Amazon EventBridge: User Guide
Copyright © 2020 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>What Is Amazon EventBridge?</td>
<td>1</td>
</tr>
<tr>
<td>Concepts</td>
<td>1</td>
</tr>
<tr>
<td>Related AWS Services</td>
<td>2</td>
</tr>
<tr>
<td>Setting Up</td>
<td>3</td>
</tr>
<tr>
<td>Sign Up for Amazon Web Services (AWS)</td>
<td>3</td>
</tr>
<tr>
<td>Sign in to the Amazon EventBridge Console</td>
<td>3</td>
</tr>
<tr>
<td>Account Credentials</td>
<td>3</td>
</tr>
<tr>
<td>Set Up the Command Line Interface</td>
<td>4</td>
</tr>
<tr>
<td>Regional Endpoints</td>
<td>4</td>
</tr>
<tr>
<td>Getting Started</td>
<td>5</td>
</tr>
<tr>
<td>Creating an EventBridge Rule That Triggers on an Event from an AWS Resource</td>
<td>5</td>
</tr>
<tr>
<td>Creating a Rule That Triggers on an AWS API Call via CloudTrail</td>
<td>6</td>
</tr>
<tr>
<td>Creating a Rule That Triggers on a Schedule</td>
<td>7</td>
</tr>
<tr>
<td>Receiving Events from a SaaS Partner</td>
<td>8</td>
</tr>
<tr>
<td>Creating a Rule That Triggers on a SaaS Partner Event</td>
<td>9</td>
</tr>
<tr>
<td>Creating an Event Bus</td>
<td>10</td>
</tr>
<tr>
<td>Creating a Custom Event Bus</td>
<td>10</td>
</tr>
<tr>
<td>Deleting or Disabling a Rule</td>
<td>11</td>
</tr>
<tr>
<td>Tutorials</td>
<td>12</td>
</tr>
<tr>
<td>Tutorial: Relay Events to Systems Manager Run Command</td>
<td>12</td>
</tr>
<tr>
<td>Tutorial: Log EC2 Instance States</td>
<td>13</td>
</tr>
<tr>
<td>Step 1: Create an AWS Lambda Function</td>
<td>13</td>
</tr>
<tr>
<td>Step 2: Create a Rule</td>
<td>14</td>
</tr>
<tr>
<td>Step 3: Test the Rule</td>
<td>14</td>
</tr>
<tr>
<td>Tutorial: Log Auto Scaling Group States</td>
<td>15</td>
</tr>
<tr>
<td>Step 1: Create an AWS Lambda Function</td>
<td>15</td>
</tr>
<tr>
<td>Step 2: Create a Rule</td>
<td>16</td>
</tr>
<tr>
<td>Step 3: Test the Rule</td>
<td>16</td>
</tr>
<tr>
<td>Tutorial: Log S3 Object Level Operations</td>
<td>17</td>
</tr>
<tr>
<td>Step 1: Configure Your AWS CloudTrail Trail</td>
<td>17</td>
</tr>
<tr>
<td>Step 2: Create an AWS Lambda Function</td>
<td>17</td>
</tr>
<tr>
<td>Step 3: Create a Rule</td>
<td>18</td>
</tr>
<tr>
<td>Step 4: Test the Rule</td>
<td>18</td>
</tr>
<tr>
<td>Tutorial: Use Input Transformer to Customize What Is Passed to the Event Target</td>
<td>19</td>
</tr>
<tr>
<td>Create a Rule</td>
<td>20</td>
</tr>
<tr>
<td>Tutorial: Log AWS API Calls</td>
<td>20</td>
</tr>
<tr>
<td>Prerequisite</td>
<td>20</td>
</tr>
<tr>
<td>Step 1: Create an AWS Lambda Function</td>
<td>20</td>
</tr>
<tr>
<td>Step 2: Create a Rule</td>
<td>21</td>
</tr>
<tr>
<td>Step 3: Test the Rule</td>
<td>22</td>
</tr>
<tr>
<td>Tutorial: Schedule Automated Amazon EBS Snapshots</td>
<td>22</td>
</tr>
<tr>
<td>Step 1: Create a Rule</td>
<td>22</td>
</tr>
<tr>
<td>Step 2: Test the Rule</td>
<td>23</td>
</tr>
<tr>
<td>Tutorial: Schedule AWS Lambda Functions</td>
<td>23</td>
</tr>
<tr>
<td>Step 1: Create an AWS Lambda Function</td>
<td>23</td>
</tr>
<tr>
<td>Step 2: Create a Rule</td>
<td>24</td>
</tr>
<tr>
<td>Step 3: Verify the Rule</td>
<td>25</td>
</tr>
<tr>
<td>Tutorial: Set Systems Manager Automation as a Target</td>
<td>26</td>
</tr>
<tr>
<td>Tutorial: Relay Events to a Kinesis Stream</td>
<td>26</td>
</tr>
<tr>
<td>Prerequisite</td>
<td>26</td>
</tr>
<tr>
<td>Step 1: Create an Amazon Kinesis Stream</td>
<td>27</td>
</tr>
<tr>
<td>Step 2: Create a Rule</td>
<td>27</td>
</tr>
<tr>
<td>Step 3: Test the Rule</td>
<td>27</td>
</tr>
<tr>
<td>Step 4: Verify That the Event Is Relayed</td>
<td>28</td>
</tr>
</tbody>
</table>
What Is Amazon EventBridge?

Amazon EventBridge is a serverless event bus service that makes it easy to connect your applications with data from a variety of sources. EventBridge delivers a stream of real-time data from your own applications, Software-as-a-Service (SaaS) applications, and AWS services and routes that data to targets such as AWS Lambda. You can set up routing rules to determine where to send your data to build application architectures that react in real time to all of your data sources. EventBridge allows you to build event driven architectures, which are loosely coupled and distributed.

EventBridge was formerly called Amazon CloudWatch Events. It includes new features that enable you to receive events from SaaS partners and your own applications. Existing CloudWatch Events users can access their existing default bus, rules, and events in the new EventBridge console and in the CloudWatch Events console. EventBridge uses the same CloudWatch Events API, so all of your existing CloudWatch Events API usage remains the same.

You can configure the following AWS resources as targets for EventBridge:

- Lambda functions
- Amazon EC2 instances
- Streams in Amazon Kinesis Data Streams
- Delivery streams in Amazon Kinesis Data Firehose
- Log groups in Amazon CloudWatch Logs
- Amazon ECS tasks
- Systems Manager Run Command
- Systems Manager Automation
- AWS Batch jobs
- AWS Step Functions state machines
- Pipelines in AWS CodePipeline
- AWS CodeBuild projects
- Amazon Inspector assessment templates
- Amazon SNS topics
- Amazon SQS queues
- Built-in targets: EC2 CreateSnapshot API call, EC2 RebootInstances API call, EC2 StopInstances API call, and EC2 TerminateInstances API call
- The default event bus of another AWS account

Concepts

Before you begin using EventBridge, you should understand the following concepts:

- **Events** – An event indicates a change in an environment. This can be an AWS environment, an SaaS partner service or application, or one of your own custom applications or services. For example, Amazon EC2 generates an event when the state of an EC2 instance changes from pending to running, and Amazon EC2 Auto Scaling generates events when it launches or terminates instances. AWS CloudTrail publishes events when you make API calls. You can also set up scheduled events that are generated on a periodic basis. For a list of services that generate events, and sample events from each service, see EventBridge Event Examples from Supported AWS Services (p. 57).
Rules – A rule matches incoming events and routes them to targets for processing. A single rule can route to multiple targets, all of which are processed in parallel. Rules aren’t processed in a particular order. This enables different parts of an organization to look for and process the events that are of interest to them. A rule can customize the JSON sent to the target, by passing only certain parts or by overwriting it with a constant.

Targets – A target processes events. Targets can include Amazon EC2 instances, Lambda functions, Kinesis streams, Amazon ECS tasks, Step Functions state machines, Amazon SNS topics, Amazon SQS queues, and built-in targets. A target receives events in JSON format.

A rule’s targets must be in the same Region as the rule.

Event buses – An event bus receives events. When you create a rule, you associate it with a specific event bus, and the rule is matched only to events received by that event bus.

Your account has one default event bus, which receives events from AWS services. You can create custom event buses to receive events from your custom applications. You can also create partner event buses to receive events from SaaS partner applications.

Partner event sources – A partner event source is used by an AWS partner to send events to an AWS customer account. To receive these events, the customer must associate an event bus with the partner event source.

Related AWS Services

You can use the following services with EventBridge:

- **AWS CloudTrail** enables you to monitor the calls made to the EventBridge API for your account, including calls made by the AWS Management Console, the AWS CLI and other services. When CloudTrail logging is turned on, EventBridge writes log files to an S3 bucket. Each log file contains one or more records, depending on how many actions are performed to satisfy a request. For more information, see Logging and Monitoring in Amazon EventBridge (p. 149).

- **AWS CloudFormation** enables you to model and set up your AWS resources. You create a template that describes the AWS resources you want, and AWS CloudFormation takes care of provisioning and configuring those resources for you. You can use EventBridge rules in your AWS CloudFormation templates. For more information, see AWS::Events::Rule in the AWS CloudFormation User Guide.

- **AWS Config** enables you to record configuration changes to your AWS resources. This includes how resources relate to one another and how they were configured in the past, so that you can see how the configurations and relationships change over time. You can also create AWS Config rules to check whether your resources are compliant or noncompliant with your organization's policies. For more information, see the AWS Config Developer Guide.

- **AWS Identity and Access Management (IAM)** helps you securely control access to AWS resources for your users. Use IAM to control who can use your AWS resources (authentication), what resources they can use, and how they can use them (authorization). For more information, see Identity and Access Management in Amazon EventBridge (p. 121).

- **Amazon Kinesis Data Streams** enables rapid and nearly continuous data intake and aggregation. The type of data used includes IT infrastructure log data, application logs, social media, market data feeds, and web clickstream data. Because the response time for the data intake and processing is in real time, processing is typically lightweight. For more information, see the Amazon Kinesis Data Streams Developer Guide.

- **AWS Lambda** enables you to build applications that respond quickly to new information. Upload your application code as Lambda functions, and Lambda runs your code on high-availability compute infrastructure. Lambda performs all the administration of the compute resources, including server and operating system maintenance, capacity provisioning, automatic scaling, code and security patch deployment, and code monitoring and logging. For more information, see the AWS Lambda Developer Guide.
Setting Up Amazon EventBridge

To use Amazon EventBridge you need an AWS account. Your AWS account allows you to use services (for example, Amazon EC2) to generate events that you can view in the CloudWatch console, a web-based interface. In addition, you can install and configure the AWS Command Line Interface (AWS CLI) to use a command-line interface.

Sign Up for Amazon Web Services (AWS)

When you create an AWS account, we automatically sign up your account for all AWS services. You pay only for the services that you use.

If you have an AWS account already, skip to the next step. If you don't have an AWS account, use the following procedure to create one.

**To sign up for an AWS account**

2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

Sign in to the Amazon EventBridge Console

**To sign in to the Amazon EventBridge console**

- Sign in to the AWS Management Console and open the Amazon EventBridge console at https://console.aws.amazon.com/events/.

Account Credentials

Although you can use your root user credentials to access EventBridge, we recommend that you use an AWS Identity and Access Management (IAM) account. If you're using an IAM account to access EventBridge, you must have the following permissions:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "events:*",
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": "**"
    }
  ]
}
```
For more information, see Authentication (p. 122).

Set Up the Command Line Interface

You can use the AWS CLI to perform EventBridge operations.

For information about how to install and configure the AWS CLI, see Getting Set Up with the AWS Command Line Interface in the AWS Command Line Interface User Guide.

Regional Endpoints

You must enable regional endpoints (the default) in order to use EventBridge. For more information, see Activating and Deactivating AWS STS in an AWS Region in the IAM User Guide.
Getting Started with Amazon EventBridge

Use the procedures in this section to create and delete EventBridge rules and event buses. For tutorials on specific scenarios and specific targets, see Amazon EventBridge Tutorials (p. 12).

Contents
- Creating an EventBridge Rule That Triggers on an Event from an AWS Resource (p. 5)
- Creating an EventBridge Rule That Triggers on an AWS API Call Using AWS CloudTrail (p. 6)
- Creating an EventBridge Rule That Triggers on a Schedule (p. 7)
- Receiving Events from a SaaS Partner (p. 8)
- Creating an Event Bus (p. 10)
- Deleting or Disabling an EventBridge Rule (p. 11)

Limitations
- The targets you associate with a rule must be in the same Region as the rule.
- Some target types might not be available in every Region. For more information, see Regions and Endpoints in the Amazon Web Services General Reference.
- Creating rules with built-in targets is supported only in the AWS Management Console.
- If you create a rule with an encrypted Amazon SQS queue as a target, you must have the following section included in your AWS Key Management Service key policy. It allows the event to be successfully delivered to the encrypted queue.

```json
{
    "Sid": "Allow CWE to use the key",
    "Effect": "Allow",
    "Principal": {
        "Service": "events.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*"
}
```

Creating an EventBridge Rule That Triggers on an Event from an AWS Resource

Use the following steps to create an EventBridge rule that triggers on an event emitted by an AWS service.

When an AWS service in your account emits an event, it always goes to your account’s default event bus. To write a rule that triggers on events from AWS services in your account, you must associate it
with the default event bus. You can create a rule on a custom event bus that looks for events from AWS services, but this rule will trigger only when you receive such an event from another account via cross-account event delivery. For more information, see Sending and Receiving Events Between AWS Accounts (p. 104).

To create a rule that triggers on an event

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
   A rule can't have the same name as another rule in the same Region and on the same event bus.
5. For Define pattern, choose Event pattern.
6. Choose Pre-defined pattern by service.
7. For Service provider, choose AWS.
8. For Service name, choose the name of the service that emits the event.
9. For Event type, choose All Events or choose the type of event to use for this rule. If you choose All Events, all events emitted by this AWS service will match the rule.
   If you want to customize the event pattern, choose Edit, make your changes, and choose Save.
10. For Select event bus, choose the event bus that you want to associate with this rule. If you want this rule to trigger on matching events that come from your own AWS account, select AWS default event bus. When an AWS service in your account emits an event, it always goes to your account's default event bus.
11. For Select targets, choose the AWS service that is to act when an event of the selected type is detected.
12. In the other fields in this section, enter information specific to this target type, if any is needed.
13. For many target types, EventBridge needs permissions to send events to the target. In these cases, EventBridge can create the IAM role needed for your rule to run:
   • To create an IAM role automatically, choose Create a new role for this specific resource
   • To use an IAM role that you created before, choose Use existing role
14. Optionally, choose Add target to add another target for this rule.
15. (Optional) Enter one or more tags for the rule. For more information, see Tagging Your Amazon EventBridge Resources (p. 154).
16. Choose Create.

Creating an EventBridge Rule That Triggers on an AWS API Call Using AWS CloudTrail

To create a rule that triggers on an action by an AWS service that does not emit events, you can base the rule on API calls made by that service. The API calls are recorded by AWS CloudTrail. For more information about the API calls that you can use as triggers for rules, see Services Supported by CloudTrail Event History.

Rules in EventBridge work only in the Region where they’re created. If you configure CloudTrail to track API calls in multiple Regions and you want a rule based on CloudTrail to trigger in each of those Regions, you must create a separate rule in each Region that you want to track.
All events that are delivered via CloudTrail have \texttt{AWS API Call via CloudTrail} as the value for \texttt{detail-type}.

\textbf{Note}

You might accidentally create rules that lead to infinite loops, where a rule is fired repeatedly. For example, a rule might detect that ACLs have changed on an S3 bucket and trigger software to change them to the desired state. If you don't write the rule carefully, the subsequent change to the ACLs fires the rule again, creating an infinite loop.

To prevent this, write the rules so that the triggered actions don't refire the same rule. For example, your rule could fire only if ACLs are found to be in a bad state instead of after any change.

An infinite loop can quickly cause higher than expected charges. We recommend that you use budgeting, which alerts you when charges exceed your specified quota. For more information, see \textit{Managing Your Costs with Budgets} in the \textit{AWS Billing and Cost Management User Guide}.

\textbf{To create a rule that triggers on an API call via CloudTrail}

1. Open the Amazon EventBridge console at \url{https://console.aws.amazon.com/events/}.
2. In the navigation pane, choose \textbf{Rules}.
3. Choose \textbf{Create rule}.
4. Enter a name and description for the rule. A rule can't have the same name as another rule in the same Region and on the same event bus.
5. For \textbf{Define pattern}, choose \textbf{Event pattern}.
6. Choose \textbf{Pre-defined pattern by service}.
7. For \textbf{Service provider}, choose \texttt{AWS}.
8. For \textbf{Service name}, choose the name of the service that emits the event.
9. For \textbf{Event type}, choose \texttt{AWS API Call via CloudTrail}.

   If you want to customize the event pattern, choose \textbf{Edit}, make your changes, and choose \textbf{Save}.
10. For \textbf{Select event bus}, choose the event bus that you want to associate with this rule.
11. For \textbf{Select event bus}, choose the event bus that you want to associate with this rule. If you want this rule to trigger on matching events that come from your own AWS account, select \texttt{AWS default event bus}. When an AWS service in your account emits an event, it always goes to your account's default event bus. >
12. In the other fields in this section, enter information specific to this target type, if any is needed.
13. For many target types, EventBridge needs permissions to send events to the target. In these cases, EventBridge can create the IAM role needed for your rule to run:
   - To create an IAM role automatically, choose \textit{Create a new role for this specific resource}
   - To use an IAM role that you created before, choose \textit{Use existing role}
14. (Optional) Choose \textbf{Add target} to add another target for this rule.
15. (Optional) Enter one or more tags for the rule. For more information, see \textit{Tagging Your Amazon EventBridge Resources} (p. 154).
16. Choose \textbf{Create}.

\textbf{Creating an EventBridge Rule That Triggers on a Schedule}

Use the following steps to create an EventBridge rule that triggers on a regular schedule. You can create scheduled rules only using the default event bus.
To create a rule that triggers on a regular schedule

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.

A rule can’t have the same name as another rule in the same Region and on the same event bus.

5. For Define pattern, choose Schedule.
6. Either choose Fixed rate of and specify how often the task is to run, or choose Cron expression and specify a cron expression that defines when the task is to be triggered. For more information about cron expression syntax, see Schedule Expressions for Rules (p. 32).
7. For Select event bus, choose AWS default event bus. Scheduled rules are supported only on the default event bus.
8. For Select targets, choose the AWS service that is to act on the specified schedule.
9. In the other fields in this section, enter information specific to this target type, if any is needed.
10. For many target types, EventBridge needs permissions to send events to the target. In these cases, EventBridge can create the IAM role needed for your rule to run:
    • To create an IAM role automatically, choose Create a new role for this specific resource
    • To use an IAM role that you created before, choose Use existing role
11. (Optional) Choose Add target to add another target for this rule.
12. (Optional) Enter one or more tags for the rule. For more information, see Tagging Your Amazon EventBridge Resources (p. 154).
13. Choose Create.

Receiving Events from an SaaS Partner

To be able to receive events from SaaS partner applications and services, you must have a partner event source offered to you from the partner. You can then create a partner event bus and match it to the corresponding partner event source.

Partner event sources are available in the following Regions.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>us-east-1</td>
<td>US East (N. Virginia)</td>
</tr>
<tr>
<td>us-east-2</td>
<td>US East (Ohio)</td>
</tr>
<tr>
<td>us-west-1</td>
<td>US West (N. California)</td>
</tr>
<tr>
<td>us-west-2</td>
<td>US West (Oregon)</td>
</tr>
<tr>
<td>ca-central-1</td>
<td>Canada (Central)</td>
</tr>
<tr>
<td>eu-central-1</td>
<td>Europe (Frankfurt)</td>
</tr>
<tr>
<td>eu-west-1</td>
<td>Europe (Ireland)</td>
</tr>
<tr>
<td>eu-west-2</td>
<td>Europe (London)</td>
</tr>
<tr>
<td>eu-west-3</td>
<td>Europe (Paris)</td>
</tr>
</tbody>
</table>
Creating a Rule That Triggers on an SaaS Partner Event

To begin to receive events from an SaaS partner

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Partner event sources.
3. Find the partner that you want and choose Set up for that partner.
4. Choose Copy to copy your account ID to the clipboard.
5. In the navigation pane, choose Partner event sources.
6. Go to the partner's website and follow the instructions to create a partner event source. Use your account ID for this. The event source that you create will be available to only your account.
7. Go back to the EventBridge console and choose Partner event sources in the navigation pane.
8. Select the button next to the partner event source, and choose Associate with event bus.

The status of that event source changes from Pending to Active, and the name of the event bus is updated to match the partner event source name. You can now start creating rules that trigger on events from that partner event source. For more information, see Creating a Rule That Triggers on an SaaS Partner Event (p. 9).

Creating a Rule That Triggers on an SaaS Partner Event

Before you can create rules for events from SaaS partner applications and services, you must create a partner event bus and match it to that partner event source. For more information, see Receiving Events from an SaaS Partner (p. 8).

To create a rule that triggers on an event from an SaaS partner

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
5. For Define pattern, choose Event pattern.
6. Choose Pre-defined pattern by service.
7. For Service provider, choose Service partners.
8. For Service name, choose the name of the partner.
9. For **Event type**, choose **All Events** or choose the type of event to use for this rule. If you choose **All Events**, all events emitted by this partner event source will match the rule.

   If you want to customize the event pattern, choose **Edit**, make your changes, and choose **Save**.

10. For **Service event bus**, confirm that the event bus that corresponds to this partner is selected.

11. For **Select targets**, choose the AWS service that is to act when an event of the selected type is detected.

12. In the other fields in this section, enter information specific to this target type, if any is needed.

13. For many target types, EventBridge needs permissions to send events to the target. In these cases, EventBridge can create the IAM role needed for your rule to run:

   - To create an IAM role automatically, choose **Create a new role for this specific resource**
   - To use an IAM role that you created before, choose **Use existing role**

14. (Optional) Choose **Add target** to add another target for this rule.

15. (Optional) Enter one or more tags for the rule. For more information, see [Tagging Your Amazon EventBridge Resources](p. 154).

16. Choose **Create**.

---

### Creating an Event Bus

Your account includes one default event bus, which receives events emitted by AWS services. You can also configure your custom applications to send events to the default event bus.

You can create two types of additional event buses in your account:

- **Partner event buses**, which can receive events from applications and services created by AWS software as a service (SaaS) partners. To receive events from SaaS partners, you need to create a partner event bus for each partner event source that you want to receive events from.

  For more information, see [Receiving Events from an SaaS Partner](p. 8).

- **Custom event buses**, which can receive events from your custom applications and services.

  Each event bus in your account can have up to 100 EventBridge rules associated with it, so if your account has many rules, you might want to create custom event buses to associate with some of the rules for your custom application events. Another reason to create custom event buses is to apply different permissions to different event buses. When you set permissions on an event bus, you can specify which other accounts or entire organizations can send events to the event bus.

### Creating a Custom Event Bus

You can create a custom event bus to receive events from your custom applications. Your applications can also send events to your default event bus.

**To create a custom event bus**

1. Open the Amazon EventBridge console at [https://console.aws.amazon.com/events/](https://console.aws.amazon.com/events/).
2. In the navigation pane, choose **Event buses**.
3. Choose **Create event bus**.
4. Enter a name for the new event bus.
5. To enable other accounts or entire organizations to send events to this event bus, choose **Other AWS account, Organization**, or both.
a. If you choose Other AWS account, choose either Individual AWS account ID or All AWS accounts. If you choose Individual AWS account ID, enter the account ID. To add more accounts, choose Add account.

If you choose All AWS accounts, be careful to create rules that match only the events that you want to receive from other accounts. To create more secure rules, make sure that the event pattern for each rule contains an Account field with the account IDs of one or more accounts to receive events from. Rules that have an event pattern containing an Account field don't match events sent from other accounts.

b. If you choose Organization, choose My organization to grant permissions to all accounts in the organization that your account is a member of. Or choose Other organization and enter the organization ID including the o- prefix. My organization is available only if your account is a member of an organization.

If you choose Other organization and want to add more organizations, choose Add organization.

6. Choose Create.

Deleting or Disabling an EventBridge Rule

Use the following steps to delete or disable an EventBridge rule.

To delete or disable a rule

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.

   Under Event bus, select the event bus that is associated with the rule.
3. Do one of the following:

   a. To delete a rule, select the button next to the rule and choose Actions, Delete, Delete.

      If the rule is a managed rule, you must enter the name of the rule to acknowledge that it is a managed rule and that deleting it may stop functionality in the service that created the rule. To continue, enter the rule name and choose Force delete. For more information, see Amazon EventBridge Managed Rules (p. 119).

   b. To temporarily disable a rule, select the button next to the rule and choose Disable, Disable.

      You can't disable a managed rule.
Amazon EventBridge Tutorials

The following tutorials show you how to create EventBridge rules for certain tasks and targets.

Tutorials:
- Tutorial: Use EventBridge to Relay Events to AWS Systems Manager Run Command (p. 12)
- Tutorial: Log the State of an Amazon EC2 Instance Using EventBridge (p. 13)
- Tutorial: Log the State of an Auto Scaling Group Using EventBridge (p. 15)
- Tutorial: Log Amazon S3 Object-Level Operations Using EventBridge (p. 17)
- Tutorial: Use Input Transformer to Customize What Is Passed to the Event Target (p. 19)
- Tutorial: Log AWS API Calls Using EventBridge (p. 20)
- Tutorial: Schedule Automated Amazon EBS Snapshots Using EventBridge (p. 22)
- Tutorial: Schedule AWS Lambda Functions Using EventBridge (p. 23)
- Tutorial: Set AWS Systems Manager Automation as an EventBridge Target (p. 26)
- Tutorial: Relay Events to an Amazon Kinesis Stream Using EventBridge (p. 26)
- Tutorial: Run an Amazon ECS Task When a File Is Uploaded to an Amazon S3 Bucket (p. 28)
- Tutorial: Schedule Automated Builds Using AWS CodeBuild (p. 29)
- Tutorial: Log State Changes of Amazon EC2 Instances (p. 30)
- Tutorial: Download Code Bindings for Events using the EventBridge Schema Registry (p. 31)

Tutorial: Use EventBridge to Relay Events to AWS Systems Manager Run Command

You can use Amazon EventBridge to invoke AWS Systems Manager Run Command and perform actions on Amazon EC2 instances when certain events happen. In this tutorial, set up Systems Manager Run Command to run shell commands and configure each new instance that is launched in an Amazon EC2 Auto Scaling group. This tutorial assumes that you have already assigned a tag to the Amazon EC2 Auto Scaling group, with environment as the key and production as the value.

To create the EventBridge rule

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Event pattern.
   b. Choose Pre-defined pattern by service.
   c. For Service provider, choose AWS.
   d. For Service Name, choose Auto Scaling.
e. For Event type, choose **Instance Launch and Terminate**.

f. Choose **Specific instance event(s), EC2 Instance-launch Lifecycle Action**.

g. By default, the rule matches any Amazon EC2 Auto Scaling group in the Region. To make the rule match a specific group, choose **Specific group name(s)** and select one or more groups.

6. For **Select event bus**, choose **AWS default event bus**. When an AWS service in your account emits an event, it always goes to your account's default event bus.

7. For **Target**, select **SSM Run Command**.

8. For **Document**, choose **AWS-RunShellScript**.

   For **Target key**, type **tag:environment**. For **Target value(s)**, enter **production** and choose **Add**.

9. Under **Configure automation parameter(s)**, do the following:
   a. Choose **Constant**.
   b. For **Commands**, enter a shell command and choose **Add**. Repeat this step for all commands to run when an instance launches.
   c. If necessary, enter the appropriate information in **WorkingDirectory** and **ExecutionTimeout**.

10. EventBridge can create the IAM role needed for your event to run:
    - To create an IAM role automatically, choose **Create a new role for this specific resource**
    - To use an IAM role that you created before, choose **Use existing role**

11. Choose **Create rule**.

---

**Tutorial: Log the State of an Amazon EC2 Instance Using EventBridge**

You can create an AWS Lambda function that logs the changes in state for an Amazon EC2 instance. You can choose to create a rule that runs the function whenever there is a state transition or a transition to one or more states that are of interest. In this tutorial, you log the launch of any new instance.

**Step 1: Create an AWS Lambda Function**

Create a Lambda function to log the state change events. You specify this function when you create your rule.

**To create a Lambda function**

2. If you're new to Lambda, you see a welcome page. Choose **Get Started Now**. Otherwise, choose **Create a Lambda function**.
3. On the **Select blueprint** page, enter **hello** for the filter and choose the **hello-world** blueprint.
4. On the **Configure triggers** page, choose **Next**.
5. On the **Configure function** page, do the following:
   a. Enter a name and description for the Lambda function. For example, name the function **LogEC2InstanceStateChange**.
   b. Edit the sample code for the Lambda function. For example:

```javascript
'use strict';
```
exports.handler = (event, context, callback) => {
  console.log('LogEC2InstanceStateChange');
  console.log('Received event:', JSON.stringify(event, null, 2));
  callback(null, 'Finished');
};

c. For Role, choose **Choose an existing role**. For **Existing role**, select your basic execution role. Otherwise, create a basic execution role.
d. Choose **Next**.

6. On the **Review** page, choose **Create function**.

---

**Step 2: Create a Rule**

Create a rule to run your Lambda function whenever you launch an Amazon EC2 instance.

**To create the EventBridge rule**

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose **Rules**.
3. Choose **Create rule**.
4. Enter a name and description for the rule.
5. For **Define pattern**, do the following:
   a. Choose **Event pattern**.
   b. Choose **Pre-defined pattern by service**.
   c. For **Service provider**, choose **AWS**.
   d. For **Service Name**, choose **EC2**.
   e. For **Event type**, choose **EC2 Instance State-change Notification**.
   f. Choose **Specific state(s)**, **running**.
   g. By default, the rule matches any instance group in the Region. To make the rule match a specific instance, choose **Specific instance ID(s)** and enter one or more instance IDs.
6. For **Select event bus**, choose **AWS default event bus**. When an AWS service in your account emits an event, it always goes to your account’s default event bus.
7. For **Target**, choose **Lambda function**.
8. For **Function**, select the Lambda function that you created.
9. Choose **Create**.

---

**Step 3: Test the Rule**

To test your rule, launch an Amazon EC2 instance. After waiting a few minutes for the instance to launch and initialize, verify that your Lambda function was invoked.

**To test your rule by launching an instance**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Launch an instance. For more information, see Launch Your Instance in the Amazon EC2 User Guide for Linux Instances.
3. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
4. In the navigation pane, choose **Rules**, choose the name of the rule that you created, and choose **Metrics for the rule**.
5. To view the output from your Lambda function, do the following:
   b. In the navigation pane, choose Logs.
   c. Choose the name of the log group for your Lambda function (/aws/lambda/function-name).
   d. Choose the name of the log stream to view the data provided by the function for the instance that you launched.

6. (Optional) When you’re finished, you can open the Amazon EC2 console and stop or terminate the instance that you launched. For more information, see Terminate Your Instance in the Amazon EC2 User Guide for Linux Instances.

Tutorial: Log the State of an Auto Scaling Group Using EventBridge

You can run an AWS Lambda function that logs an event whenever an Auto Scaling group launches or terminates an Amazon EC2 instance and whether the launch or terminate event was successful.

For information about additional EventBridge scenarios using Amazon EC2 Auto Scaling events, see Getting CloudWatch Events When Your Auto Scaling Group Scales in the Amazon EC2 Auto Scaling User Guide.

Step 1: Create an AWS Lambda Function

Create a Lambda function to log the scale-out and scale-in events for your Auto Scaling group. Specify this function when you create your rule.

To create a Lambda function

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. If you’re new to Lambda, you see a welcome page. Choose Get Started Now. Otherwise, choose Create a Lambda function.
3. On the Select blueprint page, enter hello for the filter and choose the hello-world blueprint.
4. On the Configure triggers page, choose Next.
5. On the Configure function page, do the following:
   a. Enter a name and description for the Lambda function. For example, name the function LogAutoScalingEvent.
   b. Edit the sample code for the Lambda function. For example:

   ```javascript
   'use strict';
   exports.handler = (event, context, callback) => {
     console.log('LogAutoScalingEvent');
     console.log('Received event:', JSON.stringify(event, null, 2));
     callback(null, 'Finished');
   };
   ```
   c. For Role, choose Choose an existing role. For Existing role, select your basic execution role. Otherwise, create a basic execution role.
   d. Choose Next.
6. Choose Create function.
Step 2: Create a Rule

Create a rule to run your Lambda function whenever your Auto Scaling group launches or terminates an instance.

To create a rule

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Event Pattern.
   b. Choose Pre-defined by service.
   c. For Service provider, choose AWS.
   d. For Service Name, choose Auto Scaling.
   e. For Event type, choose Instance Launch and Terminate.
   f. To capture all successful and unsuccessful instance launch and terminate events, choose Any instance event.
   g. By default, the rule matches any Auto Scaling group in the Region. To make the rule match a specific group, choose Specific group name(s) and select one or more groups.
   h. By default, the rule matches any Auto Scaling group in the Region. To make the rule match a specific Auto Scaling group, choose Specific group name(s) and select one or more Auto Scaling groups.
6. For Select event bus, choose AWS default event bus. When an AWS service in your account emits an event, it always goes to your account’s default event bus.
7. For Target, choose Lambda function.
8. For Function, select the Lambda function that you created.
9. Choose Create.

Step 3: Test the Rule

You can test your rule by manually scaling an Auto Scaling group so that it launches an instance. After waiting a few minutes for the scale-out event to occur, verify that your Lambda function was invoked.

To test your rule using an Auto Scaling group

1. To increase the size of your Auto Scaling group, do the following:
   a. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
   b. In the navigation pane, choose Auto Scaling, Auto Scaling Groups.
   c. Select the check box for your Auto Scaling group.
   d. On the Details tab, choose Edit. For Desired, increase the desired capacity by one. For example, if the current value is 2, enter 3. The desired capacity must be less than or equal to the maximum size of the group. If your new value for Desired is greater than Max, you must update Max. When you're finished, choose Save.
2. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
3. In the navigation pane, choose Rules, choose the name of the rule that you created, and choose Metrics for the rule.
4. To view the output from your Lambda function, do the following:
   b. In the navigation pane, choose Logs.
   c. Select the name of the log group for your Lambda function (/aws/lambda/function-name).
   d. Select the name of the log stream to view the data provided by the function for the instance that you launched.

5. (Optional) When you're finished, you can decrease the desired capacity by one so that the Auto Scaling group returns to its previous size.

---

### Tutorial: Log Amazon S3 Object-Level Operations Using EventBridge

You can log the object-level API operations on your S3 buckets. Before Amazon EventBridge can match these events, you must use AWS CloudTrail to set up a trail configured to receive these events.

#### Step 1: Configure Your AWS CloudTrail Trail

To log data events for an S3 bucket to AWS CloudTrail and EventBridge, create a trail. A trail captures API calls and related events in your account and delivers the log files to an S3 bucket that you specify. You can update an existing trail or create one.

**To create a trail**

1. Open the CloudTrail console at https://console.aws.amazon.com/cloudtrail/.
2. In the navigation pane, choose Trails, Create trail.
3. For Trail name, enter a name for the trail.
4. For Data events, enter the bucket name and prefix (optional). For each trail, you can add up to 250 Amazon S3 objects.
   - To log data events for all Amazon S3 objects in a bucket, specify an S3 bucket and an empty prefix. When an event occurs on an object in that bucket, the trail processes and logs the event.
   - To log data events for specific Amazon S3 objects, choose Add S3 bucket and specify an S3 bucket and optionally the object prefix. When an event occurs on an object in that bucket and the object starts with the specified prefix, the trail processes and logs the event.
5. For each resource, specify whether to log Read events, Write events, or both.
6. For Storage location, create or choose an existing S3 bucket to designate for log file storage.
7. Choose Create.

For more information, see Data Events in the AWS CloudTrail User Guide.

#### Step 2: Create an AWS Lambda Function

Create a Lambda function to log data events for your S3 buckets. You specify this function when you create your rule.

**To create a Lambda function**

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. If you’re new to Lambda, you see a welcome page. Choose Create a function. Otherwise, choose Create function.
3. Choose Author from scratch.
4. Under Author from scratch, do the following:
   a. Enter a name for the Lambda function. For example, name the function LogS3DataEvents.
   b. For Role, choose Create a custom role.
      A new window opens. Change the Role name if necessary and choose Allow.
   c. Back on the Lambda console, choose Create function.
5. Edit the code for the Lambda function to the following and choose Save.

   ```javascript
   'use strict';
   exports.handler = (event, context, callback) => {
     console.log('LogS3DataEvents');
     console.log('Received event:', JSON.stringify(event, null, 2));
     callback(null, 'Finished');
   };
   ```

Step 3: Create a Rule

Create a rule to run your Lambda function in response to an Amazon S3 data event.

To create a rule
1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Event Pattern.
   b. Choose Pre-defined pattern by service.
   c. For Service provider, choose AWS.
   d. For Service Name, choose Simple Storage Service (S3).
   e. For Event type, choose Object Level Operations.
   f. Choose Specific operation(s), PutObject.
   g. By default, the rule matches data events for all buckets in the Region. To match data events for specific buckets, choose Specify bucket(s) by name and enter one or more buckets.
6. For Select event bus, choose AWS default event bus. When an AWS service in your account emits an event, it always goes to your account’s default event bus.
7. For Targets, choose Lambda function.
8. For Function, select the Lambda function that you created.
9. Choose Create.

Step 4: Test the Rule

To test the rule, put an object in your S3 bucket. You can verify that your Lambda function was invoked.
To view the logs for your Lambda function

2. In the navigation pane, choose Logs.
3. Select the name of the log group for your Lambda function (/aws/lambda/function-name).
4. Select the name of the log stream to view the data provided by the function for the instance that you launched.

You can also check the contents of your CloudTrail logs in the S3 bucket that you specified for your trail. For more information, see Getting and Viewing Your CloudTrail Log Files in the AWS CloudTrail User Guide.

Tutorial: Use Input Transformer to Customize What Is Passed to the Event Target

You can use the input transformer feature of EventBridge to customize the text that is taken from an event before it's input to the target of a rule.

You can define multiple JSON paths from the event and assign their outputs to different variables. Then you can use those variables in the input template as `<variable-name>`. The characters `<` and `>` cannot be escaped.

If you specify a variable to match a JSON path that doesn't exist in the event, that variable isn't created and doesn't appear in the output.

In this tutorial, we extract the `instance-id` and `state` of an Amazon EC2 instance from the instance state change event. We use the input transformer to put that data into an easy-to-read message that is sent to an Amazon SNS topic. The rule is triggered when any instance changes to any state. For example, with this rule, the following Amazon EC2 instance state-change notification event produces the Amazon SNS message **The EC2 instance i-1234567890abcdef0 has changed state to stopped.**

```
{
  "id":"7bf73129-1428-4cd3-a780-95db273d1602",
  "detail-type":"EC2 Instance State-change Notification",
  "source":"aws.ec2",
  "account":"123456789012",
  "time":"2015-11-11T21:29:54Z",
  "region":"us-east-1",
  "resources":[
    "arn:aws:ec2:us-east-1:123456789012:instance/ i-1234567890abcdef0"
  ],
  "detail":{
    "instance-id":" i-1234567890abcdef0",
    "state":"stopped"
  }
}
```

We achieve this by mapping the `instance` variable to the `.detail.instance-id` JSON path from the event, and the `state` variable to the `.detail.state` JSON path. We then set the input template as "The EC2 instance <instance> has changed state to <state>.”

**Note**

For more information on the event transformer, see Transforming Target Input (p. 47)
Create a Rule

To customize instance state change information sent to a target using the input transformer

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Type a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Event pattern.
   b. Choose Pre-defined pattern by service.
   c. For Service provider, choose AWS.
   d. For Service Name, choose EC2.
   e. For Event type, choose EC2 Instance State-change Notification.
   f. Choose Any state, Any instance.
6. For Select event bus, choose AWS default event bus. When an AWS service in your account emits an event, it always goes to your account’s default event bus.
7. For Target, choose SNS topic.
8. For Topic, select the Amazon SNS topic that is to be notified when Amazon EC2 instances change state.
10. For Input Path, enter "{"state" : "$detail.state", "instance" : "$detail.instance-id"}.
11. For Input Template, enter "The EC2 instance <instance> has changed state to <state>.
12. Choose Create.

Tutorial: Log AWS API Calls Using EventBridge

You can use an AWS Lambda function that logs each AWS API call. For example, you can create a rule to log any operation in Amazon EC2, or you can limit this rule to log only a specific API call. In this tutorial, you log every time an Amazon EC2 instance is stopped.

Prerequisite

Before you can match these events, you must use AWS CloudTrail to set up a trail. If you don’t have a trail, complete the following procedure.

To create a trail

1. Open the CloudTrail console at https://console.aws.amazon.com/cloudtrail/.
2. Choose Trails, Create trail.
3. For Trail name, type a name for the trail.
4. For Storage location, in Create a new S3 bucket type the name for the new bucket that CloudTrail should deliver logs to.
5. Choose Create.

Step 1: Create an AWS Lambda Function

Create a Lambda function to log the API call events. Specify this function when you create your rule.
To create a Lambda function

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. If you're new to Lambda, you see a welcome page. Choose Get Started Now. Otherwise, choose Create a Lambda function.
3. On the Select blueprint page, enter hello for the filter and choose the hello-world blueprint.
4. On the Configure triggers page, choose Next.
5. On the Configure function page, do the following:
   a. Enter a name and description for the Lambda function. For example, name the function LogEC2StopInstance.
   b. Edit the sample code for the Lambda function. For example:
      ```javascript
      'use strict';
      exports.handler = (event, context, callback) => {
        console.log('LogEC2StopInstance');
        console.log('Received event:', JSON.stringify(event, null, 2));
        callback(null, 'Finished');
      };
      ```
   c. For Role, choose Choose an existing role. For Existing role, select your basic execution role. Otherwise, create a basic execution role.
   d. Choose Next.

Step 2: Create a Rule

Create a rule to run your Lambda function whenever you stop an Amazon EC2 instance.

To create a rule

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Event pattern.
   b. Choose Pre-defined pattern by service.
   c. For Service provider, choose AWS.
   d. For Service Name, choose EC2.
   e. For Event type, choose AWS API Call via CloudTrail.
   f. Choose Specific operations(s) and enter StopInstances in the box.
   g. By default, the rule matches any Amazon EC2 Auto Scaling group in the Region. To make the rule match a specific group, choose Specific group name(s) and select one or more groups.
6. For Select event bus, choose AWS default event bus. When an AWS service in your account emits an event, it always goes to your account's default event bus.
7. For Targets, choose Add target, Lambda function.
8. For Function, select the Lambda function that you created.
9. Choose Create.
Step 3: Test the Rule

You can test your rule by stopping an Amazon EC2 instance using the Amazon EC2 console. After waiting a few minutes for the instance to stop, check your AWS Lambda metrics on the CloudWatch console to verify that your function was invoked.

To test your rule by stopping an instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Launch an instance. For more information, see Launch Your Instance in the Amazon EC2 User Guide for Linux Instances.
3. Stop the instance. For more information, see Stop and Start Your Instance in the Amazon EC2 User Guide for Linux Instances.
4. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
5. In the navigation pane, choose Rules, choose the name of the rule that you created, and choose Metrics for the rule.
6. To view the output from your Lambda function, do the following:
   b. In the navigation pane, choose Logs.
   c. Select the name of the log group for your Lambda function (/aws/lambda/function-name).
   d. Select the name of the log stream to view the data provided by the function for the instance that you stopped.
7. (Optional) When you're finished, terminate the stopped instance. For more information, see Terminate Your Instance in the Amazon EC2 User Guide for Linux Instances.

Tutorial: Schedule Automated Amazon EBS Snapshots Using EventBridge

You can run EventBridge rules according to a schedule. In this tutorial, you create an automated snapshot of an existing Amazon Elastic Block Store (Amazon EBS) volume on a schedule. You can choose a fixed rate to create a snapshot every few minutes or use a cron expression to specify that the snapshot is made at a specific time of day.

Important
Creating rules with built-in targets is supported only on the AWS Management Console.

Step 1: Create a Rule

Create a rule that takes snapshots on a schedule. You can use a rate expression or a cron expression to specify the schedule. For more information, see Schedule Expressions for Rules (p. 32).

To create a rule

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Schedule.
b. Choose **Fixed rate every** and specify the schedule interval (for example, 5 minutes). Alternatively, choose **Cron expression** and specify a cron expression (for example, every 15 minutes Monday through Friday, starting at the current time).

6. For **Select event bus**, choose **AWS default event bus**. Scheduled rules are supported only on the default event bus.

7. For **Target**, choose **select EC2 CreateSnapshot API call**.

8. For **Volume ID**, enter the volume ID of the targeted Amazon EBS volume.

9. Choose **Create a new role for this specific resource**. The new role grants the target permissions to access resources on your behalf.

10. Choose **Create**.

### Step 2: Test the Rule

You can verify your rule by viewing your first snapshot after it’s taken.

**To test your rule**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Elastic Block Store, Snapshots**.
3. Verify that the first snapshot appears in the list.
4. (Optional) When you're finished, disable the rule to prevent additional snapshots from being taken.
   a. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
   b. In the navigation pane, choose **Rules**.
   c. Select the button next to the rule and choose **Disable**.
   d. When prompted for confirmation, choose **Disable**.

### Tutorial: Schedule AWS Lambda Functions Using EventBridge

You can set up a rule to run an AWS Lambda function on a schedule. This tutorial shows how to use the AWS Management Console or the AWS CLI to create the rule. If you want to use the AWS CLI but haven’t installed it, see the **AWS Command Line Interface User Guide**.

EventBridge doesn’t provide second-level precision in schedule expressions. The finest resolution using a cron expression is 1 minute. Due to the distributed nature of the EventBridge and the target services, the delay between the time the scheduled rule is triggered and the time the target service honors the execution of the target resource might be several seconds. Your scheduled rule is triggered in that minute but not on the precise zeroth second.

### Step 1: Create an AWS Lambda Function

Create a Lambda function to log the scheduled events. Specify this function when you create your rule.

**To create a Lambda function**

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. If you’re new to Lambda, you see a welcome page. Choose **Get Started Now**. Otherwise, choose **Create a Lambda function**.
Step 2: Create a Rule

Create a rule to run your Lambda function on a schedule.

To create a rule using the console

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Schedule.
   b. Choose Fixed rate every and specify the schedule interval (for example, 5 minutes).
6. For Select event bus, choose AWS default event bus. Scheduled rules are supported only on the default event bus.
7. For Target, choose Lambda function.
8. For Function, select the Lambda function that you created.
9. Choose Create.

If you prefer, you can create the rule using the AWS CLI. First, you must grant the rule permission to invoke your Lambda function. Then you can create the rule and add the Lambda function as a target.

To create a rule using the AWS CLI

1. To create a rule that triggers itself on a schedule, use the following put-rule command.

```bash
aws events put-rule \
  --name my-scheduled-rule \
  --schedule-expression 'rate(5 minutes)'
```
Step 3: Verify the Rule

At least 5 minutes after completing step 2, you can verify that your Lambda function was invoked.

To test your rule

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
   Choose the name of the rule that you created and choose Metrics for the rule.
3. To view the output from your Lambda function, do the following:
   a. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/
   b. In the navigation pane, choose Logs.
c. Select the name of the log group for your Lambda function (/aws/lambda/function-name).

d. Select the name of the log stream to view the data provided by the function for the instance that you launched.

Tutorial: Set AWS Systems Manager Automation as an EventBridge Target

You can use EventBridge to invoke AWS Systems Manager Automation on a regular timed schedule or when specified events are detected. This tutorial assumes that you’re invoking Systems Manager Automation based on certain events.

To create the EventBridge rule

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Event pattern
   b. Choose Pre-defined pattern by service.
   c. For Service provider, choose AWS.
   d. For Service Name and Event type, choose the service and event type to use as the trigger. Depending on the service and event type that you choose, you might need to specify additional options.
6. For Select event bus, choose AWS default event bus. When an AWS service in your account emits an event, it always goes to your account’s default event bus.
7. For Target, choose SSM Automation.
8. For Document, choose the Systems Manager document to run when the target is triggered.
9. (Optional) To specify a certain version of the document, choose Configure document version.
10. Under Configure automation parameter(s), choose No Parameter(s) or Constant.

   If you choose Constant, specify the constants to pass to the document execution.
11. EventBridge can create the IAM role needed for your event to run:
   - To create an IAM role automatically, choose Create a new role for this specific resource
   - To use an IAM role that you created before, choose Use existing role
12. Choose Create.

Tutorial: Relay Events to an Amazon Kinesis Stream Using EventBridge

You can relay AWS API call events in EventBridge to a stream in Amazon Kinesis.

Prerequisite

Install the AWS CLI. For more information, see the AWS Command Line Interface User Guide.
Step 1: Create an Amazon Kinesis Stream

To create a stream, use the following `create-stream` command.

```
aws kinesis create-stream --stream-name test --shard-count 1
```

When the stream status is `ACTIVE`, the stream is ready. To check the stream status, use the following `describe-stream` command.

```
aws kinesis describe-stream --stream-name test
```

Step 2: Create a Rule

As an example, create a rule to send events to your stream when you stop an Amazon EC2 instance.

To create a rule

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Event pattern.
   b. Choose Pre-defined pattern by service.
   c. For Service provider, choose AWS.
   d. For Service Name, choose EC2.
   e. For Event type, choose Instance State-change Notification.
   f. Choose Specific state(s), running.
6. For Select event bus, choose AWS default event bus. When an AWS service in your account emits an event, it always goes to your account's default event bus.
7. For Targets, choose Kinesis stream.
8. For Stream, select the stream that you created.
9. Choose Create a new role for this specific resource.
10. Choose Create.

Step 3: Test the Rule

To test your rule, stop an Amazon EC2 instance. After waiting a few minutes for the instance to stop, check your CloudWatch metrics to verify that your function was invoked.

To test your rule by stopping an instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Launch an instance. For more information, see Launch Your Instance in the Amazon EC2 User Guide for Linux Instances.
3. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
4. In the navigation pane, choose Rules.
   - Choose the name of the rule that you created and choose Metrics for the rule.
Step 4: Verify That the Event Is Relayed

You can get the record from the stream to verify that the event was relayed.

**To get the record**

1. To start reading from your Kinesis stream, use the following `get-shard-iterator` command.

   ```bash
   aws kinesis get-shard-iterator --shard-id shardId-000000000000 --shard-iterator-type TRIM_HORIZON --stream-name test
   
   The following is example output.
   ```
   ```json
   {
     "ShardIterator": "AAAAAAAAAAHSywljv02EgPX4NyKdZ5wryMzP9yALs8NeKbUjp1IxtZs1Sp
+KEd9I6AJj541NR1EMi+9m/d/nHvtLxysfxEzYkT4D9QVz/mBYWRO6OTZKsNo9gd
+efGNZaHFDkH1rJ4BL9Wyrk+ghYG22D2T1Da2EyNSH1+LAbK33gQweTJADB9yMwlo5r6PqcP2dzhg="
   }
   ```

2. To get the record, use the following `get-records` command. The shard iterator is the one that you got in the previous step.

   ```bash
   aws kinesis get-records --shard-iterator AAAAAAAAASywljv02EgPX4NyKdZ5wryMzP9yALs8NeKbUjp1IxtZs1Sp
+KEd9I6AJj541NR1EMi+9m/d/nHvtLxysfxEzYkT4D9QVz/mBYWRO6OTZKsNo9gd+efGNZaHFDkH1rJ4BL9Wyrk
+ghYG22D2T1Da2EyNSH1+LAbK33gQweTJADB9yMwlo5r6PqcP2dzhg=
   ```

   If the command is successful, it requests records from your stream for the specified shard. You can receive zero or more records. Any records returned might not represent all records in your stream. If you don't receive the data that you expect, keep calling `get-records`.

   Records in Kinesis are Base64-encoded. However, the streams support in the AWS CLI doesn't provide Base64 decoding. If you use a Base64 decoder to manually decode the data, you see that it's the event relayed to the stream in JSON form.

**Tutorial: Run an Amazon ECS Task When a File Is Uploaded to an Amazon S3 Bucket**

You can use EventBridge to run Amazon ECS tasks when certain AWS events occur. In this tutorial, you set up a EventBridge rule that runs an Amazon ECS task whenever a file is uploaded to a certain Amazon S3 bucket using the Amazon S3 PUT operation.

This tutorial assumes that you have already created the task definition in Amazon ECS.

**To run an Amazon ECS task whenever a file is uploaded to an S3 bucket using the PUT operation**

1. Open the Amazon EventBridge console at [https://console.aws.amazon.com/events/](https://console.aws.amazon.com/events/).
2. In the navigation pane, choose **Rules**.
3. Choose **Create rule**.
4. Enter a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Event pattern.
   b. Choose Pre-defined pattern by service.
   c. For Service provider, choose AWS.
   d. For Service Name, choose Simple Storage Service (S3).
   e. For Event type, choose Object Level Operations.
   f. Choose Specific operation(s), Put Object.
   g. Choose Specific bucket(s) by name and enter the name of the bucket.
6. For Select event bus, choose AWS default event bus. When an AWS service in your account emits an event, it always goes to your account’s default event bus.
7. For Targets, do the following:
   a. Choose ECS task.
   b. For Cluster and Task Definition, select the resources that you created.
   c. For Launch Type, choose FARGATE or EC2. FARGATE is shown only in Regions where AWS Fargate is supported.
   d. (Optional) Specify a value for Task Group. If the Launch Type is FARGATE, optionally specify a Platform Version. Specify only the numeric portion of the platform version, such as 1.1.0.
   e. (Optional) Specify a task definition revision and task count. If you don’t specify a task definition revision, the latest is used.
   f. If your task definition uses the awsvpc network mode, you must specify subnets and security groups. All subnets and security groups must be in the same VPC.
      If you specify more than one security group or subnet, separate them with commas but not spaces.
      For Subnets, specify the entire subnet-id value for each subnet, as in the following example.

      subnet-123abcd, subnet-789abcd
   g. Choose whether to allow the public IP address to be auto-assigned.
   h. EventBridge can create the IAM role needed for your task to run:
      • To create an IAM role automatically, choose Create a new role for this specific resource.
      • To use an IAM role that you created before, choose Use existing role. This must be a role that already has sufficient permissions to invoke the build. EventBridge doesn’t grant additional permissions to the role that you select.
8. Choose Create.

Tutorial: Schedule Automated Builds Using AWS CodeBuild

In the example in this tutorial, you schedule CodeBuild to run a build every week night at 20:00 GMT. You also pass a constant to CodeBuild to be used for this scheduled build.

To create a rule scheduling a CodeBuild project build nightly at 20:00 GMT

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Schedule.
   b. Choose Cron expression and specify the following as the expression: 0 20 ? * MON-FRI *.
      (for example, 5 minutes).
6. For Select event bus, choose AWS default event bus. When an AWS service in your account emits an event, it always goes to your account’s default event bus.
7. For Targets, choose CodeBuild project.
8. For Project ARN, enter the ARN of the build project.
9. In this tutorial, we add the optional step of passing a parameter to CodeBuild, to override the default. This isn't required when you set CodeBuild as the target. To pass the parameter, choose Configure input, Constant (JSON text).
   In the box under Constant (JSON text), enter the following to set the timeout override to 30 minutes for these scheduled builds: { "timeoutInMinutesOverride": 30 }.
   For more information about the parameters that you can pass, see StartBuild in the AWS CodeBuild API Reference. You can't pass the projectName parameter in this field. Instead, specify the project using the ARN in Project ARN.
10. EventBridge can create the IAM role needed for your build project to run:
   • To create an IAM role automatically, choose Create a new role for this specific resource.
   • To use an IAM role that you created before, choose Use existing role. This must be a role that already has sufficient permissions to invoke the build. EventBridge doesn't grant additional permissions to the role that you select.
11. Choose Create.

Tutorial: Log State Changes of Amazon EC2 Instances

In the example in this tutorial, you create a rule causing state-change notifications in Amazon EC2 to be logged in Amazon CloudWatch Logs.

To create a rule to log Amazon EC2 state-change notifications in CloudWatch Logs
1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Event pattern.
   b. Choose Pre-defined pattern by service.
   c. For Service provider, choose AWS.
   d. For Service Name, choose EC2.
   e. For Event type, choose EC2 Instance State-change Notification.
   f. Choose Any state and Any instance.
6. For Target, select CloudWatch log group.
7. For Log group, enter a name for the log group to receive the state-change notifications.
8. Choose Create.

Tutorial: Download Code Bindings for Events using the EventBridge Schema Registry

You can generate code bindings for event schemas to speed development for Java, Python, and TypeScript. You can get code bindings for existing AWS services, schemas you create, and for schemas you generate based on events on an event bus. You can generate code bindings for a schema using the EventBridge console, the EventBridge Schema Registry API, and directly in your IDE with the AWS Toolkit.

In this tutorial you will generate and download code bindings from an EventBridge Schema for the events of an AWS service.

To generate code bindings from an EventBridge schema

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Schemas.
3. Select the AWS event schema registry tab.
4. Find a schema for an AWS service that you would like code bindings for, either by browsing through the schema registry, or by searching for a schema.
5. Select the schema name to display the Schema details page.
6. In the Version section, select Download code bindings.
7. On the Download code bindings page, select the language of the code bindings you want to download.
8. Select Download.

It may take a few seconds for your download to begin. The download file will be a zip file of code bindings for the language you selected.

Use these code bindings in your own code to help quickly build applications using this EventBridge event.
Schedule Expressions for Rules

You can create rules that self-trigger on an automated schedule in EventBridge using cron or rate expressions. All scheduled events use UTC time zone and the minimum precision for schedules is 1 minute.

EventBridge supports cron expressions and rate expressions. Rate expressions are simpler to define but don’t offer the fine-grained schedule control that cron expressions support. For example, with a cron expression, you can define a rule that triggers at a specified time on a certain day of each week or month. In contrast, rate expressions trigger a rule at a regular rate, such as once every hour or once every day.

**Note**
EventBridge does not provide second-level precision in schedule expressions. The finest resolution using a cron expression is a minute. Due to the distributed nature of the EventBridge and the target services, the delay between the time the scheduled rule is triggered and the time the target service honors the execution of the target resource might be several seconds. Your scheduled rule is triggered within that minute, but not on the precise 0th second.

**Formats**
- Cron Expressions (p. 32)
- Rate Expressions (p. 34)

**Cron Expressions**
Cron expressions have six required fields, which are separated by white space.

**Syntax**

```
cron(fields)
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Values</th>
<th>Wildcards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minutes</td>
<td>0-59</td>
<td>,-<em>/</em></td>
</tr>
<tr>
<td>Hours</td>
<td>0-23</td>
<td>,-<em>/</em></td>
</tr>
<tr>
<td>Day-of-month</td>
<td>1-31</td>
<td>,-* ? / L W</td>
</tr>
<tr>
<td>Month</td>
<td>1-12 or JAN-DEC</td>
<td>,-*</td>
</tr>
<tr>
<td>Day-of-week</td>
<td>1-7 or SUN-SAT</td>
<td>,-* ? L #</td>
</tr>
<tr>
<td>Year</td>
<td>1970-2199</td>
<td>,-*</td>
</tr>
</tbody>
</table>

**Wildcards**
- The , (comma) wildcard includes additional values. In the Month field, JAN,FEB,MAR would include January, February, and March.
• The - (dash) wildcard specifies ranges. In the Day field, 1-15 would include days 1 through 15 of the specified month.
• The * (asterisk) wildcard includes all values in the field. In the Hours field, * would include every hour. You cannot use * in both the Day-of-month and Day-of-week fields. If you use it in one, you must use ? in the other.
• The / (forward slash) wildcard specifies increments. In the Minutes field, you could enter 1/10 to specify every tenth minute, starting from the first minute of the hour (for example, the 11th, 21st, and 31st minute, and so on).
• The ? (question mark) wildcard specifies one or another. In the Day-of-month field you could enter 7 and if you didn’t care what day of the week the 7th was, you could enter ? in the Day-of-week field.
• The L wildcard in the Day-of-month or Day-of-week fields specifies the last day of the month or week.
• The W wildcard in the Day-of-month field specifies a weekday. In the Day-of-month field, 3W specifies the weekday closest to the third day of the month.
• The # wildcard in the Day-of-week field specifies a certain instance of the specified day of the week within a month. For example, 3#2 would be the second Tuesday of the month: the 3 refers to Tuesday because it is the third day of each week, and the 2 refers to the second day of that type within the month.

Limitations

• You can’t specify the Day-of-month and Day-of-week fields in the same cron expression. If you specify a value (or a *) in one of the fields, you must use a ? (question mark) in the other.
• Cron expressions that lead to rates faster than 1 minute are not supported.

Examples

You can use the following sample cron strings when creating a rule with schedule.

<table>
<thead>
<tr>
<th>Minutes</th>
<th>Hours</th>
<th>Day of month</th>
<th>Month</th>
<th>Day of week</th>
<th>Year</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>*</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>Run at 10:00 am (UTC) every day</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>*</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>Run at 12:15 pm (UTC) every day</td>
</tr>
<tr>
<td>0</td>
<td>18</td>
<td>?</td>
<td>*</td>
<td>MON-FRI</td>
<td>*</td>
<td>Run at 6:00 pm (UTC) every Monday through Friday</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>1</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>Run at 8:00 am (UTC) every 1st day of the month</td>
</tr>
<tr>
<td>0/15</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>Run every 15 minutes</td>
</tr>
<tr>
<td>Minutes</td>
<td>Hours</td>
<td>Day of month</td>
<td>Month</td>
<td>Day of week</td>
<td>Year</td>
<td>Meaning</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>--------------</td>
<td>-------</td>
<td>-------------</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>0/10</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>MON-FRI</td>
<td>*</td>
<td>Run every 10 minutes Monday through Friday</td>
</tr>
<tr>
<td>0/5</td>
<td>8-17</td>
<td>?</td>
<td>*</td>
<td>MON-FRI</td>
<td>*</td>
<td>Run every 5 minutes Monday through Friday between 8:00 am and 5:55 pm (UTC)</td>
</tr>
</tbody>
</table>

The following examples show how to use Cron expressions with the AWS CLI `put-rule` command. The first example creates a rule that is triggered every day at 12:00pm UTC.

```shell
aws events put-rule --schedule-expression "cron(0 12 * * ? *)" --name MyRule1
```

The next example creates a rule that is triggered every day, at 5 and 35 minutes past 2:00pm UTC.

```shell
aws events put-rule --schedule-expression "cron(5,35 14 * * ? *)" --name MyRule2
```

The next example creates a rule that is triggered at 10:15am UTC on the last Friday of each month during the years 2019 to 2022.

```shell
```

### Rate Expressions

A rate expression starts when you create the scheduled event rule, and then runs on its defined schedule.

Rate expressions have two required fields. Fields are separated by white space.

**Syntax**

```
rate(value unit)
```

**value**

A positive number.

**unit**

The unit of time. Different units are required for values of 1, such as `minute`, and values over 1, such as `minutes`.

Valid values: `minute` | `minutes` | `hour` | `hours` | `day` | `days`
Limitations

If the value is equal to 1, then the unit must be singular. Similarly, for values greater than 1, the unit must be plural. For example, rate(1 hours) and rate(5 hour) are not valid, but rate(1 hour) and rate(5 hours) are valid.

Examples

The following examples show how to use rate expressions with the AWS CLI put-rule command. The first example triggers the rule minute, the next triggers it every five minutes, the third example triggers it once an hour, and the final example triggers it once per day.

```bash
aws events put-rule --schedule-expression "rate(1 minute)" --name MyRule2

aws events put-rule --schedule-expression "rate(5 minutes)" --name MyRule3

aws events put-rule --schedule-expression "rate(1 hour)" --name MyRule4

aws events put-rule --schedule-expression "rate(1 day)" --name MyRule5
```
Events and Event Patterns in EventBridge

Events in Amazon EventBridge are represented as JSON objects. All events have a similar structure, and the same top-level fields.

EventBridge rules use event patterns to match on AWS events on an event bus. When a pattern matches, the rule routes that event to a target.

Topics
- AWS Events (p. 36)
- Event Patterns (p. 37)
- Matching Null Values and Empty Strings in EventBridge Event Patterns (p. 41)
- Arrays in EventBridge Event Patterns (p. 42)
- Content-based Filtering with Event Patterns (p. 43)

AWS Events

The following is an example event in Amazon EventBridge.

```json
{
    "version": "0",
    "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
    "detail-type": "EC2 Instance State-change Notification",
    "source": "aws.ec2",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "us-west-1",
    "resources": [
        "arn:aws:ec2:us-west-1:123456789012:instance/ i-1234567890abcdef0"
    ],
    "detail": {
        "instance-id": " i-1234567890abcdef0",
        "state": "terminated"
    }
}
```

It is important to remember the following details about an event:

- They all have the same top-level fields – the ones appearing in the example above – which are never absent.
- The contents of the `detail` top-level field are different depending on which service generated the event and what the event is. The combination of the `source` and `detail-type` fields serves to identify the fields and values found in the `detail` field. For examples of events generated by AWS services, see EventBridge Event Examples from Supported AWS Services (p. 57).

Each event field is described below.

**version**

By default, this is set to 0 (zero) in all events.
id

A unique value is generated for every event. This can be helpful in tracing events as they move through rules to targets, and are processed.

detail-type

Identifies, in combination with the source field, the fields and values that appear in the detail field.

All events that are delivered via CloudTrail have AWS API Call via CloudTrail as the value for detail-type. For more information, see Events Delivered Via CloudTrail (p. 102).

source

Identifies the service that sourced the event. All events sourced from within AWS begin with "aws." Customer-generated events can have any value here, as long as it doesn’t begin with "aws." We recommend the use of Java package-name style reverse domain-name strings.

To find the correct value for source for an AWS service, see the table in AWS Service Namespaces. For example, the source value for Amazon CloudFront is aws.cloudfront.

account

The 12-digit number identifying an AWS account.

time

The event timestamp, which can be specified by the service originating the event. If the event spans a time interval, the service might choose to report the start time, so this value can be noticeably before the time the event is actually received.

region

Identifies the AWS region where the event originated.

resources

This JSON array contains ARNs that identify resources that are involved in the event. Inclusion of these ARNs is at the discretion of the service. For example, Amazon EC2 instance state-changes include Amazon EC2 instance ARNs, Auto Scaling events include ARNs for both instances and Auto Scaling groups, but API calls with AWS CloudTrail do not include resource ARNs.

detail

A JSON object, whose content is at the discretion of the service originating the event. The detail content in the example above is very simple, just two fields. AWS API call events have detail objects with around 50 fields nested several levels deep.

To see a list of all event types available from AWS services, see EventBridge Event Examples from Supported AWS Services (p. 57).

Event Patterns

Event patterns have the same structure as the Events they match. They look much like the events they are filtering. Rules use event patterns to select events and route them to targets. A pattern either matches an event or it doesn’t. The following is an example of a simple AWS Event which you might encounter on EventBridge.

```json
{
    "version": "0",
    "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
    "detail-type": "EC2 Instance State-change Notification",
    "detail": {}
}
```
Event patterns have the same structure as the events they match. For example, the following event pattern allows you to subscribe to only events from Amazon EC2.

```json
{
  "source": [ "aws.ec2" ]
}
```

The pattern simply quotes the fields you want to match and provides the values you are looking for.

The sample event above, like most events, has a nested structure. Suppose you want to process all instance-termination events. Create an event pattern like the following.

```json
{
  "source": [ "aws.ec2" ],
  "detail-type": [ "EC2 Instance State-change Notification" ],
  "detail": {
    "state": [ "terminated" ]
  }
}
```

**Specify Fields to Match**

Only specify fields that you care about. In the previous example, you only provide values for three fields: The top-level fields “source” and “detail-type”, and the “state” field inside the “detail” object field. EventBridge ignores all the other fields in the event while applying the filter.

**Match Values**

Match values are always in arrays. Note that the value to match is in a JSON array, surrounded by “[“ and “]”. This is so you can provide multiple values. For example, if you were interested in events from Amazon EC2 or Fargate, you could specify the following.

```json
{
  "source": [ "aws.ec2", "aws.fargate" ]
}
```

This matches on events where the value for the “source” field is either “aws.ec2” or “aws.fargate”.

**Match on All JSON Data Types**

You can match on all of the JSON data types. Consider the following example Amazon EC2 Auto Scaling event.
For the above example, you can match on the "responseElements" field as follows.

```
{
    "source": [ "aws.autoscaling" ],
    "detail-type": [ "EC2 Instance Launch Successful" ],
    "detail": {
        "responseElements": [ null ]
    }
}
```

This works for numbers too. Consider the following Amazon Macie event (truncated for brevity).

```
{
    "version": "0",
    "id": "3e355723-fca9-4de3-9fd7-154c289d6b59",
    "detail-type": "Macie Alert",
    "source": "aws.macie",
    "account": "123456789012",
    "time": "2017-04-24T22:28:49Z",
    "region": "us-east-1",
    "detail": {
        "notification-type": "ALERT_CREATED",
        "name": "Scanning bucket policies",
        "tags": [ "Custom_Alert", "Insider" ],
        "url": "https://lb00.us-east-1.macie.aws.amazon.com/1111222333/posts/alert_id",
        "risk-score": 80,
        "trigger": {
            "alert-type": "basic",
            "created-at": "2017-01-02 19:54:00.644000",
            "description": "Alerting on failed enumeration of large number of bucket policies"
        }
    },
    "created-at": "2017-04-18T00:21:12.059000",
    ...
}
```

If you want to match anything that has a risk score of 80 and a trigger risk of 8, do the following.
Simple Matching with Event Patterns

The following example event is used to show how the subsequent event patterns would match with this event JSON.

```json
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "EC2 Instance State-change Notification",
  "source": "aws.ec2",
  "account": "111122223333",
  "time": "2017-12-22T18:43:48Z",
  "region": "us-west-1",
  "resources": [
    "arn:aws:ec2:us-west-1:123456789012:instance/ i-1234567890abcdef0"
  ],
  "detail": {
    "instance-id": " i-1234567890abcdef0",
    "state": "terminated"
  }
}
```

Event patterns are represented as JSON objects with a structure that is similar to that of events, for example:

```json
{
  "source": [ "aws.ec2" ],
  "detail-type": [ "EC2 Instance State-change Notification" ],
  "detail": {
    "state": [ "running" ]
  }
}
```

This event pattern would not match on the example event, as the value of the "state" field matches on "running", but the value in the example event is "terminated".

It is important to remember the following about event pattern matching:

- For a pattern to match an event, the event must contain all the field names listed in the pattern. The field names must appear in the event with the same nesting structure.
- Other fields of the event not mentioned in the pattern are ignored; effectively, there is a "*": "*" wildcard for fields not mentioned.
- The matching is exact (character-by-character), without case-folding or any other string normalization.
- The values being matched follow JSON rules: Strings enclosed in quotes, numbers, and the unquoted keywords `true`, `false`, and `null`.
- Number matching is at the string representation level. For example, 300, 300.0, and 3.0e2 are not considered equal.
When you write patterns to match events, you can use the TestEventPattern API or the test-event-pattern CLI command to make sure that your pattern will match the desired events. For more information, see TestEventPattern or test-event-pattern.

The following event patterns would match the previous example event. The first pattern matches because one of the instance values specified in the pattern matches the event (and the pattern does not specify any additional fields not contained in the event). The second one matches because the "terminated" state is contained in the event.

```
{
    "resources": [
        "arn:aws:ec2:us-east-1:123456789012:instance/i-12345678",
        "arn:aws:ec2:us-east-1:123456789012:instance/i-abcdefgh"
    ]
}
```

```
{
    "detail": {
        "state": [ "terminated" ]
    }
}
```

These event patterns do not match the event at the top of this page. The first pattern does not match because the pattern specifies a "pending" value for state, and this value does not appear in the event. The second pattern does not match because the resource value specified in the pattern does not appear in the event.

```
{
    "source": [ "aws.ec2" ],
    "detail-type": [ "EC2 Instance State-change Notification" ],
    "detail": {
        "state": [ "pending" ]
    }
}
```

```
{
    "source": [ "aws.ec2" ],
    "detail-type": [ "EC2 Instance State-change Notification" ],
    "resources": [ "arn:aws:ec2:us-east-1::image/ami-12345678" ]
}
```

Matching Null Values and Empty Strings in EventBridge Event Patterns

You can create a pattern that matches an event field that has a null value or an empty string. To see how this works, consider the following example event:

```
{
    "version": "0",
    "id": "3e3c153a-8339-4e30-8c35-687ebebf853fe",
    "detail-type": "EC2 Instance Launch Successful",
    "source": "aws.autoscaling",
    "account": "123456789012",
}
```
Arrays in Event Patterns

The value of each field in a pattern is an array containing one or more values, and the pattern matches if any of the values in the array match the value in the event. If the value in the event is an array, then the pattern matches if the intersection of the pattern array and the event array is non-empty.

For example, consider an event pattern that includes the following.

```
"resources": [
    "arn:aws:ec2:us-east-1:123456789012:instance/i-b188560f",
    "arn:aws:ec2:us-east-1:111122223333:instance/i-b188560f",
    "arn:aws:ec2:us-east-1:444455556666:instance/i-b188560f",
]
```

This example pattern matches an event that includes the following text, because the first item in the pattern array matches the second item in the event array.

```
"resources": [
    "arn:aws:autoscaling:us-east-1:123456789012:autoScalingGroup:eb56d16b-bbf0-401d-b893-
    d5978e4a025:autoScalingGroupName/ASGTerminate",
    "arn:aws:ec2:us-east-1:123456789012:instance/i-b188560f"
]
```
Content-based Filtering with Event Patterns

Amazon EventBridge supports declarative filtering using event patterns. With event pattern content filtering you can write complex rules that only trigger under very specific conditions. For instance, you might want a rule that will trigger only when a field of the event is within a specific numeric range, if the event comes from a specific IP address, or only if a specific field does not exist in the event JSON. Content filtering allows you to create complex rules in your event patterns, so that the rule will only call a target if your filtering conditions are met.

Topics
- Prefix Matching (p. 43)
- Anything-but Matching (p. 43)
- Numeric Matching (p. 45)
- IP Address Matching (p. 45)
- Exists Matching (p. 45)
- Complex Example with Multiple Matching (p. 46)

Prefix Matching

You can match on the prefix of a value in the event source. For example, the following event pattern would match on any event where the "time" field started with "2017-10-02".

```
{   "time": [ { "prefix": "2017-10-02" } ],
}
```

The above event pattern would match on any event with that date in the time field, including "time": "2017-10-02T18:43:48Z".

**Note**
Prefix matching only works on string-valued fields.

Prefix Matching Example

Suppose you want to process all AWS Auto Scaling events from European regions. The following event pattern shows how to match on that.

```
{   "source": [ "aws.autoscaling" ],
    "region": [ { "prefix": "eu-" } ]
}
```

Anything-but Matching

Anything-but matching matches anything except what's provided in the rule.

You can use anything-but with strings and numeric values, including lists that contain only strings, or only numbers.

The following shows single anything-but matching, first with strings and then with numbers.

```
{}
```
"detail": {
    "state": [ { "anything-but": "initializing" } ]
}
}

{ "detail": {
    "x-limit": [ { "anything-but": 123 } ]
}
}

The following shows anything-but matching with a list of strings.

{ "detail": {
    "state": [ { "anything-but": [ "stopped", "overloaded" ] } ]
}
}

The following shows anything-but matching with a list of numbers.

{ "detail": {
    "x-limit": [ { "anything-but": [ 100, 200, 300 ] } ]
}
}

The following shows an anything-but event pattern that matches on a prefix. It will match on any event, except those that have the prefix init for the "state" field.

{ "detail": {
    "state": [ { "anything-but": { "prefix": "init" } ]
}
}

Anything-but Matching Example

Sometimes you want to exclude rather than include a particular field value. Suppose you want to process all events except those that are AWS CloudTrail reports of API calls.

{ "detail-type": [ { "anything-but": "AWS API Call via CloudTrail" } ]
}

The anything-but match expression can blacklist literal strings or also a list of values. The list must contain either all strings, or all numbers. To see all the events except those that came from Amazon EC2 or Amazon S3, do the following.

{ "source": [ { "anything-but": [ "aws.ec2", "aws.s3" ] } ]
}

The anything-but match expression can also use a nested match expression to exclude prefixes. For example, the EventBridge main event bus has a large number of events coming from all AWS services. But, you can also inject your own events using the PutEvents API. You can distinguish AWS events and process only your own because the "source" field in all AWS events begins with the string "aws.".
Numeric Matching

The following shows numeric matching for an event pattern.

```
{
  "detail": {
    "c-count": [ { "numeric": [ ">", 0, "\leq", 5 ] } ],
    "d-count": [ { "numeric": [ "<", 10 ] } ],
    "x-limit": [ { "numeric": [ "=", 3.018e2 ] } ]
  }
}
```

This pattern will only match evaluations that are true for all the fields. Numeric matching only works with values that are JSON numbers, and is limited to values between -1.0e9 and +1.0e9 inclusive, with 15 digits of precision (six digits to the right of the decimal point).

IP Address Matching

IP address matching is available for both IPv4 and IPv6 addresses.

```
{
  "detail": {
    "source-ip": [ { "cidr": "10.0.0.0/24" } ]
  }
}
```

Exists Matching

Exists matching works on the presence or absence of a field in the JSON of the event.

The following event pattern will match any event which does not have a `detail.c-count` field.

```
{
  "detail": {
    "c-count": [ { "exists": false } ]
  }
}
```

**Note**

Exists matching **only works on leaf nodes**. It does not work on intermediate nodes.

For example, the above pattern would match the following event.

```
{
  "detail-type": [ "EC2 Instance State-change Notification" ],
  "resources": [ "arn:aws:ec2:us-east-1:123456789012:instance/i-02ebd4584a2ebd341" ],
  "detail": {
    "state": [ "initializing", "running" ]
  }
}
```

But, it would also match the following event because `c-count` is not a leaf node.
Existence Matching Example

Suppose you wanted to make an Elasticsearch full-text index of a bunch of events. To do this, you select all the events that have a description field as follows.

```json
{  
  "detail": {  
    "description": [ { "exists": true } ],
  }
}
```

You could also use `{ "exists": false }` to select events that do not contain a particular field.

Complex Example with Multiple Matching

You can combine multiple matching rules into a more complex event pattern. For example, the following combines `anything-but` and `numeric` in a single event pattern.

```json
{  
  "time": [ { "prefix": "2017-10-02" } ],
  "detail": {  
    "state": [ { "anything-but": "initializing" } ],
    "c-count": [ { "numeric": [ "">", 0, "<=" , 5 ] } ],
    "d-count": [ { "numeric": [ "<", 10 ] } ],
    "x-limit": [ { "anything-but": [ 100, 200, 300 ] } ]
  }
}
```
Transforming Target Input

The Input transformer feature of EventBridge customizes the text from an event before it is passed to the target of a rule. You can define variables that use JSON path to reference values in the original event source. You can define multiple variables, assigning each a value from the input. Then you can use those variables in the Input Template as `<variable-name>`. The characters "<" and ">" cannot be escaped.

There are three pre-defined variables you can use without defining a JSON path. These variables are reserved, and you cannot create variables with these names.

- `aws.events.rule-arn` — The Amazon Resource Name (ARN) of the EventBridge rule.
- `aws.events.rule-name` — The Name of the EventBridge rule.
- `aws.events.event` — A copy of the original event.

Input Transform Examples

The following is an example Amazon EC2 event.

```json
{
   "version": "0",
   "id": "7bf73129-1428-4cd3-a780-95db273d1602",
   "detail-type": "EC2 Instance State-change Notification",
   "source": "aws.ec2",
   "account": "123456789012",
   "time": "2015-11-11T21:29:54Z",
   "region": "us-east-1",
   "resources": [
      "arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1111"
   ],
   "detail": {
      "instance-id": "i-0123456789",
      "state": "RUNNING"
   }
}
```

When defining a rule in the console, select the Input Transformer option under Configure input. This option displays two text boxes: one for the Input Path and one for the Input Template.

The Input Path is used to define variables. You use JSON path to reference items in your event and store those values in variables. For instance, you could create an Input Path to reference values in the example event by entering the following in the first text box.

```json
{
   "version": "0",
   "id": "7bf73129-1428-4cd3-a780-95db273d1602",
   "detail-type": "EC2 Instance State-change Notification",
   "source": "aws.ec2",
   "account": "123456789012",
   "time": "2015-11-11T21:29:54Z",
   "region": "us-east-1",
   "resources": [
      "arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1111"
   ],
   "detail": {
      "instance-id": "i-0123456789",
      "state": "RUNNING"
   }
}
```
This defines two variables, `<instance>` and `<state>`. You can reference these variables as you create your Input Template.

The Input Template is a template for the information you want to pass to your target. You can create a template that passes either a string or JSON to the target. Using the previous event and Input Path, the following Input Template examples will transform the event to the example output before routing it to a target.

<table>
<thead>
<tr>
<th>Description</th>
<th>Template</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple string</td>
<td>&quot;instance &lt;instance&gt; is in &lt;state&gt;&quot;</td>
<td>&quot;instance i-0123456789 is in RUNNING&quot;</td>
</tr>
<tr>
<td>String with escaped quotes</td>
<td>&quot;instance &quot;&lt;instance&gt;&quot; is in &lt;state&gt;&quot;</td>
<td>&quot;instance &quot;i-0123456789&quot; is in RUNNING&quot;</td>
</tr>
<tr>
<td>Simple JSON</td>
<td>{ &quot;instance&quot; : &lt;instance&gt;, &quot;state&quot;: &lt;state&gt; }</td>
<td>{ &quot;instance&quot; : &quot;i-0123456789&quot;, &quot;state&quot;: &quot;RUNNING&quot; }</td>
</tr>
<tr>
<td>JSON with a mix of variables and static information</td>
<td>{ &quot;instance&quot; : &lt;instance&gt;, &quot;state&quot;: [ 9, &lt;state&gt;, true ], &quot;Transformed&quot; : &quot;Yes&quot; }</td>
<td>{ &quot;instance&quot; : &quot;i-0123456789&quot;, &quot;state&quot;: [ 9, &quot;RUNNING&quot;, true ], &quot;Transformed&quot; : &quot;Yes&quot; }</td>
</tr>
<tr>
<td>Including reserved variables in JSON</td>
<td>{ &quot;instance&quot; : &lt;instance&gt;, &quot;state&quot;: &lt;state&gt;, &quot;ruleArn&quot; : &lt;aws.events.rule-arn&gt;, &quot;ruleName&quot; : &lt;aws.events.rule-name&gt;, &quot;originalEvent&quot; : &lt;aws.events.event&gt; }</td>
<td>{ &quot;instance&quot; : &quot;i-0123456789&quot;, &quot;state&quot;: &quot;RUNNING&quot;, &quot;ruleArn&quot; : &quot;arn:aws:events:us-east-2:123456789012:rule/example&quot;, &quot;ruleName&quot; : &quot;example&quot;, &quot;originalEvent&quot; : { ... // commented for brevity }</td>
</tr>
</tbody>
</table>
Transforming Input Using the EventBridge API

For information and examples on using input transformations with EventBridge API, see Use Input Transformer to extract data from an event and input that data to the target.

Common Issues with Transforming Input

These are some common issues when transforming input in EventBridge.

- For Strings, quotes are required.
- There is no validation when creating JSON path for your template.
- If you specify a variable to match a JSON path that doesn't exist in the event, that variable isn’t created and will not appear in the output.
- The JSON that is passed to the target is minified and escaped.
- EventBridge does not escape values extracted by Input Path, when populating the Input Template for a target.
- EventBridge does not support variables inside of quotes within JSON, such as: {"value" : "instance <instance> is in <state>"}.

<table>
<thead>
<tr>
<th>Description</th>
<th>Template</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Including reserved variables in a string</td>
<td><code>&lt;aws.events.rule-name&gt; triggered</code></td>
<td>&quot;example triggered&quot;</td>
</tr>
</tbody>
</table>
Amazon EventBridge Schema Registry

The EventBridge Schema Registry allows you to discover, create, and manage OpenAPI schemas for events on EventBridge. You can find schemas for existing AWS services, create and upload custom schemas, or generate a schema based on events on an event bus. For all schemas in EventBridge you can generate and download code bindings to help quickly build applications that use those events.

Schemas are available for the events of all AWS services on Amazon EventBridge. You can also create or upload schemas, or automatically infer schemas directly from events on an event bus. Once you have found or created a schema for an event, you can download code bindings for popular programming languages. You can browse, search, create, upload, and generate code bindings for schemas. You can manage schemas from the Amazon EventBridge console, using the API, or directly in your IDE using the AWS Toolkits. You can quickly build serverless apps that use events using the AWS Serverless Application Model.

For information about using the EventBridge Schema Registry with the API, or through Amazon CloudFront, see the following.

- Amazon EventBridge Schema Registry API Reference
- EventSchemas Resource Type Reference in AWS CloudFormation

Topics
- Discover Existing AWS Event Schemas (p. 50)
- Schema Registries (p. 51)
- Upload or Create Schemas (p. 51)
- Generate a Schema from Event JSON (p. 53)
- Generate a Schema Based on Events on an Event Bus (p. 55)
- Generate Code Bindings for EventBridge Schemas (p. 55)
- EventBridge Schema Registry Integration with AWS Toolkits (p. 56)

Discover Existing AWS Event Schemas

Amazon EventBridge includes schemas for all AWS services on EventBridge. You can search or browse these schemas in the Amazon EventBridge console, or by using API actions (see SearchSchemas).

To find schemas for AWS services

The Schema details page is displayed.

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Schemas.
3. On the Schemas page, select AWS event schema registry.

The first page of available schemas is displayed.
To search for a schema, enter a search term in the Search AWS event schemas. A search returns matches for both the name and contents of the available schemas, and will display which versions of the schema it was found in.

5. Open an event schema by selecting the name of the schema.

Schema Registries

Schema registries are containers for Schemas. Registries collect and organize schemas so that your schemas are in logical groups. You can view All schemas, or the built-in schemas, AWS event schema registry and Discovered schema registry. You can also create custom registries to collect and organize the schemas you create or upload.

To create a custom registry

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Schemas and then Create registry.
3. On the Registry details page enter a Name.
4. Optionally, enter a description for your new registry
5. Choose Create.

You can select Create custom schema from your new registry, or select that registry when you are creating a new schema.

To create a registry using the Amazon EventBridge Schema Registry API, use the CreateRegistry API action.

Upload or Create Schemas

Schemas are defined using JSON files, using the OpenAPI Specification. You can create or upload your own event schemas in EventBridge using this specification. You can download a template, or you can edit a template directly in the EventBridge console.

To create a schema from a downloaded template

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Schema registry.
3. In the Getting started section under Schema template, choose Download.
4. Alternatively, you can download the JSON from the following code example.

```json
{
    "openapi": "3.0.0",
    "info": {
        "version": "1.0.0",
        "title": "Event"
    },
```
5. Edit the template so that the schema matches your events. For more information on events, see Events and Event Patterns in EventBridge (p. 36).

**To upload a schema file**

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Schemas and then Create schema.
3. Optionally, select or create a schema registry.
4. Under Schema details enter a name for your schema.
5. Optionally, enter a description for the schema you created.
6. With the Create tab selected, either drag your schema file to the text box, or paste the schema source.
7. Select Create.

**To edit a template directly in the console**

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Schemas and then Create schema.
3. Optionally, select or create a schema registry.
4. Under Schema details enter a name for your schema.
5. Optionally, enter a description for the schema you created.
6. With the Create tab selected, choose Load template.
7. Edit the template so that the schema matches your events. For more information on events, see Events and Event Patterns in EventBridge (p. 36).

8. Select Create.

To create a schema using the EventBridge Schema Registry API, use the CreateSchema API action.

Generate a Schema from Event JSON

With the JSON of an event, you can automatically generate a schema for those types of events. Given existing event code, you can quickly generate a custom schema. Once the schema has been generated, you can download code bindings to help create applications for those types of events.

To generate an EventBridge schema based on event JSON

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Schemas and then Create schema.
3. Optionally, select or create a schema registry.
4. Under Schema details enter a name for your schema.
5. Optionally, enter a description for the schema you created.
6. Select Discover from JSON
7. In the text box under JSON, paste or drag the JSON source of an event.

As an example, you could paste in the source from this AWS Step Functions event for a failed execution.

```json
{
    "version": "0",
    "id": "315c1398-40ff-a850-213b-158f73e60175",
    "detail-type": "Step Functions Execution Status Change",
    "source": "aws.states",
    "account": "012345678912",
    "time": "2019-02-26T19:42:21Z",
    "region": "us-east-1",
    "resources": [
    ],
    "detail": {
        "name": "execution-name",
        "status": "FAILED",
        "startDate": 1551225146847,
        "stopDate": 1551225151881,
        "input": "{}",
        "output": null
    }
}
```

8. Select Discover schema.
9. EventBridge will generate an OpenAPI schema for the event. For example, the following is the generated schema for the event you pasted in.

```json
{
```

53
"openapi": "3.0.0",
"info": {
  "version": "1.0.0",
  "title": "StepFunctionsExecutionStatusChange"
},
"paths": {},
"components": {
  "schemas": {
    "AWSEvent": {
      "type": "object",
      "required": ["detail-type", "resources", "detail", "id", "source", "time",
                    "region", "version", "account"],
      "x-amazon-events-detail-type": "Step Functions Execution Status Change",
      "x-amazon-events-source": "aws.states",
      "properties": {
        "detail": {
          "$ref": "#/components/schemas/StepFunctionsExecutionStatusChange"
        },
        "account": {
          "type": "string"
        },
        "detail-type": {
          "type": "string"
        },
        "id": {
          "type": "string"
        },
        "region": {
          "type": "string"
        },
        "resources": {
          "type": "array",
          "items": {
            "type": "string"
          }
        },
        "source": {
          "type": "string"
        },
        "time": {
          "type": "string",
          "format": "date-time"
        },
        "version": {
          "type": "string"
        }
      }
    },
    "StepFunctionsExecutionStatusChange": {
      "type": "object",
      "required": ["output", "input", "executionArn", "name", "stateMachineArn",
                    "startDate", "stopDate", "status"],
      "properties": {
        "executionArn": {
          "type": "string"
        },
        "input": {
          "type": "string"
        },
        "name": {
          "type": "string"
        },
        "output": {},
        "startDate": {
          "type": "integer",
          "format": "int64"
Generate a Schema Based on Events on an Event Bus

Amazon EventBridge can infer schemas based on events on an event bus. Enabling event discovery on an event bus will generate schemas for events on that bus.

**Note**
Enabling event discovery on an event bus may incur a cost. The first five million ingested events in each month is free.

**To enable schema discovery on an event bus**
1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose **Event buses**.
3. To enable discovery on the **Default event bus**, select **Start discovery**.
4. To enable discovery on a **Custom event bus**, select the radio button for the custom event bus and choose **Start Discovery**.

Discovered schemas will show up in the **Discovered schemas registry** on the **Schemas** page. Changes to the contents of events on that bus will create new versions of the related EventBridge schema in the **Discovered schemas registry**.

Generate Code Bindings for EventBridge Schemas

You can generate code bindings for event schemas to speed development for Java, Python, and TypeScript. You can get code bindings for existing AWS services, schemas you create, and for schemas you generate based on events on an event bus. You can generate code bindings for a schema using the EventBridge console, the EventBridge Schema Registry API, and directly in your IDE with the AWS Toolkit.

You must enable discovery on an event bus to generate code bindings.

**To generate code bindings from an EventBridge schema**
1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose **Schemas**.

10. Once the schema has been generated, select **Create**.
3. Find a schema that you would like code bindings for, either by looking through the schema registries, or by searching for a schema.
4. Select the schema name to display the Schema details page.
5. In the Version section, select Download code bindings.
6. On the Download code bindings page, select the language of the code bindings you want to download.
7. Select Download.

   It may take a few seconds for your download to begin. The download file will be a zip file of code bindings for the language you selected.

EventBridge Schema Registry Integration with AWS Toolkits

EventBridge Schema Registry integrates with some AWS Toolkits, letting you browse or search for schemas and download code bindings for schemas directly in your IDE.

For more information, see the following AWS Toolkit documentation links.

- AWS Toolkit for JetBrains
- AWS Toolkit for VS Code
EventBridge Event Examples from Supported AWS Services

The AWS services in the following list emit events that can be detected by EventBridge.

Additionally, you can also use EventBridge with services that do not emit events and are not listed on this page, by watching for events delivered via CloudTrail. For more information, see Events Delivered Via CloudTrail (p. 102).

Event Types
- Amazon Augmented AI Events (p. 58)
- Application Auto Scaling Events (p. 58)
- AWS Batch Events (p. 58)
- Amazon EventBridge Scheduled Events (p. 58)
- Amazon Chime Events (p. 58)
- Events from CloudWatch (p. 58)
- CodeBuild Events (p. 59)
- CodeCommit Events (p. 59)
- AWS CodeDeploy Events (p. 59)
- CodePipeline Events (p. 60)
- AWS Config Events (p. 61)
- Amazon EBS Events (p. 61)
- Amazon EC2 Auto Scaling Events (p. 61)
- Amazon EC2 Spot Instance Interruption Events (p. 61)
- Amazon EC2 State Change Events (p. 62)
- Amazon Elastic Container Registry (Amazon ECR) Events (p. 62)
- Amazon Elastic Container Service (Amazon ECS) Events (p. 62)
- AWS Elemental MediaConvert Events (p. 62)
- AWS Elemental MediaPackage Events (p. 63)
- AWS Elemental MediaStore Events (p. 63)
- Amazon EMR Events (p. 63)
- Amazon GameLift Event (p. 65)
- AWS Glue Events (p. 72)
- AWS IoT Greengrass Events (p. 76)
- AWS Ground Station Events (p. 76)
- Amazon GuardDuty Events (p. 77)
- AWS Health Events (p. 77)
- AWS KMS Events (p. 78)
- Amazon Macie Events (p. 79)
- AWS Management Console Sign-in Events (p. 84)
- AWS OpsWorks Stacks Events (p. 85)
- Amazon SageMaker Events (p. 87)
- AWS Security Hub Events (p. 91)
- AWS Server Migration Service Events (p. 91)
Amazon Augmented AI Events

For examples of events generated by Amazon Augmented AI, see Use Events in Amazon Augmented AI.

Application Auto Scaling Events

For examples of events generated by Application Auto Scaling, see Application Auto Scaling Events and EventBridge.

AWS Batch Events

For examples of events generated by AWS Batch, see AWS Batch Events.

Amazon EventBridge Scheduled Events

The following is an example of a scheduled event:

```
{
  "id": "53dc4d37-cffa-4f76-80c9-8b7d4a4d2eaa",
  "detail-type": "Scheduled Event",
  "source": "aws.events",
  "account": "123456789012",
  "time": "2015-10-08T16:53:06Z",
  "region": "us-east-1",
  "resources": [ "arn:aws:events:us-east-1:123456789012:rule/MyScheduledRule" ],
  "detail": {}
}
```

Amazon Chime Events

For examples of events generated by Amazon Chime, see Automating Amazon Chime with EventBridge.

Events from CloudWatch

For sample events from CloudWatch, see Alarm Events and EventBridge in the AWS CodeBuild User Guide.
CodeBuild Events

For CodeBuild sample events, see Build Notifications Input Format Reference in the AWS CodeBuild User Guide.

CodeCommit Events

For CodeCommit sample events, see Monitoring CodeCommit Events in EventBridge and CloudWatch Events in the AWS CodeCommit User Guide.

AWS CodeDeploy Events

The following are examples of the events for CodeDeploy. For more information, see Monitoring Deployments with CloudWatch Events in the AWS CodeDeploy User Guide.

**CodeDeploy Deployment State-change Notification**

There was a change in the state of a deployment.

```json
{
  "account": "123456789012",
  "region": "us-east-1",
  "detail-type": "CodeDeploy Deployment State-change Notification",
  "source": "aws.codedeploy",
  "version": "0",
  "time": "2016-06-30T22:06:31Z",
  "id": "c071bfbf-83c4-49ca-a6ff-3df053957145",
  "resources": [
    "arn:aws:codedeploy:us-east-1:123456789012:application:myApplication",
  ],
  "detail": {
    "instanceGroupId": "9fd2fbef-2157-40d8-91e7-6845af69e2d2",
    "region": "us-east-1",
    "application": "myApplication",
    "deploymentId": "d-123456789",
    "state": "SUCCESS",
    "deploymentGroup": "myDeploymentGroup"
  }
}
```

**CodeDeploy Instance State-change Notification**

There was a change in the state of an instance that belongs to a deployment group.

```json
{
  "account": "123456789012",
  "region": "us-east-1",
  "detail-type": "CodeDeploy Instance State-change Notification",
  "source": "aws.codedeploy",
  "version": "0",
  "time": "2016-06-30T23:18:50Z",
  "id": "fb1d3015-c091-4bf9-95e2-d98521ab2ecb",
  "resources": [
    "arn:aws:ec2:us-east-1:123456789012:instance/i-0000000aaaabbbbbb",
  ]
}
```
CodePipeline Events

The following are examples of events for CodePipeline.

**Pipeline Execution State Change**

```
{
  "version": "0",
  "id": "CWE-event-id",
  "detail-type": "CodePipeline Pipeline Execution State Change",
  "source": "aws.codepipeline",
  "account": "123456789012",
  "time": "2017-04-22T03:31:47Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:codepipeline:us-east-1:123456789012:pipeline:myPipeline"
  ],
  "detail": {
    "pipeline": "myPipeline",
    "version": "1",
    "state": "STARTED",
    "execution-id": "01234567-0123-0123-0123-012345678901"
  }
}
```

**Stage Execution State Change**

```
{
  "version": "0",
  "id": "CWE-event-id",
  "detail-type": "CodePipeline Stage Execution State Change",
  "source": "aws.codepipeline",
  "account": "123456789012",
  "time": "2017-04-22T03:31:47Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:codepipeline:us-east-1:123456789012:pipeline:myPipeline"
  ],
  "detail": {
    "pipeline": "myPipeline",
    "version": "1",
    "state": "STARTED"
  }
}
```
**Action Execution State Change**

In this sample, there are two `region` fields. The one at the top is the name of the AWS Region where the action in the target pipeline is executed. In this example, this is `us-east-1`. The `region` in the detail section is the AWS Region where the event was created. This is the same as the Region where the pipeline was created. In this example, this is `us-west-2`.

```json
{
"version": "0",
"id": "CWE-event-id",
"detail-type": "CodePipeline Action Execution State Change",
"source": "aws.codepipeline",
"account": "123456789012",
"time": "2017-04-22T03:31:47Z",
"region": "us-east-1",
"resources": [
  "arn:aws:codepipeline:us-east-1:123456789012:pipeline:myPipeline"
],
"detail": {
  "pipeline": "myPipeline",
  "version": 1,
  "execution-id": "01234567-0123-0123-0123-012345678901",
  "stage": "Prod",
  "action": "myAction",
  "state": "STARTED",
  "region": "us-west-2",
  "type": {
    "owner": "AWS",
    "category": "Deploy",
    "provider": "CodeDeploy",
    "version": 1
  }
}
}
```

**AWS Config Events**

For information about the AWS Config events, see Monitoring AWS Config with Amazon CloudWatch Events in the AWS Config Developer Guide.

**Amazon EBS Events**

For information about the Amazon EBS events, see Amazon CloudWatch Events for Amazon EBS in the Amazon EC2 User Guide for Linux Instances.

**Amazon EC2 Auto Scaling Events**

For information about the Auto Scaling events, see Getting CloudWatch Events When Your Auto Scaling Group Scales in the Amazon EC2 Auto Scaling User Guide.

**Amazon EC2 Spot Instance Interruption Events**

For information about the events for Spot Instance interruptions, see Spot Instance Interruption Notices in the Amazon EC2 User Guide for Linux Instances.
Amazon EC2 State Change Events

The following is an example of the events for Amazon EC2 instances when the instance state changes.

**EC2 Instance State-change Notification**

This example is for an instance in the **pending** state. The other possible values for **state** include **running**, **shutting-down**, **stopped**, **stopping**, and **terminated**.

```json
{
    "id":"7bf73129-1428-4cd3-a780-95db273d1602",
    "detail-type":"EC2 Instance State-change Notification",
    "source":"aws.ec2",
    "account":"123456789012",
    "time":"2015-11-11T21:29:54Z",
    "region":"us-east-1",
    "resources":
    [
        "arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1111"
    ],
    "detail":{
        "instance-id":"i-abcd1111",
        "state":"pending"
    }
}
```

Amazon Elastic Container Registry (Amazon ECR) Events

Amazon ECR sends image actions events to EventBridge. Events are sent when images are pushed, scanned, or deleted.

For Amazon ECS sample events, see Amazon ECR Events in the Amazon Elastic Container Registry User Guide.

Amazon Elastic Container Service (Amazon ECS) Events

Amazon ECS sends two types of events to EventBridge: container instance events and task events. Container instance events are only sent if you are using the EC2 launch type for your tasks. For tasks using the Fargate launch type, you only receive task state events. Amazon ECS tracks the state of container instances and tasks. If either resources changes, an event is triggered. These events are classified as either container instance state change events or task state change events.

For Amazon ECS sample events, see Amazon ECS Events in the Amazon Elastic Container Service Developer Guide.

AWS Elemental MediaConvert Events

For MediaConvert sample events, see Using CloudWatch Events to Monitor AWS Elemental MediaConvert Jobs in the AWS Elemental MediaConvert User Guide.
AWSElementalMediaPackageEvents

ForMediaPackagesampleevents,seeMonitoringAWSElementalMediaPackagewithAmazonCloudWatchEventsintheAWSElementalMediaPackageUserGuide.

AWSElementalMediaStoreEvents

ForMediaStoresampleevents,seeAutomatingAWSElementalMediaStorewithCloudWatchEventsintheAWSElementalMediaStoreUserGuide.

AmazonEMREvents

EventsreportedbyAmazonEMRhaveaws.emrasthevalueforSource,whileAmazonEMRAPIeventsreroutedbyCloudTrailhaveaws.elasticmapreduceasthevalueforSource.

ThefollowingareexamplesofeventsreportedbyAmazonEMR.

AmazonEMRAutoScalingPolicyStateChange

```json
{
"version": "0",
"id": "2f8147ab-8c48-47c6-b0b6-3ee23ec8d300",
"detail-type": "EMR Auto Scaling Policy State Change",
"source": "aws.emr",
"account": "123456789012",
"time": "2016-12-16T20:42:44Z",
"region": "us-east-1",
"resources": [],
"detail": {
"resourceId": "ig-X2LBMHTGPCBU",
"clusterId": "j-1YONHTCP3YFKC",
"state": "PENDING",
"message": "AutoScaling policy modified by user request",
"scalingResourceType": "INSTANCE_GROUP"
}
}
```

AmazonEMRClusterStateChange–Starting

```json
{
"version": "0",
"id": "999cccaa-eaaa-0000-1111-123456789012",
"detail-type": "EMR Cluster State Change",
"source": "aws.emr",
"account": "123456789012",
"time": "2016-12-16T20:43:05Z",
"region": "us-east-1",
"resources": [],
"detail": {
"severity": "INFO",
"stateChangeReason": "{\"code\":\"\"}",
"name": "Development Cluster",
"clusterId": "j-123456789ABC",
"state": "STARTING",
"message": "Amazon EMR cluster j-123456789ABC (Development Cluster) was requested at 2016-12-16 20:42 UTC and is being created."
}
```
Amazon EMR Cluster State Change – Terminated

```json
{"version": "0",
"id": "1234abb0-f87e-1234-b7b6-000000123456",
"detail-type": "EMR Cluster State Change",
"source": "aws.emr",
"account": "123456789012",
"time": "2016-12-16T21:00:23Z",
"region": "us-east-1",
"resources": [],
"detail": {
  "severity": "INFO",
  "stateChangeReason": "{\"code\":\"USER_REQUEST\",\"message\":\"Terminated by user request\"}",
  "name": "Development Cluster",
  "clusterId": "j-123456789ABCD",
  "state": "TERMINATED",
  "message": "Amazon EMR Cluster jj-123456789ABCD (Development Cluster) has terminated at 2016-12-16 21:00 UTC with a reason of USER_REQUEST."
}
```

Amazon EMR Instance Group State Change

```json
{"version": "0",
"id": "999cccaa-eaaa-0000-1111-123456789012",
"detail-type": "EMR Instance Group State Change",
"source": "aws.emr",
"account": "123456789012",
"time": "2016-12-16T20:57:47Z",
"region": "us-east-1",
"resources": [],
"detail": {
  "market": "ON_DEMAND",
  "severity": "INFO",
  "requestedInstanceCount": "2",
  "instanceType": "m3.xlarge",
  "instanceGroupType": "CORE",
  "instanceGroupId": "ig-ABCDEFGHIJKL",
  "clusterId": "j-123456789ABCD",
  "runningInstanceCount": "2",
  "state": "RUNNING",
  "message": "The resizing operation for instance group ig-ABCDEFGHJKL in Amazon EMR cluster j-123456789ABCD (Development Cluster) is complete. It now has an instance count of 2. The resize started at 2016-12-16 20:57 UTC and took 0 minutes to complete."
}
```

Amazon EMR Step Status Change

```json
{"version": "0",
"id": "999cccaa-eaaa-0000-1111-123456789012",
"detail-type": "EMR Step Status Change",
"source": "aws.emr",
"account": "123456789012",
"time": "2016-12-16T20:53:09Z",
"region": "us-east-1",
"resources": []
}```
Amazon EventBridge User Guide

Amazon GameLift Event

The following are examples of Amazon GameLift events. For more information, see FlexMatch Events Reference in the Amazon GameLift Developer Guide.

Matchmaking Searching

```json
```

Potential Match Created

```json
```
"account": "123456789012",
"time": "2017-08-08T21:17:41.178Z",
"region": "us-west-2",
"resources": [
  "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/SampleConfiguration"
],
"detail": {
  "tickets": [
    {
      "ticketId": "ticket-1",
      "startTime": "2017-08-08T21:15:35.676Z",
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        }
      ],
    },
    {
      "ticketId": "ticket-2",
      "startTime": "2017-08-08T21:17:40.657Z",
      "players": [
        {
          "playerId": "player-2",
          "team": "blue"
        }
      ],
    }
  ],
  "acceptanceTimeout": 600,
  "ruleEvaluationMetrics": [
    {
      "ruleName": "EvenSkill",
      "passedCount": 3,
      "failedCount": 0
    },
    {
      "ruleName": "EvenTeams",
      "passedCount": 3,
      "failedCount": 0
    },
    {
      "ruleName": "FastConnection",
      "passedCount": 3,
      "failedCount": 0
    },
    {
      "ruleName": "NoobSegregation",
      "passedCount": 3,
      "failedCount": 0
    }
  ],
  "acceptanceRequired": true,
  "type": "PotentialMatchCreated",
  "gameSessionInfo": {
    "players": [
      {
        "playerId": "player-1",
        "team": "red"
      },
      {
        "playerId": "player-2",
        "team": "blue"
      }
    ]
  }
}
Accept Match

```json
{
  "version": "0",
  "id": "b3f76d66-c8e5-416a-aa4c-aa1278153edc",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-09T20:04:42.660Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/SampleConfiguration"
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-09T20:01:35.305Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      },
      {
        "ticketId": "ticket-2",
        "startTime": "2017-08-09T20:04:16.637Z",
        "players": [
          {
            "playerId": "player-2",
            "team": "blue",
            "accepted": false
          }
        ]
      }
    ],
    "type": "AcceptMatch",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        },
        {
          "playerId": "player-2",
          "team": "blue",
          "accepted": false
        }
      ]
    },
    "matchId": "848b5f1f-0460-488e-8631-2960934d13e5"
  }
}
```

Accept Match Completed

```json
{
  "version": "0",
  "id": "b1990d3d-f737-4d6c-b150-af5ace8c35d3",
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-09T20:01:35.305Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      },
      {
        "ticketId": "ticket-2",
        "startTime": "2017-08-09T20:04:16.637Z",
        "players": [
          {
            "playerId": "player-2",
            "team": "blue",
            "accepted": false
          }
        ]
      }
    ],
    "type": "AcceptMatch",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        },
        {
          "playerId": "player-2",
          "team": "blue",
          "accepted": false
        }
      ]
    },
    "matchId": "848b5f1f-0460-488e-8631-2960934d13e5"
  }
}
```
"detail-type": "GameLift Matchmaking Event",
"source": "aws.gamelift",
"account": "123456789012",
"time": "2017-08-08T20:43:14.621Z",
"region": "us-west-2",
"resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/SampleConfiguration"
],
"detail": {
    "tickets": [
        {
            "ticketId": "ticket-1",
            "startTime": "2017-08-08T20:30:40.972Z",
            "players": [
                {
                    "playerId": "player-1",
                    "team": "red"
                }
            ]
        },
        {
            "ticketId": "ticket-2",
            "startTime": "2017-08-08T20:33:14.111Z",
            "players": [
                {
                    "playerId": "player-2",
                    "team": "blue"
                }
            ]
        }
    ],
    "acceptance": "TimedOut",
    "type": "AcceptMatchCompleted",
    "gameSessionInfo": {
        "players": [
            {
                "playerId": "player-1",
                "team": "red"
            },
            {
                "playerId": "player-2",
                "team": "blue"
            }
        ],
        "matchId": "a0d9bd24-4695-4f12-876f-ea6386dd6dce"
    }
}
"ticketId": "ticket-1",
"startTime": "2017-08-09T19:58:59.277Z",
"players": [
  {
    "playerId": "player-1",
    "playerSessionId": "psess-6e7c13cf-10d6-4756-a53f-db7de782ed67",
    "team": "red"
  }
],
"ticketId": "ticket-2",
"startTime": "2017-08-09T19:59:08.663Z",
"players": [
  {
    "playerId": "player-2",
    "playerSessionId": "psess-786b342f-9c94-44eb-bb9e-c1de46c472ce",
    "team": "blue"
  }
],
"type": "MatchmakingSucceeded",
"gameSessionInfo": {
  "gameSessionArn": "arn:aws:gamelift:us-west-2:123456789012:gamesession/836cf48d-bcb0-4a2c-bec1-9c456541352a",
  "ipAddress": "192.168.1.1",
  "port": 10777,
  "players": [
    {
      "playerId": "player-1",
      "playerSessionId": "psess-6e7c13cf-10d6-4756-a53f-db7de782ed67",
      "team": "red"
    },
    {
      "playerId": "player-2",
      "playerSessionId": "psess-786b342f-9c94-44eb-bb9e-c1de46c472ce",
      "team": "blue"
    }
  ],
  "matchId": "c0ec1a54-7fec-4b55-8583-76d67adb7754"
}

Matchmaking Timed Out

{
  "version": "0",
  "id": "fe528a7d-46ad-4bdc-96cb-b094b5f6bf36",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-09T20:11:35.598Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/SampleConfiguration"
  ],
  "detail": {
    "reason": "TimedOut",
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-09T20:01:35.305Z",
        "players": [
          
        ]
      }
    ]
  }
}
Matchmaking Cancelled

```json
{
  "version": "0",
  "id": "8d6f84da-5e15-4741-8d5c-5ac99091c27f",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-09T20:00:07.843Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/SampleConfiguration"
  ],
  "detail": {
    "reason": "Cancelled",
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-09T19:59:26.118Z",
        "players": [
          {
            "playerId": "player-1"
          }
        ]
      }
    ]
  }
}
```
Matchmaking Failed

```json
{
  "version": "0",
  "id": "025b55a4-41ac-4cf4-89d1-f2b3c6fd8f9d",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-16T18:41:09.970Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/SampleConfiguration"
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-16T18:41:02.631Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      }
    ],
    "customEventData": "foo",
    "type": "MatchmakingFailed",
```
"reason": "UNEXPECTED_ERROR",
"message": "An unexpected error was encountered during match placing.",
"gameSessionInfo": {
  "players": [
    {"playerId": "player-1",
      "team": "red"
    }
  ],
  "matchId": "3ea83c13-218b-43a3-936e-135cc570c9a7"
}

AWS Glue Events

The following is the format for AWS Glue events.

Successful Job Run

{
  "version": "0",
  "id": "abcdef00-1234-5678-9abc-def012345678",
  "detail-type": "Glue Job State Change",
  "source": "aws.glue",
  "account": "123456789012",
  "time": "2017-09-07T18:57:21Z",
  "region": "us-west-2",
  "resources": [],
  "detail": {
    "jobName": "MyJob",
    "severity": "INFO",
    "state": "SUCCEEDED",
    "jobRunId": "jr_abcdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789",
    "message": "Job run succeeded"
  }
}

Failed Job Run

{
  "version": "0",
  "id": "abcdef01-1234-5678-9abc-def012345678",
  "detail-type": "Glue Job State Change",
  "source": "aws.glue",
  "account": "123456789012",
  "time": "2017-09-07T06:02:03Z",
  "region": "us-west-2",
  "resources": [],
  "detail": {
    "jobName": "MyJob",
    "severity": "ERROR",
    "state": "FAILED",
    "jobRunId": "jr_0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcde",
    "message": "JobName:MyJob and JobRunId:jr_0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcde failed to execute with exception Role arn:aws:iam::123456789012:role/Glue_Role should be given assume role permissions for Glue Service."
  }
}
Timeout

```
{
  "version":"0",
  "id":"abcdef00-1234-5678-9abc-def012345678",
  "detail-type":"Glue Job State Change",
  "source":"aws.glue",
  "account":"123456789012",
  "time":"2017-11-20T20:22:06Z",
  "region":"us-east-1",
  "resources":[],
  "detail":{
    "jobName":"MyJob",
    "severity":"WARN",
    "state":"TIMEOUT",
    "jobRunId":"jr_abc0123456789abcdef0123456789abcdef0123456789abcdef0123456789def",
    "message":"Job run timed out"
  }
}
```

Stopped Job Run

```
{
  "version":"0",
  "id":"abcdef00-1234-5678-9abc-def012345678",
  "detail-type":"Glue Job State Change",
  "source":"aws.glue",
  "account":"123456789012",
  "time":"2017-11-20T20:22:06Z",
  "region":"us-east-1",
  "resources":[],
  "detail":{
    "jobName":"MyJob",
    "severity":"INFO",
    "state":"STOPPED",
    "jobRunId":"jr_abc0123456789abcdef0123456789abcdef0123456789abcdef0123456789def",
    "message":"Job run stopped"
  }
}
```

Crawler Started

```
{
  "version":"0",
  "id":"05efe8a2-c309-6884-a41b-3508bdc96995",
  "detail-type":"Glue Crawler State Change",
  "source":"aws.glue",
  "account":"561226563745",
  "time":"2017-11-11T01:09:46Z",
  "region":"us-east-1",
  "resources":[
    ],
  "detail":{
    "accountId":"561226563745",
    "crawlerName":"S3toS3AcceptanceTestCrawlera470bd94-9e00-4518-8942-e80c8431c322",
    "startTime":"2017-11-11T01:09:46Z",
    "state":"STARTED",
    "message":"Crawler Started"
  }
}
```

Crawler Succeeded
Amazon EventBridge User Guide
AWS Glue Events

```

Crawler Failed

```

```

Job Run is in Starting State

```
"time":"2018-04-24T20:57:34Z",
"region":"us-east-1",
"resources":[]
"detail":{
  "jobName":"MyJob",
  "severity":"INFO",
  "notificationCondition":{
    "NotifyDelayAfter":1.0
  },
  "state":"STARTING",
  "jobRunId":"jr_6aa58e7a3aa44e2e4c7db2c50e2f7396cb57901729e4b702dcdb2cfbbbe3f7a86",
  "message":"Job is in STARTING state",
  "startedOn":"2018-04-24T20:55:47.941Z"
}

Job Run is in Running State

{
  "version":"0",
  "id":"66fbc5e1-aac3-5e85-63d0-856ec669a050",
  "detail-type":"Glue Job Run Status",
  "source":"aws.glue",
  "account":"123456789012",
  "time":"2018-04-24T20:57:34Z",
  "region":"us-east-1",
  "resources":[]
  "detail":{
    "jobName":"MyJob",
    "severity":"INFO",
    "notificationCondition":{
      "NotifyDelayAfter":1.0
    },
    "state":"RUNNING",
    "jobRunId":"jr_6aa58e7a3aa44e2e4c7db2c50e2f7396cb57901729e4b702dcdb2cfbbbe3f7a86",
    "message":"Job is in RUNNING state",
    "startedOn":"2018-04-24T20:55:47.941Z"
}

Job Run is in Stopping State

{
  "version":"0",
  "id":"66fbc5e1-aac3-5e85-63d0-856ec669a050",
  "detail-type":"Glue Job Run Status",
  "source":"aws.glue",
  "account":"123456789012",
  "time":"2018-04-24T20:57:34Z",
  "region":"us-east-1",
  "resources":[]
  "detail":{
    "jobName":"MyJob",
    "severity":"INFO",
    "notificationCondition":{
      "NotifyDelayAfter":1.0
    },
    "state":"STOPPING",
    "jobRunId":"jr_6aa58e7a3aa44e2e4c7db2c50e2f7396cb57901729e4b702dcdb2cfbbbe3f7a86",
    "message":"Job is in STOPPING state",
    "startedOn":"2018-04-24T20:55:47.941Z"
}
**AWS Glue Data Catalog Table State Change**

```json
{
  "version": "0",
  "id": "2617428d-715f-edef-70b8-d210da0317a0",
  "detail-type": "Glue Data Catalog Table State Change",
  "source": "aws.glue",
  "account": "123456789012",
  "time": "2019-01-16T18:16:01Z",
  "region": "eu-west-1",
  "resources": [
    "arn:aws:glue:eu-west-1:123456789012:table/d1/t1"
  ],
  "detail": {
    "databaseName": "d1",
    "changedPartitions": [
      "[C.pdf, dir3]",
      "[D.doc, dir4]"
    ],
    "typeOfChange": "BatchCreatePartition",
    "tableName": "t1"
  }
}
```

**AWS Glue Data Catalog Database State Change**

In the following example, the `typeOfChange` is `CreateTable`. Other possible values for this field are `CreateDatabase` and `UpdateTable`.

```json
{
  "version": "0",
  "id": "60e7ddc2-a588-5328-220a-21c060f6c3f4",
  "detail-type": "Glue Data Catalog Database State Change",
  "source": "aws.glue",
  "account": "123456789012",
  "time": "2019-01-16T18:08:48Z",
  "region": "eu-west-1",
  "resources": [
    "arn:aws:glue:eu-west-1:123456789012:table/d1/t1"
  ],
  "detail": {
    "databaseName": "d1",
    "typeOfChange": "CreateTable",
    "changedTables": [
      "t1"
    ]
  }
}
```

**AWS IoT Greengrass Events**

For information about AWS IoT Greengrass events, see Get Deployment Notifications in the *AWS IoT Greengrass Developer Guide*.

**AWS Ground Station Events**

For information about example AWS Ground Station events, see Automating AWS Ground Station with CloudWatch Events in the *AWS Ground Station User Guide*.
Amazon GuardDuty Events

For information about example Amazon GuardDuty events, see Monitoring Amazon GuardDuty with Amazon CloudWatch Events in the Amazon GuardDuty User Guide.

AWS Health Events

The following is the format for the AWS Personal Health Dashboard (AWS Health) events. For more information, see Managing AWS Health Events with Amazon CloudWatch Events in the AWS Health User Guide.

**AWS Health Event Format**

```json
{
  "version": "0",
  "id": "7bf73129-1428-4cd3-a780-95db273d1602",
  "detail-type": "AWS Health Event",
  "source": "aws.health",
  "account": "123456789012",
  "time": "2016-06-05T06:27:57Z",
  "region": "region",
  "resources": [],
  "detail": {
    "eventArn": "arn:aws:health:region::event/id",
    "service": "service",
    "eventTypeCode": "AWS_service_code",
    "eventTypeCategory": "category",
    "startTime": "Sun, 05 Jun 2016 05:01:10 GMT",
    "endTime": "Sun, 05 Jun 2016 05:30:57 GMT",
    "eventDescription": [
      {
        "language": "lang-code",
        "latestDescription": "description"
      }
    ]
  }
}
```

**eventTypeCategory**

The category code of the event. The possible values are `issue`, `accountNotification`, and `scheduledChange`.

**eventTypeCode**

The unique identifier for the event type. Examples include `AWS_EC2_INSTANCE_NETWORK_MAINTENANCE_SCHEDULED` and `AWS_EC2_INSTANCE_REBOOT_MAINTENANCE_SCHEDULED`. Events that include `MAINTENANCE_SCHEDULED` are usually pushed out about two weeks before the `startTime`.

**id**

The unique identifier for the event.

**service**

The AWS service affected by the event. For example, EC2, S3, REDSHIFT, or RDS.

**Elastic Load Balancing API Issue**

```json
{}
```
“version”: "0",
“id”: "121345678-1234-1234-1234-123456789012",
“detail-type”: "AWS Health Event",
“source”: "aws.health",
“account”: "123456789012",
“time”: "2016-06-05T06:27:57Z",
“region”: "ap-southeast-2",
“resources”: [],
“detail”: {
  "eventArn": "arn:aws:health:ap-southeast-2::event/AWS_ELASTICLOADBALANCING_API_ISSUE_90353408594353980",
  "service": "ELASTICLOADBALANCING",
  "eventTypeCode": "AWS_ELASTICLOADBALANCING_API_ISSUE",
  "eventTypeCategory": "issue",
  "startTime": "Sat, 11 Jun 2016 05:01:10 GMT",
  "endTime": "Sat, 11 Jun 2016 05:30:57 GMT",
  "eventDescription": [{
    "language": "en_US",
    "latestDescription": "A description of the event will be provided here"
  }]
}

Amazon EC2 Instance Store Drive Performance Degraded

{
 “version”: "0",
 “id”: "121345678-1234-1234-1234-123456789012",
 “detail-type”: "AWS Health Event",
 “source”: "aws.health",
 “account”: "123456789012",
 “time": "2016-06-05T06:27:57Z",
 “region": "us-west-2",
 “resources": ["i-abcd1111"],
 “detail": {
  "eventArn": "arn:aws:health:us-west-2::event/AWS_EC2_INSTANCE_STORE_DRIVE_PERFORMANCE_DEGRADED_90353408594353980",
  "service": "EC2",
  "eventTypeCode": "AWS_EC2_INSTANCE_STORE_DRIVE_PERFORMANCE_DEGRADED",
  "eventTypeCategory": "issue",
  "startTime": "Sat, 05 Jun 2016 15:10:09 GMT",
  "eventDescription": [{
    "language": "en_US",
    "latestDescription": "A description of the event will be provided here"
  }],
  "affectedEntities": [{
    "entityValue": "i-abcd1111",
    "tags": {
      "stage": "prod",
      "app": "my-app"
    }
  }
}

AWS KMS Events

The following are examples of the AWS Key Management Service (AWS KMS) events. For more information, see AWS KMS Events in the AWS Key Management Service Developer Guide.

KMS CMK Rotation

AWS KMS automatically rotated a CMK’s key material.
Amazon EventBridge User Guide
Amazon Macie Events

KMS Imported Key Material Expiration
AWS KMS deleted a CMK's expired key material.

KMS CMK Deletion
AWS KMS completed a scheduled CMK deletion.

The following are examples of Amazon Macie events.
Alert Created

```
{
  "version": "0",
  "id": "CWE-event-id",
  "detail-type": "Macie Alert",
  "source": "aws.macie",
  "account": "123456789012",
  "time": "2017-04-24T22:28:49Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:macie:us-east-1:123456789012:trigger/trigger_id"
  ],
  "detail": {
    "notification-type": "ALERT_CREATED",
    "name": "Scanning bucket policies",
    "tags": [
      "Custom_Alert",
      "Insider"
    ],
    "url": "https://lb00.us-east-1.macie.aws.amazon.com/1111222333/posts/alert_id",
    "risk-score": 80,
    "trigger": {
      "alert-type": "basic",
      "created-at": "2017-01-02 19:54:00.644000",
      "description": "Alerting on failed enumeration of large number of bucket policies",
      "risk": 8
    },
    "created-at": "2017-04-18T00:21:12.059000",
    "actor": "555566667777:assumed-role:superawesome:aroaidp1dc7nsefsnheji",
    "summary": {
      "Description": "Alerting on failed enumeration of large number of bucket policies",
      "IP": {
        "34.199.185.34": 121,
        "34.205.153.2": 2,
        "72.21.196.70": 2
      },
      "Time Range": [
        {
          "count": 125,
          "start": "2017-04-24T20:23:49Z",
          "end": "2017-04-24T20:25:54Z"
        }
      ],
      "Source ARN": "arn:aws:sts::123456789012:assumed-role/RoleName",
      "Record Count": 1,
      "Location": {
        "us-east-1": 125
      },
      "Event Count": 125,
      "Events": {
        "GetBucketLocation": {
          "count": 48,
          "ISP": {
            "Amazon": 48
          }
        },
        "ListRoles": {
          "count": 2,
          "ISP": {
            "Amazon": 2
          }
        }
      }
    }
  }
}```
null
Alert Updated

{
"version": "0",
"id": "CWE-event-id",
"detail-type": "Macie Alert",
"source": "aws.macie",
"account": "123456789012",
"time": "2017-04-18T17:47:48Z",
"region": "us-east-1",
"resources": [
  "arn:aws:macie:us-east-1:123456789012:trigger/trigger_id"
],
"detail": {
  "notification-type": "ALERT_UPDATED",
  "name": "Public bucket contains high risk object",
  "tags": [
    "Custom_Alert",
    "Audit"
  ],
  "url": "https://lb00.us-east-1.macie.aws.amazon.com/111122223333/posts/alert_id",
  "risk-score": 100,
  "trigger": {
    "alert-type": "basic",
    "created-at": "2017-04-08 00:23:39.138000",
    "description": "Public bucket contains high risk object",
    "risk": 10
  },
  "created-at": "2017-04-08T00:23:39.138000",
  "actor": "public_bucket",
  "summary": {
    "Description": "Public bucket contains high risk object",
  }
}
"Object": {
  "public_bucket/secret_key.txt": 1,
  "public_bucket/financial_summary.txt": 1
},
"Record Count": 2,
"Themes": {
  "Secret Markings": 1,
  "Corporate Proposals": 1,
  "Confidential Markings": 1
},
"Event Count": 2,
"DLP risk": {
  "7": 2
},
"Owner": {
  "bucket_owner": 2
},
"Timestamps": {
  "2017-04-03T16:12:53+00:00": 2
}
}

{
  "version": "0",
  "id": "CWE-event-id",
  "detail-type": "Macie Alert",
  "source": "aws.macie",
  "account": "123456789012",
  "time": "2017-04-22T03:31:47Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:macie:us-east-1:123456789012:trigger/macie"
  ],
  "detail": {
    "notification-type": "ALERT_UPDATED",
    "name": "Lists the instance profiles that have the specified associated IAM role, Lists the names of the inline policies that are embedded in the specified IAM role",
    "tags": [
      "Predictive",
      "Behavioral_Anomaly"
    ],
    "url": "https://lb00.us-east-1.macie.aws.amazon.com/11112223333/posts/alert_id",
    "risk-score": 20,
    "created-at": "2017-04-22T03:08:35.256000",
    "actor": "123456789012:assumed-role:rolename",
    "trigger": {
      "alert-type": "predictive",
      "features": {
        "distinctEventName": {
          "name": "distinctEventName",
          "description": "Event Names executed during a user session",
          "narrative": "A sudden increase in event names utilized by a user can be an indicator of a change in user behavior or account risk",
          "risk": 3
        },
        "ListInstanceProfilesForRole": {
          "name": "ListInstanceProfilesForRole",
          "description": "Lists the instance profiles that have the specified associated IAM role",
          "narrative": "Lists the instance profiles that have the specified associated IAM role",
          "risk": 5
        }
      }
    }
  }
}
AWS Management Console Sign-in Events

AWS Management Console sign-in events can be detected by CloudWatch Events. For information on regional sign-in events see: Logging Regional Sign-in Events.

The following is an example of a console sign-in event:

```json
{
    "id": "6f87d04b-9f74-4f04-a780-7acf4b0a9b38",
    "detail-type": "AWS Console Sign In via CloudTrail",
    "source": "aws.signin",
    "account": "123456789012",
    "time": "2016-01-05T18:21:27Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "eventVersion": "1.02",
        "userIdentity": {
            "type": "Root",
            "principalId": "123456789012",
            "arn": "arn:aws:iam::123456789012:root",
            "accountId": "123456789012"
        },
        "eventTime": "2016-01-05T18:21:27Z",
        "eventSource": "signin.amazonaws.com",
        "eventName": "ConsoleLogin",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "0.0.0.0",
        "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36",
        "requestParameters": null,
        "responseElements": {
            "ConsoleLogin": "Success"
        }
    }
}
```
AWS OpsWorks Stacks Events

The following are examples of AWS OpsWorks Stacks events.

**AWS OpsWorks Stacks instance state change**

Indicates a change in the state of an AWS OpsWorks Stacks instance. The following are instance states.

- booting
- connection_lost
- online
- pending
- rebooting
- requested
- running_setup
- setup_failed
- shutting_down
- start_failed
- stopping
- stop_failed
- stopped
- terminating
- terminated

```json
{
    "version": "0",
    "id": "dc5fa8df-48f1-2108-b1b9-1fe5ebcf2296",
    "detail-type": "OpsWorks Instance State Change",
    "source": "aws.opsworks",
    "account": "123456789012",
    "time": "2018-01-25T11:12:23Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:opsworks:us-east-1:123456789012:instance/a648d98f-fdd8-4323-952a-a503e4z500z"
    ],
    "detail": {
        "initiated_by": "user",
        "hostname": "testing1",
        "stack-id": "acd3df16-e859-4598-8414-377b12a902da",
        "layer-ids": [
            "d1a0cb7f-c7e9-4a63-811c-976f0267b2c8"
        ],
        "instance-id": "a648d98f-fdd8-4323-952a-a503e4z500z",
        "ec2-instance-id": "i-08b1c2b67aa292276",
        "status": "requested"
    }
}
```
The `initiated_by` field is only populated when the instance is in the requested, terminating, or stopping states. The `initiated_by` field can contain one of the following values.

- **user** - A user requested the instance state change by using either the API or AWS Management Console.
- **auto-scaling** - The AWS OpsWorks Stacks automatic scaling feature initiated the instance state change.
- **auto-healing** - The AWS OpsWorks Stacks automatic healing feature initiated the instance state change.

**AWS OpsWorks Stacks command state change**

A change occurred in the state of an AWS OpsWorks Stacks command. The following are command states.

- **expired** - A command timed out.
- **failed** - A general command failure occurred.
- **skipped** - A command was skipped because the instance has a different state in AWS OpsWorks Stacks than in Amazon EC2.
- **successful** - A command succeeded.
- **superseded** - A command was skipped because it would have applied configuration changes that have already been applied.

```
{
  "version": "0",
  "id": "96c778b6-a40e-c8c1-aafe-c9852a3a7b52",
  "detail-type": "OpsWorks Command State Change",
  "source": "aws.opsworks",
  "account": "123456789012",
  "time": "2018-01-26T08:54:40Z",
  "region": "us-east-1",
  "resources": ["arn:aws:opsworks:us-east-1:123456789012:instance/a648d98f-fdd8-4323-952a-a50a3e4e500f"],
  "detail": {
    "command-id": "acc9f4f3-a3ec-4fab-b70f-c7d04e71e3ec",
    "instance-id": "a648d98f-fdd8-4323-952a-a50a3e4e500f",
    "type": "setup",
    "status": "successful"
  }
}
```

**AWS OpsWorks Stacks deployment state change**

A change occurred in the state of an AWS OpsWorks Stacks deployment. The following are deployment states.

- **running**
- **successful**
- **failed**

```
{
  "version": "0",
  "id": "96c778b6-a40e-c8c1-aafe-c9852a3a7b52",
  "detail-type": "OpsWorks Deployment State Change",
  "source": "aws.opsworks",
  "account": "123456789012",
  "time": "2018-01-26T08:54:40Z",
  "region": "us-east-1",
  "resources": ["arn:aws:opsworks:us-east-1:123456789012:instance/a648d98f-fdd8-4323-952a-a50a3e4e500f"],
  "detail": {
    "deployment-id": "de9662a2-fc4c-4a84-a731-9f4b42a4f588",
    "instance-count": 1,
    "instance-type": "t2.micro",
    "status": "deployed"
  }
}
```
The duration field is only populated when a deployment is finished, and shows time in seconds.

AWS OpsWorks Stacks alert

An AWS OpsWorks Stacks service error was raised.

```
{  
  "version": "0",  
  "id": "f99faa6f-0e27-e398-95bb-8f190806d275",  
  "detail-type": "OpsWorks Alert",  
  "source": "aws.opsworks",  
  "account": "123456789012",  
  "time": "2018-01-20T16:51:29Z",  
  "region": "us-east-1",  
  "resources": [],  
  "detail": {  
    "stack-id": "2f48f2be-ac7d-4dd5-80bb-88375f94db7b",  
    "instance-id": "986efb74-69e8-4c6d-878e-5b77c084cb0",  
    "type": "InstanceStop",  
    "message": "The shutdown of the instance timed out. Please try stopping it again."
  }
}
```

Amazon SageMaker Events

The following are examples of Amazon SageMaker events.

Amazon SageMaker training job state change

Indicates a change in the status of an Amazon SageMaker training job.

If the value of TrainingJobStatus is Failed, the event contains the FailureReason field, which provides a description of why the training job failed.

```
{  
  "version": "0",  
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",  
  "detail-type": "SageMaker Training Job State Change",  
  "source": "aws.sagemaker",  
  "account": "123456789012",
}
```
Amazon SageMaker HyperParameter tuning job state change

Indicates a change in the status of an Amazon SageMaker hyperparameter tuning job.
{
    "version": "0",
    "id": "84e2571-85d4-695f-b930-0153b71dcb42",
    "detail-type": "SageMaker HyperParameter Tuning Job State Change",
    "source": "aws.sagemaker",
    "account": "123456789012",
    "time": "2018-10-06T12:26:13Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:sagemaker:us-east-1:123456789012:tuningJob/x"
    ],
    "detail": {
        "HyperParameterTuningJobName": "016bffd3-6d71-4d3a-9710-0a332b2759fc",
        "HyperParameterTuningJobArn": "arn:aws:sagemaker:us-east-1:123456789012:tuningJob/x",
        "TrainingJobDefinition": {
            "StaticHyperParameters": {},
            "AlgorithmSpecification": {
                "TrainingImage": "trainingImageName",
                "TrainingInputMode": "inputModeFile",
                "MetricDefinitions": [
                    {
                        "Name": "metricName",
                        "Regex": "regex"
                    }
                ]
            },
            "RoleArn": "roleArn",
            "InputDataConfig": [
                {
                    "ChannelName": "channelName",
                    "DataSource": {
                        "S3DataSource": {
                            "S3DataFormat": "s3DataType",
                            "S3Uri": "s3Uri",
                            "S3DataDistributionType": "s3DistributionType"
                        }
                    },
                    "ContentType": "contentType",
                    "CompressionType": "gz",
                    "RecordWrapperType": "RecordWrapper"
                }
            ],
            "VpcConfig": {
                "SecurityGroupIds": [<securityGroupIds>],
                "Subnets": [<subnets>]
            },
            "OutputDataConfig": {
                "KmsKeyId": "kmsKeyId",
                "S3OutputPath": "s3OutputPath"
            },
            "ResourceConfig": {
                "InstanceType": "instanceType",
                "InstanceCount": 10,
                "VolumeSizeInGB": 500,
                "VolumeKmsKeyId": "volumeKeyId"
            },
            "StoppingCondition": {
                "MaxRuntimeInSeconds": 3600
            }
        },
        "HyperParameterTuningJobStatus": "status",
    }
}
Amazon EventBridge User Guide
Amazon SageMaker Events

Amazon SageMaker transform job state change

Indicates a change in the state of an Amazon SageMaker batch transform job.

If the value of TransformJobStatus is Failed, the event contains the FailureReason field, which provides a description of why the training job failed.

```json
{
    "version": "0",
    "id": "844e2571-85d4-695f-b930-0153b71dcb42",
    "detail-type": "SageMaker Transform Job State Change",
    "source": "aws.sagemaker",
    "account": "123456789012",
    "time": "2018-10-06T12:26:13Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:sagemaker:us-east-1:123456789012:transform-job/myjob"
    ],
    "detail": {
        "TransformJobName": "4b52bd8f-e034-4345-818d-884b9d7c9724",
        "TransformJobStatus": "Completed",
        "ModelName": "ModelName",
        "MaxConcurrentTransforms": 5,
        "MaxPayloadInMB": 10,
        "BatchStrategy": "Strategy",
        "Environment": {
            "environment1": "environment2"
        },
        "TransformInput": {
            "DataSource": {
                "S3DataSource": {
                    "S3DataType": "s3DataType",
                    "S3Uri": "s3Uri"
                },
                "ContentType": "content type",
                "CompressionType": "compression type",
                "SplitType": "split type"
            },
            "TransformOutput": {
                "S3OutputPath": "s3Uri",
                "Accept": "accept",
                "AssembleWith": "assemblyType",
                "KmsKeyId": "kmsKeyId"
            }
        }
    }
}
```
AWS Security Hub Events

For information about example Security Hub events, see Monitoring AWS Security Hub with Amazon CloudWatch Events in the AWS Security Hub User Guide.

AWS Server Migration Service Events

The following are examples of the events for AWS Server Migration Service.

Deleted replication job notification

```
{
  "version": "0",
  "id": "5630992d-92cd-439f-f2a8-92c8212aee24",
  "detail-type": "Server Migration Job State Change",
  "source": "aws.sms",
  "account": "123456789012",
  "time": "2018-02-07T22:30:11Z",
  "region": "us-west-1",
  "resources": [
    "arn:aws:sms:us-west-1:123456789012:sms-job-21a64348"
  ],
  "detail": {
    "state": "Deleted",
    "replication-run-id": "N/A",
    "replication-job-id": "sms-job-21a64348",
    "version": "1.0"
  }
}
```

Completed replication job notification

```
{
  "version": "0",
  "id": "3f9c59cc-f941-522a-be6d-f08e44ff1715",
  "detail-type": "Server Migration Job State Change",
  "source": "aws.sms",
  "account": "123456789012",
  "time": "2018-02-07T22:30:11Z",
  "region": "us-west-1",
  "resources": [
    "arn:aws:sms:us-west-1:123456789012:sms-run-e1a64388"
  ],
  "detail": {
```
AWS Systems Manager Events

The following are examples of the events for AWS Systems Manager. For more information, see Log Command Execution Status Changes for Run Command in the Amazon EC2 User Guide for Linux Instances.

Run Command Status-change Notification

```
{
    "version": "0",
    "id": "51c0891d-0e34-45b1-83d6-95db273d1602",
    "detail-type": "EC2 Command Status-change Notification",
    "source": "aws.ssm",
    "account": "123456789012",
    "time": "2016-07-10T21:51:32Z",
    "region": "us-east-1",
    "resources": ["arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1111"],
    "detail": {
        "command-id": "e8d3c0e4-71f7-4491-898f-c9b35bee5f3b",
        "document-name": "AWS-RunPowerShellScript",
        "expire-after": "2016-07-14T22:01:30.049Z",
        "parameters": {
            "executionTimeout": ["3600"],
            "commands": ["date"]
        },
        "requested-date-time": "2016-07-10T21:51:30.049Z",
        "status": "Success"
    }
}
```

Run Command Invocation Status-change Notification

```
{
    "version": "0",
    "id": "4780e1b8-f56b-4de5-95f2-95db273d1602",
    "detail-type": "EC2 Command Invocation Status-change Notification",
    "source": "aws.ssm",
    "account": "123456789012",
    "time": "2016-07-10T21:51:32Z",
    "region": "us-east-1",
    "resources": ["arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1111"],
    "detail": {
        "command-id": "e8d3c0e4-71f7-4491-898f-c9b35bee5f3b",
        "document-name": "AWS-RunPowerShellScript",
        "instance-id": "i-9bb89e2b",
        "requested-date-time": "2016-07-10T21:51:30.049Z",
        "status": "Success"
    }
}
```

Automation Step Status-change Notification
Automation Execution Status-change Notification

```
{
  "version": "0",
  "id": "d290ece9-1088-4383-9df6-cd5b4ac42b99",
  "detail-type": "EC2 Automation Execution Status-change Notification",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2016-11-29T19:43:35Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ssm:us-east-1:123456789012:automation-execution/333ba70b-2333-48db-b17e-a5e69c6f4d1c",
  ],
  "detail": {
    "ExecutionId": "333ba70b-2333-48db-b17e-a5e69c6f4d1c",
    "Definition": "runcommand1",
    "DefinitionVersion": 1.0,
    "Status": "Success",
    "EndTime": "Nov 29, 2016 7:43:25 PM",
    "StartTime": "Nov 29, 2016 7:43:23 PM",
    "Time": 2630.0,
    "StepName": "runFixedCmds",
    "Action": "aws:runCommand"
  }
}
```

State Manager Association State Change

```
{
  "version": "0",
  "id": "db839caf-6f6c-40af-9a48-25b2ae2b7774",
  "detail-type": "EC2 State Manager Association State Change",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2017-05-16T23:01:10Z",
  "region": "us-west-1",
  "resources": [
    "arn:aws:ssm:us-west-1::document/AWS-RunPowerShellScript"
  ],
  "detail": {
```
```
State Manager Instance Association State Change

```
{
  "version": "0",
  "id": "6a7e8f6b-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "EC2 State Manager Instance Association State Change",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2017-02-23T15:23:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ec2:us-east-1:123456789012:instance/i-12345678",
  ],
  "detail": {
    "association-id": "34fcb7e0-9a14-49b4-9989-0e4ef50f8d8",
    "instance-id": "i-12345678",
    "document-name": "my-custom-document",
    "document-version": "1",
    "targets": [{"key": "instanceids", "values": ["i-12345678"]}],
    "creation-date": "2017-02-23T15:23:48Z",
    "last-successful-execution-date": "2017-02-23T16:23:48Z",
    "last-execution-date": "2017-02-23T16:23:48Z",
    "status": "Success",
    "detailed-status": "",
    "error-code": "testErrorCode",
    "execution-summary": "testExecutionSummary",
    "output-url": "sampleurl",
    "instance-association-cwe-version": "1"
  }
}
```

AWS Systems Manager Configuration Compliance Events

The following are examples of the events for Amazon EC2 Systems Manager (SSM) configuration compliance.

**Association Compliant**

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "associate": {
    "association-id": "6e37940a-23ba-4ab0-9b96-5d0a1a05464f",
    "document-name": "AWS-RunPowerShellScript",
    "association-version": "1",
    "document-version": "Optional.empty",
    "targets": [{"key": "InstanceIds", "values": ["i-12345678"]}],
    "creation-date": "2017-02-13T17:22:54.458Z",
    "last-successful-execution-date": "2017-05-16T23:00:01Z",
    "last-execution-date": "2017-05-16T23:00:01Z",
    "last-updated-date": "2017-02-13T17:22:54.458Z",
    "status": "Success",
    "association-status-aggregated-count": "{"Success":1}"
  }
}
```
"detail-type": "Configuration Compliance State Change",
"source": "aws.ssm",
"account": "123456789012",
"time": "2017-07-17T19:03:26Z",
"region": "us-west-1",
"resources": [
   "arn:aws:ssm:us-west-1:461348341421:managed-instance/i-01234567890abcdef"
],
"detail": {
   "last-runtime": "2017-01-01T10:10:10Z",
   "compliance-status": "compliant",
   "resource-type": "managed-instance",
   "resource-id": "i-01234567890abcdef",
   "compliance-type": "Association"
}
}

Association Non-Compliant

{
"version": "0",
"id": "01234567-0123-0123-0123-012345678901",
"detail-type": "Configuration Compliance State Change",
"source": "aws.ssm",
"account": "123456789012",
"time": "2017-07-17T19:02:31Z",
"region": "us-west-1",
"resources": [
   "arn:aws:ssm:us-west-1:461348341421:managed-instance/i-01234567890abcdef"
],
"detail": {
   "last-runtime": "2017-01-01T10:10:10Z",
   "compliance-status": "non_compliant",
   "resource-type": "managed-instance",
   "resource-id": "i-01234567890abcdef",
   "compliance-type": "Association"
}

Patch Compliant

{
"version": "0",
"id": "01234567-0123-0123-0123-012345678901",
"detail-type": "Configuration Compliance State Change",
"source": "aws.ssm",
"account": "123456789012",
"time": "2017-07-17T19:03:26Z",
"region": "us-west-1",
"resources": [
   "arn:aws:ssm:us-west-1:461348341421:managed-instance/i-01234567890abcdef"
],
"detail": {
   "resource-type": "managed-instance",
   "resource-id": "i-01234567890abcdef",
   "compliance-status": "compliant",
   "compliance-type": "Patch",
   "patch-baseline-id": "PB789",
   "severity": "critical"
}

Patch Non-Compliant
AWS Systems Manager Maintenance Windows Events

The following are examples of the events for Systems Manager Maintenance Windows.

Register a Target

The status could also be DEREGISTERED.

Window Execution Type

The other possibilities for status are PENDING, IN_PROGRESS, SUCCESS, FAILED, TIMED_OUT, and SKIPPED_OVERLAPPING.
Task Execution Type

The other possibilities for status are IN_PROGRESS, SUCCESS, FAILED, and TIMED_OUT.

Task Target Processed

The other possibilities for status are IN_PROGRESS, SUCCESS, FAILED, and TIMED_OUT.
Window State Change

The possibilities for status are ENABLED and DISABLED.

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-0123456789ab",
  "detail-type": "Maintenance Window State-change Notification",
  "source": "aws.ssm",
  "account": "012345678901",
  "time": "2016-11-16T00:58:37Z",
  "region": "us-east-1",
  "resources": [
  ],
  "detail": {
    "window-id": "mw-123456789012",
    "status": "DISABLED"
  }
}
```

AWS Systems Manager Parameter Store Events

The following are examples of the events for Amazon EC2 Systems Manager (SSM) Parameter Store.

Create Parameter

```
{
  "version": "0",
  "id": "6a7e4feb-b491-4cf7-a9f1-bf3703497718",
  "detail-type": "Parameter Store Change",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2017-05-22T16:43:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ssm:us-east-1:123456789012:parameter/foo"
  ],
  "detail": {
    "operation": "Create",
    "name": "foo",
    "type": "String",
    "description": "Sample Parameter"
  }
}
```

Update Parameter

```
{
  "version": "0",
  "id": "6a7e4feb-b491-4cf7-a9f1-bf3703497718",
  "detail-type": "Parameter Store Change",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2017-05-22T16:43:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ssm:us-east-1:123456789012:parameter/foo"
  ],
  "detail": {
    "operation": "Update",
    "name": "foo",
    "type": "String",
    "description": "Updated Parameter"
  }
}
```
"id": "9547ef2d-3b7e-4057-b6cb-5fdf09ee7c8f",
"detail-type": "Parameter Store Change",
"source": "aws.ssm",
"account": "123456789012",
"time": "2017-05-22T16:44:48Z",
"region": "us-east-1",
"resources": [
  "arn:aws:ssm:us-east-1:123456789012:parameter/foo"
],
"detail": {
  "operation": "Update",
  "name": "foo",
  "type": "String",
  "description": "Sample Parameter"
}
}

Delete Parameter

{
"version": "0",
"id": "80e9b391-6a9b-413c-839a-453b528053af",
"detail-type": "Parameter Store Change",
"source": "aws.ssm",
"account": "123456789012",
"time": "2017-05-22T16:45:48Z",
"region": "us-east-1",
"resources": [
  "arn:aws:ssm:us-east-1:123456789012:parameter/foo"
],
"detail": {
  "operation": "Delete",
  "name": "foo",
  "type": "String",
  "description": "Sample Parameter"
}
}

AWS Step Functions Events

For Step Functions sample events, see Step Functions Event Examples in the AWS Step Functions Developer Guide.

Tag Change Events on AWS Resources

The following is an example of a tag event.

{
"version": "0",
"id": "ffd8a6fe-32f8-ef66-c85c-111111111111",
"detail-type": "Tag Change on Resource",
"source": "aws.tag",
"account": "123456789012",
"time": "2018-09-18T20:41:06Z",
"region": "us-east-1",
"resources": [
  "arn:aws:ec2:us-east-1:123456789012:instance/i-0000000aaaaaaa"
]
AWS Trusted Advisor Events

The following are examples of the events for AWS Trusted Advisor. For more information, see Monitoring Trusted Advisor Check Results with Amazon CloudWatch Events in the AWS Support User Guide.

Low Utilization Amazon EC2 Instances

```json
{
    "version": "0",
    "id": "1234abcd-ab12-123a-1233-1234567890ab",
    "detail-type": "Trusted Advisor Check Item Refresh Notification",
    "source": "aws.trustedadvisor",
    "account": "123456789012",
    "time": "2018-01-12T20:07:49Z",
    "region": "us-east-2",
    "resources": [],
    "detail": {
        "check-name": "Low Utilization Amazon EC2 Instances",
        "check-item-detail": {
            "Day 1": "0.1% 0.00MB",
            "Day 2": "0.1% 0.00MB",
            "Day 3": "0.1% 0.00MB",
            "Region/AZ": "ca-central-1a",
            "Estimated Monthly Savings": "$9.22",
            "14-Day Average CPU Utilization": "0.1%",
            "Day 14": "0.1% 0.00MB",
            "Day 13": "0.1% 0.00MB",
            "Day 12": "0.1% 0.00MB",
            "Day 11": "0.1% 0.00MB",
            "Day 10": "0.1% 0.00MB",
            "14-Day Average Network I/O": "0.00MB",
            "Number of Days Low Utilization": "14 days",
            "Instance Type": "t2.micro",
            "Instance ID": "i-0123456789abcdef",
            "Day 8": "0.1% 0.00MB",
            "Instance Name": null,
            "Day 9": "0.1% 0.00MB",
            "Day 4": "0.1% 0.00MB",
            "Day 5": "0.1% 0.00MB",
            "Day 6": "0.1% 0.00MB",
            "Day 7": "0.1% 0.00MB"
        },
        "status": "WARN",
        "uuid": "aa12345f-55c7-498e-b7ac-123456789012"
    }
}
```
Exposed Access Keys

```json
{
  "version": "0",
  "id": "1234abcd-ab12-123a-123a-1234567890ab",
  "detail-type": "Trusted Advisor Check Item Refresh Notification",
  "source": "aws.trustedadvisor",
  "account": "123456789012",
  "time": "2018-01-12T20:07:03Z",
  "region": "us-east-2",
  "resources": [],
  "detail": {
    "check-name": "Exposed Access Keys",
    "check-item-detail": {
      "Case ID": "12345678-1234-1234-abcd-123456789012",
      "User Name (IAM or Root)": "my-username",
      "Deadline": "1440021299248",
      "Access Key ID": "AKIAIOSFODNN7EXAMPLE",
      "Time Updated": "1440021299248",
      "Fraud Type": "Exposed",
      "Location": "www.example.com"
    },
    "status": "ERROR",
    "resource_id": "",
    "uuid": "aa12345f-55c7-498e-b7ac-123456789012"
  }
}
```
Amazon WorkSpaces Events

For information about the Amazon WorkSpaces events, see Monitor Your WorkSpaces Using CloudWatch Events in the Amazon WorkSpaces Administration Guide.

Events Delivered Via CloudTrail

You can also use EventBridge with services that do not emit events and are not listed on this page. AWS CloudTrail is a service that automatically records events such as AWS API calls. You can create EventBridge rules that trigger on the information captured by CloudTrail. For more information about CloudTrail, see What is AWS CloudTrail?. For more information about creating a EventBridge rule that uses CloudTrail, see Creating an EventBridge Rule That Triggers on an AWS API Call Using AWS CloudTrail (p. 6).

All events that are delivered via CloudTrail have AWS API Call via CloudTrail as the value for detail-type.

Some occurrences in AWS can be reported to EventBridge both by the service itself and by CloudTrail, but in different ways. For example, an Amazon EC2 API call that launches or terminates an instance generates events available to EventBridge through CloudTrail. However, the Amazon EC2 instance state changes, from ‘running’ to ‘terminating’ for example, are EventBridge events themselves.

The following is an example of an event delivered via CloudTrail. The event was generated by an AWS API call to Amazon S3 to create a bucket.

```json
{
"version": "0",
"id": "36eb8523-97d0-4518-b33d-ee3579ff19f0",
"detail-type": "AWS API Call via CloudTrail",
"source": "aws.s3",
"account": "123456789012",
"time": "2016-02-20T01:09:13Z",
"region": "us-east-1",
"resources": [],
"detail": {
"eventVersion": "1.03",
"userIdentity": {
  "type": "Root",
  "principalId": "123456789012",
  "arn": "arn:aws:iam::123456789012:root",
  "accountId": "123456789012",
  "sessionContext": {
    "attributes": {
      "mfaAuthenticated": "false",
      "creationDate": "2016-02-20T01:05:59Z"
    }
  }
},
"eventTime": "2016-02-20T01:09:13Z",
"eventSource": "s3.amazonaws.com",
"eventName": "CreateBucket",
"awsRegion": "us-east-1",
"sourceIPAddress": "100.100.100.100",
"userAgent": "[S3Console/0.4]",
"requestParameters": {
  "bucketName": "bucket-test-iad"
},
"responseElements": null,
"requestID": "9D767BCC38E4E7487"
}
```
AWS API call events that are larger than 256 KB in size are not supported. For more information about the API calls that you can use as triggers for rules, see Services Supported by CloudTrail Event History.
Sending and Receiving Events Between AWS Accounts

You can set up your AWS account to send events to other AWS accounts, or to receive events from other accounts. This can be useful if the accounts belong to the same organization, or belong to organizations that are partners or have a similar relationship.

If you set up your account to send or receive events, you specify which individual AWS accounts can send events to or receive events from yours. If you use the AWS Organizations feature, you can specify an organization and grant access to all accounts in that organization. For more information, see What is AWS Organizations in the AWS Organizations User Guide.

The overall process is as follows:

- On the receiver account, edit the permissions on the default event bus to allow specified AWS accounts, an organization, or all AWS accounts to send events to the receiver account.
- On the sender account, set up one or more rules that have the receiver account's event bus as the target.
  
  If the sender account has permissions to send events because it is part of an AWS organization that has permissions, the sender account also must have an IAM role with policies that enable it to send events to the receiver account. If you use the AWS Management Console to create the rule that targets the receiver account, this is done automatically. If you use the AWS CLI, you must create the role manually.
- On the receiver account, set up one or more rules that match events that come from the sender account.

The AWS Region where the receiver account adds permissions to the event bus must be the same region where the sender account creates the rule to send events to the receiver account.

Events sent from one account to another are charged to the sending account as custom events. The receiving account is not charged. For more information, see Amazon EventBridge Pricing.

If a receiver account sets up a rule that sends events received from a sender account on to a third account, these events are not sent to the third account.

Enabling Your AWS Account to Receive Events from Other AWS Accounts

To receive events from other accounts or organizations, you must first edit the permissions on your account's default event bus. The default event bus accepts events from AWS services, other authorized AWS accounts, and PutEvents calls.

When you edit the permissions on your default event bus to grant permission to other AWS accounts, you can specify accounts by account ID or organization ID. Or you can choose to receive events from all AWS accounts.

Warning
If you choose to receive events from all AWS accounts, be careful to create rules that match only the events to receive from others. To create more secure rules, make sure that the event pattern...
for each rule contains an `<Account>` field with the account IDs of one or more accounts from which to receive events. Rules that have an event pattern containing an `<Account>` field do not match events sent from accounts that are not listed in the `<Account>` field. For more information, see **Events and Event Patterns in EventBridge** (p. 36).

**To enable your account to receive events from other AWS accounts using the console**

2. In the navigation pane, choose **Event Buses**, **Add Permission**.
3. Choose **AWS Account** or **Organization**.
   
   If you choose **AWS Account**, enter the 12-digit AWS account ID of the account from which to receive events. To receive events from all other AWS accounts, choose **Everybody(*)**.

   If you choose **Organization**, choose **My organization** to grant permissions to all accounts in the organization of which the current account is a member. Or choose **Another organization** and enter the organization ID of that organization. You must include the o- prefix when you type the organization ID.

4. Choose **Add**.
5. You can repeat these steps to add other accounts or organizations.

**To enable your account to receive events from other AWS accounts using the AWS CLI**

1. To enable one specific AWS account to send events, run the following command:

   ```bash
   aws events put-permission --action events:PutEvents --statement-id MySid --principal $SenderAccountID
   ```

   To enable an AWS organization to send events, run the following command:

   ```bash
   aws events put-permission --action events:PutEvents --statement-id MySid --principal * --condition '{"Type": "StringEquals", "Key": "aws:PrincipalOrgID", "Value": "$SenderOrganizationID"}'
   ```

   To enable all other AWS accounts to send events, run the following command:

   ```bash
   aws events put-permission --action events:PutEvents --statement-id MySid --principal *
   ```

   You can run `aws events put-permission` multiple times to grant permissions to both individual AWS accounts and organizations, but you cannot specify both an individual account and an organization in a single command.

2. After setting permissions for your default event bus, you can optionally use the `describe-event-bus` command to check the permissions:

   ```bash
   aws events describe-event-bus
   ```

**Sending Events to Another AWS Account**

To send events to another account, configure a EventBridge rule that has the default event bus of another AWS account as the target. The default event bus of that receiving account must also be configured to receive events from your account.
To send events from your account to another AWS account using the console

2. In the navigation pane, choose Events, Create Rule.
3. For Event Source, choose Event Pattern and select the service name and event types to send to the other account.
4. Choose Add Target.
5. For Target, choose Event bus in another AWS account. For Account ID, enter the 12-digit account ID of the AWS account to which to send events.
6. An IAM role is needed when this sender account has permissions to send events because the receiver account granted permissions to an entire organization.
   - To create an IAM role automatically, choose Create a new role for this specific resource.
   - Otherwise, choose Use existing role. Choose a role that already has sufficient permissions to invoke the build. EventBridge does not grant additional permissions to the role that you select.
7. At the bottom of the page, choose Configure Details.
8. Type a name and description for the rule, and choose Create Rule.

To send events to another AWS account using the AWS CLI

1. If the sender account has permissions to send events because it is part of an AWS organization to which the receiver account has granted permissions, the sender account also must have a role with policies that enable it to send events to the receiver account. This step explains how to create that role.

   If the sender account was given permission to send events by way of its AWS account ID, and not through an organization, this step is optional. You can skip to step 2.

   a. If the sender account was granted permissions through an organization, create the IAM role needed. First, create a file named assume-role-policy-document.json, with the following content:

   ```json
   {
   "Version": "2012-10-17",
   "Statement": [
   {
   "Effect": "Allow",
   "Principal": {
   "Service": "events.amazonaws.com"
   },
   "Action": "sts:AssumeRole"
   }
   ]
   }
   ```
   
   b. To create the role, enter the following command:

   ```bash
   $ aws iam create-role \
   --profile sender \
   --role-name event-delivery-role \
   --assume-role-policy-document file://assume-role-policy-document.json
   ```
   
   c. Create a file named permission-policy.json with the following content:

   ```json
   {
   "Version": "2012-10-17",
   "Statement": [
   ```
Writing Rules that Match Events from Another AWS Account

If your account is set up to receive events from other AWS accounts, you can write rules that match those events. Set the event pattern of the rule to match the events you are receiving from the other account.

Unless you specify account in the event pattern of a rule, any of your account’s rules, both new and existing, that match events you receive from other accounts trigger based on those events. If you are receiving events from another account, and you want a rule to trigger only on that event pattern when it is generated from your own account, you must add account and specify your own account ID to the event pattern of the rule.

If you set up your AWS account to accept events from all AWS accounts, we strongly recommend that you add account to every EventBridge rule in your account. This prevents rules in your account from triggering on events from unknown AWS accounts. When you specify the account field in the rule, you can specify the account IDs of more than one AWS account in the field.
To have a rule trigger on a matching event from any AWS account that you have granted permissions to, do not specify * in the account field of the rule. Doing so would not match any events, because * never appears in the account field of an event. Instead, just omit the account field from the rule.

**To write a rule matching events from another account using the console**

2. In the navigation pane, choose Events, Create Rule.
3. For Event Source, choose Event Pattern, and select the service name and event types that the rule should match.
5. In the edit window, add an Account line specifying which AWS accounts sending this event should be matched by the rule. For example, the edit window originally shows the following:

```json
{
    "source": [
        "aws.ec2"
    ],
    "detail-type": [
        "EBS Volume Notification"
    ]
}
```

Add the following to make the rule match EBS volume notifications that are sent by the AWS accounts 123456789012 and 111122223333:

```json
{
    "account": [
        "123456789012","111122223333"
    ],
    "source": [
        "aws.ec2"
    ],
    "detail-type": [
        "EBS Volume Notification"
    ]
}
```

6. After editing the event pattern, choose Save.
7. Finish creating the rule as usual, setting one or more targets in your account.

**To write a rule matching events from another AWS account using the AWS CLI**

- Use the put-rule command. In the Account field in the rule's event pattern, specify the other AWS accounts for the rule to match. The following example rule matches Amazon EC2 instance state changes in the AWS accounts 123456789012 and 111122223333:

```bash
aws events put-rule --name "EC2InstanceStateChanges" --event-pattern "{"account": [{"123456789012", "111122223333"}],"source": ["aws.ec2"],"detail-type": ["EC2 Instance State-change Notification"]}" --role-arn "arn:aws:iam::123456789012:role/MyRoleForThisRule"
```
Migrate a Sender-Receiver Relationship to Use AWS Organizations

If you have a sender account that had permissions granted directly to its account ID, and you now want to revoke those permissions and give the sending account access by granting permissions to an organization, you must take some additional steps. These steps ensure that the events from the sender account can still get to the receiver account. This is because accounts that are given permission to send events via an organization must also use an IAM role to do so.

To add the permissions necessary to migrate a sender-receiver relationship

1. In the sender account, create an IAM role with policies that enable it to send events to the receiver account.
   a. Create a file named `assume-role-policy-document.json`, with the following content:
      ```json
      {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "events.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
      }
      ```
   b. To create the IAM role, enter the following command:
      ```bash
      $ aws iam create-role
      --profile sender
      --role-name event-delivery-role
      --assume-role-policy-document file://assume-role-policy-document.json
      ```
   c. Create a file named `permission-policy.json` with the following content:
      ```json
      {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "events:PutEvents"
          ],
          "Resource": [
            "arn:aws:events:us-east-1:$\{receiver_account_id\}:event-bus/default"
          ]
        }
      ]
      }
      ```
   d. Enter the following command to attach this policy to the role:
      ```bash
      $ aws iam put-role-policy
      --profile sender
      --role-name event-delivery-role
      --policy-name EventBusDeliveryRolePolicy
      ```
2. Edit each existing rule in the sender account that has the receiver account default event bus as a target. Edit the rule by adding the role that you created in step 1 to the target information. Use the following command:

```
aws events put-targets --rule Rulename --targets
```

3. In the receiver account, run the following command to grant permissions for the accounts in the organization to send events to the receiver account:

```
aws events put-permission --action events:PutEvents --statement-id Sid-For-Organization --principal \* --condition '{"Type": "StringEquals", "Key": "aws:PrincipalOrgID", "Value": "SenderOrganizationID"}'
```

Optionally, you can also revoke the permissions originally granted directly to the sender account:

```
aws events remove-permission --statement-id Sid-for-SenderAccount
```
Adding Events with PutEvents

The `PutEvents` action sends multiple events to EventBridge in a single request. For more information, see `PutEvents` in the Amazon EventBridge API Reference and `put-events` in the AWS CLI Command Reference.

Each `PutEvents` request can support a limited number of entries. For more information, see Amazon EventBridge Quotas (p. 156). The `PutEvents` operation attempts to process all entries in the natural order of the request. Each event has a unique id that is assigned by EventBridge after you call `PutEvents`.

In the following example Java code sends two identical events to EventBridge.

**AWS SDK for Java Version 2.x**

```java
EventBridgeClient eventBridgeClient = EventBridgeClient.builder().build();

PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
    .resources("resource1", "resource2")
    .source("com.mycompany.myapp")
    .detailType("myDetailType")
    .detail("{ "key1": "value1", "key2": "value2" }")
    .build();

List<PutEventsRequestEntry> requestEntries = new ArrayList<PutEventsRequestEntry>();
requestEntries.add(requestEntry);

PutEventsRequest eventsRequest = PutEventsRequest.builder()
    .entries(requestEntries)
    .build();

PutEventsResponse result = eventBridgeClient.putEvents(eventsRequest);

for (PutEventsResultEntry resultEntry: result.entries()) {
    if (resultEntry.eventId() != null) {
        System.out.println("Event Id: " + resultEntry.eventId());
    } else {
        System.out.println("PutEvents failed with Error Code: " + resultEntry.errorCode());
    }
}
```

**AWS SDK for Java Version 1.0**

```java
PutEventsRequestEntry requestEntry = new PutEventsRequestEntry()
    .withTime(new Date())
    .withSource("com.mycompany.myapp")
    .withDetailType("myDetailType")
    .withResources("resource1", "resource2")
    .withDetail("{ "key1": "value1", "key2": "value2" }");

PutEventsRequest request = new PutEventsRequest()
    .withEntries(requestEntry, requestEntry);

PutEventsResult result = awsEventsClient.putEvents(request);
```
Handling Failures When Using PutEvents

By default, failure of individual entries within a request does not stop the processing of subsequent entries in the request. This means that a response Entries array includes both successfully and unsuccessfully processed entries. You must detect unsuccessfully processed entries and include them in a subsequent call.

Successful result entries include Id value, and unsuccessful result entries include ErrorCode and ErrorMessage values. The ErrorCode parameter reflects the type of error. ErrorMessage provides more detailed information about the error. The example below has three result entries for a PutEvents request. The second entry has failed and is reflected in the response.

Example: PutEvents Response Syntax

```
{
  "FailedEntryCount": 1,
  "Entries": [
    {
      "EventId": "11710aed-b79e-4468-a20b-bb3c0c3b4860"
    },
    {
      "ErrorCode": "InternalFailure",
      "ErrorMessage": "Internal Service Failure"
    },
    {
      "EventId": "d804d26a-88db-4b66-9eaf-9a11c708ae82"
    }
  ]
}
```

Entries that were unsuccessfully processed can be included in subsequent PutEvents requests. First, check the FailedRecordCount parameter in the PutEventsResult to confirm if there are failed records in the request. If so, each Entry that has an ErrorCode that is not null should be added to a subsequent request. For an example of this type of handler, refer to the following code.

Example: PutEvents failure handler

```
PutEventsRequestEntry requestEntry = new PutEventsRequestEntry()
  .withTime(new Date())
  .withSource("com.mycompany.myapp")
  .withDetailType("myDetailType")
  .withResources("resource1", "resource2")
  .withDetail("\"key1\": \"value1\", \"key2\": \"value2\" ");
List<PutEventsRequestEntry> putEventsRequestEntryList = new ArrayList<>();
for (int i = 0; i < 3; i++) {
```
putEventsRequestEntryList.add(requestEntry);
}

PutEventsRequest putEventsRequest = new PutEventsRequest();
putEventsRequest.withEntries(putEventsRequestEntryList);
PutEventsResult putEventsResult = awsEventsClient.putEvents(putEventsRequest);

while (putEventsResult.getFailedEntryCount() > 0) {
    final List<PutEventsRequestEntry> failedEntriesList = new ArrayList<>();
    final List<PutEventsResultEntry> PutEventsResultEntryList = putEventsResult.getEntries();
    for (int i = 0; i < PutEventsResultEntryList.size(); i++) {
        final PutEventsRequestEntry putEventsRequestEntry = putEventsResultEntryList.get(i);
        final PutEventsResultEntry putEventsResultEntry = PutEventsResultEntryList.get(i);
        if (putEventsResultEntry.getErrorCode() != null) {
            failedEntriesList.add(putEventsRequestEntry);
        }
    }
    putEventsRequestEntryList = failedEntriesList;
    putEventsRequest.setEntries(putEventsRequestEntryList);
    putEventsResult = awsEventsClient.putEvents(putEventsRequest);
}

### Sending Events Using the AWS CLI

You can use the AWS CLI to send custom events. The following example puts one custom event into EventBridge:

```bash
aws events put-events \\n   --entries '{"Time": "2016-01-14T01:02:03Z", "Source": "com.mycompany.myapp", "Resources": ["resource1", "resource2"], "DetailType": "myDetailType", "Detail": \\
{ \\
"key1": "value1", \\
"key2": "value2"
}'}
```

You can also create a file for example, `entries.json`, like the following:

```
[

  {
    "Time": "2016-01-14T01:02:03Z",
    "Source": "com.mycompany.myapp",
    "Resources": [
      "resource1",
      "resource2"
    ],
    "DetailType": "myDetailType",
    "Detail": \\
{ \\
"key1": "value1", \\
"key2": "value2"
}"

]
```

You can use the AWS CLI to read the entries from this file and send events. At a command prompt, type:

```bash
aws events put-events --entries file://entries.json
```

### Calculating PutEvents Event Entry Sizes

You can inject custom events into EventBridge using the `PutEvents` action. You can inject multiple events using the `PutEvents` action as long as the total entry size is less than 256KB. You can calculate
the event entry size beforehand by following the steps below. You can then batch multiple event entries into one request for efficiency.

**Note**
The size limit is imposed on the entry. Even if the entry is less than the size limit, it does not mean that the event in EventBridge is also less than this size. On the contrary, the event size is always larger than the entry size due to the necessary characters and keys of the JSON representation of the event. For more information, see *Events and Event Patterns in EventBridge* (p. 36).

The `PutEventsRequestEntry` size is calculated as follows:

- If the `Time` parameter is specified, it is measured as 14 bytes.
- The `Source` and `DetailType` parameters are measured as the number of bytes for their UTF-8 encoded forms.
- If the `Detail` parameter is specified, it is measured as the number of bytes for its UTF-8 encoded form.
- If the `Resources` parameter is specified, each entry is measured as the number of bytes for their UTF-8 encoded forms.

The following example Java code calculates the size of a given `PutEventsRequestEntry` object:

```java
int getSize(PutEventsRequestEntry entry) {
    int size = 0;
    if (entry.getTime() != null) {
        size += 14;
    }
    size += entry.getSource().getBytes(StandardCharsets.UTF_8).length;
    size += entry.getDetailType().getBytes(StandardCharsets.UTF_8).length;
    if (entry.getDetail() != null) {
        size += entry.getDetail().getBytes(StandardCharsets.UTF_8).length;
    }
    if (entry.getResources() != null) {
        for (String resource : entry.getResources()) {
            if (resource != null) {
                size += resource.getBytes(StandardCharsets.UTF_8).length;
            }
        }
    }
    return size;
}
```
Using EventBridge with Interface VPC Endpoints

If you use Amazon Virtual Private Cloud (Amazon VPC) to host your AWS resources, you can establish a private connection between your VPC and EventBridge. You can use this connection to enable EventBridge to communicate with your resources on your VPC without going through the public internet.

Amazon VPC is an AWS service that you can use to launch AWS resources in a virtual network that you define. With a VPC, you have control over your network settings, such as the IP address range, subnets, route tables, and network gateways. To connect your VPC to EventBridge, you define an **interface VPC endpoint** for EventBridge. This type of endpoint enables you to connect your VPC to AWS services. The endpoint provides reliable, scalable connectivity to EventBridge without requiring an internet gateway, network address translation (NAT) instance, or VPN connection. For more information, see [What is Amazon VPC in the Amazon VPC User Guide](https://docs.aws.amazon.com/vpc/userguide/vpc-introduction.html).

Interface VPC endpoints are powered by AWS PrivateLink, an AWS technology that enables private communication between AWS services using an elastic network interface with private IP addresses. For more information, see [New – AWS PrivateLink for AWS Services](https://aws.amazon.com/privatelink/).

The following steps are for users of Amazon VPC. For more information, see [Getting Started in the Amazon VPC User Guide](https://docs.aws.amazon.com/vpc/userguide/vpc-introduction.html).

### Availability

EventBridge currently supports VPC endpoints in the following Regions:

- US East (Ohio)
- US East (N. Virginia)
- US West (N. California)
- US West (Oregon)
- Asia Pacific (Mumbai)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- South America (São Paulo)

### Create a VPC Endpoint for EventBridge

To start using EventBridge with your VPC, create an interface VPC endpoint for EventBridge. The service name to choose is `com.amazonaws.Region.events`. For more information, see [Creating an Interface Endpoint in the Amazon VPC User Guide](https://docs.aws.amazon.com/vpc/userguide/vpc-introduction.html).
You do not need to change the settings for EventBridge. EventBridge calls other AWS services using either public endpoints or private interface VPC endpoints, whichever are in use. For example, if you create an interface VPC endpoint for EventBridge, and you already have a EventBridge rule that sends notifications to Amazon SNS when it is triggered, the notifications begin to flow through the interface VPC endpoint.
Monitoring Usage with CloudWatch Metrics

EventBridge sends metrics to Amazon CloudWatch every minute.

EventBridge Metrics

The AWS/Events namespace includes the following metrics.

All of these metrics use Count as the unit, so Sum and SampleCount are the most useful statistics.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeadLetterInvocations</td>
<td>Measures the number of times a rule’s target is not invoked in response to an event. This includes invocations that would result in triggering the same rule again, causing an infinite loop.</td>
</tr>
<tr>
<td></td>
<td>Valid Dimensions: RuleName</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td>Invocations</td>
<td>Measures the number of times a target is invoked for a rule in response to an event. This includes successful and failed invocations, but does not include throttled or retried attempts until they fail permanently. It does not include DeadLetterInvocations.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td>EventBridge only sends this metric to CloudWatch if it has a non-zero value.</td>
</tr>
<tr>
<td></td>
<td>Valid Dimensions: RuleName</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td>FailedInvocations</td>
<td>Measures the number of invocations that failed permanently. This does not include invocations that are retried, or that succeeded after a retry attempt. It also does not count failed invocations that are counted in DeadLetterInvocations.</td>
</tr>
<tr>
<td></td>
<td>Valid Dimensions: RuleName</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td>TriggeredRules</td>
<td>Measures the number of triggered rules that matched with any event.</td>
</tr>
<tr>
<td></td>
<td>Valid Dimensions: RuleName</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td>MatchedEvents</td>
<td>Measures the number of events that matched with any rule.</td>
</tr>
<tr>
<td></td>
<td>Valid Dimensions: None</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ThrottledRules</td>
<td>Measures the number of triggered rules that are being throttled.</td>
</tr>
<tr>
<td></td>
<td>Valid Dimensions: RuleName</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
</tbody>
</table>

### Dimensions for EventBridge Metrics

EventBridge metrics have one dimension, which is listed below.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RuleName</td>
<td>Filters the available metrics by rule name.</td>
</tr>
</tbody>
</table>
Amazon EventBridge Managed Rules

Other AWS services can create and manage EventBridge rules in your AWS account that are needed for certain functions in those services. These are called *managed rules*.

When a service creates a managed rule, it may also create an IAM policy that grants permissions to that service to create the rule. IAM policies created this way are scoped narrowly with resource-level permissions, to allow the creation of only the necessary rules.

You can delete managed rules by using the **Force delete** option. Do so only if you are sure that the other service no longer needs the rule. Otherwise, deleting a managed rule causes the features that rely on it to stop working.
Security in Amazon EventBridge

This section provides information about Amazon EventBridge security and authentication.

Topics

- Data Protection in Amazon EventBridge (p. 120)
- Tag-based Policies (p. 121)
- Identity and Access Management in Amazon EventBridge (p. 121)
- Logging and Monitoring in Amazon EventBridge (p. 149)
- Compliance Validation in Amazon EventBridge (p. 152)
- Resilience in Amazon EventBridge (p. 152)
- Infrastructure Security in Amazon EventBridge (p. 152)
- Configuration and Vulnerability Analysis in Amazon EventBridge (p. 153)

Amazon EventBridge uses IAM to control access to other AWS services and resources. For an overview of how IAM works, see Overview of Access Management in the IAM User Guide. For an overview of security credentials, see AWS Security Credentials in the Amazon Web Services General Reference.

Data Protection in Amazon EventBridge

Amazon EventBridge conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over the data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that 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.

- 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 personal data that is stored in Amazon S3.

We strongly recommend that you never put sensitive identifying information, such as your customers’ account numbers, into free-form fields such as a Name field. This includes when you work with EventBridge or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into EventBridge or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the AWS Shared Responsibility Model and GDPR blog post on the AWS Security Blog.
Encryption at Rest

The payload of events is stored internally in EventBridge. This internal data is not encrypted.

Encryption in Transit

All data that passes between EventBridge and other services is encrypted using Transport layer Security (TLS).

Tag-based Policies

Amazon EventBridge supports policies based on tags. For example, you could restrict access to resources that include a tag with the key `environment` and the value `production`.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "events:PutRule",
                "events:DescribeRule",
                "events:DeleteRule",
                "events:CreateEventBus",
                "events:DescribeEventBus",
                "events:DeleteEventBus"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/environment": "production"}
            }
        }
    ]
}
```

This policy will deny the ability to create, delete, or modify tags, rules, or event buses for resources that have been tagged `environment/production`.

For more information about tagging, see the following.

- Tagging Your Amazon EventBridge Resources (p. 154)
- Controlling Access Using IAM Tags

Identity and Access Management in Amazon EventBridge

Access to Amazon EventBridge requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to access AWS resources, such as retrieving event data from other AWS resources. The following sections provide details on how you can use AWS Identity and Access Management (IAM) and EventBridge to help secure your resources by controlling who can access them:

- Authentication (p. 122)
- Access Control (p. 123)
Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user** – When you sign up for AWS, you provide an email address and password that is associated with your account. These are your root credentials, and they provide complete access to all of your AWS resources.

  **Important**
  For security reasons, we recommend that you use the root credentials only to create an administrator, which is an IAM user with full permissions to your account. Then you can use this administrator to create other IAM users and roles with limited permissions. For more information, see IAM Best Practices and Creating an Admin User and Group in the IAM User Guide.

- **IAM user** – An IAM user is an identity within your account that has specific custom permissions (for example, permissions to send event data to a target in EventBridge). You can use an IAM user name and password to sign in to secure AWS webpages such as the AWS Management Console, AWS Discussion Forums, or the AWS Support Center.

  In addition to a user name and password, you can also generate access keys for each user. You can use these keys when you access AWS services programmatically, either through one of the several SDKs or by using the AWS Command Line Interface (AWS CLI). The SDK and AWS CLI tools use the access keys to cryptographically sign your request. If you don't use the AWS tools, you must sign the request yourself. EventBridge supports Signature Version 4, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 Signing Process in the Amazon Web Services General Reference.

- **IAM role** – An IAM role is another IAM identity that you can create in your account that has specific permissions. It's similar to an IAM user, but it isn't associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:

  - **Federated user access** – Instead of creating an IAM user, you can use preexisting identities from AWS Directory Service, your enterprise user directory, or a web identity provider (IdP). These are known as federated users. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated Users and Roles in the IAM User Guide.

  - **Cross-account access** – You can use an IAM role in your account to grant another account permissions to access your account's resources. For an example, see Tutorial: Delegate Access Across AWS Accounts Using IAM Roles in the IAM User Guide.

  - **AWS service access** – You can use an IAM role in your account to grant to an AWS service the permissions needed to access your account's resources. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data stored in the bucket into an Amazon Redshift cluster. For more information, see Creating a Role to Delegate Permissions to an AWS Service in the IAM User Guide.

  - **Applications running on Amazon EC2** – Instead of storing access keys in the EC2 instance for use by applications running on the instance and making AWS API requests, you can use an IAM role to
manage temporary credentials for these applications. To assign an AWS role to an EC2 instance and make it available to all of its applications, you can create an instance profile that is attached to the instance. An instance profile contains the role and enables programs running on the EC2 instance to get temporary credentials. For more information, see Using Roles for Applications on Amazon EC2 in the IAM User Guide.

Access Control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access EventBridge resources. For example, you must have permissions to invoke AWS Lambda, Amazon Simple Notification Service (Amazon SNS), and Amazon Simple Queue Service (Amazon SQS) targets associated with your EventBridge rules.

The following sections describe how to manage permissions for EventBridge. We recommend that you read the overview first.

- Overview of Managing Access Permissions to Your EventBridge Resources (p. 123)
- Using Identity-Based Policies (IAM Policies) for EventBridge (p. 126)
- Using Resource-Based Policies for EventBridge (p. 134)
- EventBridge Permissions Reference (p. 138)

Overview of Managing Access Permissions to Your EventBridge Resources

Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles), and some services (such as AWS Lambda) also support attaching permissions policies to resources.

Note
An account administrator (or administrator IAM user) is a user with administrator privileges. For more information, see IAM Best Practices in the IAM User Guide.

When granting permissions, you decide who is getting the permissions, the resources they get permissions for, and the specific actions that you want to allow on those resources.

Topics
- EventBridge Resources and Operations (p. 123)
- Understanding Resource Ownership (p. 124)
- Managing Access to Resources (p. 125)
- Specifying Policy Elements: Actions, Effects, and Principals (p. 126)
- Specifying Conditions in a Policy (p. 126)

EventBridge Resources and Operations

In EventBridge, the primary resource is a rule. EventBridge supports other resources that can be used with the primary resource, such as events. These are referred to as subresources. These resources and subresources have unique Amazon Resource Names (ARNs) associated with them. For more information about ARNs, see Amazon Resource Names (ARN) and AWS Service Namespaces in the Amazon Web Services General Reference.
### Resource Type | ARN Format
--- | ---
Rule | arn:aws:events:region:account:rule/[event-bus-name]/rule-name
Event bus | arn:aws:events:region:account:event-bus/event-bus-name
All EventBridge resources | arn:aws:events:*
All EventBridge resources owned by the specified account in the specified Region | arn:aws:events:region:account:*

**Note**
Most services in AWS treat a colon (:) or a forward slash (/) as the same character in ARNs. However, EventBridge uses an exact match in event patterns and rules. Be sure to use the correct ARN characters when creating event patterns so that they match the ARN syntax in the event that you want to match.

For example, you can indicate a specific rule (myRule) in your statement using its ARN as follows.

```json
"Resource": "arn:aws:events:us-east-1:123456789012:rule/myRule"
```

You can also specify all rules that belong to a specific account by using the asterisk (*) wildcard as follows.

```json
"Resource": "arn:aws:events:us-east-1:123456789012:rule/*"
```

To specify all resources or if a specific API action doesn't support ARNs, use the asterisk (*) wildcard in the `Resource` element as follows.

```json
"Resource": "*
```

Some EventBridge API actions accept multiple resources (that is, `PutTargets`). To specify multiple resources in a single statement, separate their ARNs with commas, as follows.

```json
"Resource": ["arn1", "arn2"]
```

EventBridge provides a set of operations to work with the EventBridge resources. For a list of available operations, see [EventBridge Permissions Reference (p. 138)](#).

### Understanding Resource Ownership

The account owns the resources that are created in the account, regardless of who created the resources. Specifically, the resource owner is the account of the principal entity (that is, the account root user, an IAM user, or an IAM role) that authenticates the resource creation request. The following examples illustrate how this works:

- If you use the root user credentials of your account to create a rule, your account is the owner of the EventBridge resource.
- If you create an IAM user in your account and grant permissions to create EventBridge resources to that user, the user can create EventBridge resources. However, your account, which the user belongs to, owns the EventBridge resources.
• If you create an IAM role in your account with permissions to create EventBridge resources, anyone who can assume the role can create EventBridge resources. Your account, which the role belongs to, owns the EventBridge resources.

Managing Access to Resources

A permissions policy describes who has access to what. The following section explains the available options for creating permissions policies.

Note
This section discusses using IAM in the context of EventBridge. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see What Is IAM? in the IAM User Guide. For information about IAM policy syntax and descriptions, see IAM Policy Reference in the IAM User Guide.

Policies attached to an IAM identity are referred to as identity-based policies (IAM policies) and policies attached to a resource are referred to as resource-based policies. EventBridge supports both identity-based (IAM policies) and resource-based policies.

Topics
• Identity-Based Policies (IAM Policies) (p. 125)
• Resource-Based Policies (IAM Policies) (p. 125)

Identity-Based Policies (IAM Policies)

You can attach policies to IAM identities. For example, you can do the following:

• Attach a permissions policy to a user or a group in your account – To grant a user permissions to view rules in the Amazon CloudWatch console, you can attach a permissions policy to a user or group that the user belongs to.

• Attach a permissions policy to a role (grant cross-account permissions) – You can attach an identity-based permissions policy to an IAM role to grant cross-account permissions. For example, the administrator in account A can create a role to grant cross-account permissions to another account (for example, account B) or an AWS service as follows:
  1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in account A.
  2. Account A administrator attaches a trust policy to the role identifying account B as the principal who can assume the role.
  3. Account B administrator can then delegate permissions to assume the role to any users in account B. Doing this allows users in account B to create or access resources in account A. The principal in the trust policy can also be an AWS service principal to grant to an AWS service the permissions needed to assume the role.

For more information about using IAM to delegate permissions, see Access Management in the IAM User Guide.

You can create specific IAM policies to restrict the calls and resources that users in your account have access to, and then attach those policies to IAM users. For more information about how to create IAM roles and to explore example IAM policy statements for EventBridge, see Overview of Managing Access Permissions to Your EventBridge Resources (p. 123).

Resource-Based Policies (IAM Policies)

When a rule is triggered in EventBridge, all the targets associated with the rule are invoked. Invocation means invoking the AWS Lambda functions, publishing to the Amazon SNS topics, and relaying the
event to the Amazon Kinesis streams. To be able to make API calls against the resources that you own,
EventBridge needs the appropriate permissions. For Lambda, Amazon SNS, and Amazon SQS resources,
EventBridge relies on resource-based policies. For Kinesis streams, EventBridge relies on IAM roles.

For more information about how to create IAM roles and to explore example resource-based policy
statements for EventBridge, see Using Resource-Based Policies for EventBridge (p. 134).

Specifying Policy Elements: Actions, Effects, and Principals

For each EventBridge resource, the service defines a set of API operations. To grant permissions for
these API operations, EventBridge defines a set of actions that you can specify in a policy. Some API
operations can require permissions for more than one action to perform the API operation. For more
information about resources and API operations, see EventBridge Resources and Operations (p. 123)
and EventBridge Permissions Reference (p. 138).

The following are the basic policy elements:

- **Resource** – You use an Amazon Resource Name (ARN) to identify the resource that the policy applies
to. For more information, see EventBridge Resources and Operations (p. 123).
- **Action** – You use action keywords to identify resource operations that you want to allow or deny. For
example, the `events:Describe` permission allows the user permissions to perform the `Describe`
operation.
- **Effect** – You specify the effect, either allow or deny, when the user requests the specific action. If you
don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly
deny access to a resource, which you might do to make sure that a user can't access it, even if a
different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the
implicit principal. For resource-based policies, you specify the user, account, service, or other entity
that you want to receive permissions (applies to resource-based policies only).

To learn more about IAM policy syntax and descriptions, see IAM JSON Policy Reference in the IAM User
Guide.

For a table showing all of the EventBridge API actions and the resources that they apply to, see
EventBridge Permissions Reference (p. 138).

Specifying Conditions in a Policy

When you grant permissions, you can use the access policy language to specify the conditions when a
policy should take effect. For example, you might want a policy to be applied only after a specific date.
For more information about specifying conditions in a policy language, see Condition in the IAM User
Guide.

To express conditions, you use predefined condition keys. There are AWS-wide condition keys and
EventBridge–specific keys that you can use as appropriate. For a complete list of AWS-wide keys, see
Available Keys for Conditions in the IAM User Guide. For a complete list of EventBridge–specific keys, see

Using Identity-Based Policies (IAM Policies) for
EventBridge

This topic provides examples of identity-based policies in which an account administrator can attach
permissions policies to IAM identities (that is, users, groups, and roles).
The following shows an example of a permissions policy that allows a user to put event data into Kinesis.

```json

{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "CloudWatchEventsInvocationAccess",  
      "Effect": "Allow",  
      "Action": [  
        "kinesis:PutRecord"  
      ],  
      "Resource": "*"  
    }  
  ]
}
```

The sections in this topic cover the following:

**Topics**

- Permissions Required to Use the CloudWatch Console (p. 127)
- AWS Managed (Predefined) Policies for EventBridge (p. 128)
- Permissions Required for EventBridge to Access Certain Targets (p. 129)
- Customer Managed Policy Examples (p. 131)

**Permissions Required to Use the CloudWatch Console**

For a user to work with EventBridge in the CloudWatch console, that user must have a minimum set of permissions that allow the user to describe other AWS resources for their account. To use EventBridge in the CloudWatch console, you must have permissions from the following services:

- Automation
- Amazon EC2 Auto Scaling
- AWS CloudTrail
- CloudWatch
- EventBridge
- IAM
- Kinesis
- Lambda
- Amazon SNS
- Amazon SWF

If you create an IAM policy that is more restrictive than the minimum required permissions, the console won’t function as intended for users with that IAM policy. To ensure that those users can still use the CloudWatch console, also attach the `CloudWatchEventsReadOnlyAccess` managed policy to the user, as described in AWS Managed (Predefined) Policies for EventBridge (p. 128).

You don’t need to allow minimum console permissions for users that are making calls only to the AWS CLI or the CloudWatch API.

The full set of permissions required to work with the CloudWatch console is the following:

- `automation:CreateAction`
• automation:DescribeAction
• automation:UpdateAction
• autoscaling:DescribeAutoScalingGroups
• cloudtrail:DescribeTrails
• ec2:DescribeInstances
• ec2:DescribeVolumes
• events:DeleteRule
• events:DescribeRule
• events:DisableRule
• events:EnableRule
• events:ListRuleNamesByTarget
• events:ListRules
• events:ListTargetsByRule
• events:PutEvents
• events:PutRule
• events:PutTargets
• events:RemoveTargets
• events:TestEventPattern
• iam:ListRoles
• kinesis:ListStreams
• lambda:AddPermission
• lambda:ListFunctions
• lambda:RemovePermission
• sns:GetTopicAttributes
• sns:ListTopics
• sns:SetTopicAttributes
• swf:DescribeAction
• swf:ReferenceAction
• swf:RegisterAction
• swf:RegisterDomain
• swf:UpdateAction

AWS Managed (Predefined) Policies for EventBridge

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. Managed policies grant necessary permissions for common use cases so you can avoid having to investigate what permissions are needed. For more information, see AWS Managed Policies in the IAM User Guide.

The following AWS managed policies, which you can attach to users in your account, are specific to EventBridge:

• CloudWatchEventsFullAccess – Grants full access to EventBridge
• CloudWatchEventsInvocationAccess – Allows EventBridge to relay events to the streams in Amazon Kinesis Data Streams in your account
Using Identity-Based Policies (IAM Policies)

- **CloudWatchEventsReadOnlyAccess** – Grants read-only access to EventBridge
- **CloudWatchEventsBuiltInTargetExecutionAccess** – Allows built-in targets in EventBridge to perform Amazon EC2 actions on your behalf

### IAM Roles for Sending Events

In order for EventBridge to relay events to your Kinesis stream targets, you must create an IAM role.

**To create an IAM role for sending EventBridge**

2. Follow the steps in Creating a Role to Delegate Permissions to an AWS Service in the IAM User Guide to create an IAM role. As you follow the steps to create a role, do the following:
   - In **Role Name**, use a name that is unique within your account (for example, CloudWatchEventsSending).
   - In **Select Role Type**, choose AWS Service Roles, and then choose Amazon EventBridge. This grants EventBridge permissions to assume the role.
   - In **Attach Policy**, choose CloudWatchEventsInvocationAccess.

You can also create your own custom IAM policies to allow permissions for EventBridge actions and resources. You can attach these custom policies to the IAM users or groups that require those permissions. For more information about IAM policies, see Overview of IAM Policies in the IAM User Guide. For more information about managing and creating custom IAM policies, see Managing IAM Policies in the IAM User Guide.

### Permissions Required for EventBridge to Access Certain Targets

For EventBridge to access certain targets, you must specify an IAM role for accessing that target, and that role must have a certain policy attached.

If the target is a Kinesis stream, the role used to send event data to that target must include the following policy.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "kinesis:PutRecord"
         ],
         "Resource": "*"
      }
   ]
}
```

If the target is Systems Manager Run Command and you're specifying one or more InstanceIds values for the command, the role that you specify must include the following policy.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Action": "ssm:SendCommand",
         "Effect": "Allow",
         "Resource": "*"
      }
   ]
}
```
If the target is Systems Manager Run Command and you’re specifying one or more tags for the command, the role that you specify must include the following policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "ssm:SendCommand",
            "Effect": "Allow",
            "Resource": [
                "arn:aws:ec2:{region}:{accountId}:instance/**
            ],
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/*": ["{tagValues}"
                }
            }
        },
        {
            "Action": "ssm:SendCommand",
            "Effect": "Allow",
            "Resource": [
                "arn:aws:ssm:{region}::*:document/{documentName}"
            ]
        }
    ]
}
```

If the target is an AWS Step Functions state machine, the role that you specify must include the following policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["states:StartExecution"],
            "Resource": ["arn:aws:states::*:stateMachine:*"]
        }
    ]
}
```

If the target is an ECS task, the role that you specify must include the following policy.

```
{
    "Version": "2012-10-17",
    "Statement": [ {
        "Effect": "Allow",
        "Action": ["ecs:RunTask"
```
Customer Managed Policy Examples

In this section, you can find example user policies that grant permissions for various EventBridge actions. These policies work when you are using the EventBridge API, AWS SDKs, or the AWS CLI.

**Note**
All examples use the US West (Oregon) Region (us-west-2) and contain fictitious account IDs.

You can use the following sample IAM policies listed to limit the EventBridge access for your IAM users and roles.

**Examples**
- Example 1: CloudWatchEventsBuiltInTargetExecutionAccess (p. 131)
- Example 2: CloudWatchEventsInvocationAccess (p. 132)
- Example 3: CloudWatchEventsConsoleAccess (p. 132)
- Example 4: CloudWatchEventsFullAccess (p. 133)
- Example 5: CloudWatchEventsReadOnlyAccess (p. 133)
- Example 6: Use Tagging to Control Access to Specific Rules (p. 133)

**Example 1: CloudWatchEventsBuiltInTargetExecutionAccess**

The following policy allows built-in targets in EventBridge to perform Amazon EC2 actions on your behalf.

**Important**
Creating rules with built-in targets is supported only in the AWS Management Console.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchEventsBuiltInTargetExecutionAccess",
            "Effect": "Allow",
            "Action": [
                "ec2:Describe*",
                "ec2:RebootInstances",
                "ec2:StopInstances",
                "ec2:TerminateInstances",
                "ec2:CreateSnapshot"
            ],
            "Resource": "*"
        }
    ]
}
```
Example 2: CloudWatchEventsInvocationAccess

The following policy allows EventBridge to relay events to the streams in Kinesis streams in your account.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudWatchEventsInvocationAccess",
      "Effect": "Allow",
      "Action": [
        "kinesis:PutRecord"
      ],
      "Resource": "*
    }
  ]
}
```

Example 3: CloudWatchEventsConsoleAccess

The following policy ensures that IAM users can use the EventBridge console.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudWatchEventsConsoleAccess",
      "Effect": "Allow",
      "Action": [
        "automation:CreateAction",
        "automation:DescribeAction",
        "automation:UpdateAction",
        "autoscaling:DescribeAutoScalingGroups",
        "cloudtrail:DescribeTrails",
        "ec2:DescribeInstances",
        "ec2:DescribeVolume",
        "events:*",
        "iam:ListRoles",
        "kinesis:ListStreams",
        "lambda:ListFunctions",
        "lambda:RemovePermission",
        "sns:GetTopicAttributes",
        "sns:ListTopics",
        "sns:ListTopicAttributes",
        "swf:DescribeAction",
        "swf:ReferenceAction",
        "swf:RegisterAction",
        "swf:RegisterDomain",
        "swf:UpdateAction"
      ],
      "Resource": "*
    },
    {
      "Sid": "IAMPassRoleForCloudWatchEvents",
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": [
        "arn:aws:iam::*:role/AWS_Events_Invoke_Targets",
        "arn:aws:iam::*:role/AWS_Events_Actions_Execution"
      ]
    }
  ]
}
```
Example 4: CloudWatchEventsFullAccess

The following policy allows performing actions against EventBridge through the AWS CLI and SDK.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "CloudWatchEventsFullAccess",
         "Effect": "Allow",
         "Action": "events:*",
         "Resource": "*"
      },
      {
         "Sid": "IAMPassRoleForCloudWatchEvents",
         "Effect": "Allow",
         "Action": "iam:PassRole",
         "Resource": "arn:aws:iam::*:role/AWS_Events_Invoke_Targets"
      }
   ]
}
```

Example 5: CloudWatchEventsReadOnlyAccess

The following policy provides read-only access to EventBridge.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "CloudWatchEventsReadOnlyAccess",
         "Effect": "Allow",
         "Action": [
            "events:Describe*",
            "events:List*",
            "events:TestEventPattern"
         ],
         "Resource": "*"
      }
   ]
}
```

Example 6: Use Tagging to Control Access to Specific Rules

You can grant users access to specified EventBridge rules while preventing them from accessing other rules. To do so, tag these rules and use IAM policies that refer to those tags.

For more information about tagging EventBridge resources, see Tagging Your Amazon EventBridge Resources (p. 154).

When you tag EventBridge rules, you can grant an IAM policy to a user to allow access to only the rules with a particular tag. For example, the following policy statement grants access to only rules with the value of Prod for the tag key Stack.

```json
{
   "Statement": [
      {
         "Effect": "Allow",
```
Using Resource-Based Policies

When a rule is triggered in EventBridge, all the targets associated with the rule are invoked. Invocation means invoking the AWS Lambda functions, publishing to the Amazon SNS topics, and relaying the event to the Kinesis streams. In order to be able to make API calls against the resources you own, EventBridge needs the appropriate permissions. For Lambda, Amazon SNS, Amazon SQS, and Amazon CloudWatch Logs resources, EventBridge relies on resource-based policies. For Kinesis streams, EventBridge relies on IAM roles.

You can use the following permissions to invoke the targets associated with your EventBridge rules. The following procedures use the AWS CLI to add permissions to your targets. For information about how to install and configure the AWS CLI, see Getting Set Up with the AWS Command Line Interface in the AWS Command Line Interface User Guide.

Topics
- AWS Lambda Permissions (p. 134)
- Amazon SNS Permissions (p. 135)
- Amazon SQS Permissions (p. 136)
- CloudWatch Logs Permissions (p. 137)

AWS Lambda Permissions

To invoke your AWS Lambda function using a EventBridge rule, add the following permission to the policy of your Lambda function.

```
{
  "Effect": "Allow",
  "Action": "lambda:InvokeFunction",
  "Principal": {
    "Service": "events.amazonaws.com"
  },
  "Condition": {
    "ArnLike": {
      "AWS:SourceArn": "arn:aws:events:region:account-id:rule/rule-name"
    },
    "Sid": "TrustCWEToInvokeMyLambdaFunction"
  }
}
```

To add permissions that enable EventBridge to invoke Lambda functions

- At a command prompt, enter the following command.
aws lambda add-permission --statement-id "TrustCWEToInvokeMyLambdaFunction" \
--action "lambda:InvokeFunction" \
--principal "events.amazonaws.com" \
--function-name "arn:aws:lambda:region:account-id:function:function-name" \
--source-arn "arn:aws:events:region:account-id:rule:rule-name"

For more information about setting permissions that enable EventBridge to invoke Lambda functions, 
see AddPermission and Using Lambda with Scheduled Events in the AWS Lambda Developer Guide.

Amazon SNS Permissions

To allow EventBridge to publish an Amazon SNS topic, use the aws sns get-topic-attributes and 
the aws sns set-topic-attributes commands.

Note
EventBridge does not support the use of Condition blocks in Amazon SNS topic policies.

To add permissions that enable EventBridge to publish SNS topics

1. First, list SNS topic attributes. At a command prompt, enter the following.


The command returns all attributes of the SNS topic. The following example shows the result of a 
newly created SNS topic.

```
{
    "Attributes": {
        "SubscriptionsConfirmed": "0",
        "DisplayName": "",
        "SubscriptionsDeleted": "0",
        "EffectiveDeliveryPolicy": "{"http":{"defaultHealthyRetryPolicy":{
            "minDelayTarget":20,"maxDelayTarget":20,"numRetries":3,"numMaxDelayRetries
            ":0,"numNoDelayRetries":0,"numMinDelayRetries":0,"backoffFunction":"linear"},
            "disableSubscriptionOverrides":false}}",
        "Owner": "account-id",
    }
}
```

2. Next, convert the following statement to a string and add it to the "Statement" collection inside 
the "Policy" attribute.

```
{
    "Sid": "TrustCWEToPublishEventsToMyTopic",
    "Effect": "Allow",
    "Principal": {
        "Service": "events.amazonaws.com"
    },
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:region:account-id:topic-name"
}
```
After you convert the statement to a string, it should look like the following.

```json
{"Sid":"TrustCWEToPublishEventsToMyTopic","Effect":"Allow","Principal":
{"Service":"events.amazonaws.com"},"Action":"sns:Publish","Resource":
"arn:aws:sns:region:account-id:topic-name"}
```

3. After you've added the statement string to the statement collection, use the `aws sns set-topic-attributes` command to set the new policy.

```bash
aws sns set-topic-attributes --topic-arn "arn:aws:sns:region:account-id:topic-name" \
--attribute-name Policy \n--attribute-value 
{"Version":"2012-10-17","Id":"__default_policy_ID","Statement":[{"Sid":"__default_statement_ID","Effect":"Allow","Principal":
{"AWS":"*"},"Action":["SNS:GetTopicAttributes","SNS:SetTopicAttributes"],"SNS:AddPermission","SNS:RemovePermission","SNS:DeleteTopic","SNS:Subscribe","SNS:ListSubscriptionsByTopic","SNS:Publish","SNS:Receive"},
{"Sid":"TrustCWEToPublishEventsToMyTopic","Effect":"Allow","Principal":
{"Service":"events.amazonaws.com"},"Action":
"sns:Publish","Resource":
"arn:aws:sns:region:account-id:topic-name"},{"Sid":"TrustCWEToPublishEventsToMyTopic","Effect":"Allow","Principal":
{"Service":"events.amazonaws.com"},"Action":
"sns:Publish","Resource":
"arn:aws:sns:region:account-id:topic-name"}]
```

For more information, see the `SetTopicAttributes` action in the [Amazon Simple Notification Service API Reference](https://docs.aws.amazon.com/sns/latest/api/)

## Amazon SQS Permissions

To allow an EventBridge rule to invoke an Amazon SQS queue, use the `aws sqs get-queue-attributes` and the `aws sqs set-queue-attributes` commands.

### To add permissions that enable EventBridge rules to invoke an SQS queue

1. First, list SQS queue attributes. At a command prompt, enter the following.

```bash
aws sqs get-queue-attributes \
--queue-url https://sqs.region.amazonaws.com/account-id/queue-name \
--attribute-names Policy
```

For a new SQS queue, its policy is empty by default. In addition to adding a statement, you also need to create a policy that contains this statement.

2. The following statement enables EventBridge to send messages to an SQS queue.

```json
{
"Sid": "TrustCWEToSendEventsToMyQueue",
"Effect": "Allow",
"Principal": {
"Service": "events.amazonaws.com"
},
"Action": "sqs:SendMessage",
"Condition": {
 "ArnEquals": {
 "aws:SourceArn": "arn:aws:events:region:account-id:rule/rule-name"
 } 

```
3. Next, convert the preceding statement into a string. After you convert the policy to a string, it should look like the following.

```json
{"Sid": "TrustCWEToSendsEventsToMyQueue", "Effect": "Allow", "Principal": {
"aws:SourceArn": "arn:aws:events:region:account-id:rule/rule-name"}}}
```

4. Create a file called `set-queue-attributes.json` with the following content.

```json
{
  "Policy": {
    "Version": "2012-10-17",
    "Id": "arn:aws:sqs:region:account-id:SQSDefaultPolicy",
    "Statement": [
      {
        "Sid": "TrustCWEToSendsEventsToMyQueue",
        "Effect": "Allow",
        "Principal": {
          "Service": "events.amazonaws.com"
        },
        "Action": "sqs:SendMessage",
        "Condition": {"ArnEquals": {
          "aws:SourceArn": "arn:aws:events:region:account-id:rule/rule-name"
        }}
      }
    ]
  }
}
```

5. Set the policy attribute using the `set-queue-attributes.json` file as the input. At a command prompt, enter the following.

```bash
aws sqs set-queue-attributes \
--queue-url https://sqs.region.amazonaws.com/account-id/queue-name \
--attributes file://set-queue-attributes.json
```

If the SQS queue already has a policy, you need to copy the original policy and combine it with a new statement in the `set-queue-attributes.json` file and run the preceding command to update the policy.

For more information, see Amazon SQS Policy Examples in the Amazon Simple Queue Service Developer Guide.

CloudWatch Logs Permissions

When CloudWatch Logs is the target of a rule, EventBridge creates log streams, and CloudWatch Logs stores the text from the triggering events as log entries. To allow EventBridge to create the log stream and log the events, CloudWatch Logs must include a resource-based policy that enables EventBridge to write to CloudWatch Logs. If you use the AWS Management Console to add CloudWatch Logs as the target of a rule, this policy is created automatically. If you use the AWS CLI to add the target, you must create this policy if it doesn’t exist. The following example shows the necessary policy. This example allows EventBridge to write to all log groups that have names that start with `/aws/events/`. If you use a different log group naming policy for these types of logs, adjust the policy accordingly.

```json
{
  "Statement": [
    {
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Resource": "arn:aws:logs:{{region}}:{{account}}:log-group:/aws/events/*:*",
      "Sid": "TrustEventsToStoreLogEvent"
    }
  ]
}
```
For more information, see Controlling Access to Resources in the IAM User Guide.

EventBridge Permissions Reference

When you are setting up Access Control (p. 123) and writing permissions policies that you can attach to an IAM identity (identity-based policies), you can use the following table as a reference. The table lists each EventBridge API operation and the corresponding actions for which you can grant permissions to perform the action. You specify the actions in the policy's Action field, and you specify a wildcard character (*) as the resource value in the policy's Resource field.

You can use AWS-wide condition keys in your EventBridge policies to express conditions. For a complete list of AWS-wide keys, see Available Keys in the IAM User Guide.

**Note**
To specify an action, use the events: prefix followed by the API operation name. For example: events:PutRule, events:EnableRule, or events:* (for all EventBridge actions).

To specify multiple actions in a single statement, separate them with commas as follows.

```
"Action": ["events:action1", "events:action2"]
```

You can also specify multiple actions using wildcards. For example, you can specify all actions whose name begins with the word "Put" as follows.

```
"Action": "events:Put*"
```

To specify all EventBridge API actions, use the * wildcard as follows.

```
"Action": "events:*"
```

The following table lists the actions that you can specify in an IAM policy for use with EventBridge.

### EventBridge API Operations and Required Permissions for Actions

<table>
<thead>
<tr>
<th>EventBridge API Operations</th>
<th>Required Permissions (API Actions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeleteRule</td>
<td>events:DeleteRule</td>
</tr>
<tr>
<td></td>
<td>Required to delete a rule.</td>
</tr>
<tr>
<td>DescribeEventBus</td>
<td>events:DescribeEventBus</td>
</tr>
<tr>
<td></td>
<td>Required to list accounts that are allowed to write events to the current account's event bus.</td>
</tr>
<tr>
<td>DescribeRule</td>
<td>events:DescribeRule</td>
</tr>
<tr>
<td></td>
<td>Required to list the details about a rule.</td>
</tr>
<tr>
<td>DisableRule</td>
<td>events:DisableRule</td>
</tr>
<tr>
<td></td>
<td>Required to disable a rule.</td>
</tr>
<tr>
<td>EnableRule</td>
<td>events:EnableRule</td>
</tr>
<tr>
<td></td>
<td>Required to enable a rule.</td>
</tr>
</tbody>
</table>
### EventBridge API Operations

<table>
<thead>
<tr>
<th>EventBridge API Operation</th>
<th>Required Permissions (API Actions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListRuleNamesByTarget</td>
<td>events:ListRuleNamesByTarget Required to list rules associated with a target.</td>
</tr>
<tr>
<td>ListRules</td>
<td>events:ListRules Required to list all rules in your account.</td>
</tr>
<tr>
<td>ListTagsForResource</td>
<td>events:ListTagsForResource Required to list all tags associated with an EventBridge resource. Currently, only rules can be tagged.</td>
</tr>
<tr>
<td>ListTargetsByRule</td>
<td>events:ListTargetsByRule Required to list all targets associated with a rule.</td>
</tr>
<tr>
<td>PutEvents</td>
<td>events:PutEvents Required to add custom events that can be matched to rules.</td>
</tr>
<tr>
<td>PutPermission</td>
<td>events:PutPermission Required to give another account permission to write events to this account’s default event bus.</td>
</tr>
<tr>
<td>PutRule</td>
<td>events:PutRule Required to create or update a rule.</td>
</tr>
<tr>
<td>PutTargets</td>
<td>events:PutTargets Required to add targets to a rule.</td>
</tr>
<tr>
<td>RemovePermission</td>
<td>events:RemovePermission Required to revoke another account’s permissions for writing events to this account’s default event bus.</td>
</tr>
<tr>
<td>RemoveTargets</td>
<td>events:RemoveTargets Required to remove a target from a rule.</td>
</tr>
<tr>
<td>TestEventPattern</td>
<td>events:TestEventPattern Required to test an event pattern against a given event.</td>
</tr>
</tbody>
</table>

### Using IAM Policy Conditions for Fine-Grained Access Control

When you grant permissions, you can use the IAM policy language to specify the conditions when a policy should take effect. In a policy statement, you can optionally specify conditions that control when
it is in effect. Each condition contains one or more key-value pairs. Condition keys aren't case sensitive. For example, you might want a policy to be applied only after a specific date.

If you specify multiple conditions or multiple keys in a single condition, they're evaluated using a logical AND operation. If you specify a single condition with multiple values for one key, they're evaluated using a logical OR operation. For permission to be granted, all conditions must be met.

You can also use placeholders when you specify conditions. For more information, see Policy Variables in the IAM User Guide. For more information about specifying conditions in an IAM policy language, see Condition in the IAM User Guide.

By default, IAM users and roles can't access the events in your account. To consume events, a user must be authorized for the \texttt{PutRule} API action. If you allow an IAM user or role for the \texttt{events:PutRule} action in their policy, they can create a rule that matches certain events. You must add a target to a rule; otherwise, a rule without a target does nothing except publish a CloudWatch metric when it matches an incoming event. Your IAM user or role must have permissions for the \texttt{events:PutTargets} action.

It's possible to limit access to the events by scoping the authorization to specific sources and types of events (using the \texttt{events:source} and \texttt{events:detail-type} condition keys). You can provide a condition in the policy statement of the IAM user or role that allows them to create a rule that only matches a specific set of sources and detail types.

Similarly, through setting conditions in your policy statements, you can decide which specific resources in your accounts can be added to a rule by an IAM user or role (using the \texttt{events:TargetArn} condition key). For example, if you turn on CloudTrail in your account and you have a CloudTrail stream, CloudTrail events are also available to the users in your account through EventBridge. If you want your users to use EventBridge and access all the events but the CloudTrail events, you can add a deny statement on the \texttt{PutRule} API action with a condition that any rule created by that user or role can't match the CloudTrail event type.

For CloudTrail events, you can limit the access to a specific principal that the original API call was originated from (using the \texttt{events:detail.userIdentity.principalId} condition key). For example, you can allow a user to see all the CloudTrail events, except the ones that are made by a specific IAM role in your account that you use for auditing or forensics.

<table>
<thead>
<tr>
<th>Condition Key</th>
<th>Key/Value Pair</th>
<th>Evaluation Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{events:source}</td>
<td>\texttt{source}</td>
<td>Source, Null</td>
</tr>
<tr>
<td></td>
<td>Where \texttt{source} is the literal string for</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the source field of the event such as \texttt{aws.ec2} and \texttt{aws.s3}. To see more possible values for \texttt{source}, see the example events in EventBridge Event Examples from Supported AWS Services (p. 57).</td>
<td></td>
</tr>
<tr>
<td>\texttt{events:detail-type}</td>
<td>\texttt{detail-type}</td>
<td>Detail Type, Null</td>
</tr>
<tr>
<td></td>
<td>Where \texttt{detail-type} is the literal string for the \texttt{detail-type} field of the event such as \texttt{&quot;AWS API Call via CloudTrail&quot;} and \texttt{&quot;EC2 Instance State-change Notification&quot;}. To see more possible values for \texttt{detail-type}, see the example events in</td>
<td></td>
</tr>
</tbody>
</table>
## Using Conditions

<table>
<thead>
<tr>
<th>Condition Key</th>
<th>Key/Value Pair</th>
<th>Evaluation Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>events:</td>
<td>&quot;events:detail.userIdentity.principalId&quot;</td>
<td>Principal Id, Null</td>
</tr>
<tr>
<td>detail.userIdentity.principalId</td>
<td>principal-id</td>
<td>Where principal-id is the literal string for the detail.userIdentity.principalId field of the event with detail-type &quot;AWS API Call via CloudTrail&quot; such as &quot;AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName.&quot;.</td>
</tr>
<tr>
<td>events:</td>
<td>&quot;events:detail.service&quot;</td>
<td>service, Null</td>
</tr>
<tr>
<td>detail.service</td>
<td>service</td>
<td>Where service is the literal string for the detail.service field of the event, such as &quot;ABUSE&quot;.</td>
</tr>
<tr>
<td>events:</td>
<td>&quot;events:detail.eventTypeCode&quot;</td>
<td>eventTypeCode, Null</td>
</tr>
<tr>
<td>detail.eventTypeCode</td>
<td>eventTypeCode</td>
<td>Where eventTypeCode is the literal string for the detail.eventTypeCode field of the event, such as &quot;AWS_ABUSE_DOS_REPORT&quot;.</td>
</tr>
<tr>
<td>events:</td>
<td>&quot;events:TargetArn&quot;</td>
<td>ARN, Null</td>
</tr>
<tr>
<td>TargetArn</td>
<td>target-arn</td>
<td>Where target-arn is the ARN of the target that can be put to a rule such as &quot;arn:aws:lambda:<em>:</em>:function:*&quot;.</td>
</tr>
</tbody>
</table>

For example policy statements for EventBridge, see [Overview of Managing Access Permissions to Your EventBridge Resources (p. 123)](#).

### Topics

- Example 1: Limit Access to a Specific Source (p. 142)
- Example 2: Define Multiple Sources That Can Be Used in an Event Pattern Individually (p. 143)
- Example 3: Define a Source and a DetailType That Can Be Used in an Event Pattern (p. 144)
- Example 4: Ensure That the Source Is Defined in the Event Pattern (p. 145)
- Example 5: Define a List of Allowed Sources in an Event Pattern with Multiple Sources (p. 146)
- Example 6: Limiting PutRule Access by detail.service (p. 147)
- Example 7: Limiting PutRule Access by detail.eventTypeCode (p. 147)
- Example 8: Ensure That AWS CloudTrail Events for API Calls from a Certain PrincipalId Are Consumed (p. 148)
- Example 9: Limiting Access to Targets (p. 149)
Example 1: Limit Access to a Specific Source

The following example policies can be attached to an IAM user. Policy A allows the `PutRule` API action for all events, whereas Policy B allows `PutRule` only if the event pattern of the rule being created matches Amazon EC2 events.

Policy A:—allow any events

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPutRuleForAllEvents",
      "Effect": "Allow",
      "Action": "events:PutRule",
      "Resource": "*"
    }
  ]
}
```

Policy B:—allow events only from Amazon EC2

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPutRuleForAllEC2Events",
      "Effect": "Allow",
      "Action": "events:PutRule",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "events:source": "aws.ec2"
        }
      }
    }
  ]
}
```

*EventPattern* is a mandatory argument to `PutRule`. Hence, if the user with Policy B calls `PutRule` with an event pattern like the following.

```
{
  "source": [ "aws.ec2" ]
}
```

The rule would be created because the policy allows for this specific source: that is, "aws.ec2". However, if the user with Policy B calls `PutRule` with an event pattern like the following, the rule creation would be denied because the policy doesn't allow for this specific source: that is, "aws.s3".

```
{
  "source": [ "aws.s3" ]
}
```

Essentially, the user with Policy B is only allowed to create a rule that would match the events originating from Amazon EC2; hence, they're only allowed access to the events from Amazon EC2.

See the following table for a comparison of Policy A and Policy B.
<table>
<thead>
<tr>
<th>Event Pattern</th>
<th>Allowed by Policy A</th>
<th>Allowed by Policy B</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ &quot;source&quot;: [ &quot;aws.ec2&quot; ]</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>{ &quot;source&quot;: [ &quot;aws.ec2&quot;, &quot;aws.s3&quot; ]</td>
<td>Yes</td>
<td>No (Source aws.s3 is not allowed)</td>
</tr>
<tr>
<td>{ &quot;source&quot;: [ &quot;aws.ec2&quot; ], &quot;detail-type&quot;: [ &quot;EC2 Instance State-change Notification&quot; ]</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>{ &quot;detail-type&quot;: [ &quot;EC2 Instance State-change Notification&quot; ]</td>
<td>Yes</td>
<td>No (Source must be specified)</td>
</tr>
</tbody>
</table>

**Example 2: Define Multiple Sources That Can Be Used in an Event Pattern Individually**

The following policy allows events from Amazon EC2 or Amazon ECS. In other words, it allows an IAM user or role to create a rule where the source in the EventPattern is specified as either "aws.ec2" or "aws.ecs". Not defining the source results in a "deny".

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "AllowPutRuleIfSourceIsEC2OrECS",
         "Effect": "Allow",
         "Action": "events:PutRule",
         "Resource": "+",
         "Condition": {
            "StringEquals": {
               "events:source": [ "aws.ec2", "aws.ecs" ]
            }
         }
      }
   ]
}
```

See the following table for examples of event patterns that would be allowed or denied by this policy.
**Event Pattern** | **Allowed by the Policy**
---|---
{ 
  "source": [ "aws.ec2" ]
} | Yes

{ 
  "source": [ "aws.ecs" ]
} | Yes

{ 
  "source": [ "aws.s3" ]
} | No

{ 
  "source": [ "aws.ec2", 
  "aws.ecs" ]
} | No

{ 
  "detail-type": [ "AWS API Call via CloudTrail" ]
} | No

---

**Example 3: Define a Source and a DetailType That Can Be Used in an Event Pattern**

The following policy allows events only from the `aws.ec2` source with `detail-type` equal to `EC2 instance state change notification`.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPutRuleIfSourceIsEC2AndDetailTypeIsInstanceStateChangeNotification",
      "Effect": "Allow",
      "Action": "events:PutRule",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "events:source": "aws.ec2",
          "events:detail-type": "EC2 Instance State-change Notification"
        }
      }
    }
  ]
}
```

See the following table for examples of event patterns that would be allowed or denied by this policy.
### Event Pattern | Allowed by the Policy
---|---
```json
{"source": ["aws.ec2"]}
```
<table>
<thead>
<tr>
<th>No</th>
</tr>
</thead>
</table>
```json
{"source": ["aws.ecs"]}
```
<table>
<thead>
<tr>
<th>No</th>
</tr>
</thead>
</table>
```json
{"source": ["aws.ec2"],
"detail-type": ["EC2 Instance State-change Notification"]}
```
<table>
<thead>
<tr>
<th>Yes</th>
</tr>
</thead>
</table>
```json
{"source": ["aws.ec2"],
"detail-type": ["EC2 Instance Health Failed"]}
```
<table>
<thead>
<tr>
<th>No</th>
</tr>
</thead>
</table>
```json
{"detail-type": ["EC2 Instance State-change Notification"]}
```
<table>
<thead>
<tr>
<th>No</th>
</tr>
</thead>
</table>

### Example 4: Ensure That the Source Is Defined in the Event Pattern

The following policy allows creating rules with EventPatterns that must have the source field. In other words, an IAM user or role can't create a rule with an EventPattern that doesn't provide a specific source.

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

{"Sid": "AllowPutRuleIfSourceIsSpecified",
"Effect": "Allow",
"Action": "events:PutRule",
"Resource": "*",
"Condition": {
"Null": {
"events:source": "false"
}

}

]}
```

See the following table for examples of event patterns that would be allowed or denied by this policy.
### Using Conditions

#### Event Pattern

<table>
<thead>
<tr>
<th>Event Pattern</th>
<th>Allowed by the Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>`{ &quot;source&quot;: [ &quot;aws.ec2&quot; ],</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;detail-type&quot;: [ &quot;EC2 Instance State-change Notification&quot; ] }</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;source&quot;: [ &quot;aws.ecs&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;aws.ec2&quot; ]</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td>No</td>
</tr>
<tr>
<td>&quot;detail-type&quot;: [ &quot;EC2 Instance State-change Notification&quot; ] }</td>
<td></td>
</tr>
</tbody>
</table>

### Example 5: Define a List of Allowed Sources in an Event Pattern with Multiple Sources

The following policy allows creating rules with EventPatterns that can have multiple sources in them. Each source listed in the event pattern must be a member of the list provided in the condition. When using the `ForAllValues` condition, make sure that at least one of the items in the condition list is defined.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
        "Sid": "AllowPutRuleIfSourceIsSpecifiedAndIsEitherS3OrEC2OrBoth",
        "Effect": "Allow",
        "Action": "events:PutRule",
        "Resource": "*",
        "Condition": {
          "ForAllValues:StringEquals": {
            "events:source": [ "aws.ec2", "aws.s3" ]
          },
          "Null": {
            "events:source": "false"
          }
        }
      }
   ]
}
```

See the following table for examples of event patterns that would be allowed or denied by this policy.

<table>
<thead>
<tr>
<th>Event Pattern</th>
<th>Allowed by the Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>`{ &quot;source&quot;: [ &quot;aws.ec2&quot; ] }</td>
<td>Yes</td>
</tr>
<tr>
<td>Event Pattern</td>
<td>Allowed by the Policy</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>{</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;source&quot;: [ &quot;aws.ec2&quot;, &quot;aws.s3&quot; ]</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td>No</td>
</tr>
<tr>
<td>&quot;source&quot;: [ &quot;aws.ec2&quot;,</td>
<td></td>
</tr>
<tr>
<td>&quot;aws.autoscaling&quot; ]</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td>No</td>
</tr>
<tr>
<td>&quot;detail-type&quot;: [ &quot;EC2 Instance</td>
<td></td>
</tr>
<tr>
<td>State-change Notification&quot; ]</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

Example 6: Limiting PutRule Access by detail.service

You can restrict an IAM user or role to creating rules only for events that have a certain value in the `events:details.service` field. The value of `events:details.service` is not necessarily the name of an AWS service.

This policy condition is helpful when working with events from AWS Health that relate to security or abuse. By using this policy condition, you can limit access to these sensitive alerts to only those users who need to see them.

For example, the following policy allows the creation of rules only for events where the value of `events:details.service` is `ABUSE`.

```json
{   "Version": "2012-10-17",
    "Statement": [
        {           "Sid": "AllowPutRuleEventsWithDetailServiceEC2",
            "Effect": "Allow",
            "Action": "events:PutRule",
            "Resource": "+",
            "Condition": {
                "StringEquals": {
                    "events:detail.service": "ABUSE"
                }
            }
        }
    ]
}
```

Example 7: Limiting PutRule Access by detail.eventTypeCode

You can restrict an IAM user or role to creating rules only for events that have a certain value in the `events:details.eventTypeCode` field. This policy condition is helpful when working with events from AWS Health that relate to security or abuse. By using this policy condition, you can limit access to these sensitive alerts to only those users who need to see them.

For example, the following policy allows the creation of rules only for events where the value of `events:details.eventTypeCode` is `AWS_ABUSE_DOS_REPORT`.

```json
{   "Version": "2012-10-17",
    "Statement": [
        {           "Sid": "AllowPutRuleEventsWithEventTypeDOS",
            "Effect": "Allow",
            "Action": "events:PutRule",
            "Resource": "+",
            "Condition": {
                "StringEquals": {
                    "events:detail.eventTypeCode": "AWS_ABUSE_DOS_REPORT"
                }
            }
        }
    ]
}
```
Example 8: Ensure That AWS CloudTrail Events for API Calls from a Certain PrincipalId Are Consumed

All AWS CloudTrail events have the ID of the user who made the API call (PrincipalId) in the detail.userIdentity.principalId path of an event. With the help of the events:detail.userIdentity.principalId condition key, you can limit the access of IAM users or roles to the CloudTrail events for only those coming from a specific account.

See the following table for examples of event patterns that would be allowed or denied by this policy.

<table>
<thead>
<tr>
<th>Event Pattern</th>
<th>Allowed by the Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
<td>No</td>
</tr>
<tr>
<td>&quot;detail-type&quot;: [ &quot;AWS API Call via CloudTrail&quot; ]</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;detail-type&quot;: [ &quot;AWS API Call via CloudTrail&quot; ],</td>
<td></td>
</tr>
<tr>
<td>&quot;detail.userIdentity.principalId&quot;: [ &quot;AIDAJ45Q7YFFAREXAMPLE&quot; ]</td>
<td></td>
</tr>
</tbody>
</table>
### Event Pattern

<table>
<thead>
<tr>
<th>Event Pattern</th>
<th>Allowed by the Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>`{</td>
<td>No</td>
</tr>
<tr>
<td>&quot;detail-type&quot;: [ &quot;AWS API Call via CloudTrail&quot; ],</td>
<td></td>
</tr>
<tr>
<td>&quot;detail.userIdentity.principalId&quot;: [ &quot;AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName&quot; ]</td>
<td></td>
</tr>
</tbody>
</table>

### Example 9: Limiting Access to Targets

If an IAM user or role has `events:PutTargets` permission, they can add any target under the same account to the rules that they are allowed to access. For example, the following policy limits adding targets to only a specific rule (`MyRule` under account `123456789012`).

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutTargetsOnASpecificRule",
            "Effect": "Allow",
            "Action": "events:PutTargets",
            "Resource": "arn:aws:events:us-east-1:123456789012:rule/MyRule"
        }
    ]
}
```

To limit what target can be added to the rule, use the `events:TargetArn` condition key. For example, you can limit targets to only Lambda functions, as in the following example.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutTargetsOnASpecificRuleAndOnlyLambdaFunctions",
            "Effect": "Allow",
            "Action": "events:PutTargets",
            "Condition": {
                "ArnLike": {
                    "events:TargetArn": "arn:aws:lambda:*:*:function:*"
                }
            }
        }
    ]
}
```

### Logging and Monitoring in Amazon EventBridge

Amazon EventBridge is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in EventBridge. CloudTrail captures API calls made by or on behalf of your AWS account. The calls captured include calls from the CloudWatch console and code calls to the
EventBridge API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for EventBridge. 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 EventBridge, 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, including how to configure and enable it, see the AWS CloudTrail User Guide.

Topics
- EventBridge Information in CloudTrail (p. 150)
- Example: EventBridge Log File Entries (p. 151)

EventBridge Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in EventBridge, 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 EventBridge, 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:

- Overview for Creating a Trail
- 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

EventBridge supports logging the following actions as events in CloudTrail log files:

- DeleteRule
- DescribeEventBus
- DescribeRule
- DisableRule
- EnableRule
- ListRuleNamesByTarget
- ListRules
- ListTargetsByRule
- PutPermission
- PutRule
- PutTargets
- RemoveTargets
- TestEventPattern

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 AWS Identity and Access Management (IAM) 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.

Example: EventBridge 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 CloudTrail log file entry shows that a user called the EventBridge `PutRule` action.

```json
{
    "eventVersion": "1.03",
    "userIdentity": {
        "type": "Root",
        "principalId": "123456789012",
        "arn": "arn:aws:iam::123456789012:root",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2015-11-17T23:56:15Z"
            }
        }
    },
    "eventTime": "2015-11-18T00:11:28Z",
    "eventSource": "events.amazonaws.com",
    "eventName": "PutRule",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS CloudWatch Console",
    "requestParameters": {
        "description": "",
        "name": "cttest2",
        "state": "ENABLED",
        "eventPattern": {
            "source": ["aws.ec2"],
            "detail-type": ["EC2 Instance State-change Notification"]
        }
    },
    "responseElements": {
        "ruleArn": "arn:aws:events:us-east-1:123456789012:rule/cttest2"
    },
    "requestID": "e9caf887-8d88-11e5-a331-3332aa445952",
    "eventID": "49d14f36-6450-44a5-a501-b0fdcdfaeb98",
    "eventType": "AwsApiCall",
    "apiVersion": "2015-10-07",
    "recipientAccountId": "123456789012"
}
```
Compliance Validation in Amazon EventBridge

Third-party auditors assess the security and compliance of Amazon EventBridge as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using EventBridge is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating Resources with Rules in the AWS Config Developer Guide – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in Amazon EventBridge

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, EventBridge offers several features to help support your data resiliency and backup needs.

Infrastructure Security in Amazon EventBridge

As a managed service, Amazon EventBridge is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access EventBridge through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). 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.

You can call these API operations from any network location, but EventBridge does support resource-based access policies, which can include restrictions based on the source IP address. You can also use EventBridge policies to control access from specific Amazon Virtual Private Cloud (Amazon VPC) endpoints or specific VPCs. Effectively, this isolates network access to a given EventBridge resource from only the specific VPC within the AWS network.

**Configuration and Vulnerability Analysis in Amazon EventBridge**

Configuration and IT controls are a shared responsibility between AWS and you, our customer. For more information, see the AWS shared responsibility model.
Tagging Your Amazon EventBridge Resources

A *tag* is a custom attribute label that you assign or that AWS assigns to an AWS resource. Each tag has two parts:

- A *tag key* (for example, CostCenter, Environment, or Project). Tag keys are case sensitive.
- An optional field known as a *tag value* (for example, 111122223333 or Production). Omitting the tag value is the same as using an empty string. Like tag keys, tag values are case sensitive.

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. For example, you could assign the same tag to a EventBridge rule that you assign to an EC2 instance.
- Track your AWS costs. You activate these tags on the AWS Billing and Cost Management dashboard. AWS uses the tags to categorize your costs and deliver a monthly cost allocation report to you. For more information, see Use Cost Allocation Tags in the AWS Billing and Cost Management User Guide.

The following sections provide more information about tags for EventBridge.

Supported Resources in EventBridge

The following resources in EventBridge support tagging:

- Rules
- Event buses

For information about adding and managing tags, see Managing Tags (p. 154).

Managing Tags

Tags consist of the *Key* and *Value* properties on a resource. You can use the CloudWatch console, the AWS CLI, or the EventBridge API to add, edit, or delete the values for these properties. For information about working with tags, see the following:

- `TagResource`, `UntagResource`, and `ListTagsForResource` in the *Amazon CloudWatch Events API Reference*
- `tag-resource`, `untag-resource`, and `list-tags-for-resource` in the *Amazon CloudWatch CLI Reference*
- `Working with Tag Editor` in the *Resource Groups User Guide*

Tag Naming and Usage Conventions

The following basic naming and usage conventions apply to using tags with EventBridge resources:
- Each resource can have a maximum of 50 tags.
- For each resource, each tag key must be unique, and each tag key can have only one value.
- The maximum tag key length is 128 Unicode characters in UTF-8.
- The maximum tag value length is 256 Unicode characters in UTF-8.
- Allowed characters are letters, numbers, spaces representable in UTF-8, and the following characters: . : + = @ _ / - (hyphen).
- Tag keys and values are case sensitive. As a best practice, decide on a strategy for capitalizing tags and consistently implement that strategy across all resource types. For example, decide whether to use Costcenter, costcenter, or CostCenter and use the same convention for all tags. Avoid using similar tags with inconsistent case treatment.
- The aws: prefix is prohibited for tags because it's reserved for AWS use. You can't edit or delete tag keys or values with this prefix. Tags with this prefix don't count against your tags per resource limit.
Amazon EventBridge Quotas

EventBridge has the following quotas.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default Limit</th>
</tr>
</thead>
</table>
| Event Publishing API requests     | PutEvents operations are limited based on AWS region. See PutEvents Quotas by Region (p. 157).  
You can request a quota increase. For instructions, see AWS Service Quotas |
| All other API requests            | All EventBridge API other than PutEvents are limited to 50 requests per second by default.  
You can request a quota increase. For instructions, see AWS Service Quotas |
| Event buses                       | Up to 100 event buses per account.                                             |
| Event buses - other quotas        | There is no restriction on the rate of events that can be received from AWS services or other AWS accounts. If you send custom events to your event bus using the PutEvents API, the PutEvents API quotas (p. 157) apply. Any events that are sent on to the targets of the rules in your account count against your invocations quota.  
The policy size of an event bus is limited to 10240 characters. This policy size increases each time you grant access to another account. You can see your current policy and its size by using the DescribeEventBus API. You can request a quota increase. For instructions, see AWS Service Quotas. |
| Event pattern                     | 2048 characters maximum.                                                      |
| Invocations                       | An invocation is an event matching a rule and being sent on to the rule's targets. Quotas vary by region.  
See Invocation Quotas by Region (p. 158).  
You can request a quota increase. For instructions, see AWS Service Quotas |
| ListRuleNamesByTarget             | Up to 100 results per page for requests.                                       |
| ListRules                         | Up to 100 results per page for requests.                                       |
| ListTargetsByRule                 | Up to 100 results per page for requests.                                       |
| PutTargets                        | 10 entries per request. Up to 5 targets per rule.                             |
| RemoveTargets                     | 10 entries per request.                                                       |
| Rules                             | 300 per event bus. You can request a quota increase. For instructions, see AWS Service Quotas |
Before requesting a quota increase, examine your rules. You may have multiple rules each matching to very specific events. Consider broadening their scope by using fewer identifiers in your Events and Event Patterns in EventBridge (p. 36). In addition, a rule can invoke several targets each time it matches an event. Consider adding more targets to your rules.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Systems Manager Run Command target</td>
<td>1 target key and 1 target value</td>
</tr>
<tr>
<td></td>
<td>Systems Manager Run Command does not currently support multiple target values.</td>
</tr>
<tr>
<td>Targets</td>
<td>Up to 5 targets per rule.</td>
</tr>
</tbody>
</table>

**PutEvents Quotas by Region**

<table>
<thead>
<tr>
<th>Regions</th>
<th>Transactions per Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>• US East (N. Virginia)</td>
<td><strong>PutEvents</strong> is limited to 2400 requests per second by default in these regions.</td>
</tr>
<tr>
<td>• US West (Oregon)</td>
<td></td>
</tr>
<tr>
<td>• US East (Ohio)</td>
<td>You can request a quota increase. For instructions, see AWS Service Quotas</td>
</tr>
<tr>
<td>• Europe (Ireland)</td>
<td></td>
</tr>
<tr>
<td>• Europe (Frankfurt)</td>
<td></td>
</tr>
<tr>
<td>• US West (N. California)</td>
<td><strong>PutEvents</strong> is limited to 1200 requests per second by default in these regions.</td>
</tr>
<tr>
<td>• Europe (London)</td>
<td></td>
</tr>
<tr>
<td>• Asia Pacific (Sydney)</td>
<td>You can request a quota increase. For instructions, see AWS Service Quotas</td>
</tr>
<tr>
<td>• Asia Pacific (Tokyo)</td>
<td></td>
</tr>
<tr>
<td>• Asia Pacific (Singapore)</td>
<td></td>
</tr>
<tr>
<td>• Canada (Central)</td>
<td><strong>PutEvents</strong> is limited to 600 requests per second by default in these regions.</td>
</tr>
<tr>
<td>• Europe (Paris)</td>
<td></td>
</tr>
<tr>
<td>• Europe (Stockholm)</td>
<td>You can request a quota increase. For instructions, see AWS Service Quotas</td>
</tr>
<tr>
<td>• South America (São Paulo)</td>
<td></td>
</tr>
<tr>
<td>• Asia Pacific (Seoul)</td>
<td></td>
</tr>
<tr>
<td>• Asia Pacific (Mumbai)</td>
<td></td>
</tr>
<tr>
<td>• Asia Pacific (Hong Kong)</td>
<td></td>
</tr>
<tr>
<td>• Middle East (Bahrain)</td>
<td></td>
</tr>
<tr>
<td>• China (Ningxia)</td>
<td><strong>PutEvents</strong> is limited to 400 requests per second by default in these regions.</td>
</tr>
<tr>
<td>• China (Beijing)</td>
<td></td>
</tr>
<tr>
<td>• Asia Pacific (Osaka-Local)</td>
<td>You can request a quota increase. For instructions, see AWS Service Quotas</td>
</tr>
</tbody>
</table>
Invocation Quotas by Region

An invocation is an event matching a rule and being sent on to the rule’s targets. If the invocation of a target fails due to a problem with the target service, account throttling, etc., new attempts are made for up to 24 hours for a specific invocation.

If you are receiving events from another account, each of those events that matches a rule in your account and is sent on to the rule’s targets counts against your account’s quota of invocations per second.

After your reach the one of the following invocation quotas in your region invocations are throttled. They still happen but they are delayed.

<table>
<thead>
<tr>
<th>Regions</th>
<th>Invocations per Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>• US East (N. Virginia)</td>
<td>The invocation quota for these regions is limited to 4500 requests per second by default in these regions.</td>
</tr>
<tr>
<td>• US West (Oregon)</td>
<td></td>
</tr>
<tr>
<td>• US East (Ohio)</td>
<td>You can request a quota increase. For instructions, see AWS Service Quotas</td>
</tr>
<tr>
<td>• Europe (Ireland)</td>
<td></td>
</tr>
<tr>
<td>• Europe (Frankfurt)</td>
<td></td>
</tr>
<tr>
<td>• US West (N. California)</td>
<td>The invocation quota for these regions is limited to 2250 requests per second by default in these regions.</td>
</tr>
<tr>
<td>• Europe (London)</td>
<td>You can request a quota increase. For instructions, see AWS Service Quotas</td>
</tr>
<tr>
<td>• Asia Pacific (Sydney)</td>
<td></td>
</tr>
<tr>
<td>• Asia Pacific (Tokyo)</td>
<td></td>
</tr>
<tr>
<td>• Asia Pacific (Singapore)</td>
<td></td>
</tr>
<tr>
<td>• Canada (Central)</td>
<td>The invocation quota for these regions is limited to 1100 requests per second by default in these regions.</td>
</tr>
<tr>
<td>• South America (São Paulo)</td>
<td>You can request a quota increase. For instructions, see AWS Service Quotas</td>
</tr>
<tr>
<td>• Europe (Paris)</td>
<td></td>
</tr>
<tr>
<td>• Europe (Stockholm)</td>
<td></td>
</tr>
<tr>
<td>• Asia Pacific (Seoul)</td>
<td></td>
</tr>
<tr>
<td>• Asia Pacific (Mumbai)</td>
<td></td>
</tr>
<tr>
<td>• Asia Pacific (Hong Kong)</td>
<td></td>
</tr>
<tr>
<td>• Middle East (Bahrain)</td>
<td></td>
</tr>
<tr>
<td>• China (Ningxia)</td>
<td>The invocation quota for these regions is limited to 750 requests per second by default in these regions.</td>
</tr>
<tr>
<td>• China (Beijing)</td>
<td>You can request a quota increase. For instructions, see AWS Service Quotas</td>
</tr>
<tr>
<td>• Asia Pacific (Osaka-Local)</td>
<td></td>
</tr>
</tbody>
</table>
Troubleshooting Amazon EventBridge

You can use the steps in this section to troubleshoot Amazon EventBridge.

Topics
- My rule was triggered but my Lambda function was not invoked (p. 159)
- I have just created/modified a rule but it did not match a test event (p. 160)
- My rule did not self-trigger at the time specified in the ScheduleExpression (p. 161)
- My rule did not trigger at the time that I expected (p. 161)
- My rule matches IAM API calls but my rule was not triggered (p. 161)
- My rule is not working because the IAM role associated with the rule is ignored when the rule is triggered (p. 162)
- I created a rule with an EventPattern that is supposed to match a resource, but I don't see any events that match the rule (p. 162)
- My event's delivery to the target experienced a delay (p. 162)
- Some events were never delivered to my target (p. 162)
- My rule was triggered more than once in response to one event. What guarantee does EventBridge offer for triggering rules or delivering events to the targets? (p. 163)
- Preventing Infinite Loops (p. 163)
- My events are not delivered to the target Amazon SQS queue (p. 163)
- My rule is being triggered but I don't see any messages published into my Amazon SNS topic (p. 163)
- My Amazon SNS topic still has permissions for EventBridge even after I deleted the rule associated with the Amazon SNS topic (p. 165)
- Which IAM condition keys can I use with EventBridge (p. 165)
- How can I tell when EventBridge rules are broken (p. 165)

My rule was triggered but my Lambda function was not invoked

Make sure you have the right permissions set for your Lambda function. Run the following command using AWS CLI (replace the function name with your function and use the AWS Region your function is in):

```
aws lambda get-policy --function-name MyFunction --region us-east-1
```

You should see an output similar to the following:

```
{
  "Policy": "{"Version":"2012-10-17"},
  "Statement": [
    {"Condition":{"ArnLike":{"AWS:SourceArn":"arn:aws:events:us-east-1:123456789012:rule/MyRule"}}}
}
```
I have just created/modified a rule but it did not match a test event

```
{"Action":"lambda:InvokeFunction\",  
"Effect":"Allow\",  
"Principal":{"Service":"events.amazonaws.com"},  
"Sid":"MyId\"}  
\"Id\":\"default\"}
```

If you see the following:

A client error (ResourceNotFoundException) occurred when calling the GetPolicy operation: The resource you requested does not exist.

Or, you see the output but you can't locate events.amazonaws.com as a trusted entity in the policy, run the following command:

```
aws lambda add-permission  
  --function-name MyFunction  
  --statement-id MyId  
  --action 'lambda:InvokeFunction'  
  --principal events.amazonaws.com  
  --source-arn arn:aws:events:us-east-1::rule/MyRule
```

**Note**

If the policy is incorrect, you can also edit the rule in the EventBridge console by removing and then adding it back to the rule. The EventBridge console will set the correct permissions on the target.

If you're using a specific Lambda alias or version, you must add the --qualifier parameter in the aws lambda get-policy and aws lambda add-permission commands.

```
aws lambda add-permission  
  --function-name MyFunction  
  --statement-id MyId  
  --action 'lambda:InvokeFunction'  
  --principal events.amazonaws.com  
  --source-arn arn:aws:events:us-east-1:123456789012:rule/MyRule  
  --qualifier alias or version
```

Another reason the Lambda function would fail to trigger is if the policy you see when running get-policy contains a SourceAccount field. A SourceAccount setting prevents EventBridge from being able to invoke the function.

**I have just created/modified a rule but it did not match a test event**

When you make a change to a rule or to its targets, incoming events might not immediately start or stop matching to new or updated rules. Allow a short period of time for changes to take effect. If, after this short period, events still do not match, you can also check CloudWatch metrics for your rule such as TriggeredRules, Invocations, and FailedInvocations for further debugging. For more information about these metrics, see Amazon CloudWatch Events Metrics and Dimensions in the Amazon CloudWatch User Guide.

If the rule is triggered by an event from an AWS service, you can also use the TestEventPattern action to test the event pattern of your rule with a test event to make sure the event pattern of your
My rule did not self-trigger at the time specified in the ScheduleExpression

ScheduleExpressions are in UTC. Make sure you have set the schedule for rule to self-trigger in the UTC timezone. If the ScheduleExpression is correct, then follow the steps under I have just created/modified a rule but it did not match a test event (p. 160).

Note
Depending on caching, the first instance of a scheduled rule may be dropped. It can take a short period for rules to take effect. Any trigger that arrives within this period may not match your created or updated schedule.

My rule did not trigger at the time that I expected

EventBridge doesn’t support setting an exact start time when you create a rule to run every time period. The count down to run time begins as soon as you create the rule.

You can use a cron expression to invoke targets at a specified time. For example, you can use a cron expression to create a rule that is triggered every 4 hours exactly on 0 minute. In the CloudWatch console, you’d use the cron expression 0 0/4 * * ? *, and with the AWS CLI you’d use the cron expression cron(0 0/4 * * ? *). For example, to create a rule named TestRule that is triggered every 4 hours using the AWS CLI, you would type the following at a command prompt:

```
aws events put-rule --name TestRule --schedule-expression 'cron(0 0/4 * * ? *)'
```

You can use the 0/5 * * * ? * cron expression to trigger a rule every 5 minutes. For example:

```
aws events put-rule --name TestRule --schedule-expression 'cron(0/5 * * * ? *)'
```

EventBridge does not provide second-level precision in schedule expressions. The finest resolution using a cron expression is a minute. Due to the distributed nature of the EventBridge and the target services, the delay between the time the scheduled rule is triggered and the time the target service honors the execution of the target resource might be several seconds. Your scheduled rule will be triggered within that minute but not on the precise 0th second.

My rule matches IAM API calls but my rule was not triggered

The IAM service is only available in the US East (N. Virginia) Region, so any AWS API call events from IAM are only available in that region. For more information, see EventBridge Event Examples from Supported AWS Services (p. 57).
My rule is not working because the IAM role associated with the rule is ignored when the rule is triggered

IAM roles for rules are only used for relating events to Kinesis streams. For Lambda functions and Amazon SNS topics, you need to provide resource-based permissions.

Make sure your regional AWS STS endpoints are enabled. EventBridge talks to the regional AWS STS endpoints when assuming the IAM role you provided. For more information, see Activating and Deactivating AWS STS in an AWS Region in the IAM User Guide.

I created a rule with an EventPattern that is supposed to match a resource, but I don't see any events that match the rule

Most services in AWS treat the colon (:) or forward slash (/) as the same character in Amazon Resource Names (ARNs). However, EventBridge uses an exact match in event patterns and rules. Be sure to use the correct ARN characters when creating event patterns so that they match the ARN syntax in the event to match.

Moreover, not every event has the resources field populated (such as AWS API call events from CloudTrail).

My event's delivery to the target experienced a delay

EventBridge tries to deliver an event to a target for up to 24 hours, except in scenarios where your target resource is constrained. The first attempt is made as soon as the event arrives in the event stream. However, if the target service is having problems, EventBridge automatically reschedules another delivery in the future. If 24 hours has passed since the arrival of event, no more attempts are scheduled and the FailedInvocations metric is published in CloudWatch. We recommend that you create a CloudWatch alarm on the FailedInvocations metric.

Some events were never delivered to my target

If a target of a EventBridge rule is constrained for a prolonged time, EventBridge may not retry delivery. For example, if the target is not provisioned to handle the incoming event traffic and the target service is throttling the requests that EventBridge makes on your behalf, then EventBridge may not retry delivery.
My rule was triggered more than once in response to one event. What guarantee does EventBridge offer for triggering rules or delivering events to the targets?

In rare cases, the same rule can be triggered more than once for a single event or scheduled time, or the same target can be invoked more than once for a given triggered rule.

Preventing Infinite Loops

In EventBridge, it is possible to create rules that lead to infinite loops, where a rule is fired repeatedly. For example, a rule might detect that ACLs have changed on an S3 bucket, and trigger software to change them to the desired state. If the rule is not written carefully, the subsequent change to the ACLs fires the rule again, creating an infinite loop.

To prevent this, write the rules so that the triggered actions do not re-fire the same rule. For example, your rule could fire only if ACLs are found to be in a bad state, instead of after any change.

An infinite loop can quickly cause higher than expected charges. We recommend that you use budgeting, which alerts you when charges exceed your specified limit. For more information, see Managing Your Costs with Budgets.

My events are not delivered to the target Amazon SQS queue

The Amazon SQS queue may be encrypted. If you create a rule with an encrypted Amazon SQS queue as a target, you must have the following section included in your KMS key policy for the event to be successfully delivered to the encrypted queue.

```json
{
    "Sid": "Allow CWE to use the key",
    "Effect": "Allow",
    "Principal": {
        "Service": "events.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*"
}
```

My rule is being triggered but I don't see any messages published into my Amazon SNS topic

Make sure you have the right permission set for your Amazon SNS topic. Run the following command using AWS CLI (replace the topic ARN with your topic and use the AWS Region your topic is in):
My rule is being triggered but I don't see any messages published into my Amazon SNS topic

```
```

You should see policy attributes similar to the following:

```
{"Version":"2012-10-17",
 "Id":"__default_policy_ID",
 "Statement": [{"Sid":"__default_statement_ID",
 "Effect":"Allow",
 "Principal":{"AWS":"*"},
 "SNS:Publish", "SNS:RemovePermission", "SNS:AddPermission", "SNS:Receive", "SNS:GetTopicAttributes"],
 "Condition": {"StringEquals":{"AWS:SourceOwner":"123456789012"}}},
 {"Sid":"Allow_Publish_Events",
 "Effect":"Allow",
 "Principal": {"Service": "events.amazonaws.com"},
 "Action": ["sns:Publish"],
```

If you see a policy similar to the following, you have only the default policy set:

```
{"Version":"2008-10-17",
 "Id":"__default_policy_ID",
 "Statement": [{"Sid":"__default_statement_ID",
 "Effect":"Allow",
 "Principal":{"AWS":"*"},
 "SNS:Publish", "SNS:RemovePermission", "SNS:AddPermission", "SNS:Receive", "SNS:GetTopicAttributes"],
 "Condition": {"StringEquals":{"AWS:SourceOwner":"123456789012"}}]}
```

If you don't see `events.amazonaws.com` with `Publish` permission in your policy, use the AWS CLI to set topic policy attribute.

Copy the current policy and add the following statement to the list of statements:

```
{"Sid":"Allow_Publish_Events",
 "Effect":"Allow",
 "Principal": {"Service": "events.amazonaws.com"},
 "Action": ["sns:Publish"],
```

The new policy should look like the one described earlier.

Set topic attributes with the AWS CLI:
My Amazon SNS topic still has permissions for EventBridge even after I deleted the rule associated with the Amazon SNS topic

When you create a rule with Amazon SNS as the target, EventBridge adds the permission to your Amazon SNS topic on your behalf. If you delete the rule shortly after you create it, EventBridge might be unable to remove the permission from your Amazon SNS topic. If this happens, you can remove the permission from the topic using the `aws sns set-topic-attributes` command. For more information about resource-based permissions for sending events, see Using Resource-Based Policies for EventBridge (p. 134).

Which IAM condition keys can I use with EventBridge

EventBridge supports the AWS-wide condition keys (see Available Keys in the IAM User Guide), plus the following service-specific condition keys. For more information, see Using IAM Policy Conditions for Fine-Grained Access Control (p. 139).

How can I tell when EventBridge rules are broken

You can use the following alarm to notify you when your EventBridge rules are broken.

To create an alarm to alert when rules are broken

2. Choose Create Alarm. In the CloudWatch Metrics by Category pane, choose Events Metrics.
3. In the list of metrics, select FailedInvocations.
4. Above the graph, choose Statistic, Sum.
5. For Period, choose a value, for example 5 minutes. Choose Next.
6. Under Alarm Threshold, for Name, type a unique name for the alarm, for example myFailedRules. For Description, type a description of the alarm, for example Rules are not delivering events to targets.
7. For is, choose >= and 1. For for, enter 10.
8. Under Actions, for Whenever this alarm, choose State is ALARM.
9. For Send notification to, select an existing Amazon SNS topic or create a new one. To create a new topic, choose New list. Type a name for the new Amazon SNS topic, for example: myFailedRules.
10. For Email list, type a comma-separated list of email addresses to be notified when the alarm changes to the ALARM state.
11. Choose **Create Alarm**.
# Document History

The following table describes important changes in each release of the *Amazon EventBridge User Guide*, beginning in July 2019. For notification about updates to this documentation, you can subscribe to an RSS feed.

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tags for Event Buses</td>
<td>This release allows you to create and manage tags for event buses. You can add tags when creating an event bus, and add or manage existing tags by calling the related API. For more information, see the following.</td>
<td>February 24, 2020</td>
</tr>
<tr>
<td></td>
<td>• <a href="#">Tagging Your Amazon EventBridge Resources</a> (p. 154)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <a href="#">Tag-based Policies</a> (p. 121)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <a href="#">TagResource</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <a href="#">UntagResource</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <a href="#">ListTagsForResource</a></td>
<td></td>
</tr>
<tr>
<td>Increased service quotas</td>
<td>Amazon EventBridge has increased quotas for invocations and for <a href="#">PutEvents</a>. Quotas vary by region, and can be increased if necessary.</td>
<td>February 11, 2020</td>
</tr>
<tr>
<td></td>
<td>• <a href="#">Amazon EventBridge Quotas</a> (p. 156)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <a href="#">PutEvents Quotas by Region</a> (p. 157)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <a href="#">Invocation Quotas by Region</a> (p. 158)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <a href="#">Request a Quota Increase</a></td>
<td></td>
</tr>
<tr>
<td>Added a new topic on transforming target input, and added a link to Application Auto Scaling Events.</td>
<td>Improved documentation on the input transformer. Added a link to Application Auto Scaling Events.</td>
<td>December 20, 2019</td>
</tr>
<tr>
<td></td>
<td>• <a href="#">Transforming Target Input</a> (p. 47)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <a href="#">Use Input Transformer to extract data from an event and input that data to the target</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <a href="#">Tutorial: Use Input Transformer to Customize What Is Passed to the Event Target</a> (p. 19)</td>
<td></td>
</tr>
<tr>
<td>Content-based filtering</td>
<td>Amazon EventBridge now supports content-based filtering with event patterns. For more information see <a href="#">Content-based Filtering with Event Patterns</a> (p. 43).</td>
<td>December 19, 2019</td>
</tr>
<tr>
<td>Added links to Amazon Augmented AI event examples.</td>
<td>Added a link to the Amazon Augmented AI topic in the Amazon SageMaker Developer Guide that provides example events for Amazon Augmented AI. For more information, see the following.</td>
<td>December 13, 2019</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Release Date</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Added links to Amazon Chime event examples.</td>
<td>Added a link to the Amazon Chime topic that provides example events for that service. For more information, see the following.</td>
<td>December 12, 2019</td>
</tr>
</tbody>
</table>
| · Use Events in Amazon Augmented AI | · Automating Amazon Chime with EventBridge  
· EventBridge Event Examples from Supported AWS Services (p. 57) |                  |
| Amazon EventBridge Schemas | You can now manage schemas and generate code bindings for events in Amazon EventBridge. For more information, see the following. | December 1, 2019 |
| · Amazon EventBridge Schema Registry (p. 50)  
· EventBridge Schemas API Reference  
· EventSchemas Resource Type Reference in AWS CloudFormation |                  |
| AWS CloudFormation support for Event Buses | AWS CloudFormation now supports the EventBus resource. It also supports the EventBusName parameter in both the EventBusPolicy and Rule resources. For more information, see Amazon EventBridge Resource Type Reference. | October 7, 2019  |
| New service                | Initial release of Amazon EventBridge.                                       | July 11, 2019    |
|                           |                                                                             |                  |
AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the AWS General Reference.