action is compatible with all policy levels.

Permissions

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

AWS managed policy

- AWSFaultInjectionSimulatorEKSAccess

aws:eks:pod-io-stress

Runs I/O stress on the target pods. For more information, see Use the EKS pod actions.

Resource type

- **aws:eks:pod**

Parameters

- **duration** – The duration of the stress test, in ISO 8601 format.

- **workers** – Optional. The number of workers. Workers perform a mix of sequential, random, and memory-mapped read/write operations, forced synchronizing, and cache dropping. Multiple child processes perform different I/O operations on the same file. The default is 1.
- **percent** – Optional. The percentage of free space on the file system to use during the stress test. The default is 80%.
- **kubernetesServiceAccount** – The Kubernetes service account. For information about the required permissions, see the section called "Configure the Kubernetes service account".
- **fisPodContainerImage** – Optional. The container image used to create the fault injector pod. The default is to use the images provided by AWS FIS. For more information, see the section called "Pod container images".
- **maxErrorsPercent** – Optional. The percentage of targets that can fail before the fault injection fails. The default is 0.
- **fisPodLabels** – Optional. The Kubernetes labels that are attached to the fault orchestration pod created by FIS.
- **fisPodAnnotations** – Optional. The Kubernetes annotations that are attached to the fault orchestration pod created by FIS.
- **fisPodSecurityPolicy** – Optional. The Kubernetes Security Standards policy to use for the fault orchestration pod created by FIS and the ephemeral containers. Possible values are `privileged`, `baseline` and `restricted`. This action is compatible with all policy levels.

Permissions

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

AWS managed policy

- AWSFaultInjectionSimulatorEKSAccess

aws:eks:pod-memory-stress

Runs memory stress on the target pods. For more information, see Use the EKS pod actions.

Resource type

- **aws:eks:pod**

Parameters

- **duration** – The duration of the stress test, in ISO 8601 format.
- **workers** – Optional. The number of stressors to use. The default is 1.
- **percent** – Optional. The percentage of virtual memory to use during the stress test. The default is 80%.
- **kubernetesServiceAccount** – The Kubernetes service account. For information about the required permissions, see the section called "Configure the Kubernetes service account".
- **fisPodContainerImage** – Optional. The container image used to create the fault injector pod. The default is to use the images provided by AWS FIS. For more information, see the section called "Pod container images".
- **maxErrorsPercent** – Optional. The percentage of targets that can fail before the fault injection fails. The default is 0.
- **fisPodLabels** – Optional. The Kubernetes labels that are attached to the fault orchestration pod created by FIS.
- **fisPodAnnotations** – Optional. The Kubernetes annotations that are attached to the fault orchestration pod created by FIS.
- **fisPodSecurityPolicy** – Optional. The Kubernetes Security Standards policy to use for the fault orchestration pod created by FIS and the ephemeral containers. Possible values are `privileged`, `baseline` and `restricted`. This action is compatible with all policy levels.

Permissions

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

AWS managed policy

- AWSFaultInjectionSimulatorEKSAccess

aws:eks:pod-network-blackhole-port

Drops inbound or outbound traffic for the specified protocol and port. Only compatible with the Kubernetes Security Standards `privileged`policy. For more information, see Use the EKS pod actions.

Resource type

- **aws:eks:pod**

Parameters

- **duration** – The duration of the test, in ISO 8601 format.
- **protocol** – Optional. The protocol. The possible values are `tcp` and udp. The default is `tcp`.
- **trafficType** – The type of traffic. The possible values are `ingress` and `egress`.
- **port** – The port number.
- **kubernetesServiceAccount** – The Kubernetes service account. For information about the required permissions, see the section called "Configure the Kubernetes service account".
- **fisPodContainerImage** – Optional. The container image used to create the fault injector pod. The default is to use the images provided by AWS FIS. For more information, see the section called "Pod container images".
- **maxErrorsPercent** – Optional. The percentage of targets that can fail before the fault injection fails. The default is 0.
- **fisPodLabels** – Optional. The Kubernetes labels that are attached to the fault orchestration pod created by FIS.
- **fisPodAnnotations** – Optional. The Kubernetes annotations that are attached to the fault orchestration pod created by FIS.

Permissions

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

AWS managed policy

- AWSFaultInjectionSimulatorEKSAccess

aws:eks:pod-network-latency

Adds latency and jitter to the network interface using the **tc** tool for traffic to or from specific sources. Only compatible with the Kubernetes Security Standards `privileged`policy. For more information, see Use the EKS pod actions.

Resource type

- **aws:eks:pod**

Parameters

- **duration** – The duration of the test, in ISO 8601 format.
- **interface** – Optional. The network interface. The default is `eth0`.
- **delayMilliseconds** – Optional. The delay, in milliseconds. The default is 200.
- **jitterMilliseconds** – Optional. The jitter, in milliseconds. The default is 10.
- **sources** – Optional. The sources, separated by commas. The possible values are: an IPv4 address, an IPv4 CIDR block, a domain name, DYNAMODB, and S3. If you specify DYNAMODB or S3, this applies only to the Regional endpoint in the current Region. The default is 0.0.0.0/0, which matches all IPv4 traffic.
- **kubernetesServiceAccount** – The Kubernetes service account. For information about the required permissions, see the section called “Configure the Kubernetes service account”.
- **fisPodContainerImage** – Optional. The container image used to create the fault injector pod. The default is to use the images provided by AWS FIS. For more information, see the section called “Pod container images”.
- **maxErrorsPercent** – Optional. The percentage of targets that can fail before the fault injection fails. The default is 0.
- **fisPodLabels** – Optional. The Kubernetes labels that are attached to the fault orchestration pod created by FIS.
- **fisPodAnnotations** – Optional. The Kubernetes annotations that are attached to the fault orchestration pod created by FIS.

Permissions

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

AWS managed policy

- AWSFaultInjectionSimulatorEKSAccess

aws:eks:pod-network-packet-loss

Adds packet loss to the network interface using the **tc** tool. Only compatible with the Kubernetes Security Standards `privileged`policy. For more information, see Use the EKS pod actions.

Resource type

- **aws:eks:pod**

Parameters

- **duration** – The duration of the test, in ISO 8601 format.
- **interface** – Optional. The network interface. The default is `eth0`.
- **lossPercent** – Optional. The percentage of packet loss. The default is 7%.
- **sources** – Optional. The sources, separated by commas. The possible values are: an IPv4 address, an IPv4 CIDR block, a domain name, DYNAMODB, and S3. If you specify DYNAMODB or S3, this applies only to the Regional endpoint in the current Region. The default is 0.0.0.0/0, which matches all IPv4 traffic.
- **kubernetesServiceAccount** – The Kubernetes service account. For information about the required permissions, see the section called "Configure the Kubernetes service account".
- **fisPodContainerImage** – Optional. The container image used to create the fault injector pod. The default is to use the images provided by AWS FIS. For more information, see the section called "Pod container images".
- **maxErrorsPercent** – Optional. The percentage of targets that can fail before the fault injection fails. The default is 0.

- **fisPodLabels** – Optional. The Kubernetes labels that are attached to the fault orchestration pod created by FIS.
- **fisPodAnnotations** – Optional. The Kubernetes annotations that are attached to the fault orchestration pod created by FIS.

Permissions

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

AWS managed policy

- AWSFaultInjectionSimulatorEKSAccess

aws:eks:terminate-nodegroup-instances

Runs the Amazon EC2 API action TerminateInstances on the target node group.

Resource type

- **aws:eks:nodegroup**

Parameters

- **instanceTerminationPercentage** – The percentage (1-100) of instances to terminate.

Permissions

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

AWS managed policy

- AWSFaultInjectionSimulatorEKSAccess

Amazon ElastiCache actions

AWS FIS supports the following ElastiCache action.

aws:elasticache:interrupt-cluster-az-power

Interrupts power to nodes in the specified Availability Zone for target Redis Replication Groups. When a primary node is targeted, the corresponding read replica with the least replication lag is promoted to primary. Read replica replacements in the specified Availability Zone are blocked for the duration of this action, which means that target Replication Groups operate with reduced capacity.

Resource type

- **aws:elasticache:redis-replicationgroup**

Parameters

- **duration** – The duration, from one minute to 12 hours. In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.

Permissions

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

Network actions

AWS FIS supports the following network actions.

Actions

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

aws:network:disrupt-connectivity

Denies the specified traffic to the target subnets. Uses network ACLs.

Resource type

- **aws:ec2:subnet**

Parameters

- **scope** – The type of traffic to deny. When the scope is not `all`, the maximum number of entries in network ACLs is 20. The possible values are:
  - `all` – Denies all traffic entering and leaving the subnet. Note that this option allows intra-subnet traffic, including traffic to and from network interfaces in the subnet.
  - `availability-zone` – Denies intra-VPC traffic to and from subnets in other Availability Zones. The maximum number of subnets that can be targeted in a VPC is 30.
  - `dynamodb` – Denies traffic to and from the Regional endpoint for DynamoDB in the current Region.
  - `prefix-list` – Denies traffic to and from the specified prefix list.
  - `s3` – Denies traffic to and from the Regional endpoint for Amazon S3 in the current Region.
  - `vpc` – Denies traffic entering and leaving the VPC.
- **duration** – The duration, from one minute to 12 hours. In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.
- **prefixListIdentifier** – If the scope is `prefix-list`, this is the identifier of the customer managed prefix list. You can specify a name, an ID, or an ARN. The prefix list can have at most 10 entries.

Permissions

- `ec2:CreateNetworkAcl` – Creates the network ACL with the tag managedByFIS=true.

- `ec2:CreateNetworkAclEntry` – The network ACL must have the tag managedByFIS=true.
- `ec2:CreateTags`
- `ec2:DeleteNetworkAcl` – The network ACL must have the tag managedByFIS=true.
- `ec2:DescribeManagedPrefixLists`
- `ec2:DescribeNetworkAcls`
- `ec2:DescribeSubnets`
- `ec2:DescribeVpcs`
- `ec2:GetManagedPrefixListEntries`
- `ec2:ReplaceNetworkAclAssociation`

AWS managed policy

- AWSFaultInjectionSimulatorNetworkAccess

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

Blocks traffic that originates in the target subnets and is destined for the specified Region. Creates route tables that include all routes for the Region to isolate. To allow FIS to create these route tables, raise the Amazon VPC quota for `routes per route table` to 250 plus the number of routes in your existing route tables.

Resource type

- **aws:ec2:subnet**

Parameters

- `region` – The code of the Region to isolate (for example, eu-west-1).
- `duration` – The length of time the action lasts. In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.

Permissions

- `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

† Scoped using the tag managedByFIS=true.

AWS managed policy

- AWSFaultInjectionSimulatorNetworkAccess

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

Blocks traffic from the target transit gateway peering attachments that is destined for the specified Region.

Resource type

- **aws:ec2:transit-gateway**

Parameters

- `region` – The code of the Region to isolate (for example, eu-west-1).
- `duration` – The length of time the action lasts. In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.

Permissions

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

AWS managed policy

- AWSFaultInjectionSimulatorNetworkAccess

Amazon RDS actions

AWS FIS supports the following Amazon RDS actions.

Actions

- aws:rds:failover-db-cluster
- aws:rds:reboot-db-instances

aws:rds:failover-db-cluster

Runs the Amazon RDS API action FailoverDBCluster on the target Aurora DB cluster.

Resource type

- **aws:rds:cluster**

Parameters

- None

Permissions

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

AWS managed policy

- AWSFaultInjectionSimulatorRDSAccess

aws:rds:reboot-db-instances

Runs the Amazon RDS API action RebootDBInstance on the target DB instance.

Resource type

- **aws:rds:db**

Parameters

- **forceFailover** – Optional. If the value is true, and if instances are Multi-AZ, forces failover from one Availability Zone to another. The default is false.

Permissions

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

AWS managed policy

- AWSFaultInjectionSimulatorRDSAccess

Amazon S3 actions

AWS FIS supports the following Amazon S3 action.

Actions

- aws:s3:bucket-pause-replication

aws:s3:bucket-pause-replication

Pauses replication from target source buckets to destination buckets. Destination buckets can be in different AWS Regions or within the same Region as the source bucket. Existing objects may continue to be replicated for up to one hour after action begins. This action only supports targeting by tags. To learn more about Amazon S3 Replication, see the Amazon S3 user guide.

Resource type

- **aws:s3:bucket**

Parameters

- **duration** – The duration, from one minute to 12 hours. In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.
- **region** – The AWS region where destination buckets are located.
- **destinationBuckets** – Optional. Comma separated list of destination S3 bucket(s).
- **prefixes** – Optional. Comma separated list of S3 object key prefixes from replication rule filters. Replication rules of target buckets with a filter based on the prefix(es) will be paused.

Permissions

- `S3:PutReplicationConfiguration` with condition key `S3:IsReplicationPauseRequest` set to `True`
- `S3:GetReplicationConfiguration` with condition key `S3:IsReplicationPauseRequest` set to `True`
- `S3:PauseReplication`
- `S3:ListAllMyBuckets`

- `tag:GetResources`

For an example policy, see Example: Use condition keys for `aws:s3:bucket-pause-replication`.

Systems Manager actions

AWS FIS supports the following Systems Manager actions.

Actions

- aws:ssm:send-command
- aws:ssm:start-automation-execution

aws:ssm:send-command

Runs the Systems Manager API action SendCommand on the target EC2 instances. The Systems Manager document (SSM document) defines the actions that Systems Manager performs on your instances. For more information, see Use the aws:ssm:send-command action.

Resource type

- **aws:ec2:instance**

Parameters

- **documentArn** – The Amazon Resource Name (ARN) of the document. In the console, this parameter is completed for you if you choose a value from **Action type** that corresponds to one of the pre-configured AWS FIS SSM documents.
- **documentVersion** – Optional. The version of the document. If empty, the default version runs.
- **documentParameters** – Conditional. The required and optional parameters that the document accepts. The format is a JSON object with keys that are strings and values that are either strings or arrays of strings.
- **duration** – The duration, from one minute to 12 hours. In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.

Permissions

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

AWS managed policy

- AWSFaultInjectionSimulatorEC2Access

aws:ssm:start-automation-execution

Runs the Systems Manager API action StartAutomationExecution.

Resource type

- None

Parameters

- **documentArn** – The Amazon Resource Name (ARN) of the automation document.
- **documentVersion** – Optional. The version of the document. If empty, the default version runs.
- **documentParameters** – Conditional. The required and optional parameters that the document accepts. The format is a JSON object with keys that are strings and values that are either strings or arrays of strings.
- **maxDuration** – The maximum time allowed for the automation execution to complete, from one minute to 12 hours. In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.

Permissions

- `ssm:GetAutomationExecution`
- `ssm:StartAutomationExecution`
- `ssm:StopAutomationExecution`
- `iam:PassRole` – Optional. Required if the automation document assumes a role.

AWS managed policy

- AWSFaultInjectionSimulatorSSMAccess

Use Systems Manager SSM documents with AWS FIS

AWS FIS supports custom fault types through the AWS Systems Manager SSM Agent and the AWS FIS action aws:ssm:send-command. Pre-configured Systems Manager SSM documents (SSM documents) that can be used to create common fault injection actions are available as public AWS documents that begin with the AWSFIS- prefix.

SSM Agent is Amazon software that can be installed and configured on Amazon EC2 instances, on-premises servers, or virtual machines (VMs). This makes it possible for Systems Manager to manage these resources. The agent processes requests from Systems Manager, and then runs them as specified in the request. You can include your own SSM document to inject custom faults, or reference one of the public Amazon-owned documents.

Requirements

For actions that require SSM Agent to run the action on the target, you must ensure the following:

- The agent is installed on the target. SSM Agent is installed by default on some Amazon Machine Images (AMIs). Otherwise, you can install the SSM Agent on your instances. For more information, see Manually install SSM Agent for EC2 instances in the *AWS Systems Manager User Guide*.
- Systems Manager has permission to perform actions on your instances. You grant access using an IAM instance profile. For more information, see Create an IAM instance profile for Systems Manager and Attach an IAM instance profile to an EC2 instance in the *AWS Systems Manager User Guide*.

Use the aws:ssm:send-command action

An SSM document defines the actions that Systems Manager performs on your managed instances. Systems Manager includes a number of pre-configured documents, or you can create your own. For more information about creating your own SSM document, see Creating Systems Manager documents in the *AWS Systems Manager User Guide*. For more information about SSM documents in general, see AWS Systems Manager documents in the *AWS Systems Manager User Guide*.

AWS FIS provides pre-configured SSM documents. You can view the pre-configured SSM documents under **Documents** in the AWS Systems Manager console: https://console.aws.amazon.com/systems-manager/documents. You can also choose from a selection of pre-configured documents in the AWS FIS console. For more information, see Pre-configured AWS FIS SSM documents.

To use an SSM document in your AWS FIS experiments, you can use the aws:ssm:send-command action. This action fetches and runs the specified SSM document on your target instances.

When you use the `aws:ssm:send-command` action in your experiment template, you must specify additional parameters for the action, including the following:

- **documentArn** – Required. The Amazon Resource Name (ARN) of the SSM document.
- **documentParameters** – Conditional. The required and optional parameters that the SSM document accepts. The format is a JSON object with keys that are strings and values that are either strings or arrays of strings.
- **documentVersion** – Optional. The version of the SSM document to run.

You can view the information for an SSM document (including the parameters for the document) by using the Systems Manager console or the command line.

To view information about an SSM document using the console

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Documents**.
3. Select the document, and choose the **Details** tab.

To view information about an SSM document using the command line

Use the SSM describe-document command.

Pre-configured AWS FIS SSM documents

You can use pre-configured AWS FIS SSM documents with the `aws:ssm:send-command` action in your experiment templates.

Requirements

- The pre-configured SSM documents provided by AWS FIS are supported only on the following operating systems:
  - Amazon Linux 2023, Amazon Linux 2, Amazon Linux
  - Ubuntu
  - RHEL 7, 8, 9
  - CentOS 7, 8, 9
- The pre-configured SSM documents provided by AWS FIS are supported only on EC2 instances. They are not supported on other types of managed nodes, such as on-premises servers.

To use these SSM documents in experiments on ECS tasks, use the corresponding the section called "Amazon ECS actions". For example, the **aws:ecs:task-cpu-stress** action uses the AWSFIS-Run-CPU-Stress document.

Documents

- 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

AWSFIS-Run-CPU-Stress

Runs CPU stress on an instance using the **stress-ng** tool. Uses the AWSFIS-Run-CPU-Stress SSM document.

Action type (console only)

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

ARN

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

Document parameters

- **DurationSeconds** – Required. The duration of the CPU stress test, in seconds.
- **CPU** – Optional. The number of CPU stressors to use. The default is 0, which uses all CPU stressors.
- **LoadPercent** – Optional. The target CPU load percentage, from 0 (no load) to 100 (full load). The default is 100.
- **InstallDependencies** – Optional. If the value is `True`, Systems Manager installs the required dependencies on the target instances if they are not already installed. The default is `True`. The dependency is **stress-ng**.

The following is an example of the string you can enter in the console.

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

AWSFIS-Run-Disk-Fill

Allocates disk space on the root volume of an instance to simulate a disk full fault. Uses the AWSFIS-Run-Disk-Fill SSM document.

If the experiment injecting this fault is stopped, either manually or through a stop condition, AWS FIS attempts to roll back by canceling the running SSM document. However, if the disk is 100% full, either due to the fault or the fault plus application activity, Systems Manager might be unable to complete the cancel operation. Therefore, if you might need to stop the experiment, ensure that the disk will not become 100% full.

Action type (console only)

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

ARN

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

Document parameters

- **DurationSeconds** – Required. The duration of the disk fill test, in seconds.
- **Percent** – Optional. The percentage of the disk to allocate during the disk fill test. The default is 95%.
- **InstallDependencies** – Optional. If the value is `True`, Systems Manager installs the required dependencies on the target instances if they are not already installed. The default is `True`. The dependencies are **atd** and **fallocate**.

The following is an example of the string you can enter in the console.

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

AWSFIS-Run-IO-Stress

Runs IO stress on an instance using the **stress-ng** tool. Uses the AWSFIS-Run-IO-Stress SSM document.

Action type (console only)

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

ARN

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

Document parameters

- **DurationSeconds** – Required. The duration of the IO stress test, in seconds.
- **Workers** – Optional. The number of workers that perform a mix of sequential, random, and memory-mapped read/write operations, forced synchronizing, and cache dropping. Multiple child processes perform different I/O operations on the same file. The default is 1.
- **Percent** – Optional. The percentage of free space on the file system to use during the IO stress test. The default is 80%.
- **InstallDependencies** – Optional. If the value is `True`, Systems Manager installs the required dependencies on the target instances if they are not already installed. The default is `True`. The dependency is **stress-ng**.

The following is an example of the string you can enter in the console.

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

AWSFIS-Run-Kill-Process

Stops the specified process in the instance, using the **killall** command. Uses the AWSFIS-Run-Kill-Process SSM document.

Action type (console only)

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

ARN

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

Document parameters

- **ProcessName** – Required. The name of the process to stop.
- **Signal** – Optional. The signal to send along with the command. The possible values are `SIGTERM` (which the receiver can choose to ignore) and `SIGKILL` (which cannot be ignored). The default is `SIGTERM`.
- **InstallDependencies** – Optional. If the value is `True`, Systems Manager installs the required dependencies on the target instances if they are not already installed. The default is `True`. The dependency is **killall**.

The following is an example of the string you can enter in the console.

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

AWSFIS-Run-Memory-Stress

Runs memory stress on an instance using the **stress-ng** tool. Uses the AWSFIS-Run-Memory-Stress SSM document.

Action type (console only)

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

ARN

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

Document parameters

- **DurationSeconds** – Required. The duration of the memory stress test, in seconds.
- **Workers** – Optional. The number of virtual memory stressors. The default is 1.
- **Percent** – Required. The percentage of virtual memory to use during the memory stress test.
- **InstallDependencies** – Optional. If the value is `True`, Systems Manager installs the required dependencies on the target instances if they are not already installed. The default is `True`. The dependency is **stress-ng**.

The following is an example of the string you can enter in the console.

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

AWSFIS-Run-Network-Blackhole-Port

Drops inbound or outbound traffic for the protocol and port using the **iptables** tool. Uses the AWSFIS-Run-Network-Blackhole-Port SSM document.

Action type (console only)

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

ARN

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

Document parameters

- **Protocol** – Required. The protocol. The possible values are `tcp` and udp.
- **Port** – Required. The port number.
- **TrafficType** – Optional. The type of traffic. The possible values are `ingress` and `egress`. The default is `ingress`.
- **DurationSeconds** – Required. The duration of the network blackhole test, in seconds.
- **InstallDependencies** – Optional. If the value is `True`, Systems Manager installs the required dependencies on the target instances if they are not already installed. The default is `True`. The dependencies are **atd**, **dig**, and **iptables**.

The following is an example of the string you can enter in the console.

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

AWSFIS-Run-Network-Latency

Adds latency to the network interface using the **tc** tool. Uses the AWSFIS-Run-Network-Latency SSM document.

Action type (console only)

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

ARN

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

Document parameters

- **Interface** – Optional. The network interface. The default is `eth0`.
- **DelayMilliseconds** – Optional. The delay, in milliseconds. The default is 200.
- **DurationSeconds** – Required. The duration of the network latency test, in seconds.
- **InstallDependencies** – Optional. If the value is `True`, Systems Manager installs the required dependencies on the target instances if they are not already installed. The default is `True`. The dependencies are **atd**, **dig**, and **tc**.

The following is an example of the string you can enter in the console.

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

AWSFIS-Run-Network-Latency-Sources

Adds latency and jitter to the network interface using the **tc** tool for traffic to or from specific sources. Uses the AWSFIS-Run-Network-Latency-Sources SSM document.

Action type (console only)

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

ARN

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

Document parameters

- **Interface** – Optional. The network interface. The default is `eth0`.
- **DelayMilliseconds** – Optional. The delay, in milliseconds. The default is 200.
- **JitterMilliseconds** – Optional. The jitter, in milliseconds. The default is 10.
- **Sources** – Required. The sources, separated by commas. The possible values are: an IPv4 address, an IPv4 CIDR block, a domain name, `DYNAMODB`, and `S3`. If you specify `DYNAMODB` or `S3`, this applies only to the Regional endpoint in the current Region.
- **TrafficType** – Optional. The type of traffic. The possible values are `ingress` and `egress`. The default is `ingress`.
- **DurationSeconds** – Required. The duration of the network latency test, in seconds.
- **InstallDependencies** – Optional. If the value is `True`, Systems Manager installs the required dependencies on the target instances if they are not already installed. The default is `True`. The dependencies are **atd**, **dig**, **jq**, and **tc**.

The following is an example of the string you can enter in the 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

Adds packet loss to the network interface using the **tc** tool. Uses the AWSFIS-Run-Network-Packet-Loss SSM document.

Action type (console only)

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

ARN

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

Document parameters

- **Interface** – Optional. The network interface. The default is `eth0`.
- **LossPercent** – Optional. The percentage of packet loss. The default is 7%.
- **DurationSeconds** – Required. The duration of the network packet loss test, in seconds.
- **InstallDependencies** – Optional. If the value is `True`, Systems Manager installs the required dependencies on the target instances. The default is `True`. The dependencies are **atd**, **dig**, and **tc**.

The following is an example of the string you can enter in the console.

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

AWSFIS-Run-Network-Packet-Loss-Sources

Adds packet loss to the network interface using the **tc** tool for traffic to or from specific sources. Uses the AWSFIS-Run-Network-Packet-Loss-Sources SSM document.

Action type (console only)

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

ARN

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

Document parameters

- **Interface** – Optional. The network interface. The default is `eth0`.
- **LossPercent** – Optional. The percentage of packet loss. The default is 7%.
- **Sources** – Required. The sources, separated by commas. The possible values are: an IPv4 address, an IPv4 CIDR block, a domain name, `DYNAMODB`, and `S3`. If you specify `DYNAMODB` or `S3`, this applies only to the Regional endpoint in the current Region.
- **TrafficType** – Optional. The type of traffic. The possible values are `ingress` and `egress`. The default is `ingress`.
- **DurationSeconds** – Required. The duration of the network packet loss test, in seconds.

- **InstallDependencies** – Optional. If the value is `True`, Systems Manager installs the required dependencies on the target instances. The default is `True`. The dependencies are **atd**, **dig**, **jq**, and **tc**.

The following is an example of the string you can enter in the console.

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

Examples

For an example experiment template, see the section called "Run a pre-configured AWS FIS SSM document".

For an example tutorial, see Run CPU stress on an instance.

Troubleshooting

Use the following procedure to troubleshoot issues.

To troubleshoot issues with SSM documents

1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Node Management**, **Run Command**.
3. On the **Command history** tab, use the filters to locate the run of the document.
4. Choose the ID of the command to open its details page.
5. Choose the ID of the instance. Review the output and errors for each step.

Use the AWS FIS aws:ecs:task actions

You can use the **aws:ecs:task** actions to inject faults into your Amazon ECS tasks.

These actions use an SSM agent as a sidecar container to run SSM documents that will perform the fault injection and registers Amazon ECS tasks as SSM managed instances via the sidecar container. To use these actions, you will need to update your Amazon ECS task definitions to add the SSM agent as a sidecar container so that it registers the task where its running as an SSM managed

instance. When you run a AWS FIS experiment targeting `aws:ecs:task`, AWS FIS maps the target Amazon ECS tasks you specify on a AWS FIS experiment template to a set of SSM managed instances using a resource tag, `ECS_TASK_ARN`, that is added to the managed instance. The tag value is the ARN of the associated Amazon ECS task where the SSM documents should be executed, so should not be removed when running the experiment.

Actions

- 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”

Limitations

- The following actions do not work with AWS Fargate:
  - aws:ecs:task-kill-process
  - aws:ecs:task-network-blackhole-port
  - aws:ecs:task-network-latency
  - aws:ecs:task-network-packet-loss
- If you enabled ECS Exec, you must disable it before you can use these actions.

Requirements

- Add the following permissions to the AWS FIS experiment role:
  - `ssm:SendCommand`
  - `ssm:ListCommands`
  - `ssm:CancelCommand`
- Add the following permissions to the Amazon ECS task IAM role:
  - `ssm:CreateActivation`
  - `ssm:AddTagsToResource`

- `iam:PassRole`

  Note that you can specify the ARN of the managed instance role as the resource for `iam:PassRole`.

- Create an Amazon ECS task execution IAM role and add the AmazonECSTaskExecutionRolePolicy managed policy.
- Add the following permissions to the managed instance role attached to tasks registered as managed instances:
  - `ssm:DeleteActivation`
  - `ssm:DeregisterManagedInstance`
- Add the AmazonSSMManagedInstanceCore managed policy to the managed instance role attached to tasks registered as managed instances.
- Set the environment variable `MANAGED_INSTANCE_ROLE_NAME` to the name of the managed instance role.
- Add an SSM agent container to the ECS task definition. The command script registers ECS tasks as managed instances.

```
{
    "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; yum upgrade -y; yum 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_SIDECAR,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": []
```

```
}
```

For a more readable version of the script, see the section called “Reference version of the script”.

- When using the `aws:ecs:task-network-blackhole-port`, `aws:ecs:task-network-latency`, and `aws:ecs:task-network-packet-loss` actions, you must update the SSM Agent container in the ECS task definition using one of the following options.
  - **Option 1** – Add the specific Linux capability.

    ```
    "linuxParameters": {
        "capabilities": {
            "add": [
                "NET_ADMIN"
            ]
        }
    },
    ```

  - **Option 2** – Add all Linux capabilities.

    ```
    "privileged": true,
    ```

- When using the `aws:ecs:task-kill-process`, `aws:ecs:task-network-blackhole-port`, `aws:ecs:task-network-latency`, and `aws:ecs:task-network-packet-loss` actions, the ECS task definition must have `pidMode` set to `task`.

Reference version of the script

The following is a more readable version of the script in the Requirements section, for your reference.

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

# This is the activation script used to register ECS tasks as Managed Instances in SSM
# The script retrieves information form 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_SIDECAR: 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

yum upgrade -y
yum 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_SIDECAR,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
```

Example experiment template

The following is an example experiment template for the the section called “aws:ecs:task-cpu-stress” action.

```
{
    "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 the AWS FIS aws:eks:pod actions

You can use the **aws:eks:pod** actions to inject faults into the Kubernetes pods running in your EKS clusters.

Actions

- 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”

Limitations

- The following actions do not work with AWS Fargate:
  - aws:eks:pod-network-blackhole-port
  - aws:eks:pod-network-latency
  - aws:eks:pod-network-packet-loss
- The following actions do not support the `bridge` network mode:
  - aws:eks:pod-network-blackhole-port
  - aws:eks:pod-network-latency
  - aws:eks:pod-network-packet-loss
- You can't identify targets of type **aws:eks:pod** in your experiment template using resource ARNs or resource tags. You must identify targets using the required resource parameters.

- The actions `aws:eks:pod-network-latency` and `aws:eks:pod-network-packet-loss` should not be run in parallel and target the same pod. Depending on the value of the `maxErrors` parameter you specify, the action may end in completed or in failed state:
  - If `maxErrorsPercent` is 0 (default), the action will end in failed state.
  - Otherwise, the failure will add up to the `maxErrorsPercent` budget. If the number of failed injections do not reach the provided `maxErrors`, the action will end up in completed state.
  - You can identify these failures from the logs of the injected ephemeral container in the target pod. It will fail with `Exit Code: 16`.
- The action `aws:eks:pod-network-blackhole-port` should not be run in parallel with other actions that target the same pod and use the same `trafficType`. Parallel actions using different traffic types are supported.
- FIS can only monitor the status of fault injection when the `securityContext` of the target pods is set to `readOnlyRootFilesystem: false`. Without this configuration, all EKS pod actions will fail.

Requirements

- Install the AWS CLI on your computer. This is needed only if you'll use the AWS CLI to create IAM roles. For more information, see Installing or updating the AWS CLI.
- Install **kubectl** on your computer. This is needed only to interact with the EKS cluster to configure or monitor the target application. For more information, see https://kubernetes.io/docs/tasks/tools/.
- The minimum supported version of EKS is 1.23.

Create a service role for the Kubernetes service account

Create an IAM role to use as a service role. For more information, see the section called "Experiment role".

Configure the Kubernetes service account

Configure a Kubernetes service account to run experiments with targets in the specified Kubernetes namespace. In the following example, the service account is *`myserviceaccount`* and the namespace is *`default`*. Note that default is one of the standard Kubernetes namespaces.

To configure your Kubernetes service account

1. Create a file named `rbac.yaml` and add the following.

```
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. Run the following command.

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

Map your experiment role to the Kubernetes user

Use the following command to create an identity mapping. For more information, see Manage IAM users and roles in the **eksctl** documentation.

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

Pod container images

The pod container images provided by AWS FIS are hosted in Amazon ECR. When you reference an image from Amazon ECR, you must use the full image URI.

AWS Region	Image URI
US East (Ohio)	051821878176.dkr.ecr.us-east-2.amazonaws.com/aws-fis-pod:0.1
US East (N. Virginia)	731367659002.dkr.ecr.us-east-1.amazonaws.com/aws-fis-pod:0.1
US West (N. California)	080694859247.dkr.ecr.us-west-1.amazonaws.com/aws-fis-pod:0.1

AWS Region	Image URI
US West (Oregon)	864386544765.dkr.ecr.us-west-2.amazonaws.com/aws-fis-pod:0.1
Africa (Cape Town)	056821267933.dkr.ecr.af-south-1.amazonaws.com/aws-fis-pod:0.1
Asia Pacific (Hong Kong)	246405402639.dkr.ecr.ap-east-1.amazonaws.com/aws-fis-pod:0.1
Asia Pacific (Mumbai)	524781661239.dkr.ecr.ap-south-1.amazonaws.com/aws-fis-pod:0.1
Asia Pacific (Seoul)	526524659354.dkr.ecr.ap-northeast-2.amazonaws.com/aws-fis-pod:0.1
Asia Pacific (Singapore)	316401638346.dkr.ecr.ap-southeast-1.amazonaws.com/aws-fis-pod:0.1
Asia Pacific (Sydney)	488104106298.dkr.ecr.ap-southeast-2.amazonaws.com/aws-fis-pod:0.1
Asia Pacific (Tokyo)	635234321696.dkr.ecr.ap-northeast-1.amazonaws.com/aws-fis-pod:0.1
Canada (Central)	490658072207.dkr.ecr.ca-central-1.amazonaws.com/aws-fis-pod:0.1
Europe (Frankfurt)	713827034473.dkr.ecr.eu-central-1.amazonaws.com/aws-fis-pod:0.1
Europe (Ireland)	205866052826.dkr.ecr.eu-west-1.amazonaws.com/aws-fis-pod:0.1
Europe (London)	327424803546.dkr.ecr.eu-west-2.amazonaws.com/aws-fis-pod:0.1

AWS Region	Image URI
Europe (Milan)	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
Europe (Stockholm)	263175118295.dkr.ecr.eu-north-1.amazonaws.com/aws-fis-pod:0.1
Middle East (Bahrain)	065825543785.dkr.ecr.me-south-1.amazonaws.com/aws-fis-pod:0.1
South America (São Paulo)	767113787785.dkr.ecr.sa-east-1.amazonaws.com/aws-fis-pod:0.1
AWS GovCloud (US-East)	246533647532.dkr.ecr.us-gov-east-1.amazonaws.com/aws-fis-pod:0.1
AWS GovCloud (US-West)	246529956514.dkr.ecr.us-gov-west-1.amazonaws.com/aws-fis-pod:0.1

Example experiment template

The following is an example experiment template for the the section called “aws:eks:pod-network-latency” action.

```
{
    "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"
    }
}
```

List the AWS FIS actions using the AWS CLI

You can use the AWS Command Line Interface (AWS CLI) to view information about the actions that AWS FIS supports.

Prerequisite

Install the AWS CLI on your computer. To get started, see the AWS Command Line Interface User Guide. For more information about the commands for AWS FIS, see fis in the *AWS CLI Command Reference*.

Example: List the names of all actions

You can list the names of all actions using the list-actions command as follows.

```
aws fis list-actions --query "actions[*].[id]" --output text | sort
```

The following is example output.

```
aws:cloudwatch:assert-alarm-state
aws:dynamodb:global-table-pause-replication
aws:ebs:pause-volume-io
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:ecs:drain-container-instances
aws:ecs:stop-task
aws:eks:inject-kubernetes-custom-resource
aws:eks:terminate-nodegroup-instances
aws:elasticache:interrupt-cluster-az-power
aws:fis:inject-api-internal-error
aws:fis:inject-api-throttle-error
aws:fis:inject-api-unavailable-error
aws:fis:wait
aws:network:disrupt-connectivity
aws:network:route-table-disrupt-cross-region-connectivity
aws:network:transit-gateway-disrupt-cross-region-connectivity
aws:rds:failover-db-cluster
aws:rds:reboot-db-instances
aws:s3:bucket-pause-replication
aws:ssm:send-command
aws:ssm:start-automation-execution
```

Example: View information about an action

After you have the name of an action, you can view detailed information about the action using the get-action command as follows.

```
aws fis get-action --id aws:ec2:reboot-instances
```

The following is example output.

```
{
    "action": {
        "id": "aws:ec2:reboot-instances",
        "description": "Reboot the specified EC2 instances.",
        "targets": {
            "Instances": {
                "resourceType": "aws:ec2:instance"
            }
        },
        "tags": {}
    }
}
```

Experiment templates for AWS FIS

An experiment template contains one or more actions to run on specified targets during an experiment. It also contains the stop conditions that prevent the experiment from going out of bounds. After you create an experiment template, you can use it to run an experiment.

Template components

You'll use the following components to construct experiment templates:

Action set

The AWS FIS actions that you want to run. Actions can be run in a set order that you specify, or they can be run simultaneously. For more information, see Action set.

Targets

The AWS resources on which a specific action is carried out. For more information, see Targets.

Stop conditions

The CloudWatch alarms that define a threshold at which your application performance is not acceptable. If a stop condition is triggered while an experiment is running, AWS FIS stops the experiment. For more information, see Stop conditions.

Experiment role

An IAM role that grants AWS FIS the permissions required so that it can run experiments on your behalf. For more information, see Experiment role.

Experiment options

Options for the experiment template. For more information, see Experiment options.

Your account has quotas related to AWS FIS. For example, there is a quota on the number of actions per experiment template. For more information, see Quotas and limitations.

Template syntax

The following is the syntax for an experiment template.

```
{
```

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

For examples, see Example templates.

Get started

To create an experiment template using the AWS Management Console, see Create an experiment template.

To create an experiment template using the AWS CLI, see Example AWS FIS experiment templates.

Action set for AWS FIS

To create an experiment template, you must define one or more actions to make up the action set. For a list of predefined actions provided by AWS FIS, see Actions.

You can run an action only once during an experiment. To run the same AWS FIS action more than once in the same experiment, add it to the template multiple times using different names.

Contents

Action syntax

The following is the syntax for an action set.

```
{
    "actions": {
        "action_name": {
            "actionId": "aws:service:action-type",
```

```
            "description": "string",
            "parameters": {
                "name": "value"
             },
            "startAfter": ["action_name", ...],
            "targets": {
                "resource_type": "target_name"
            }
        }
    }
}
```

When you define an action, you provide the following:

action_name

A name for the action.

actionId

The action identifier.

description

An optional description.

parameters

Any action parameters.

startAfter

Any actions that must complete before this action can start. Otherwise, the action runs at the start of the experiment.

targets

Any action targets.

For examples, see the section called “Example actions”.

Action duration

If an action includes a parameter that you can use to specify the duration of the action, by default, the action is considered complete only after the specified duration has elapsed. If you have set

the `emptyTargetResolutionMode` experiment option to `skip`, then the action will complete immediately with status 'skipped' when no targets were resolved. For example, if you specify a duration of 5 minutes, AWS FIS considers the action complete after 5 minutes. It then starts the next action, until all actions are complete.

Duration can be either the length of time that an action condition is maintained or the length of time for which metrics are monitored. For example, latency is injected for the duration of time specified. For near instantaneous action types, such as terminating an instance, stop conditions are monitored for the duration of time specified.

If an action includes a post action within the action parameters, the post action runs after the action completes. The time it takes to complete the post action might cause a delay between the specified action duration and the beginning of the next action (or the end of the experiment, if all other actions are complete).

Example actions

The following are example actions.

Examples

- Stop EC2 instances
- Interrupt Spot Instances
- Disrupt network traffic
- Terminate EKS workers

Example: Stop EC2 instances

The following action stops the EC2 instances identified using the target named *`targetInstances`*. After two minutes, it restarts the target instances.

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

```
        }
    }
}
```

Example: Interrupt Spot Instances

The following action stops the Spot Instances identified using the target named *targetSpotInstances*. It waits two minutes before interrupting the Spot Instance.

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

Example: Disrupt network traffic

The following action denies traffic between the target subnets and subnets in other Availability Zones.

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

Example: Terminate EKS workers

The following action terminates 50% of the EC2 instances in the EKS cluster identified using the target named *targetNodeGroups*.

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

Targets for AWS FIS

A target is one or more AWS resources on which an action is performed by AWS Fault Injection Service (AWS FIS) during an experiment. Targets can be in the same AWS account as the experiment, or in a different account using a multi-account experiment. To learn more about targeting resources in a different account, see *Multi-account experiments*.

You define targets when you create an experiment template. You can use the same target for multiple actions in your experiment template.

AWS FIS identifies all targets at the start of the experiment, before starting any of the actions in the actions set. AWS FIS uses the target resources that it selects for the entire experiment. If no targets are found, the experiment fails.

Contents

- Example filters

Target syntax

The following is the syntax for a target.

```
{
    "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"
        }
    }
}
```

When you define a target, you provide the following:

target_name

A name for the target.

resourceType

The resource type.

resourceArns

The Amazon Resource Names (ARN) of specific resources.

resourceTags

The tags applied to specific resources.

parameters

The parameters that identify targets using specific attributes.

filters

The resource filters scopes the identified target resources using specific attributes.

selectionMode

The selection mode for the identified resources.

For examples, see the section called “Example targets”.

Resource types

Each AWS FIS action is performed on a specific AWS resource type. When you define a target, you must specify exactly one resource type. When you specify a target for an action, the target must be the resource type supported by the action.

The following resource types are supported by AWS FIS:

- **aws:dynamodb:global-table** – An Amazon DynamoDB global table
- **aws:ec2:autoscaling-group** – An Amazon EC2 Auto Scaling group
- **aws:ec2:ebs-volume** – An Amazon EBS volume
- **aws:ec2:instance** – An Amazon EC2 instance
- **aws:ec2:spot-instance** – An Amazon EC2 Spot Instance
- **aws:ec2:subnet** – An Amazon VPC subnet
- **aws:ec2:transit-gateway** – A transit gateway
- **aws:ecs:cluster** – An Amazon ECS cluster
- **aws:ecs:task** – An Amazon ECS task
- **aws:eks:cluster** – An Amazon EKS cluster
- **aws:eks:nodegroup** – An Amazon EKS node group
- **aws:eks:pod** – A Kubernetes pod

- **aws:elasticache:redis-replicationgroup** – An ElastiCache Redis Replication Group
- **aws:iam:role** – An IAM role
- **aws:rds:cluster** – An Amazon Aurora DB cluster
- **aws:rds:db** – An Amazon RDS DB instance
- **aws:s3:bucket** – An Amazon S3 bucket

Identify target resources

When you define a target in the AWS FIS console, you can choose specific AWS resources (of a specific resource type) to target. Or, you can let AWS FIS identify a group of resources based on the criteria that you provide.

To identify your target resources, you can specify the following:

- **Resource IDs** – The resource IDs of specific AWS resources. All resource IDs must represent the same type of resource.
- **Resource tags** – The tags applied to specific AWS resources.
- **Resource filters** – The path and values that represent resources with specific attributes. For more information, see Resource filters.
- **Resource parameters** – The parameters that represent resources that meet specific criteria. For more information, see Resource parameters.

Considerations

- You can't specify both a resource ID and a resource tag for the same target.
- You can't specify both a resource ID and a resource filter for the same target.
- If you specify a resource tag with an empty tag value, it is not equivalent to a wildcard. It matches resources that have a tag with the specified tag key and an empty tag value.

Resource filters

Resource filters are queries that identify target resources according to specific attributes. AWS FIS applies the query to the output of an API action that contains the canonical description of the AWS resource, according to the resource type that you specify. Resources that have attributes that match the query are included in the target definition.

Each filter is expressed as an attribute path and possible values. A path is a sequence of elements, separated by periods, that describe the path to reach an attribute in the output of the **Describe** action for a resource. Each element must be expressed in Pascal case, even if the output of the **Describe** action for a resource is in camel case. For example, you should use `AvailabilityZone`, not `availablityZone` as an attribute element.

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

The following table includes the API actions and AWS CLI commands that you can use to get the canonical descriptions for each resource type. AWS FIS runs these actions on your behalf to apply the filters that you specify. The corresponding documentation describes the resources that are included in the results by default. For example, the documentation for **DescribeInstances** states that recently terminated instances might appear in the results.

Resource type	API action	AWS CLI command
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
aws:ecs:cluster	DescribeClusters	describe-clusters
aws:ecs:task	DescribeTasks	describe-tasks
aws:eks:cluster	DescribeClusters	describe-clusters
aws:eks:nodegroup	DescribeNodegroup	describe-nodegroup

Resource type	API action	AWS CLI command
aws:elasticache:redis-replicationgroup	DescribeReplicationGroups	describe-replication-groups
aws:iam:role	ListRoles	list-roles
aws:rds:cluster	DescribeDBClusters	describe-db-clusters
aws:rds:db	DescribeDBInstances	describe-db-instances
aws:s3:bucket	ListBuckets	list-buckets

The following logic applies to all resource filters:

- Values inside a filter – OR
- Values across filters – AND

For examples, see the section called “Example filters”.

Resource parameters

Resource parameters identify target resources according to specific criteria.

The following resource type supports parameters.

aws:ec2:ebs-volume

- `availabilityZoneIdentifier` – The code (for example, us-east-1a) of the Availability Zone that contains the target volumes.

aws:ec2:subnet

- `availabilityZoneIdentifier` – The code (for example, us-east-1a) or AZ ID (for example, use1-az1) of the Availability Zone that contains the target subnets.
- `vpc` – The VPC that contains the target subnets. Does not support more than one VPC per account.

aws:ecs:task

- `cluster` – The cluster that contains the target tasks.

- `service` – The service that contains the target tasks.

aws:eks:pod

- `availabilityZoneIdentifier` – Optional. The Availability Zone that contains the target pods. For example, `us-east-1d`. We determine the Availability Zone of a pod by comparing its hostIP and the CIDR of the cluster subnet.
- `clusterIdentifier` – Required. The name or ARN of the target EKS cluster.
- `namespace` – Required. The Kubernetes namespace of the target pods.
- `selectorType` – Required. The selector type. The possible values are `labelSelector`, `deploymentName`, and `podName`.
- `selectorValue` – Required. The selector value. This value depends on the value of `selectorType`.
- `targetContainerName` – Optional. The name of the target container as defined in the pod spec. The default is the first container defined in each target pod spec.

aws:rds:cluster

- `writerAvailabilityZoneIdentifiers` – Optional. The Availability Zones of the writer of the DB cluster. Possible values are: a comma separated list of Availability Zone identifiers, `all`.

aws:rds:db

- `availabilityZoneIdentifiers` – Optional. The Availability Zones of the DB instance to be affected. Possible values are: a comma separated list of Availability Zone identifiers, `all`.

aws:elasticache:redis-replicationgroup

- `availabilityZoneIdentifier` – Required. The code (for example, us-east-1a) or AZ ID (for example, use1-az1) of the Availability Zone that contains the target nodes.

Selection mode

You scope the identified resources by specifying a selection mode. AWS FIS supports the following selection modes:

- `ALL` – Run the action on all targets.
- `COUNT(n)` – Run the action on the specified number of targets, chosen from the identified targets at random. For example, COUNT(1) selects one of the identified targets.

- PERCENT(n) – Run the action on the specified percentage of targets, chosen from the identified targets at random. For example, PERCENT(25) selects 25% of the identified targets.

If you have an odd number of resources and specify 50%, AWS FIS rounds down. For example, if you add five Amazon EC2 instances as targets and scope to 50%, AWS FIS rounds down to two instances. You can't specify a percentage that is less than one resource. For example, if you add four Amazon EC2 instances and scope to 5%, AWS FIS can't select an instance.

If you define multiple targets using the same target resource type, AWS FIS can select the same resource multiple times.

Regardless of which selection mode you use, if the scope that you specify identifies no resources, the experiment fails.

Example targets

The following are example targets.

Examples

- Instances in the specified VPC with the specified tags
- Tasks with the specified parameters

Example: Instances in the specified VPC with the specified tags

The possible targets for this example are Amazon EC2 instances in the specified VPC with the tag env=prod. The selection mode specifies that AWS FIS chooses one of these targets at random.

```
{
    "targets": {
        "randomInstance": {
            "resourceType": "aws:ec2:instance",
            "resourceTags": {
                "env": "prod"
            },
            "filters": [
                {
                    "path": "VpcId",
                    "values": [
```

```
                    "vpc-aabbcc11223344556"
                ]
            }
        ],
        "selectionMode": "COUNT(1)"
    }
  }
}
```

Example: Tasks with the specified parameters

The possible targets for this example are Amazon ECS tasks with the specified cluster and service. The selection mode specifies that AWS FIS choose one of these targets at random.

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

Example filters

The following are example filters.

Examples

- EC2 instances
- DB clusters

Example: EC2 instances

When you specify a filter for an action that supports the **aws:ec2:instance** resource type, AWS FIS uses the Amazon EC2 **describe-instances** command and applies the filter to identify the targets.

The **describe-instances** command returns JSON output where each instance is a structure under `Instances`. The following is partial output that includes fields marked with *`italics`*. We'll provide examples that use these fields to specify an attribute path from the structure of the JSON output.

```
{
    "Reservations": [
        {
            "Groups": [],
            "Instances": [
                {
                    "ImageId": "ami-00111111111111111",
                    "InstanceId": "i-00aaaaaaaaaaaaaa",
                    "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",
                    ...
                },
                ...
                {
                    ...
                }
            ],
```

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

To select instances in a specific Availability Zone using a resource filter, specify the attribute path for `AvailabilityZone` and the code for the Availability Zone as the value. For example:

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

To select instances in a specific subnet using a resource filter, specify the attribute path for `SubnetId` and the ID of the subnet as the value. For example:

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

To select instances that are in a specific instance state, specify the attribute path for `Name` and one of the following state names as the value: `pending` | `running` | `shutting-down` | `terminated` | `stopping` | `stopped`. For example:

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

Example: Amazon RDS cluster (DB cluster)

When you specify a filter for an action that supports the **aws:rds:cluster** resource type, AWS FIS runs the Amazon RDS **describe-db-clusters** command and applies the filter to identify the targets.

The **describe-db-clusters** command returns JSON output similar to the following for each DB cluster. The following is partial output that includes fields marked with *italics*. We'll provide examples that use these fields to specify an attribute path from the structure of the JSON output.

```
[
    {
        "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",
        ...
    }
]
```

To apply a resource filter that returns only the DB clusters that use a specific DB engine, specify the attribute path as `Engine` and the value as `aurora-postgresql` as shown in the following example.

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

To apply a resource filter that returns only the DB clusters in a specific Availability Zone, specify the attribute path and value as shown in the following example.

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

Stop conditions for AWS FIS

AWS Fault Injection Service (AWS FIS) provides controls and guardrails for you to run experiments safely on AWS workloads. A *stop condition* is a mechanism to stop an experiment if it reaches a threshold that you define as an Amazon CloudWatch alarm. If a stop condition is triggered during an experiment, AWS FIS stops the experiment. You cannot resume a stopped experiment.

To create a stop condition, first define the steady state for your application or service. The steady state is when your application is performing optimally, defined in terms of business or technical metrics. For example, latency, CPU load, or number of retries. You can use the steady state to create a CloudWatch alarm that you can use to stop an experiment if your application or service reaches a state where its performance is not acceptable. For more information, see Using Amazon CloudWatch alarms in the *Amazon CloudWatch User Guide*.

Your account has a quota on the number of stop conditions that you can specify in an experiment template. For more information, see Quotas and limitations for AWS Fault Injection Service.

Stop condition syntax

When you create an experiment template, you specify one or more stop conditions by specifying the CloudWatch alarms that you created.

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

```
}
```

The following example indicates that the experiment template does not specify a stop condition.

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

Learn more

For a tutorial that demonstrates how to create a CloudWatch alarm and add a stop condition to an experiment template, see Run CPU stress on an instance.

For more information about the CloudWatch metrics that are available for the resource types supported by AWS FIS, see the following:

- Monitor your instances using CloudWatch
- Amazon ECS CloudWatch metrics
- Monitoring Amazon RDS metrics using CloudWatch
- Monitoring Run Command metrics using CloudWatch

IAM roles for AWS FIS experiments

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. To use AWS FIS, you must create an IAM role that grants AWS FIS the permissions required so that AWS FIS can run experiments on your behalf. You specify this experiment role when you create an experiment template. For a single-account experiment, the IAM policy for the experiment role must grant permission to modify the resources that you specify as targets in your experiment template. For a multi-account experiment, the experiment role must grant the orchestrator role permission to assume the IAM role for each target account. For more information, see Permissions for multi-account experiments.

We recommend that you follow the standard security practice of granting least privilege. You can do so by specifying specific resource ARNs or tags in your policies.

To help you get started with AWS FIS quickly, we provide AWS managed policies that you can specify when you create an experiment role. Alternatively, you can also use these policies as a model as you create your own inline policy documents.

Contents

Prerequisites

Before you begin, install the AWS CLI and create the required trust policy.

Install the AWS CLI

Before you begin, install and configure the AWS CLI. When you configure the AWS CLI, you are prompted for AWS credentials. The examples in this procedure assume that you also configured a default Region. Otherwise, add the `--region` option to each command. For more information, see Installing or updating the AWS CLI and Configuring the AWS CLI.

Create a trust relationship policy

An experiment role must have a trust relationship that allows the AWS FIS service to assume the role. Create a text file named `fis-role-trust-policy.json` and add the following trust relationship policy.

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

We recommend that you use the `aws:SourceAccount` and `aws:SourceArn` condition keys to protect yourself against the confused deputy problem. The source account is the owner of the experiment and the source ARN is the ARN of the experiment. For example, you should add the following condition block to your trust policy.

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

Add permissions to assume target account roles (multi-account experiments only)

For multi-account experiments, you need permissions that allows orchestrator account to assume target account roles. You can modify the following example and add as an inline policy document to assume target account roles:

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

Option 1: Create an experiment role and attach an AWS managed policy

Use one of the AWS managed policies from AWS FIS to get started quickly.

To create an experiment role and attach an AWS managed policy

1. Verify that there is a managed policy for the AWS FIS actions in your experiment. Otherwise, you'll need to create your own inline policy document instead. For more information, see the section called “AWS managed policies”.

2. Use the following create-role command to create a role and add the trust policy that you created in the prerequisites.

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

3. Use the following attach-role-policy command to attach the AWS managed policy.

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

Where *fis-policy-arn* is one of the following:

- 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

Option 2: Create an experiment role and add an inline policy document

Use this option for actions that don't have a managed policy, or to include only the permissions that are required for your specific experiment.

To create an experiment and add an inline policy document

1. Use the following create-role command to create a role and add the trust policy that you created in the prerequisites.

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

2. Create a text file named `fis-role-permissions-policy.json` and add a permissions policy. For an example that you can use as a starting point, see the following.

- **Fault injection actions** – Start from the following policy.

```
{
```

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

- **Amazon EBS actions** – Start from the following policy.

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

- **Amazon EC2 actions** – Start from the AWSFaultInjectionSimulatorEC2Access policy.
- **Amazon ECS actions** – Start from the AWSFaultInjectionSimulatorECSAccess policy.
- **Amazon EKS actions** – Start from the AWSFaultInjectionSimulatorEKSAccess policy.
- **Network actions** – Start from the AWSFaultInjectionSimulatorNetworkAccess policy.
- **Amazon RDS actions** – Start from the AWSFaultInjectionSimulatorRDSAccess policy.

- **Systems Manager actions** – Start from the AWSFaultInjectionSimulatorSSMAccess policy.

3. Use the following put-role-policy command to add the permissions policy that you created in the previous step.

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

Experiment options

Experiment options are optional settings for an experiment. You can define certain experiment options on the experiment template. Additional experiment options are set when you begin the experiment.

The following is the syntax for experiment options that you define on the experiment template.

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

If you do not specify any experiment options when you create the experiment template, the default for each option is used.

The following is the syntax for experiment options that you set when you begin the experiment.

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

If you do not specify any experiment options when you begin the experiment, the default `run-all` is used.

Contents

- Empty target resolution mode
- Actions mode

Account targeting

If you have multiple AWS accounts with resources that you want to target in an experiment, you can define a multi-account experiment using the account targeting experiment option. You run multi-account experiments from an orchestrator account that impacts resources in multiple target accounts. The orchestrator account owns the AWS FIS experiment template and experiment. A target account is an individual AWS account with resources that can be affected by an AWS FIS experiment. For more information, see Multi-account experiments for AWS FIS.

You use account targeting to indicate the location of your target resources. You can provide two values for account targeting:

- **single-account** – Default. The experiment will only target resources in the AWS account where AWS FIS experiment runs.
- **multi-account** – The experiment can target resources in multiple AWS accounts.

Target account configurations

To run a multi-account experiment, you must define one or more target account configurations. A target account configuration specifies the accountId, roleArn, and description for each account with resources targeted in the experiment. The account IDs of the target account configurations for an experiment template must be unique.

When you create a multi-account experiment template, the experiment template will return a read-only field, `targetAccountConfigurationsCount`, that is a count of all the target account configurations for the experiment template.

The following is the syntax for a target account configuration.

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

When you create a target account configuration, you provide the following:

accountId

12-digit AWS account ID of the target account.

roleArn

An IAM Role granting AWS FIS permissions to take actions in target account.

description

An optional description.

To learn more about how to work with target account configurations, see the section called “Work with multi-account experiments”.

Empty target resolution mode

This mode gives you the option to allow experiments to complete even when a target resource is not resolved.

- **fail** – Default. If no resources are resolved for the target, the experiment is terminated immediately with a status of `failed`.
- **skip** – If no resources are resolved for the target, the experiment will continue and any actions with no resolved targets are skipped. Actions with targets that are defined using unique identifiers, such as ARNs, cannot be skipped. If a target defined using a unique identifier is not found the experiment is terminated immediately with a status of `failed`

Actions mode

Actions mode is an optional parameter that you can specify when you start an experiment. You can set actions mode to `skip-all` to generate a target preview before injecting faults into your target resources. The target preview allows you to verify the following:

- That you have configured your experiment template to target the resources you expect. The actual resources that are targeted when you start this experiment may be different from the preview because resources may be removed, updated, or sampled randomly.
- That your logging configurations are set up correctly.

- That for multi-account experiments you have correctly set up an IAM role for each of your target account configurations.

> **Note**
>
> The `skip-all` mode does not allow you to verify that you have the necessary permissions to run the AWS FIS experiment and take actions on your resources.

The actions mode parameter accepts the following values:

- `run-all` - (Default) The experiment will take actions on target resources.
- `skip-all` - The experiment will skip all actions on target resources.

To learn more about how to set the actions mode parameter when you start an experiment, see Generate a target preview from an experiment template.

Work with AWS FIS experiment templates

You can create and manage experiment templates using the AWS FIS console or the command line. After you create an experiment template, you can use it to run an experiment.

Tasks

- Create an experiment template
- View experiment templates
- Generate a target preview from an experiment template
- Start an experiment from a template
- Update an experiment template
- Tag experiment templates
- Delete an experiment template

Create an experiment template

Before you begin, complete the following tasks:

- Plan your experiment.
- Create an IAM role that grants the AWS FIS service permission to perform actions on your behalf. For more information, see IAM roles for AWS FIS experiments.
- Ensure that you have access to AWS FIS. For more information, see AWS FIS policy examples.

To create an experiment template using the console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. Choose **Create experiment template**.
4. (Optional) For **Account targeting**, choose **Multiple accounts** to configure a multi-account experiment template.
5. For **Account targeting**, choose **Confirm**.
6. For **Description and name**, enter a description and a name for the template.
7. For **Actions**, specify the set of actions for the template. For each action, choose **Add action** and complete the following:

   - For **Name**, enter a name for the action.

     Allowed characters are alphanumeric characters, hyphens (-), and underscores(_). The name must start with a letter. No spaces are allowed. Each action name must be unique in this template.

   - (Optional) For **Description**, enter a description for the action. The maximum length is 512 characters.
   - (Optional) For **Start after**, select another action defined in this template that must be completed before the current action starts. Otherwise, the action runs at the start of the experiment.
   - For **Action type**, choose the AWS FIS action.
   - For **Target**, choose a target that you defined in the **Targets** section. If you haven't defined a target for this action yet, AWS FIS creates a new target for you.
   - For **Action parameters**, specify the parameters for the action. This section appears only if the AWS FIS action has parameters.
   - Choose **Save**.

8. For **Targets**, define the target resources on which to carry out the actions. You must specify at least one resource ID or one resource tag as a target. Choose **Edit** to edit the target that AWS FIS created for you in the previous step, or choose **Add target**. For each target, do the following:

   - For **Name**, enter a name for the target.

     Allowed characters are alphanumeric characters, hyphens (-), and underscores(_). The name must start with a letter. No spaces are allowed. Each target name must be unique in this template.

   - For **Resource type**, choose a resource type that is supported for the action.
   - For **Target method**, do one of the following:
     - Choose **Resource IDs** and then choose or add the resource IDs.
     - Choose **Resource tags, filters, and parameters** and then add the tags and filters that you need. For more information, see the section called “Identify target resources”.
   - For **Selection mode**, choose **Count** to run the action on the specified number of identified targets or choose **Percent** to run the action on the specified percentage of identified targets. By default, the action runs on all identified targets.
   - Choose **Save**.

9. To update an action with the target that you created, find the action under **Actions**, choose **Edit**, and then update **Target**. You can use the same target for multiple actions.
10. (Multi-account experiments only) For **Target account configurations**, add a Role ARN and optional description for each target account. To upload the target account role ARNs with a CSV file, choose **Upload role ARNs for all target accounts** and then choose **Choose .CSV file**
11. For **Service Access**, choose **Use an existing IAM role**, and then choose the IAM role that you created as described in the prerequisites for this tutorial. If your role is not displayed, verify that it has the required trust relationship. For more information, see the section called “Experiment role”.
12. (Optional) For **Stop conditions**, select the Amazon CloudWatch alarms for the stop conditions. For more information, see Stop conditions for AWS FIS.
13. (Optional) For **Logs**, configure the destination option. To send logs to an S3 bucket, choose **Send to an Amazon S3 bucket** and enter the bucket name and prefix. To send logs to CloudWatch Logs, choose **Send to CloudWatch Logs** and enter the log group.

14. (Optional) For **Tags**, choose **Add new tag** and specify a tag key and tag value. The tags that you add are applied to your experiment template, not the experiments that are run using the template.
15. Choose **Create experiment template**. When prompted for confirmation, enter **`create`** and choose **Create experiment template**.

To create an experiment template using the CLI

Use the create-experiment-template command.

You can load an experiment template from a JSON file.

Use the `--cli-input-json` parameter.

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

For more information, see Generating a CLI skeleton template in the *AWS Command Line Interface User Guide*. For example templates, see Example AWS FIS experiment templates.

View experiment templates

You can view the experiment templates that you created.

To view an experiment template using the console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. To view information about a specific template, select the **Experiment template ID**.
4. In the **Details** section, you can view the description and stop conditions for the template.
5. To view the actions for the experiment template, choose **Actions**.
6. To view the targets for the experiment template, choose **Targets**.
7. To view the tags for the experiment template, choose **Tags**.

To view an experiment template using the CLI

Use the list-experiment-templates command to get a list of experiment templates, and use the get-experiment-template command to get information about a specific experiment template.

Generate a target preview from an experiment template

Before you start an experiment, you can generate a target preview to verify that your experiment template is configured to target the expected resources. The resources that are targeted when you begin the actual experiment may be different from those in the preview, as resources may be removed, updated, or sampled randomly. When you generate a target preview, you start an experiment that skips all actions.

> **Note**
>
> Generating a target preview does not allow you to verify that you have the necessary permissions to take actions on your resources.

To start a target preview using the console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. To view the targets for the experiment template, choose **Targets**.
4. To verify your target resources for the experiment template, choose **Generate Preview**. When you run an experiment, this target preview will be automatically updated with the targets from the most recent experiment.

To start a target preview using the CLI

- Run the following start-experiment command. Replace the values in italics with your own values.

```
aws fis start-experiment \
    --experiment-options actionsMode=skip-all \
    --experiment-template-id EXTxxxxxxxxx
```

Start an experiment from a template

After you have created an experiment template, you can start experiments using that template.

When you start an experiment, we create a snapshot of the specified template and use that snapshot to run the experiment. Therefore, if the experiment template is updated or deleted while the experiment is running, those changes have no impact on the running experiment.

When you start an experiment, AWS FIS creates a service-linked role on your behalf. For more information, see Use service-linked roles for AWS Fault Injection Service.

After you start the experiment, you can stop it at any time. For more information, see Stop an experiment.

To start an experiment using the console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. (Optional) To generate a preview to verify your targets:
   - Choose **Targets**.
   - Choose **Generate preview**.
4. Select the experiment template, and choose **Start experiment**.
5. (Optional) To add a tag to your experiment, choose **Add new tag** and enter a tag key and a tag value.
6. Choose **Start experiment**. When prompted for confirmation, enter **`start`** and choose **Start experiment**.

To start an experiment using the CLI

Use the start-experiment command.

Update an experiment template

You can update an existing experiment template. When you update an experiment template, the changes do not affect any running experiments that use the template.

To update an experiment template using the console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. Select the experiment template, and choose **Actions**, **Update experiment template**.

4. Modify the template details as needed, and choose **Update experiment template**.

To update an experiment template using the CLI

Use the update-experiment-template command.

Tag experiment templates

You can apply your own tags to experiment templates to help you organize them. You can also implement tag-based IAM policies to control access to experiment templates.

To tag an experiment template using the console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. Select the experiment template and choose **Actions**, **Manage tags**.
4. To add a new tag, choose **Add new tag**, and then specify a key and value.

   To remove a tag, choose **Remove** for the tag.
5. Choose **Save**.

To tag an experiment template using the CLI

Use the tag-resource command.

Delete an experiment template

If you no longer need an experiment template, you can delete it. When you delete an experiment template, any running experiments that use the template are not affected. The experiment continues to run until completed or stopped. However, experiment templates that are deleted are not available for viewing from the **Experiments** page in the console.

To delete an experiment template using the console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. Select the experiment template, and choose **Actions**, **Delete experiment template**.
4. When prompted for confirmation, enter **`delete`** and choose **Delete experiment template**.

To delete an experiment template using the CLI

Use the delete-experiment-template command.

Example AWS FIS experiment templates

If you're using the AWS FIS API or a command line tool to create an experiment template, you can construct the template in JavaScript Object Notation (JSON). For more information about the components of an experiment template, see Template components.

To create an experiment using one of the example templates, save it to a JSON file (for example, `my-template.json`), replace the placeholder values in *`italics`* with your own values, and then run the following create-experiment-template command.

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

Example templates

- Stop EC2 instances based on filters
- Stop a specified number of EC2 instances
- Run a pre-configured AWS FIS SSM document
- Run a predefined Automation runbook
- Throttle API actions on EC2 instances with the target IAM role
- Stress test CPU of pods in a Kubernetes cluster

Stop EC2 instances based on filters

The following example stops all running Amazon EC2 instances in the specified Region with the specified tag in the specified VPC. It restarts them after two minutes.

```
{
    "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"
}
```

Stop a specified number of EC2 instances

The following example stops three instances with the specified tag. AWS FIS selects the specific instances to stop at random. It restarts these instances after two minutes.

```
{
    "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"
}
```

Run a pre-configured AWS FIS SSM document

The following example runs a CPU fault injection for 60 seconds on the specified EC2 instance using a pre-configured AWS FIS SSM document, AWSFIS-Run-CPU-Stress. AWS FIS monitors the experiment for two minutes.

```
{
```

```
    "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"
}
```

Run a predefined Automation runbook

The following example publishes a notification to Amazon SNS using a runbook provided by Systems Manager, AWS-PublishSNSNotification. The role must have permissions to publish notifications to the specified SNS topic.

```
{
    "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"
}
```

Throttle API actions on EC2 instances with the target IAM role

The following example throttles 100% of the API calls specified in the action definition for API calls made by the IAM role(s) specified in the target definition.

> **Note**
>
> If you wish to target EC2 instances that are members of an Auto Scaling group, please use the **aws:ec2:asg-insufficient-instance-capacity-error** action, and target by Auto Scaling group instead. For more information, see
>
> Injects `InsufficientInstanceCapacity` error responses on requests made by the target Auto Scaling groups. This action only supports Auto Scaling groups using

launch templates. To learn more about insufficient instance capacity errors, see the Amazon EC2 user guide.

Resource type

- **aws:ec2:autoscaling-group**

Parameters

- **duration** – In the AWS FIS API, the value is a string in ISO 8601 format. For example, PT1M represents one minute. In the AWS FIS console, you enter the number of seconds, minutes, or hours.

- **availabilityzoneIdentifiers** – The comma separated list of Availability Zones. Supports Zone IDs (e.g. `"use1-az1, use1-az2"`) and Zone names (e.g. `"us-east-1a"`).
- **percentage** – Optional. The percentage (1-100) of the target Auto Scaling group's launch requests to inject the fault. The default is 100.

Permissions

- `ec2:InjectApiError`with condition key ec2:FisActionId value set to `aws:ec2:asg-insufficient-instance-capacity-error` and `ec2:FisTargetArns` condition key set to target Auto Scaling groups.

- `autoscaling:DescribeAutoScalingGroups`

For an example policy, see Example: Use condition keys for `ec2:InjectApiError`.

.

```
{
    "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"
}
```

Stress test CPU of pods in a Kubernetes cluster

The following example uses Chaos Mesh to stress test the CPU of pods in an Amazon EKS Kubernetes cluster for one minute.

```
{
    "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\"],
\"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": {}
}
```

The following example uses Litmus to stress test the CPU of pods in an Amazon EKS Kubernetes cluster for one minute.

```
{
    "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": "{\"engineState\":\"active\",\"appinfo\":
{\"appns\":\"default\",\"applabel\":\"run=nginx\",\"appkind\":\"deployment\"},
\"chaosServiceAccount\":\"litmus-admin\",\"experiments\":[{\"name\":\"pod-cpu-hog
\",\"spec\":{\"components\":{\"env\":[{\"name\":\"TOTAL_CHAOS_DURATION\",\"value\":
\"60\"},{\"name\":\"CPU_CORES\",\"value\":\"1\"},{\"name\":\"PODS_AFFECTED_PERC\",
\"value\":\"100\"},{\"name\":\"CONTAINER_RUNTIME\",\"value\":\"docker\"},{\"name\":
\"SOCKET_PATH\",\"value\":\"/var/run/docker.sock\"}]},\"probe\":[]}}],\"annotationCheck
\":\"false\"}"
            },
            "targets": {
                "Cluster": "MyCluster"
            }
        }
    },
    "stopConditions": [{
        "source": "none"
    }],
    "roleArn": "arn:aws:iam::111122223333:role/role-name",
    "tags": {}
}
```

Multi-account experiments for AWS FIS

With a multi-account experiment, you can set up and run real-world failure scenarios on an application that spans multiple AWS accounts within a Region. You run multi-account experiments from an *orchestrator account* that impacts resources in multiple *target accounts*.

When you run a multi-account experiment, target accounts with affected resources will be notified via their AWS Health dashboards, providing awareness to users in the target accounts. With multi-account experiments, you can:

- Run real world failure scenarios on applications that span multiple accounts with the central controls and guardrails that AWS FIS provides.
- Control the effects of a multi-account experiment using IAM roles with fine-grained permissions and tags to define the scope of each target.
- Centrally view the actions AWS FIS takes in each account from the AWS Management Console and through AWS FIS logs.
- Monitor and audit API calls AWS FIS makes in each account with AWS CloudTrail.

This section helps you get started with multi-account experiments.

Topics

Concepts for multi-account experiments

The following are the key concepts for multi-account experiments:

Orchestrator account

The orchestrator account acts as a central account to configure and manage the experiment in the AWS FIS Console, as well as to centralize logging. The orchestrator account owns the AWS FIS experiment template and experiment.

Target accounts

A target account is an individual AWS account with resources that can be affected by an AWS FIS multi-account experiment.

Target account configurations

You define the target accounts that are part of an experiment by adding target account configurations to the experiment template. A target account configuration is an element of the experiment template that is required for multi-account experiments. You define one for each target account by setting an AWS account ID, IAM role, and an optional description.

Prerequisites for multi-account experiments

To use stop conditions for a multi-account experiment, you must first configure cross-account alarms. IAM roles are defined when you create a multi-account experiment template. You can create the necessary IAM roles before you create the template.

Content

Permissions for multi-account experiments

Multi-account experiments use IAM *role chaining* to grant permissions to AWS FIS to take actions on resources in target accounts. For multi-account experiments, you set up IAM roles in each target account and the orchestrator account. These IAM roles require a trust relationship between the target accounts and the orchestrator account, and between the orchestrator account and AWS FIS.

The IAM roles for the target accounts contain the permissions required to take action on resources and are created for an experiment template by adding target account configurations. You will create an IAM role for the orchestrator account with permission to assume the roles of target accounts and establish a trust relationship with AWS FIS. This IAM role is used as the `roleArn` for the experiment template.

To learn more about role chaining, see Roles Terms and concepts. in IAM's User Guide

In the following example, you will set up permissions for an orchestrator account A to run an experiment with `aws:ebs:pause-volume-io` in target account B.

1. In account B, create an IAM role with the permissions required to run the action. For permissions required for each action, see the section called "Actions reference". The following example shows the permissions a target account grants to run the EBS Pause Volume IO action the section called "aws:ebs:pause-volume-io".

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:PauseVolumeIO"
            ],
            "Resource": "arn:aws:ec2:region:accountIdB:volume/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": "*"
        }
    ]
}
```

2. Next, add a trust policy in account B that creates a trust relationship with account A. Choose a name for the IAM role for account A, which you will create in step 3.

```
{
    "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::accountIdA:role/role_name"
                 }
             }
         }
    ]
}
```

3. In account A, create an IAM role. This role name must match the role you specified in the trust policy in step 2. To target multiple accounts, you grant the orchestrator permissions to assume each role. The following example shows the permissions for account A to assume account B. If you have additional target accounts, you will add additional role ARNs to this policy. You can only have one role ARN per target account.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": [
                "arn:aws:iam::accountIdB:role/role_name"
            ]
        }
    ]
}
```

4. This IAM role for account A is used as the `roleArn` for the experiment template. The following example shows the trust policy required in the IAM role that grants AWS FIS permissions to assume account A, the orchestrator account.

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "fis.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

You can also use Stacksets to provision multiple IAM roles at one time. To use CloudFormation StackSets, you will need to set up the necessary StackSet permissions in your AWS accounts. To learn more, see Working with AWS CloudFormation StackSets.

Stop conditions for multi-account experiments (optional)

A *stop condition* is a mechanism to stop an experiment if it reaches a threshold that you define as an alarm. To set up a stop condition for your multi-account experiment, you can use cross-account alarms. You must enable sharing in each target account to make the alarm available to the orchestrator account using read-only permissions. Once shared, you can combine metrics from different target accounts using Metric Math. Then, you can add this alarm as a stop condition for the experiment.

To learn more about cross-account dashboards, see Enabling cross-account functionality in CloudWatch.

Work with multi-account experiments

You can create and manage multi-account experiment templates using the AWS FIS console or the command line. You create a multi-account experiment by specifying the account targeting experiment option as `"multi-account"`, and adding target account configurations. After you create a multi-account experiment template, you can use it to run an experiment.

Content

- Update a target account configuration
- Delete a target account configuration

Best practices for multi-account experiments

The following are best practices to using multi-account experiments:

- When you configure targets for multi-account experiments, we recommend targeting with consistent resource tags across all target accounts. An AWS FIS experiment will resolve resources with consistent tags in each target account. An action must resolve at least one target resource in any target account or will fail, except for experiments with `emptyTargetResolutionMode` set to `skip`. Action quotas apply per account. If you want to target resources by resource ARNs, the same single-account limit per action applies.
- When you target resources in one or more availability zones using parameters or filters, you should specify an *AZ ID*, not an AZ name. The AZ ID is a unique and consistent identifier for an Availability Zone across accounts. To learn how to find the AZ ID for the availability zones in your account, see Availability Zone IDs for your AWS resources.

Create a multi-account experiment template

To learn how to create an experiment template via the AWS Management Console

See Create an experiment template.

To create an experiment template using the CLI

1. Open the AWS Command Line Interface
2. To create an experiment from a saved JSON file with the account targeting experiment option set to `"multi-account"` (for example, `my-template.json`), replace the placeholder values in *`italics`* with your own values, and then run the following create-experiment-template command.

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

   This will return the experiment template in the response. Copy the `id` from the response, which is the ID of the experiment template.

3. Run the create-target-account-configuration command to add a target account configuration to the experiment template. Replace the placeholder values in *italics* with your own values, using the `id` from step 2 as the value for the `--experiment-template-id` parameter, and then run the following. The `--description` parameter is optional. Repeat this step for each target account.

```
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. Run the get-target-account-configuration command to retrieve the details for a specific target account configuration.

```
aws fis get-target-account-configuration --experiment-template-id EXTxxxxxxxxx --
account-id 111122223333
```

5. Once you have added all your target account configurations, you can run the list-target-account-configurations command command to see that your target account configurations have been created.

```
aws fis list-target-account-configurations --experiment-template-id EXTxxxxxxxxx
```

   You can also verify that you have added target account configurations by running the get-experiment-template command. The template will return a read-only field `targetAccountConfigurationsCount` that is a count of all the target account configurations on the experiment template.

6. When you are ready, you can run the experiment template using the start-experiment command.

```
aws fis start-experiment --experiment-template-id EXTxxxxxxxxx
```

Update a target account configuration

You can update an existing target account configuration if you want to change the role ARN or description for the the account. When you update a target account configuration, the changes do not affect any running experiments that use the template.

To update a target account configuration using the AWS Management Console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**
3. Select the experiment template, and choose **Actions**, **Update experiment template**.
4. Modify the **target account configurations**, and choose **Update experiment template**.

To update a target account configuration using the CLI

Run the update-target-account-configuration command to command, replacing the placeholder values in *italics* with your own values. The `--role-arn` and `--description` parameters are optional, and will not be updated if not included.

```
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"
```

Delete a target account configuration

If you no longer need a target account configuration, you can delete it. When you delete a target account configuration, any running experiments that use the template are not affected. The experiment continues to run until completed or stopped.

To delete a target account configuration using the AWS Management Console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. Select the experiment template, and choose **Actions**, **Update**.
4. Under **Target account configurations**, select **Remove** for the target account Role ARN you want to delete.

To delete a target account configuration using the CLI

Run the delete-target-account-configuration command, replacing the placeholder values in *italics* with your own values.

```
aws fis update-target-account-configuration --experiment-template-id EXTxxxxxxxxx --
account-id 111122223333
```

AWS FIS Scenario library

Scenarios define events or conditions that customers can apply to test the resiliency of their applications, such as the interruption of compute resources on which the application is running. Scenarios are created and owned by AWS, and minimize undifferentiated heavy lifting by providing you with a group of pre-defined targets and fault actions (e.g., stopping 30% of instances in an autoscaling group) for common application impairments.

Topics

Working with AWS FIS scenarios

Scenarios are provided through a console-only scenario library and run using an AWS FIS experiment template. In order to run an experiment using a scenario, you will select the scenario from the library, specify parameters matching your workload details, and save it as an experiment template in your account.

Topics

Viewing a scenario

To view a scenario using the console:

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Scenario library**.
3. To view information about a specific scenario, select the scenario card to bring up a split panel.

- In the **Description** tab in the split panel at the bottom of the page, you can view a short description of the scenario. You can also find a short summary of pre-requisites containing a summary of the target resources required and any actions you need to take to prepare the resources for use with the scenario. Finally you can also see additional information about the targets and actions in the scenario as well as the anticipated duration when the experiment runs successfully with default settings.
- In the **Content** tab in the split panel at the bottom of the page, you can preview a partially populated version of the experiment template that will be created from the scenario.
- In the **Details** tab in the split panel at the bottom of the page, you can find a detailed explanation how the scenario is implemented. This may contain detailed information about how individual aspects of the scenario are approximated. Where applicable you can also read about what metrics to use as stop conditions and to provide observability to learn from the experiment. Finally you will find recommendations how to expand the resulting experiment template.

Using a scenario

To use a scenario using the console:

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Scenario library**.
3. To view information about a specific scenario, select the scenario card to bring up a split panel
4. To use the scenario, select the scenario card and choose **Create template with scenario**.
5. In the **Create experiment template** view fill in any missing items.
   a. Some scenarios allow you to bulk edit parameters that are shared across multiple actions or targets. This functionality will be disabled once you make any changes to the scenario, including changes by the bulk parameter editing. To use this feature select the **Edit bulk parameters** button. Edit parameters in the modal and select the **Save** button.
   b. Some experiment templates may have missing action or target parameters, highlighted on each action and target card. Select the **Edit** button for each card, add the missing information, and select the **Save** button on the card.
   c. All templates require a **Service access** execution role. You can choose an existing role or create a new role for this experiment template.

d. We recommend defining one or more optional **Stop conditions** by selecting an existing AWS CloudWatch alarm. Learn more about Stop conditions for AWS FIS. If you don't have an alarm configured yet, you can follow the instructions at Using Amazon CloudWatch Alarms and update the experiment template later.

e. We recommend enabling optional experiment **Logs** to Amazon CloudWatch logs or to an Amazon S3 bucket. Learn more about Experiment logging for AWS FIS. If you don't have appropriate resources configured yet, you can update the experiment template later.

6. In the **Create experiment template** select **Create experiment template**.
7. From the **Experiment templates** view of the AWS FIS console select **Start experiment**. Learn more about Experiments for AWS FIS.

Exporting a scenario

Scenarios are a console-only experience. While similar to experiment templates, scenarios are not complete experiment templates and can not be directly imported into AWS FIS. If you wish to use scenarios as part of your own automation, you can use one of two paths:

1. Follow the steps in Using a scenario to create a valid AWS FIS experiment template and export that template.
2. Follow the steps in Viewing a scenario and in step 3, from the **Content** tab, copy and save the scenario content, then add missing parameters manually to create a valid experiment template.

Scenarios in the AWS FIS scenarios library

Scenarios included in the scenario library are designed to use tags where possible and each scenario describes the required tags in the **Prerequisites** and **How it works** sections of the scenario description. You can tag your resources with those pre-defined tags or you can set your own tags using the bulk parameter editing experience (see Using a scenario).

This reference describes the common scenarios in the AWS FIS scenario library. You can also list the supported scenarios using the AWS FIS console.

For more information, see Working with scenarios.

AWS FIS supports the following Amazon EC2 scenarios. These scenarios target instances using tags. You can use your own tags or use the default tags included in the scenario. Some of these scenarios use SSM documents.

- **EC2 stress: instance failure** - Explore the effect of instance failure by stopping one or more EC2 instances.

  Target instances in the current region that have a specific tag attached. In this scenario we will stop those instances and restart them at the end of the action duration, by default 5 min.

- **EC2 stress: Disk** - Explore impact of increased disk utilization on your EC2 based application.

  In this scenario we will target EC2 instances in the current region that have a specific tag attached. In this scenario you can customize an increasing amount disk utilization injected on targeted EC2 instances for the action duration, by default 5 min for each disk stress action.

- **EC2 stress: CPU** - Explore impact of increased CPU on your EC2 based application.

  In this scenario we will target EC2 instances in the current region that have a specific tag attached. In this scenario you can customize an increasing amount of CPU stress injected on targeted EC2 instances for the action duration, by default 5 min for each CPU stress action.

- **EC2 stress: Memory** - Explore impact of increased memory utilization on your EC2 based application.

  In this scenario we will target EC2 instances in the current region that have a specific tag attached. In this scenario you can customize an increasing amount of memory stress injected on targeted EC2 instances for the action duration, by default 5 min for each memory stress action.

- **EC2 stress: Network Latency** - Explore impact of increased network latency on your EC2 based application.

  In this scenario we will target EC2 instances in the current region that have a specific tag attached. In this scenario you can customize an increasing amount of network latency injected on targeted EC2 instances for the action duration, by default 5 min for each latency action.

AWS FIS supports the following Amazon EKS scenarios. These scenarios target EKS pods using a Kubernetes application labels. You can use your own labels or use the default labels included in the scenario. For more information about EKS with FIS, see Use the EKS pod actions.

- **EKS stress: Pod Delete** - Explore the effect of EKS pod failure by deleting one or more pods.

  In this scenario we will target pods in the current region that are associated with an application label. In this scenario we will terminate all matched pods. Re-creation of pods will be controlled by kubernetes configuration.

- **EKS stress: CPU** - Explore impact of increased CPU on your EKS based application.

  In this scenario we will target pods in the current region that are associated with an application label. In this scenario you can customize an increasing amount of CPU stress injected on targeted EKS pods for the action duration, by default 5 min for each CPU stress action.

- **EKS stress: Disk** - Explore impact of increased disk utilization on your EKS based application.

  In this scenario we will target pods in the current region that are associated with an application label. In this scenario you can customize an increasing amount of disk stress injected on targeted EKS pods for the action duration, by default 5 min for each CPU stress action.

- **EKS stress: Memory** - Explore impact of increased memory utilization on your EKS based application.

  In this scenario we will target pods in the current region that are associated with an application label. In this scenario you can customize an increasing amount of memory stress injected on targeted EKS pods for the action duration, by default 5 min for each memory stress action.

- **EKS stress: Network latency** - Explore impact of increased network latency on your EKS based application.

  In this scenario we will target pods in the current region that are associated with an application label. In this scenario you can customize an increasing amount of network latency injected on targeted EKS pods for the action duration, by default 5 min for each latency action.

AWS FIS supports the following scenarios for multi-AZ and multi-Region applications. These scenarios target multiple resource types.

- **AZ Availability: Power Interruption** - Inject the expected symptoms of a complete interruption of power in an Availability Zone (AZ). Learn more about AZ Availability: Power Interruption.
- **Cross-Region: Connectivity** - Block application network traffic from the experiment Region to the destination Region and pause cross-Region data replication. Learn more about using Cross-Region: Connectivity.

AZ Availability: Power Interruption

You can use the AZ Availability: Power Interruption scenario to induce the expected symptoms of a complete interruption of power in an Availability Zone (AZ).

This scenario can be used to demonstrate that multi-AZ applications operate as expected during a single, complete AZ power interruption. It includes loss of zonal compute (Amazon EC2, EKS, and ECS), no re-scaling of compute in the AZ, subnet connectivity loss, RDS failover, ElastiCache failover, and unresponsive EBS volumes. By default, actions for which no targets are found will be skipped.

Actions

Together, the following actions create many of the expected symptoms of a complete power interruption in a single AZ. AZ Availability: Power Interruption only affects services that are expected to see impact during a single AZ power interruption. By default, the scenario injects power interruption symptoms for 30 minutes and then, for an additional 30 minutes, injects symptoms that may occur during recovery.

Stop-Instances

During an AZ power interruption, EC2 instances in the affected AZ will shut down. After power is restored instances will reboot. AZ Availability: Power Interruption includes aws:ec2:stop-instances to stop all instances in the affected AZ for the interruption duration. After the duration, the instances are restarted. Stopping EC2 instances managed by Amazon EKS causes dependent EKS pods to be deleted. Stopping EC2 instances managed by Amazon ECS causes dependent ECS tasks to be stopped.

This action targets EC2 instances running in the affected AZ. By default, it targets instances with a tag named `AzImpairmentPower` with a value of `StopInstances`. You can add this tag to your instances or replace the default tag with your own tag in the experiment template. By default, if no valid instances are found this action will be skipped.

Stop-ASG-Instances

During an AZ power interruption, EC2 instances managed by an Auto Scaling group in the affected AZ will shut down. After power is restored instances will reboot. AZ Availability: Power Interruption includes aws:ec2:stop-instances to stop all instances, including those managed by Auto Scaling, in the affected AZ for the interruption duration. After the duration, the instances are restarted.

This action targets EC2 instances running in the affected AZ. By default, it targets instances with a tag named `AzImpairmentPower` with a value of `IceAsg`. You can add this tag to your instances or replace the default tag with your own tag in the experiment template. By default, if no valid instances are found this action will be skipped.

Pause Instance Launches

During an AZ power interruption, EC2 API calls to provision capacity in the AZ will fail. In particular, the following APIs will be impacted: `ec2:StartInstances`, `ec2:CreateFleet`, and `ec2:RunInstances`. AZ Availability: Power Interruption includes includes aws:ec2:api-insufficient-instance-capacity-error to prevent new instances from being provisioned in the affected AZ.

This action targets IAM roles used to provision instances. These must be targeted using an ARN. By default, if no valid IAM roles are found this action will be skipped.

Pause ASG Scaling

During an AZ power interruption, EC2 API calls made by the Auto Scaling control plane to recover lost capacity in the AZ will fail. In particular, the following APIs will be impacted: `ec2:StartInstances`, `ec2:CreateFleet`, and `ec2:RunInstances`. AZ Availability: Power Interruption includes aws:ec2:asg-insufficient-instance-capacity-error to prevent new instances from being provisioned in the affected AZ. This also prevents Amazon EKS and Amazon ECS from scaling in the affected AZ.

This action targets Auto Scaling groups. By default, it targets Auto Scaling groups with a tag named `AzImpairmentPower` with a value of `IceAsg`. You can add this tag to your Auto Scaling groups or replace the default tag with your own tag in the experiment template. By default, if no valid Auto Scaling groups are found this action will be skipped.

Pause Network Connectivity

During an AZ power interruption, networking in the AZ will be unavailable. When this happens some AWS services may take up to a few minutes to update DNS to reflect that private endpoints in the affected AZ are not available. During this time, DNS lookups may return inaccessible IP addresses. AZ Availability: Power Interruption includes aws:network:disrupt-connectivity to block all network connectivity for all subnets in the affected AZ for 2 minutes. This will force timeouts and DNS refreshes for most applications. Ending the action after 2 minutes allows for subsequent recovery of regional service DNS while the AZ continues to be unavailable.

This action targets subnets. By default, it targets clusters with a tag named `AzImpairmentPower` with a value of `DisruptSubnet`. You can add this tag to your subnets or replace the default tag with your own tag in the experiment template. By default, if no valid subnets are found this action will be skipped.

Failover RDS

During an AZ power interruption, RDS nodes in the affected AZ will shut down. Single AZ RDS nodes in the affected AZ will be fully unavailable. For multi-AZ clusters, the writer node will failover into an unaffected AZ and reader nodes in the affected AZ will be unavailable. For multi-AZ clusters, AZ Availability: Power Interruption includes aws:rds:failover-db-cluster to failover if the writer is in the affected AZ.

This action targets RDS clusters. By default, it targets clusters with a tag named `AzImpairmentPower` with a value of `DisruptRds`. You can add this tag to your clusters or replace the default tag with your own tag in the experiment template. By default, if no valid clusters are found this action will be skipped.

Pause ElastiCache Redis

During an AZ power interruption, ElastiCache nodes in the AZ are unavailable. AZ Availability: Power Interruption includes aws:elasticache:interrupt-cluster-az-power to terminate ElastiCache nodes in the affected AZ. For the duration of the interruption, new instances will not be provisioned in the affected AZ, so the cluster will remain at reduced capacity.

This action targets ElastiCache clusters. By default, it targets clusters with a tag named `AzImpairmentPower` with a value of `ElasticacheImpact`. You can add this tag to your clusters or replace the default tag with your own tag in the experiment template. By default, if no valid clusters are found this action will be skipped. Note that only clusters with writer nodes in the affected AZ will be considered valid targets.

Pause EBS I/O

After an AZ power interruption, once power is restored a very small percentage of instances may experience unresponsive EBS volumes. AZ Availability: Power Interruption includes aws:ebs:pause-io to leave 1 EBS volume in an unresponsive state.

By default, only volumes set to persist after the instance is terminated are targeted. This action targets volumes with a tag named `AzImpairmentPower` with a value of `APIPauseVolume`. You can add this tag to your volumes or replace the default tag with your own tag in the experiment template. By default, if no valid volumes are found this action will be skipped.

Limitations

- This scenario does not include stop conditions. The correct stop conditions for your application should be added to the experiment template.
- Amazon EKS Pods running on AWS Fargate are not supported.
- Amazon ECS tasks running on AWS Fargate are not supported.
- Amazon RDS Multi-AZ with two readable standby DB instances is not supported. In this case, the instances will be terminated, RDS will failover, and capacity will immediately be provisioned back in the affected AZ. The readable standby in the affected AZ will remain available.

Requirements

- Add the required permission to the AWS FIS experiment role.
- Resource tags must be applied to resources that are to be targeted by the experiment. These can use your own tagging convention or the default tags defined in the scenario.

Permissions

The following policy grants AWS FIS the necessary permissions to execute an experiment with the AZ Availability: Power Interruption scenario. This policy must be attached to the experiment role.

```
{
    "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": [
                "*"
            ]
        }
    ]
}
```

Scenario Content

The following content defines the scenario. This JSON can be saved and used to create an experiment template using the create-experiment-template command from the AWS Command

Line Interface (AWS CLI). For the most recent version of the scenario, visit the scenario library in the FIS console.

```
{
    "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:redis-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:interrupt-cluster-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."
}
```

Cross-Region: Connectivity

You can use the Cross-Region: Connectivity scenario to block application network traffic from the experiment Region to the destination Region and pause cross-Region replication for Amazon S3 and Amazon DynamoDB. Cross Region: Connectivity affects outbound application traffic from the Region in which you run the experiment (*experiment Region*). Stateless inbound traffic from the Region you wish to isolate from the *experiment region* (*destination Region*) may not be blocked. Traffic from AWS managed services may not be blocked.

This scenario can be used to demonstrate that multi-Region applications operate as expected when resources in the destination Region are not accessible from the experiment Region. It includes blocking network traffic from the experiment Region to the destination Region by targeting transit gateways and route tables. It also pauses cross-Region replication for S3 and DynamoDB. By default, actions for which no targets are found will be skipped.

Actions

Together, the following actions block cross-Region connectivity for the included AWS services. The actions are run in parallel. By default, the scenario blocks traffic for 3 hours, which you can increase up to a maximum 12 Hour duration.

Disrupt Transit Gateway Connectivity

Cross Region: Connectivity includes aws:network:transit-gateway-disrupt-cross-region-connectivity to block cross-Region network traffic from VPCs in the *experiment Region* to VPCs in the *destination Region* connected by a transit gateway. This does not affect access to VPC endpoints within the *experiment Region* but will block traffic from the *experiment Region* destined for a VPC endpoint in the *destination Region*.

This action targets transit gateways connecting the *experiment Region* and the *destination Region*. By default, it targets transit gateways with a tag named `DisruptTransitGateway` with a value of `Allowed`. You can add this tag to your transit gateways or replace the default tag with your own tag in the experiment template. By default, if no valid transit gateways are found this action will be skipped.

Disrupt Subnet Connectivity

Cross Region: Connectivity includes aws:network:route-table-disrupt-cross-region-connectivity to block cross-Region network traffic from VPCs in the *experiment Region* to public AWS IP

blocks in the *destination Region*. These public IP blocks include AWS service endpoints in the *destination Region*, e.g. the S3 regional endpoint, and AWS IP blocks for managed services, e.g. the IP addresses used for load balancers and Amazon API Gateway. This action also blocks network connectivity over cross-Region VPC Peering connections from the *experiment Region* to the *destination Region*. It does not affect access to VPC endpoints in the *experiment Region* but will block traffic from the *experiment Region* destined for a VPC endpoint in the *destination Region*.

This action targets subnets in the experiment Region. By default, it targets subnets with a tag named `DisruptSubnet` with a value of `Allowed`. You can add this tag to your subnets or replace the default tag with your own tag in the experiment template. By default, if no valid subnets are found this action will be skipped.

Pause S3 Replication

Cross Region: Connectivity includes aws:s3:bucket-pause-replication to pause S3 replication from the *experiment Region* to the *destination Region* for the targeted buckets. Replication from the *destination Region* to the *experiment Region* will be unaffected. After the scenario ends, bucket replication will resume from the point it was paused. Note that the time it takes for replication to keep all objects in sync will vary based on the duration of the experiment, and the rate of object upload to the bucket.

This action targets S3 buckets in the experiment Region with Cross-Region Replication (CRR) enabled to an S3 bucket in the destination Region. By default, it targets buckets with a tag named `DisruptS3` with a value of `Allowed`. You can add this tag to your buckets or replace the default tag with your own tag in the experiment template. By default, if no valid buckets are found this action will be skipped.

Pause DynamoDB Replication

Cross-Region: Connectivity includes aws:dynamodb:global-table-pause-replication to pause replication between the experiment Region and all other Regions, including the destination Region. This prevents replication into and out of the *experiment Region* but does not affect replication between other Regions. After the scenario ends, table replication will resume from the point it was paused. Note that the time it takes for replication to keep all data in sync will vary based on the duration of the experiment and the rate of changes to the table.

This action targets DynamoDB global tables in the experiment Region. By default, it targets tables with a tag named `DisruptDynamoDb` with a value of `Allowed`. You can add this tag to your tables

or replace the default tag with your own tag in the experiment template. By default, if no valid global tables are found this action will be skipped.

Limitations

- This scenario does not include stop conditions. The correct stop conditions for your application should be added to the experiment template.

Requirements

- Add the required permission to the AWS FIS experiment role.
- Resource tags must be applied to resources that are to be targeted by the experiment. These can use your own tagging convention or the default tags defined in the scenario.

Permissions

The following policy grants AWS FIS the necessary permissions to execute an experiment with the Cross-Region: Connectivity scenario. This policy must be attached to the experiment role.

```
{
    "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",
            "Resource": "arn:aws:ec2:*:*:prefix-list/*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/managedByFIS": "true"
                }
            }
        },
        {
            "Sid": "RouteTableDisruptConnectivity12",
            "Effect": "Allow",
            "Action": "ec2:ModifyManagedPrefixList",
            "Resource": "arn:aws:ec2:*:*:prefix-list/*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/managedByFIS": "true"
                }
            }
        },
        {
            "Sid": "RouteTableDisruptConnectivity13",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcs",
                "ec2:DescribeVpcPeeringConnections",
                "ec2:DescribeManagedPrefixLists",
```

```
                "ec2:DescribeSubnets",
                "ec2:DescribeRouteTables",
                "ec2:DescribeVpcEndpoints"
            ],
            "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": "RouteTableDisruptConnectivity20",
            "Effect": "Allow",
            "Action": "ec2:ModifyVpcEndpoint",
            "Resource": [
                "arn:aws:ec2:*:*:vpc-endpoint/*"
            ]
        },
        {
            "Sid": "TransitGatewayDisruptConnectivity1",
            "Effect": "Allow",
            "Action": [
                "ec2:DisassociateTransitGatewayRouteTable",
                "ec2:AssociateTransitGatewayRouteTable"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:transit-gateway-route-table/*",
                "arn:aws:ec2:*:*:transit-gateway-attachment/*"
            ]
        },
        {
            "Sid": "TransitGatewayDisruptConnectivity2",
            "Effect": "Allow",
            "Action": [
```

```
                "ec2:DescribeTransitGatewayPeeringAttachments",
                "ec2:DescribeTransitGatewayAttachments",
                "ec2:DescribeTransitGateways"
            ],
            "Resource": "*"
        },
        {
            "Sid": "S3CrossRegion1",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "S3CrossRegion2",
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "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": "DdbCrossRegion1",
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": "*"
        },
        {
            "Sid": "DdbCrossRegion2",
            "Effect": "Allow",
            "Action": [
                "dynamodb:DescribeTable",
                "dynamodb:DescribeGlobalTable"
            ],
            "Resource": [
                "arn:aws:dynamodb:*:*:table/*",
                "arn:aws:dynamodb:*:*:global-table/*"
            ]
        },
        {
            "Sid": "DdbCrossRegion3",
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey",
                "kms:GetKeyPolicy",
                "kms:PutKeyPolicy"
            ],
            "Resource": "arn:aws:kms:*:*:key/*"
        }
    ]
}
```

Scenario Content

The following content defines the scenario. This JSON can be saved and used to create an experiment template using the create-experiment-template command from the AWS Command

Line Interface (AWS CLI). For the most recent version of the scenario, visit the scenario library in the FIS console.

```
{
        "targets": {
                "Transit-Gateway": {
                        "resourceType": "aws:ec2:transit-gateway",
                        "resourceTags": {
                                "TgwTag": "TgwValue"
                        },
                        "selectionMode": "ALL"
                },
                "Subnet": {
                        "resourceType": "aws:ec2:subnet",
                        "resourceTags": {
                                "SubnetKey": "SubnetValue"
                        },
                        "selectionMode": "ALL",
                        "parameters": {}
                },
                "S3-Bucket": {
                        "resourceType": "aws:s3:bucket",
                        "resourceTags": {
                                "S3Impact": "Allowed"
                        },
                        "selectionMode": "ALL"
                },
                "DynamoDB-Global-Table": {
                        "resourceType": "aws:dynamodb:encrypted-global-table",
                        "resourceTags": {
                                "DisruptDynamoDb": "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"
                        }
                },
                "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:encrypted-global-table-pause-
replication",
                        "parameters": {
                                "duration": "PT3H"
                        },
                        "targets": {
                                "Tables": "DynamoDB-Global-Table"
                        }
                }
        },
        "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"
}
```

Experiments for AWS FIS

AWS FIS enables you to perform fault injection experiments on your AWS workloads. To get started, create an experiment template. After you create an experiment template, you can use it to start an experiment.

An experiment is finished when one of the following occurs:

- All actions in the template completed successfully.
- A stop condition is triggered.
- An action cannot be completed because of an error. For example, if the target cannot be found.
- The experiment is stopped manually.

You cannot resume a stopped or failed experiment. You also cannot rerun a completed experiment. However, you can start a new experiment from the same experiment template. You can optionally update the experiment template before you specify it again in a new experiment.

Tasks

- Start an experiment
- View your experiments
- Tag an experiment
- Stop an experiment
- List resolved targets

Start an experiment

You start an experiment from an experiment template. For more information, see Start an experiment from a template.

You can schedule your experiments as a one-time task or recurring tasks using Amazon EventBridge. For more information, see Tutorial: Schedule a recurring experiment.

You can monitor your experiment using any of the following features:

- View your experiments in the AWS FIS console. For more information, see View your experiments.

- View Amazon CloudWatch metrics for the target resources in your experiments or view AWS FIS usage metrics. For more information, see Monitor using CloudWatch.
- Enable experiment logging to capture detailed information about your experiment as it runs. For more information see Experiment logging.

View your experiments

You can view the progress of a running experiment, and you can view experiments that have completed, stopped, or failed.

Stopped, completed, and failed experiments are automatically removed from your account after 120 days.

To view experiments using the console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiments**.
3. Choose the **Experiment ID** of the experiment to open its details page.
4. Do one or more of the following:
   - Check **Details**, **State** for the state of the experiment.
   - Choose the **Actions** tab for information about the experiment actions.
   - Choose the **Targets** tab for information about the experiment targets.
   - Choose the **Timeline** tab for a visual representation of the actions based on their start and end times.

To view experiments using the CLI

Use the list-experiments command to get a list of experiments, and use the get-experiment command to get information about a specific experiment.

Experiment states

An experiment can be in one of the following states:

- **pending** – The experiment is pending.
- **initiating** – The experiment is preparing to start.

- **running** – The experiment is running.
- **completed** – All actions in the experiment completed successfully.
- **stopping** – The stop condition was triggered or the experiment was stopped manually.
- **stopped** – All running or pending actions in the experiment are stopped.
- **failed** – The experiment failed due to an error, such as insufficient permissions or incorrect syntax.

Action states

An action can be in one of the following states:

- **pending** – The action is pending, either because the experiment hasn't started or the action is to start later in the experiment.
- **initiating** – The action is preparing to start.
- **running** – The action is running.
- **completed** – The action completed successfully.
- **cancelled** – The experiment stopped before the action started.
- **skipped** – The action has been skipped.
- **stopping** – The action is stopping.
- **stopped** – All running or pending actions in the experiment are stopped.
- **failed** – The action failed due to a client error, such as insufficient permissions or incorrect syntax.

Tag an experiment

You can apply tags to experiments to help you organize them. You can also implement tag-based IAM policies to control access to experiments.

To tag an experiment using the console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiments**.
3. Select the experiment and choose **Actions**, **Manage tags**.
4. To add a new tag, choose **Add new tag**, and specify a key and value.

To remove a tag, choose **Remove** for the tag.

5. Choose **Save**.

To tag an experiment using the CLI

Use the tag-resource command.

Stop an experiment

You can stop a running experiment at any time. When you stop an experiment, any post actions that have not been completed for an action are completed before the experiment stops. You cannot resume a stopped experiment.

To stop an experiment using the console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiments**.
3. Select the experiment, and choose **Stop experiment**.
4. In the confirmation dialog box, choose **Stop experiment**.

To stop an experiment using the CLI

Use the stop-experiment command.

List resolved targets

You can view information for resolved targets for an experiment after target resolution has ended.

To view resolved targets using the console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiments**.
3. Select the experiment, and choose **Report**.
4. View resolved target information under **Resources**.

To view resolved targets using the CLI

Use the list-experiment-resolved-targets command.

Experiment scheduler

With AWS Fault Injection Service (FIS), you can perform fault injection experiments on your AWS workloads. These experiments run on templates that contain one or more actions to run on specified targets. You can now schedule your experiments as a one-time task or recurring tasks natively from the FIS Console. In addition to scheduled rules, FIS now offers a new scheduling capability. FIS now integrates with EventBridge Scheduler and creates rules on your behalf. EventBridge Scheduler is a serverless scheduler that allows you to create, run, and manage tasks from one central, managed service.

> **Important**
>
> Experiment Scheduler with AWS Fault Injection Service is not available in AWS GovCloud (US-East) and AWS GovCloud (US-West).

Topics

Getting started

An execution role is an IAM role that AWS Fault Injection Service assumes in order to interact with EventBridge scheduler and for Event Bridge scheduler to Start FIS Experiment. You attach permission policies to this role to grant EventBridge Scheduler access to invoke FIS Experiment. The following steps describe how to create a new execution role and a policy to allow EventBridge to Start an Experiment.

Create scheduler role using the AWS CLI

This is IAM role that is needed for Event Bridge to be able to schedule experiment on behalf of the customer.

1. Copy the following assume role JSON policy and save it locally as `fis-execution-role.json`. This trust policy allows EventBridge Scheduler to assume the role on your behalf.

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

2. From the AWS Command Line Interface (AWS CLI), enter the following command to create a new role. Replace `FisSchedulerExecutionRole` with the name you want to give this role.

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

If successful, you'll see the following output:

```
{
    "Role": {
        "Path": "/",
        "RoleName": "FisSchedulerExecutionRole",
        "RoleId": "AROAZL22PDN5A6WKRBQNU",
        "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"
                }
```

```
            ]
        }
    }
}
```

3. To create a new policy that allows EventBridge Scheduler to invoke the experiment, copy the following JSON and save it locally as `fis-start-experiment-permissions.json`. The following policy allows EventBridge Scheduler to call the `fis:StartExperiment` action on all experiment templates in your account. Replace the `*` at the end of `"arn:aws:fis:*:*:experiment-template/*"` with the ID of your experiment template if you want to limit the role to a single experiment template.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "fis:StartExperiment",
            "Resource": [
                "arn:aws:fis:*:*:experiment-template/*",
                "arn:aws:fis:*:*:experiment/*"
            ]
        }
    ]
}
```

4. Run the following command to create the new permission policy. Replace `FisSchedulerPolicy` with the name you want to give this policy.

```
aws iam create-policy --policy-name FisSchedulerPolicy --policy-document file://fis-
start-experiment-permissions.json
```

If successful, you'll see the following output. Note the policy ARN. You use this ARN in the next step to attach the policy to our execution role.

```
{
    "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"
    }
}
```

5. Run the following command to attach the policy to your execution role. Replace `your-policy-arn` with the ARN of the policy you created in the previous step. Replace `FisSchedulerExecutionRole` with the name of your execution role.

```
aws iam attach-role-policy --policy-arn your-policy-arn --role-name
 FisSchedulerExecutionRole
```

   The `attach-role-policy` operation doesn't return a response on the command line.

6. You can restrict the scheduler to only run AWS FIS experiments that have a specific tag value. For example, the following policy grants the `fis:StartExperiment` permission for all AWS FIS experiment templates, but restricts the scheduler to only run experiments that are tagged `Purpose=Schedule`.

```
{
    "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"
                }
```

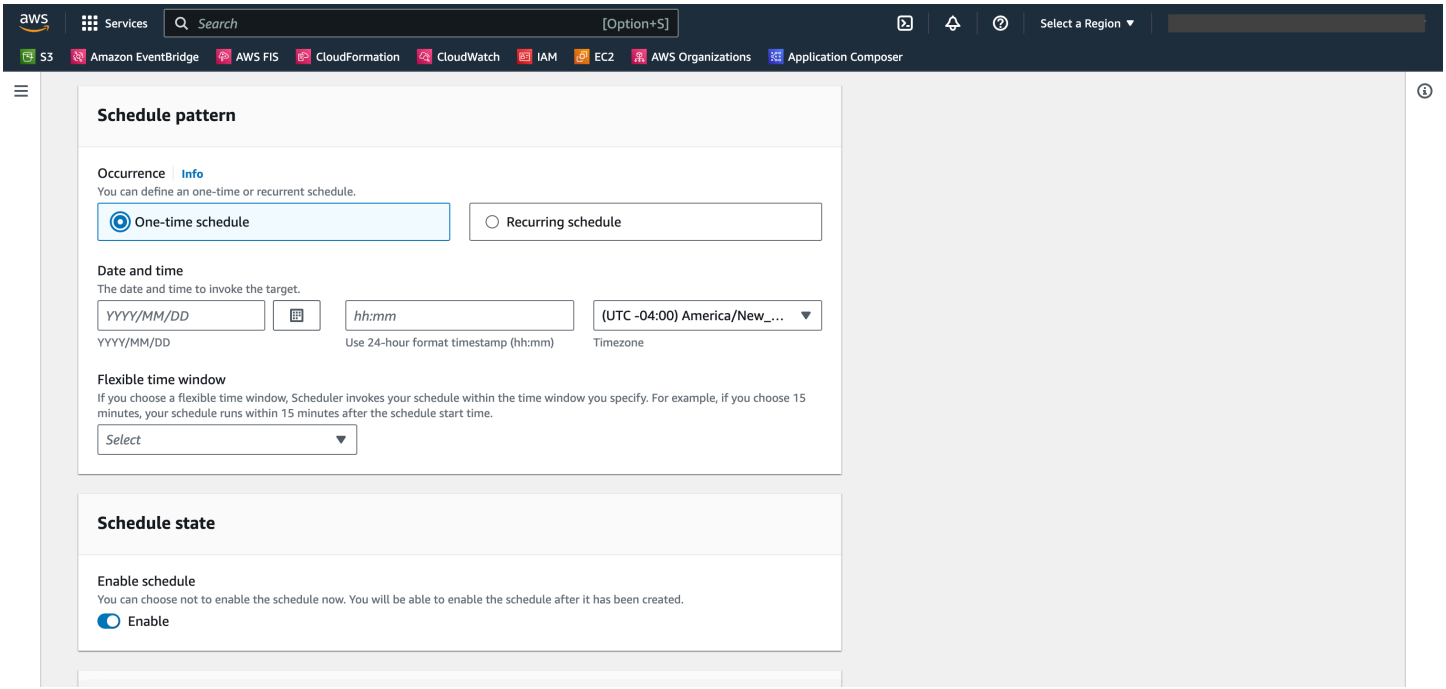
```
            }
        }
    ]
}
```

Schedule an FIS experiment

Before you schedule an experiment, you need one or more Experiment templates for your schedule to invoke. You can use an existing AWS resource, or create a new one.

Once experiment template is created, click on **Actions** and select **Schedule experiment**. You will be redirected to schedule experiment page. The name of the schedule will be filled in for you.

Follow to the schedule pattern section and choose either one-time schedule or recurring. Fill in required input fields and navigate to permissions.



Schedule state will be enabled by default. Note: if you disable **schedule state**, the experiment will not be scheduled even if you create a schedule.

AWS FIS Experiment Scheduler is built on top of EventBridge Scheduler. You can refer the documentation for the various schedule types supported.

To update schedule using the console

1. Open the AWS FIS console.
2. In the left navigation pane, choose **Experiment Templates** .
3. Choose **Experiment Template** for which you want to create the schedule.
4. Click **Actions**, and select **Schedule Experiment** from the dropdown.
   a. Under **Schedule name**, name is auto populated.
   b. Under **Schedule pattern**, select **Recurring schedule**.
   c. Under **Schedule type**, you can select a **Rate-based schedule**, see schedule types .
   d. Under **Rate expression**, choose a rate that is slower than the execution time of your experiment, e.g. **5 minutes**.
   e. Under **Timeframe**, select your **Time Zone** .
   f. Under **Start Date and Time**, specify a start date and time.
   g. Under **End Date and Time**, specify an end date and time
   h. Under **Schedule State**, toggle the **Enable Schedule Option**.
   i. Under **Permissions**, select **Use existing role**, and then search for `FisSchedulerExecutionRole`.
   j. Choose **Next**.
5. Select **Review and create schedule**, review your scheduler details, and then choose **Create schedule**.

Updating the Experiment Schedule

You can update an experiment schedule so that it occurs at a specific date and time that suits you.

To update an experiment execution using the console

1. Open the Amazon FIS console.
2. In the navigation pane, choose **Experiment Templates**.
3. Choose **Resource type: Experiment Template** for which a schedule is already created.
4. Click on the Experiment ID for the template. Then navigate to schedules Tab.
5. Check if there is a existing schedule associated with the experiment. Select the schedule associated and Click the button **Update Schedule**.

Disable or Delete an Experiment Execution using the console

To stop an experiment from executing or running on a schedule, you can delete or disable the rule. The following steps walk you through how to delete or disable an Experiment Execution.

To delete or disable a rule

1. Open the Amazon FIS console.
2. In the navigation pane, choose **Experiment Templates**.
3. Choose **Resource type: Experiment Template** for which a schedule is already created.
4. Click on the Experiment ID for the template. Then navigate to schedules Tab.
5. Check if there is a existing schedule associated with the experiment. Select the schedule associated and Click the button **Update Schedule**.
6. Do one of the following:
   a. To delete the schedule, select the button next to the rule **Delete Schedule**. Type `delete` and click the **Delete Schedule** button.
   b. To disable the schedule, select the button next to the rule **Disable Schedule**. Type `disable` and click the **Disable Schedule** button.

Monitoring AWS FIS

You can use the following tools to monitor the progress and impact of your AWS Fault Injection Service (AWS FIS) experiments.

AWS FIS console and AWS CLI

Use the AWS FIS console or the AWS CLI to monitor the progress of a running experiment. You can view the status of each action in the experiment, and the results of each action. For more information, see the section called “View your experiments”.

CloudWatch usage metrics and alarms

Use CloudWatch usage metrics to provide visibility into your account's usage of resources. AWS FIS usage metrics correspond to AWS service quotas. You can configure alarms that alert you when your usage approaches a service quota. For more information, see Monitor using CloudWatch.

You can also create stop conditions for your AWS FIS experiments by creating CloudWatch alarms that define when an experiment goes out of bounds. When the alarm is triggered, the experiment stops. For more information, see Stop conditions. For more information about creating CloudWatch alarms, see Create a CloudWatch Alarm Based on a Static Threshold and Creating a CloudWatch Alarm Based on Anomaly Detection in the *Amazon CloudWatch User Guide*.

AWS FIS experiment logging

Enable experiment logging to capture detailed information about your experiment as it runs. For more information see Experiment logging.

Experiment state change events

Amazon EventBridge enables you to respond automatically to system events or resource changes. AWS FIS emits a notification when the state of an experiment changes. You can create rules for the events that you are interested in that specify the automated action to take when an event matches a rule. For example, sending a notification to an Amazon SNS topic or invoking a Lambda function. For more information, see Monitor using EventBridge.

CloudTrail logs

Use AWS CloudTrail to capture detailed information about the calls made to the AWS FIS API and store them as log files in Amazon S3. CloudTrail also logs calls made to service APIs for the

resources on which you're running experiments. You can use these CloudTrail logs to determine which calls were made, the source IP address where the call came from, who made the call, when the call was made, and so on.

AWS Health Dashboard Notifications

AWS Health provides ongoing visibility into your resource performance and the availability of your AWS services and accounts. When you start an experiment, AWS FIS emits a notification to your AWS Health Dashboard. The notification is present for the duration of the experiment in each account that contains resources targeted in an experiment, including multi-account experiments. Multi-account experiments with only actions that do not include targets, such as `aws:ssm:start-automation-execution` and `aws:fis:wait`, will not emit a notification. Information about the role used to allow the experiment will be listed under **Affected resources**. To learn more about the AWS Health Dashboard, see AWS Health Dashboard in the AWS Health User Guide.

> **Note**
>
> AWS Health delivers events on a *best effort* basis.

Monitor AWS FIS usage metrics using Amazon CloudWatch

You can use Amazon CloudWatch to monitor the impact of AWS FIS experiments on targets. You can also monitor your AWS FIS usage.

For more information about viewing the state of an experiment, see View your experiments.

Monitor AWS FIS experiments

As you plan your AWS FIS experiments, identify the CloudWatch metrics that you can use to identify the baseline or "steady state" for the target resource types for the experiment. After you start an experiment, you can monitor those CloudWatch metrics for the targets selected through the experiment template.

For more information about the available CloudWatch metrics for a target resource type supported by AWS FIS, see the following:

- Monitor your instances using CloudWatch
- Amazon ECS CloudWatch metrics

- Monitoring Amazon RDS metrics using CloudWatch
- Monitoring Run Command metrics using CloudWatch

AWS FIS usage metrics

You can use CloudWatch usage metrics to provide visibility into your account's usage of resources. Use these metrics to visualize your current service usage on CloudWatch graphs and dashboards.

AWS FIS usage metrics correspond to AWS service quotas. You can configure alarms that alert you when your usage approaches a service quota. For more information about CloudWatch alarms, see the Amazon CloudWatch User Guide.

AWS FIS publishes the following metric in the **AWS/Usage** namespace.

Metric	Description
`ResourceCount`	The total number of the specified resource running on your account. The resource is defined by the dimensions associated with the metric.

The following dimensions are used to refine the usage metrics that are published by AWS FIS.

Dimension	Description
`Service`	The name of the AWS service containing the resource. For AWS FIS usage metrics, the value for this dimension is `FIS`.
`Type`	The type of entity that is being reported. Currently, the only valid value for AWS FIS usage metrics is `Resource`.
`Resource`	The type of resource that is running. The possible values are `ExperimentTemplates` for experiment templates, and `ActiveExperiments` for active experiments.

Dimension	Description
Class	This dimension is reserved for future use.

Monitor AWS FIS experiments using Amazon EventBridge

When the state of an experiment changes, AWS FIS emits a notification. These notifications are made available as events through Amazon EventBridge (formerly called CloudWatch Events). AWS FIS emits these events on a best effort basis. Events are delivered to EventBridge in near real time.

With EventBridge, you can create rules that trigger programmatic actions in response to an event. For example, you can configure a rule that invokes an SNS topic to send an email notification or invokes a Lambda function to take some action.

For more information about EventBridge, see Getting started with Amazon EventBridge in the *Amazon EventBridge User Guide*.

The following is the syntax of an experiment state change event:

```
{
    "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`

The ID of the experiment whose state changed.

`experiment-template-id`

The ID of the experiment template used by the experiment.

`new_value`

The new state of the experiment. The possible values are:

- `completed`
- `failed`
- `initiating`
- `running`
- `stopped`
- `stopping`

`old_value`

The previous state of the experiment. The possible values are:

- `initiating`
- `pending`
- `running`
- `stopping`

Experiment logging for AWS FIS

You can use experiment logging to capture detailed information about your experiment as it runs.

You are charged for experiment logging based on the costs associated with each log destination type. For more information, see Amazon CloudWatch Pricing (under **Paid Tier**, **Logs**, **Vended Logs**) and Amazon S3 Pricing.

Permissions

You must grant AWS FIS permissions to send logs to each log destination that you configure. For more information, see the following in the *Amazon CloudWatch Logs User Guide*:

- Logs sent to CloudWatch Logs
- Logs sent to Amazon S3

Log schema

The following is the schema used in experiment logging. The current schema version is 2. The fields for `details` depend on the value of `log_type`. The fields for `resolved_targets` depend on the value of `target_type`. For more information, see the section called "Example log records".

```
{
    "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

    }
}
```

Release notes

- Version 2 introduces:
  - The `target_type` field and changes the `resolved_targets` field from a list of ARNs to a list of objects. The valid fields for the `resolved_targets` object depend on the value of `target_type`, which is the resource type of the targets.
  - The `action-error` and `target-resolution-detail` event types which add the `account_id` field.
- Version 1 is the initial release.

Log destinations

AWS FIS supports log delivery to the following destinations:

- An Amazon S3 bucket
- An Amazon CloudWatch Logs log group

S3 log delivery

The logs are delivered to the following location.

```
bucket-and-optional-prefix/AWSLogs/account-id/fis/region/experiment-
id/YYYY/MM/DD/account-id_awsfislogs_region_experiment-id_YYYYMMDDHHMMZ_hash.log
```

It can take several minutes before the logs are delivered to the bucket.

CloudWatch Logs log delivery

The logs are delivered to a log stream named /aws/fis/*experiment-id*.

Logs are delivered to the log group in less than one minute.

Example log records

The following are example log records for an experiment that runs the **aws:ec2:reboot-instances** action on an EC2 instance selected at random.

Records

- experiment-start
- target-resolution-start
- target-resolution-detail
- target-resolution-end
- action-start
- action-end
- action-error
- experiment-end

experiment-start

The following is an example record for the `experiment-start` event.

```
{
    "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

The following is an example record for the `target-resolution-start` event.

```
{
    "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

The following is an example record for the `target-resolution-detail` event. If target resolution fails, the record also includes the `error_information` field.

```
{
    "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

If target resolution fails, the record also includes the `error_information` field. If `total_pages` is greater than 1, the number of resolved targets exceeded the size limit for one record. There are additional `target-resolution-end` records that contain the remaining resolved targets.

The following is example record for the `target-resolution-end` event for an EC2 action.

```
{
    "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
    }
}
```

The following is example record for the `target-resolution-end` event for an EKS action.

```
{
    "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

The following is an example record for the `action-start` event. If the experiment template specifies parameters for the action, the record also includes the `parameters` field.

```
{
    "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

The following is an example record for the `action-error` event. This event is only returned when an action fails. It is returned for each account where the action fails.

```
{
    "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

The following is an example record for the `action-end` event.

```
{
    "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

The following is an example record for the `experiment-end` event.

```
{
    "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"
        }
    }
```

```
}
```

Enable experiment logging

Experiment logging is disabled by default. To receive experiment logs for an experiment, you must create the experiment from an experiment template with logging enabled. The first time that you run an experiment that is configured to use a destination that hasn't been used previously for logging, we delay the experiment to configure log delivery to this destination, which takes about 15 seconds.

To enable experiment logging using the console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. Select the experiment template, and choose **Actions**, **Update experiment template**.
4. For **Logs**, configure the destination options. To send logs to an S3 bucket, choose **Send to an Amazon S3 bucket** and enter the bucket name and prefix. To send logs to CloudWatch Logs, choose **Send to CloudWatch Logs** and enter the log group.
5. Choose **Update experiment template**.

To enable experiment logging using the AWS CLI

Use the update-experiment-template command and specify a log configuration.

Disable experiment logging

If you no longer want to receive logs for your experiments, you can disable experiment logging.

To disable experiment logging using the console

1. Open the AWS FIS console at https://console.aws.amazon.com/fis/.
2. In the navigation pane, choose **Experiment templates**.
3. Select the experiment template, and choose **Actions**, **Update experiment template**.
4. For **Logs**, clear **Send to an Amazon S3 bucket** and **Send to CloudWatch Logs**.
5. Choose **Update experiment template**.

To disable experiment logging using the AWS CLI

Use the update-experiment-template command and specify an empty log configuration.

Log API calls with AWS CloudTrail

AWS Fault Injection Service (AWS FIS) is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, a role, or an AWS service in AWS FIS. CloudTrail captures all API calls for AWS FIS as events. The calls that are captured include calls from the AWS FIS console and code calls to the AWS FIS API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS FIS. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to AWS FIS, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Use CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS FIS, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for AWS FIS, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Create a Trail for Your AWS Account
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

All AWS FIS actions are logged by CloudTrail and are documented in the AWS Fault Injection Service API Reference. For the experiment actions that are carried out on a target resource, view

the API reference documentation for the service that owns the resource. For example, for actions that are carried out on an Amazon EC2 instance, see the Amazon EC2 API Reference.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element.

Understand AWS FIS log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following is an example CloudTrail log entry for a call to the AWS FIS `StopExperiment` action.


```
{
  "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"

}
```

The following is an example CloudTrail log entry for an API action that AWS FIS invoked as part of an experiment that includes the `aws:ssm:send-command` AWS FIS action. The `userIdentity` element reflects a request made with temporary credentials obtained by assuming a role. The name of the assumed role appears in `userName`. The ID of the experiment, EXP21nT17WMzA6dnUgz, appears in `principalId` and as part of the ARN of the assumed role.


```
{
    "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"
}
```

Security in AWS Fault Injection Service

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to AWS Fault Injection Service, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using AWS FIS. The following topics show you how to configure AWS FIS to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS FIS resources.

Contents

Data protection in AWS Fault Injection Service

The AWS shared responsibility model applies to data protection in AWS Fault Injection Service. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks

for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with AWS FIS or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Encryption at rest

AWS FIS always encrypts your data at rest. Data in AWS FIS is encrypted at rest using transparent server-side encryption. This helps reduce the operational burden and complexity involved in protecting sensitive data. With encryption at rest, you can build security-sensitive applications that meet encryption compliance and regulatory requirements.

Encryption in transit

AWS FIS encrypts data in transit between the service and other integrated AWS services. All data that passes between AWS FIS and integrated services is encrypted using Transport Layer Security (TLS). For more information about other integrated AWS services, see Supported AWS services.

Identity and access management for AWS Fault Injection Service

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use AWS FIS resources. IAM is an AWS service that you can use with no additional charge.

Contents

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in AWS FIS.

Service user – If you use the AWS FIS service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS FIS features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator.

Service administrator – If you're in charge of AWS FIS resources at your company, you probably have full access to AWS FIS. It's your job to determine which AWS FIS features and resources your

service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS FIS.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing AWS API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center User Guide* and Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account.

We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

- **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
- **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A

user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS Fault Injection Service works with IAM

Before you use IAM to manage access to AWS FIS, learn what IAM features are available to use with AWS FIS.

IAM features you can use with AWS Fault Injection Service

IAM feature	AWS FIS support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Yes
Temporary credentials	Yes
Principal permissions	Yes
Service roles	Yes
Service-linked roles	Yes

To get a high-level view of how AWS FIS and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for AWS FIS

Supports identity-based policies	Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for AWS FIS

To view examples of AWS FIS identity-based policies, see AWS Fault Injection Service policy examples.

Resource-based policies within AWS FIS

Supports resource-based policies	No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for AWS FIS

Supports policy actions	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of AWS FIS actions, see Actions defined by AWS Fault Injection Service in the *Service Authorization Reference*.

Policy actions in AWS FIS use the following prefix before the action:

```
fis
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
    "fis:action1",
    "fis:action2"
]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `List`, include the following action:

```
"Action": "fis:List*"
```

Policy resources for AWS FIS

Supports policy resources	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice,

specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

Some AWS FIS API actions support multiple resources. To specify multiple resources in a single statement, separate the ARNs with commas.

```
"Resource": [
    "resource1",
    "resource2"
]
```

To see a list of AWS FIS resource types and their ARNs, see Resource types defined by AWS Fault Injection Service in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by AWS Fault Injection Service.

Policy condition keys for AWS FIS

Supports service-specific policy condition keys	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of AWS FIS condition keys, see Condition keys for AWS Fault Injection Service in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by AWS Fault Injection Service.

To view examples of AWS FIS identity-based policies, see AWS Fault Injection Service policy examples.

ACLs in AWS FIS

Supports ACLs	No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with AWS FIS

Supports ABAC (tags in policies)	Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see What is ABAC? in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

To view an example identity-based policy for limiting access to a resource based on the tags for that resource, see Example: Use tags to control resource usage.

Using temporary credentials with AWS FIS

Supports temporary credentials	Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for AWS FIS

Supports forward access sessions (FAS)	Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a

different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

Service roles for AWS FIS

Supports service roles	Yes

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

Service-linked roles for AWS FIS

Supports service-linked roles	Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing AWS FIS service-linked roles, see Use service-linked roles for AWS Fault Injection Service.

AWS Fault Injection Service policy examples

By default, users and roles don't have permission to create or modify AWS FIS resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

For details about actions and resource types defined by AWS FIS, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for AWS Fault Injection Service in the *Service Authorization Reference*.

Contents

Policy best practices

Identity-based policies determine whether someone can create, access, or delete AWS FIS resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to

specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.

- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see IAM Access Analyzer policy validation in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Example: Use the AWS FIS console

To access the AWS Fault Injection Service console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the AWS FIS resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

The following example policy grants permission to list and view all AWS FIS resources using AWS FIS console, but not to create, update, or delete them. It also grants permissions to view the available resources used by all AWS FIS actions that you could specify in an experiment template.

```
{
    "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"
                }
            }
        }
    ]
}
```

Example: List available AWS FIS actions

The following policy grants permission to list the available AWS FIS actions.

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

```
        {
            "Effect": "Allow",
            "Action": [
                "fis:ListActions"
            ],
            "Resource": "arn:aws:fis:*:*:action/*"
        }
    ]
}
```

Example: Create an experiment template for a specific action

The following policy grants permission to create an experiment template for the action `aws:ec2:stop-instances`.

```
{
  "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::account-id:role/role-name"
      ]
    }
  ]
}
```

Example: Start an experiment

The following policy grants permission to start an experiment using the specified IAM role and experiment template. It also allows AWS FIS to create a service-linked role on the user's behalf. For more information, see Use service-linked roles for AWS Fault Injection Service.

```
{
  "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"
            }
        }
    }
  ]
}
```

Example: Use tags to control resource usage

The following policy grants permission to run experiments from experiment templates that have the tag `Purpose=Test`. It does not grant permission to create or modify experiment templates, or run experiments using templates that do not have the specified tag.

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "fis:StartExperiment",
            "Resource": "arn:aws:fis:*:*:experiment-template/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Purpose": "Test"
                }
            }
        }
    ]
}
```

Example: Delete an experiment template with a specific tag

The following policy grants permission to delete an experiment template with tag `Purpose=Test`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "fis:DeleteExperimentTemplate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Purpose": "Test"
                }
            }
        }
    ]
}
```

Example: Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "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": "*"
        }
    ]
}
```

Example: Use condition keys for `ec2:InjectApiError`

The following example policy uses the `ec2:FisTargetArns` condition key to scope target resources. This policy allows the AWS FIS actions `aws:ec2:api-insufficient-instance-capacity-error` and `aws:ec2:asg-insufficient-instance-capacity-error`.

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

```
        {
            "Effect": "Allow",
            "Action": "ec2:InjectApiError",
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "ec2:FisActionId": [
                        "aws:ec2:api-insufficient-instance-capacity-error",
                    ],
                    "ec2:FisTargetArns": [
                        "arn:aws:iam:*:*:role:role-name"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:InjectApiError",
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "ec2:FisActionId": [
                        "aws:ec2:asg-insufficient-instance-capacity-error"
                    ],
                    "ec2:FisTargetArns": [

 "arn:aws:autoscaling:*:*:autoScalingGroup:uuid:autoScalingGroupName/asg-name"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:DescribeAutoScalingGroups",
            "Resource": "*"
        }
    ]
}
```

Example: Use condition keys for `aws:s3:bucket-pause-replication`

The following example policy uses the `S3:IsReplicationPauseRequest` condition key to allow `PutReplicationConfiguration` and `GetReplicationConfiguration` only when used by AWS FIS in the context of the AWS FIS action `aws:s3:bucket-pause-replication`.

```
{
    "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 service-linked roles for AWS Fault Injection Service

AWS Fault Injection Service uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to AWS FIS. Service-linked roles are predefined by AWS FIS and include all of the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up AWS FIS easier because you don't have to manually add the necessary permissions to manage monitoring and resource selection for experiments. AWS FIS defines the permissions of its service-linked roles, and unless defined otherwise, only AWS FIS can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

In addition to the service-linked role, you must also specify an IAM role that grants permission to modify the resources that you specify as targets in an experiment template. For more information, see IAM roles for AWS FIS experiments.

You can delete a service-linked role only after first deleting the related resources. This protects your AWS FIS resources because you can't inadvertently remove permission to access the resources.

Service-linked role permissions for AWS FIS

AWS FIS uses the service-linked role named **AWSServiceRoleForFIS** to enable it to manage monitoring and resource selection for experiments.

The **AWSServiceRoleForFIS** service-linked role trusts the following services to assume the role:

- `fis.amazonaws.com`

The **AWSServiceRoleForFIS** service-linked role uses the managed policy **AmazonFISServiceRolePolicy**. This policy enables AWS FIS to manage monitoring and resource

selection for experiments. For more information, see AmazonFISServiceRolePolicy in the *AWS Managed Policy Reference*.

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For the **AWSServiceRoleForFIS** service-linked role to be successfully created, the IAM identity that you use AWS FIS with must have the required permissions. To grant the required permissions, attach the following policy to the IAM identity.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "fis.amazonaws.com"
                }
            }
        }
    ]
}
```

For more information, see Service-linked role permissions in the *IAM User Guide*.

Create a service-linked role for AWS FIS

You don't need to manually create a service-linked role. When you start an AWS FIS experiment in the AWS Management Console, the AWS CLI, or the AWS API, AWS FIS creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you start an AWS FIS experiment, AWS FIS creates the service-linked role for you again.

Edit a service-linked role for AWS FIS

AWS FIS does not allow you to edit the **AWSServiceRoleForFIS** service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

Delete a service-linked role for AWS FIS

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

> **Note**
>
> If the AWS FIS service is using the role when you try to clean up the resources, then the cleanup might fail. If that happens, wait for a few minutes and try the operation again.

To clean up AWS FIS resources used by the AWSServiceRoleForFIS

Make sure that none of your experiments are currently running. If necessary, stop your experiments. For more information, see Stop an experiment.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the **AWSServiceRoleForFIS** service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Supported Regions for AWS FIS service-linked roles

AWS FIS supports using service-linked roles in all of the Regions where the service is available. For more information, see AWS Fault Injection Service endpoints and quotas.

AWS managed policies for AWS Fault Injection Service

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users,

groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AmazonFISServiceRolePolicy

This policy is attached to the service-linked role named **AWSServiceRoleForFIS** to allow AWS FIS to manage monitoring and resource selection for experiments. For more information, see Use service-linked roles for AWS Fault Injection Service.

AWS managed policy: AWSFaultInjectionSimulatorEC2Access

Use this policy in an experiment role to grant AWS FIS permission to run experiments that use the AWS FIS actions for Amazon EC2. For more information, see the section called “Experiment role”.

To view the permissions for this policy, see AWSFaultInjectionSimulatorEC2Access in the *AWS Managed Policy Reference*.

AWS managed policy: AWSFaultInjectionSimulatorECSAccess

Use this policy in an experiment role to grant AWS FIS permission to run experiments that use the AWS FIS actions for Amazon ECS. For more information, see the section called “Experiment role”.

To view the permissions for this policy, see AWSFaultInjectionSimulatorECSAccess in the *AWS Managed Policy Reference*.

AWS managed policy: AWSFaultInjectionSimulatorEKSAccess

Use this policy in an experiment role to grant AWS FIS permission to run experiments that use the AWS FIS actions for Amazon EKS. For more information, see the section called “Experiment role”.

To view the permissions for this policy, see AWSFaultInjectionSimulatorEKSAccess in the *AWS Managed Policy Reference*.

AWS managed policy: AWSFaultInjectionSimulatorNetworkAccess

Use this policy in an experiment role to grant AWS FIS permission to run experiments that use the AWS FIS networking actions. For more information, see the section called “Experiment role”.

To view the permissions for this policy, see AWSFaultInjectionSimulatorNetworkAccess in the *AWS Managed Policy Reference*.

AWS managed policy: AWSFaultInjectionSimulatorRDSAccess

Use this policy in an experiment role to grant AWS FIS permission to run experiments that use the AWS FIS actions for Amazon RDS. For more information, see the section called "Experiment role".

To view the permissions for this policy, see AWSFaultInjectionSimulatorRDSAccess in the *AWS Managed Policy Reference*.

AWS managed policy: AWSFaultInjectionSimulatorSSMAccess

Use this policy in an experiment role to grant AWS FIS permission to run experiments that use the AWS FIS actions for Systems Manager. For more information, see the section called "Experiment role".

To view the permissions for this policy, see AWSFaultInjectionSimulatorSSMAccess in the *AWS Managed Policy Reference*.

AWS FIS updates to AWS managed policies

View details about updates to AWS managed policies for AWS FIS since this service began tracking these changes.

Change	Description	Date
AWSFaultInjectionSimulatorECSAccess – Update to an existing policy	Added permissions to allow AWS FIS to resolve ECS targets.	January 25, 2024
AWSFaultInjectionSimulatorNetworkAccess – Update to an existing policy	Added permissions to allow AWS FIS to run experiments using the **aws:network:route-table-disrupt-cross-region-connectivity** and **aws:network:transit-gateway-disrupt-cross-region-connectivity** actions.	January 25, 2024
AWSFaultInjectionSimulatorEC2Access – Update to an existing policy	Added permissions to allow AWS FIS to resolve EC2 instances.	November 13, 2023

Change	Description	Date
AWSFaultInjectionSimulatorEKSAccess – Update to an existing policy	Added permissions to allow AWS FIS to resolve EKS targets.	November 13, 2023
AWSFaultInjectionSimulatorRDSAccess – Update to an existing policy	Added permissions to allow AWS FIS to resolve RDS targets.	November 13, 2023
AWSFaultInjectionSimulatorEC2Access – Update to an existing policy	Added permissions to allow AWS FIS to run SSM documents on EC2 instances and to terminate EC2 instances.	June 2, 2023
AWSFaultInjectionSimulatorSSMAccess – Update to an existing policy	Added permissions to allow AWS FIS to run SSM documents on EC2 instances.	June 2, 2023
AWSFaultInjectionSimulatorECSAccess – Update to an existing policy	Added permissions to allow AWS FIS to run experiments using the new **aws:ecs:task** actions.	June 1, 2023
AWSFaultInjectionSimulatorEKSAccess – Update to an existing policy	Added permissions to allow AWS FIS to run experiments using the new **aws:eks:pod** actions.	June 1, 2023
AWSFaultInjectionSimulatorEC2Access – New policy	Added a policy to allow AWS FIS to run an experiment that uses AWS FIS actions for Amazon EC2.	October 26, 2022
AWSFaultInjectionSimulatorECSAccess – New policy	Added a policy to allow AWS FIS to run an experiment that uses AWS FIS actions for Amazon ECS.	October 26, 2022
AWSFaultInjectionSimulatorEKSAccess – New policy	Added a policy to allow AWS FIS to run an experiment that uses AWS FIS actions for Amazon EKS.	October 26, 2022

Change	Description	Date
AWSFaultInjectionSimulatorNetworkAccess – New policy	Added a policy to allow AWS FIS to run an experiment that uses AWS FIS networking actions.	October 26, 2022
AWSFaultInjectionSimulatorRDSAccess – New policy	Added a policy to allow AWS FIS to run an experiment that uses AWS FIS actions for Amazon RDS.	October 26, 2022
AWSFaultInjectionSimulatorSSMAccess – New policy	Added a policy to allow AWS FIS to run an experiment that uses AWS FIS actions for Systems Manager.	October 26, 2022
AmazonFISServiceRolePolicy – Update to an existing policy	Added permissions to allow AWS FIS to describe subnets.	October 26, 2022
AmazonFISServiceRolePolicy – Update to an existing policy	Added permissions to allow AWS FIS to describe EKS clusters.	July 7, 2022
AmazonFISServiceRolePolicy – Update to an existing policy	Added permissions to allow AWS FIS to list and describe the tasks in your clusters.	February 7, 2022
AmazonFISServiceRolePolicy – Update to an existing policy	Removed the `events:ManagedBy` condition for the `events:DescribeRule` action.	January 6, 2022
AmazonFISServiceRolePolicy – Update to an existing policy	Added permissions to allow AWS FIS to retrieve history for the CloudWatch alarms used in stop conditions.	June 30, 2021
AWS FIS started tracking changes	AWS FIS started tracking changes to its AWS managed policies	March 1, 2021

Infrastructure security in AWS Fault Injection Service

As a managed service, AWS Fault Injection Service is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access AWS FIS through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Access AWS FIS using an interface VPC endpoint (AWS PrivateLink)

You can establish a private connection between your VPC and AWS Fault Injection Service by creating an *interface VPC endpoint*. VPC endpoints are powered by AWS PrivateLink, a technology that enables you to privately access AWS FIS APIs without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to communicate with AWS FIS APIs.

Each interface endpoint is represented by one or more elastic network interfaces in your subnets.

For more information, see Access AWS services through AWS PrivateLink in the *AWS PrivateLink Guide*.

Considerations for AWS FIS VPC endpoints

Before you set up an interface VPC endpoint for AWS FIS, review Access an AWS service using an interface VPC endpoint in the *AWS PrivateLink Guide*.

AWS FIS supports making calls to all of its API actions from your VPC.

Create an interface VPC endpoint for AWS FIS

You can create a VPC endpoint for the AWS FIS service using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Create a VPC endpoint in the *AWS PrivateLink Guide*.

Create a VPC endpoint for AWS FIS using the following service name: `com.amazonaws.region.fis`.

If you enable private DNS for the endpoint, you can make API requests to AWS FIS using its default DNS name for the Region, for example, `fis.us-east-1.amazonaws.com`.

Create a VPC endpoint policy for AWS FIS

You can attach an endpoint policy to your VPC endpoint that controls access to AWS FIS. The policy specifies the following information:

- The principal that can perform actions.
- The actions that can be performed.
- The resources on which actions can be performed.

For more information, see Control access to VPC endpoints using endpoint policies in the *AWS PrivateLink Guide*.

Example: VPC endpoint policy for specific AWS FIS actions

The following VPC endpoint policy grants access to the listed AWS FIS actions on all resources to all principals.

```
{
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "fis:ListExperimentTemplates",
            "fis:StartExperiment",
            "fis:StopExperiment",
            "fis:GetExperiment"
```

```
         ],
         "Resource":"*",
         "Principal":"*"
      }
   ]
}
```

Example: VPC endpoint policy that denies access from a specific AWS account

The following VPC endpoint policy denies the specified AWS account access to all actions and resources, but grants all other AWS accounts access to all actions and resources.

```
{
   "Statement":[
      {
         "Effect": "Allow",
         "Action": "*",
         "Resource": "*",
         "Principal": "*"
      },
      {
         "Effect":"Deny",
         "Action": "*",
         "Resource": "*",
         "Principal": {
           "AWS": [ "123456789012" ]
         }
      }
   ]
}
```

Tag your AWS FIS resources

A *tag* is a metadata label that either you or AWS assigns to an AWS resource. Each tag consists of a *key* and a *value*. For tags that you assign, you define the key and the value. For example, you might define the key as `purpose` and the value as `test` for a resource.

Tags help you do the following:

- Identify and organize your AWS resources. Many AWS services support tagging, so you can assign the same tag to resources from different services to indicate that the resources are related.
- Control access to your AWS resources. For more information, see Controlling Access Using Tags in the *IAM User Guide*.

Tagging restrictions

The following basic restrictions apply to tags on AWS FIS resources:

- Maximum number of tags that you can assign to a resource: 50
- Maximum key length: 128 Unicode characters
- Maximum value length: 256 Unicode characters
- Valid characters for keys and values: a-z, A-Z, 0-9, space, and the following characters: _ . : / = + - and @
- Keys and values are case sensitive
- You cannot use `aws:` as a prefix for keys, because it's reserved for AWS use

Work with tags

The following AWS Fault Injection Service (AWS FIS) resources support tagging:

- Actions
- Experiments
- Experiment templates

You can use the console to work with tags for experiments and experiment templates. For more information, see the following:

- Tag an experiment
- Tag experiment templates

You can use the following AWS CLI commands to work with tags for actions, experiments, and experiment templates:

- tag-resource – Add tags to a resource.
- untag-resource – Remove tags from a resource.
- list-tags-for-resource – List the tags for a specific resource.

Quotas and limitations for AWS Fault Injection Service

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some quotas, but not for all quotas.

To view the quotas for AWS FIS, open the Service Quotas console. In the navigation pane, choose **AWS services** and select **AWS Fault Injection Service**.

To request a quota increase, see Requesting a quota increase in the *Service Quotas User Guide*.

Your AWS account has the following quotas related to AWS FIS.

Name	Default	Adjuse	Description
Action duration in hours	Each supported Region: 12	No	The maximum number of hours allowed to run one action in this account in the current Region.
Actions per experiment template	Each supported Region: 20	No	The maximum number of actions that you can create in an experiment template in this account in the current Region.
Active experiments	Each supported Region: 5	No	The maximum number of active experiments that you can run simultaneously in this account in the current Region.
Completed experiment data retention in days	Each supported Region: 120	No	The maximum number of days allowed for AWS FIS to retain data about completed experimen

Name	Default	Adjustable	Description
			ts in this account in the current Region.
Experiment duration in hours	Each supported Region: 12	No	The maximum number of hours allowed to run one experiment in this account in the current Region.
Experiment templates	Each supported Region: 500	No	The maximum number of experiment templates that you can create in this account in the current Region.
Maximum number of Managed Prefix Lists in aws:network:route-table-disrupt-cross-region-connectivity	Each supported Region: 15	No	The maximum number of Managed Prefix Lists that aws:network:route-table-disrupt-cross-region-connectivity will allow, per action.
Maximum number of Route Tables in aws:network:route-table-disrupt-cross-region-connectivity	Each supported Region: 10	No	The maximum number of Route Tables that aws:network:route-table-disrupt-cross-region-connectivity will allow, per action.
Maximum number of routes in aws:network:route-table-disrupt-cross-region-connectivity	Each supported Region: 200	No	The maximum number of routes that aws:network:route-table-disrupt-cross-region-connectivity will allow, per action.

Name	Default	Adjustable	Description
Parallel actions per experiment	Each supported Region: 10	No	The maximum number of actions that you can run in parallel in an experiment in this account in the current Region.
Stop conditions per experiment template	Each supported Region: 5	No	The maximum number of stop conditions that you can add to an experiment template in this account in the current Region.
Target Auto Scaling groups for aws:ec2:asg-insufficient-instance-capacity-error	Each supported Region: 5	Yes	The maximum number of Auto Scaling groups that aws:ec2:asg-insufficient-instance-capacity-error can target when you identify targets using tags, per experiment.
Target Buckets for aws:s3:bucket-pause-replication	Each supported Region: 20	Yes	The maximum number of S3 Buckets that aws:s3:bucket-pause-replication can target when you identify targets using tags, per experiment.

Name	Default	Adjustable	Description
Target Clusters for aws:ecs:drain-container-instances	Each supported Region: 5	Yes	The maximum number of Clusters that aws:ecs:drain-container-instances can target when you identify targets using tags, per experiment.
Target Clusters for aws:rds:failover-db-cluster	Each supported Region: 5	Yes	The maximum number of Clusters that aws:rds:failover-db-cluster can target when you identify targets using tags, per experiment.
Target DBInstances for aws:rds:reboot-db-instances	Each supported Region: 5	Yes	The maximum number of DBInstances that aws:rds:reboot-db-instances can target when you identify targets using tags, per experiment.
Target Instances for aws:ec2:reboot-instances	Each supported Region: 5	Yes	The maximum number of Instances that aws:ec2:reboot-instances can target when you identify targets using tags, per experiment.

Name	Default	Adjustable	Description
Target Instances for aws:ec2:stop-instances	Each supported Region: 5	Yes	The maximum number of Instances that aws:ec2:stop-instances can target when you identify targets using tags, per experiment.
Target Instances for aws:ec2:terminate-instances	Each supported Region: 5	Yes	The maximum number of Instances that aws:ec2:terminate-instances can target when you identify targets using tags, per experiment.
Target Instances for aws:ssm:send-command	Each supported Region: 5	Yes	The maximum number of Instances that aws:ssm:send-command can target when you identify targets using tags, per experiment.
Target Nodegroups for aws:eks:terminate-nodegroup-instances	Each supported Region: 5	Yes	The maximum number of Nodegroups that aws:eks:terminate-nodegroup-instances can target when you identify targets using tags, per experiment.

Name	Default	Adjustable	Description
Target Pods for aws:eks:pod-cpu-stress	Each supported Region: 50	Yes	The maximum number of Pods that aws:eks:pod-cpu-stress can target when you identify targets using parameters, per experiment.
Target Pods for aws:eks:pod-delete	Each supported Region: 50	Yes	The maximum number of Pods that aws:eks:pod-delete can target when you identify targets using parameters, per experiment.
Target Pods for aws:eks:pod-io-stress	Each supported Region: 50	Yes	The maximum number of Pods that aws:eks:pod-io-stress can target when you identify targets using parameters, per experiment.
Target Pods for aws:eks:pod-memory-stress	Each supported Region: 50	Yes	The maximum number of Pods that aws:eks:pod-memory-stress can target when you identify targets using parameters, per experiment.

Name	Default	Adjustable	Description
Target Pods for aws:eks:pod-network-blackhole-port	Each supported Region: 50	Yes	The maximum number of Pods that aws:eks:pod-network-blackhole-port can target when you identify targets using parameters, per experiment.
Target Pods for aws:eks:pod-network-latency	Each supported Region: 50	Yes	The maximum number of Pods that aws:eks:pod-network-latency can target when you identify targets using parameters, per experiment.
Target Pods for aws:eks:pod-network-packet-loss	Each supported Region: 50	Yes	The maximum number of Pods that aws:eks:pod-network-packet-loss can target when you identify targets using parameters, per experiment.
Target ReplicationGroups for aws:elasticache:interrupt-cluster-az-power	Each supported Region: 5	Yes	The maximum number of ReplicationGroups that aws:elasticache:interrupt-cluster-az-power can target when you identify targets using tags/parameters, per experiment.

Name	Default	Adjustable	Description
Target SpotInstances for aws:ec2:send-spot-instance-interruptions	Each supported Region: 5	Yes	The maximum number of SpotInstances that aws:ec2:send-spot-instance-interruptions can target when you identify targets using tags, per experiment.
Target Subnets for aws:network:disrupt-connectivity	Each supported Region: 5	Yes	The maximum number of Subnets that aws:network:disrupt-connectivity can target when you identify targets using tags, per experiment. Quotas above 5 apply only to parameter scope:all. If you require a higher quota for another scope type, contact customer support at https://console.aws.amazon.com/support/home#/.
Target Subnets for aws:network:route-table-disrupt-cross-region-connectivity	Each supported Region: 6	Yes	The maximum number of Subnets that aws:network:route-table-disrupt-cross-region-connectivity can target when you identify targets using tags, per experiment.

Name	Default	Adjustable	Description
Target Tasks for aws:ecs:stop-task	Each supported Region: 5	Yes	The maximum number of Tasks that aws:ecs:stop-task can target when you identify targets using tags, per experiment.
Target Tasks for aws:ecs:task-cpu-stress	Each supported Region: 5	Yes	The maximum number of Tasks that aws:ecs:task-cpu-stress can target when you identify targets using tags/parameters, per experiment.
Target Tasks for aws:ecs:task-io-stress	Each supported Region: 5	Yes	The maximum number of Tasks that aws:ecs:task-io-stress can target when you identify targets using tags/parameters, per experiment.
Target Tasks for aws:ecs:task-kill-process	Each supported Region: 5	Yes	The maximum number of Tasks that aws:ecs:task-kill-process can target when you identify targets using tags/parameters, per experiment.
Target Tasks for aws:ecs:task-network-blackhole-port	Each supported Region: 5	Yes	The maximum number of Tasks that aws:ecs:task-network-blackhole-port can target when you identify targets using tags/parameters, per experiment.

Name	Default	Adjustable	Description
Target Tasks for aws:ecs:task-network-latency	Each supported Region: 5	Yes	The maximum number of Tasks that aws:ecs:task-network-latency can target when you identify targets using tags/parameters, per experiment.
Target Tasks for aws:ecs:task-network-packet-loss	Each supported Region: 5	Yes	The maximum number of Tasks that aws:ecs:task-network-packet-loss can target when you identify targets using tags/parameters, per experiment.
Target TransitGateways for aws:network:transit-gateway-disrupt-cross-region-connectivity	Each supported Region: 5	Yes	The maximum number of Transit Gateways that aws:network:transit-gateway-disrupt-cross-region-connectivity can target when you identify targets using tags, per experiment.
Target account configurations per experiment template	Each supported Region: 10	Yes	The maximum number of target account configurations that you can create for an experiment template in this account in the current Region.

Name	Default	Adjustable	Description
Target tables for aws:dynamodb:global-table-pause-replication action	Each supported Region: 5	Yes	The maximum number of global tables that aws:dynamodb:global-table-pause-replication can target, per experiment.

Your usage of AWS FIS is subject to the following additional limitations:

Name	Limitation
Targets for `aws:elasticache:interrupt-cluster-az-power` action	Limited to 10 `aws:elasticache:redis-replicationgroup` clusters impaired per account per region per day. You can request an increase by creating a support case in the AWS Support Center Console.

Document history

The following table describes important documentation updates in the *AWS Fault Injection Service User Guide*.

Change	Description	Date
New action	You can now use the `aws:dynamodb:global-table-pause-replication` action to pause data replication between the target global table and its replica tables. The `aws:dynamodb:encrypted-global-table-pause-replication` action will no longer be supported.	April 24, 2024
New actions mode experiment option	You can set actions mode to `skip-all` to generate a target preview before running an experiment.	March 13, 2024
AWS managed policy updates	AWS FIS updated existing managed policies.	January 25, 2024
New scenarios and actions	You can now use AWS FIS scenarios Cross-Region:Connectivity and AZ Availability: Power Interruption.	November 30, 2023
New action	You can now use the **aws:ec2:asg-insufficient-instance-capacity-error** action.	November 30, 2023

New action	You can now use the **aws:ec2:api-insufficient-instance-capacity-error** action.	November 30, 2023
New action	You can now use the **aws:network:route-table-disrupt-cross-region-connectivity** action.	November 30, 2023
New action	You can now use the **aws:network:transit-gateway-disrupt-cross-region-connectivity** action.	November 30, 2023
New action	You can now use the **aws:dynamodb:encrypted-global-table-pause-replication** action.	November 30, 2023
New action	You can now use the **aws:s3:bucket-pause-replication** action.	November 30, 2023
New action	You can now use the **aws:elasticache:interrupt-cluster-az-power** action.	November 30, 2023
New experiment options	You can now use AWS FIS experiment options for account targeting and empty target resolution.	November 27, 2023
Name change of AWS FIS	Updated service name to AWS Fault Injection Service.	November 15, 2023
AWS managed policy updates	AWS FIS updated existing managed policies.	November 13, 2023

New scenario library	You can now use the AWS FIS scenario library feature.	November 7, 2023
New experiment scheduler	You can now use the AWS FIS experiment scheduler feature.	November 7, 2023
AWS managed policy updates	AWS FIS updated existing managed policies.	June 2, 2023
New actions	You can use the new **aws:ecs:task** and **aws:eks:pod** actions.	June 1, 2023
AWS managed policy updates	AWS FIS updated existing managed policies.	June 1, 2023
New pre-configured SSM document	You can use the following pre-configured SSM document: AWSFIS-Run-Disk-Fill.	April 28, 2023
New action	You can use the **aws:ebs:pause-volume-io** action to pause I/O between the target volumes and the instances they are attached to.	January 27, 2023
New action	You can use the **aws:network:disrupt-connectivity** action to deny specific types of traffic to the target subnets.	October 26, 2022
New action	You can use the **aws:eks:inject-kubernetes-custom-resource** action to run a ChaosMesh or Litmus experiment on a single target cluster.	July 7, 2022

Experiment logging	You can configure your experiment templates to send experiment activity logs to CloudWatch Logs or to an S3 bucket.	February 28, 2022
New notifications	When the state of an experiment changes, AWS FIS emits a notification. These notifications are made available as events through Amazon EventBridge.	February 24, 2022
New action	You can use the **aws:ecs:stop-task** action to stop the specified task.	February 9, 2022
New action	You can use the **aws:cloudwatch:assert-alarm-state** action to verify that the specified alarms are in one of the specified alarm states.	November 5, 2021
New pre-configured SSM documents	You can use the following pre-configured SSM documents: AWSFIS-Run-IO-Stress, AWSFIS-Run-Network-Blackhold-Port, AWSFIS-Run-Network-Latency-Sources, AWSFIS-Run-Network-Packet-Loss, and AWSFIS-Run-Network-Packet-Loss-Sources.	November 4, 2021

New action	You can use the **aws:ec2:send-spot-instance-interruptions** action to send a Spot Instance interruption notice to target Spot Instances and then interrupt the target Spot Instances.	October 20, 2021
New action	You can use the **aws:ssm:start-automation-execution** action to initiate the execution of an Automation runbook.	September 17, 2021
Initial release	The initial release of the AWS Fault Injection Service User Guide.	March 15, 2021