Amazon EventBridge
User Guide
Table of Contents

What Is Amazon EventBridge? ............................................................................................................................................ 1
Related AWS Services .................................................................................................................................................. 1
Terms and concepts ................................................................................................................................................... 3
Set up & prerequisites .................................................................................................................................................. 4
Sign Up for Amazon Web Services (AWS) .................................................................................................................... 4
Sign in to the Amazon EventBridge Console ............................................................................................................. 4
Account Credentials .................................................................................................................................................... 4
Set up the Command Line Interface ......................................................................................................................... 5
Regional Endpoints .................................................................................................................................................... 5
Get started ................................................................................................................................................................. 6
Create rule .................................................................................................................................................................. 6
Event buses ................................................................................................................................................................. 8
Event bus permissions ................................................................................................................................................ 9
Managing event bus permissions ........................................................................................................................................ 9
Example policy: Send events to the default bus in a different account .............................................................. 11
Example policy: Send events to a custom bus in a different account ................................................................. 11
Example policy: Send events to the same account and restrict updates ............................................................ 12
Example policy: Send events only from a specific rule to the bus in a different Region .................................................................................................................................................................................. 12
Example policy: Send events only from a specific Region to a different Region ................................................. 13
Example policy: Send events only from a specific rule to the bus in a different Region ........................................ 14
Create event bus ........................................................................................................................................................ 15
Events ............................................................................................................................................................................. 16
Event patterns .............................................................................................................................................................. 18
Specify Fields to Match ........................................................................................................................................... 18
Match Values ............................................................................................................................................................... 18
Match on All JSON Data Types ....................................................................................................................................... 19
Simple Matching with Event Patterns .......................................................................................................................... 20
Null values and empty strings ....................................................................................................................................... 22
Arrays ............................................................................................................................................................................ 23
Content-based filtering ................................................................................................................................................ 24
Adding events with PutEvents .................................................................................................................................... 28
Handling Failures When Using PutEvents ................................................................................................................... 29
Sending Events Using the AWS CLI ............................................................................................................................. 30
Calculating PutEvent event entry sizes .......................................................................................................................... 32
Rules .............................................................................................................................................................................. 33
Create rule that triggers on a schedule .......................................................................................................................... 34
Using dead-letter queues .............................................................................................................................................. 35
Considerations for using a dead-letter queue ............................................................................................................... 35
Granting permissions to the dead-letter queue ........................................................................................................... 36
How to resend events from a dead-letter queue .......................................................................................................... 36
Disable or delete rule .................................................................................................................................................... 37
Managed rules ............................................................................................................................................................... 38
Schedule expressions for rules ...................................................................................................................................... 39
Cron Expressions .......................................................................................................................................................... 39
Rate Expressions ........................................................................................................................................................ 41
Targets ............................................................................................................................................................................. 43
Target-specific Parameters ........................................................................................................................................... 43
Permissions ................................................................................................................................................................. 44
Configure targets .......................................................................................................................................................... 45
API destinations ............................................................................................................................................................ 46
API Gateway ................................................................................................................................................................. 54
Cross-account/Region events ......................................................................................................................................... 55
Transforming target input ............................................................................................................................................. 58
Input Transform Examples ........................................................................................................................................... 58
Amazon Interactive Video Service (IVS) Events ......................................................... 100
AWS IoT Analytics Events ............................................................................................. 100
AWS IoT Greengrass Events ......................................................................................... 100
AWS KMS Events ......................................................................................................... 100
Amazon Location Service events ................................................................................ 101
Amazon Macie Classic Events ...................................................................................... 102
Amazon Macie Events .................................................................................................. 106
Amazon Managed Blockchain Events ......................................................................... 106
AWS Managed Services Events .................................................................................... 106
AWS Management Console Blockchain sign-in Events .............................................. 106
AWS OpsWorks Stacks Events ..................................................................................... 107
Amazon QLDB Events .................................................................................................. 110
AWS RAM Events ........................................................................................................ 110
Amazon RDS Events .................................................................................................... 110
Amazon Redshift Events ............................................................................................. 110
Amazon SageMaker Events ........................................................................................ 110
Savings Plans Events .................................................................................................. 110
AWS Security Hub Events .......................................................................................... 110
AWS Server Migration Service Events ....................................................................... 111
AWS Signer Events ...................................................................................................... 111
AWS Step Functions Events ........................................................................................ 111
AWS Storage Gateway Events ..................................................................................... 112
Amazon SWF Events .................................................................................................... 112
AWS Systems Manager Events .................................................................................... 112
  AWS Systems Manager Automation Events .................................................................. 112
  AWS Systems Manager Change Calendar Events ...................................................... 113
  AWS Systems Manager Compliance Events .................................................................. 114
  AWS Systems Manager Maintenance Windows Events ............................................ 115
  AWS Systems Manager Parameter Store Events ....................................................... 117
  AWS Systems Manager Run Command Events .......................................................... 118
  AWS Systems Manager State Manager Events .......................................................... 119
Tag Change Events on AWS Resources ....................................................................... 120
AWS Transit Gateway Events ....................................................................................... 121
AWS Trusted Advisor Events ....................................................................................... 121
Amazon WorkSpaces Events ....................................................................................... 123
AWS X-Ray Events ....................................................................................................... 123
Related AWS services ................................................................................................. 124
Interface VPC Endpoints ............................................................................................ 125
  Availability .................................................................................................................... 125
  Create a VPC Endpoint for EventBridge ................................................................. 125
X-Ray ............................................................................................................................ 127
Receive events from a SaaS partner .............................................................................. 128
Supported SaaS partner integrations .......................................................................... 128
Configure EventBridge ................................................................................................ 130
  Creating a rule that triggers on SaaS partner events ............................................... 130
Salesforce ...................................................................................................................... 132
  Configure AppFlow to use Salesforce as a partner event source ............................ 132
  Configure EventBridge to receive Salesforce events .............................................. 133
Tutorials .......................................................................................................................... 135
  Tutorial: Relay Events to Systems Manager Run Command .................................... 135
  Tutorial: Log EC2 Instance States ............................................................................ 136
    Step 1: Create an AWS Lambda Function .............................................................. 136
    Step 2: Create a Rule .............................................................................................. 137
    Step 3: Test the Rule .............................................................................................. 137
  Tutorial: Log Auto Scaling Group States .................................................................. 138
    Step 1: Create an AWS Lambda Function .............................................................. 138
    Step 2: Create a Rule .............................................................................................. 139
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 services like AWS Lambda or Kinesis, to an HTTP invocation endpoint using an API destination, or to the event bus in another account. To define how events are processed, you set up rules in EventBridge to define a filter pattern to match incoming events. When EventBridge receives an event with an event pattern that matches the event pattern you defined in a rule, the event is sent to the target or targets you define for the rule. To learn more about available targets, see Amazon EventBridge targets (p. 43).

EventBridge was formerly called Amazon CloudWatch Events. The default event bus and any rules you created in CloudWatch Events are also displayed in the EventBridge console. EventBridge uses the same CloudWatch Events API, so all of your existing CloudWatch Events API usage remains the same. New features added to EventBridge are not added to CloudWatch Events.

Topics
- Related AWS Services (p. 1)
- Amazon EventBridge terms and concepts (p. 3)

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. 195).

- **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 non-compliant 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 Amazon EventBridge Identity and Access Management (p. 160).

- **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
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.
Amazon EventBridge terms and 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 Events from AWS services (p. 74).

- **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.
Amazon EventBridge setup and prereqs

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 EventBridge 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.

Topics
- Sign Up for Amazon Web Services (AWS) (p. 4)
- Sign in to the Amazon EventBridge Console (p. 4)
- Account Credentials (p. 4)
- Set Up the Command Line Interface (p. 5)
- Regional Endpoints (p. 5)

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:
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.
Get started with Amazon EventBridge

Use EventBridge rules to determine which events you want to trigger a target. For tutorials on specific scenarios and specific targets, see Amazon EventBridge tutorials (p. 135).

Topics
- Create a rule in Amazon EventBridge (p. 6)

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

Create a rule in Amazon EventBridge

When you create a rule for events, you specify a target action to take when EventBridge receives an event that matches the rule. When an event matches the rule, EventBridge sends the event to the specified target and triggers the action defined in the rule.

When an AWS service in your account emits an event, it always goes to the default event bus for your account. To write a rule that triggers on events from AWS services in your account, you must associate it with the default event bus.

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

To create a rule for an AWS service
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. The fields displayed vary by the service you choose. Enter information specific to this target type as 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. For Retry policy and dead-letter queue, under Retry policy:
   a. For Maximum age of event, enter a value between 1 minute (00:01) and 24 hours (24:00).
   b. For Retry attempts, enter a number between 0 and 185.
15. For Dead-letter queue, choose whether to use a standard Amazon SQS queue as a dead-letter queue. EventBridge sends events that match this rule to the dead-letter queue if they are not successfully delivered to the target. Do one of the following:
   
   • Choose None to not use a dead-letter queue.
   • Choose Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue and then select the queue to use from the drop-down list.
   • Choose Select an Amazon SQS queue in an other AWS account as a dead-letter queue and then enter the ARN of the queue to use. You must attach a resource-based policy to the queue that grants EventBridge permission to send messages to it. To learn more, see Granting permissions to the dead-letter queue (p. 36).
16. Optionally, choose Add target to add another target for this rule.
17. (Optional) Enter one or more tags for the rule. For more information, see Amazon EventBridge SaaS tagging (p. 216).
18. Choose Create.
Amazon EventBridge event buses

An event bus receives events. Rules associated with the event bus evaluate events as they are received to check whether they match a rule. When you create a rule, you associate it with a specific event bus, and the rule criteria are used to evaluate only events received by that event bus. You can configure a resource-based policy for an event bus to manage permissions for the bus. This lets you specify which events to allow, and which entities have permission to create or modify rules or targets for an event. For example, you can use a policy on an event bus to allow or deny events from specific sources, such as a rule or an event bus in a different account or Region. This lets you aggregate all events from your application or organization in a single account and Region so you can easily monitor and respond to event data for your entire system.

You can configure up to 100 EventBridge rules for each event bus. If you have more than 100 rules in your environment, you can create additional custom event buses in your account and associate an additional 100 rules with each event bus. This lets you customize how events are received in your account, such as creating event buses with different permissions for different services.

You can create a custom event bus to receive events from your custom applications. You can also create partner event buses to receive events from SaaS partner applications. The most common ways to configure an event bus include:

- EventBridge creates a default event bus in each account to receive events from AWS services.
- A custom event bus to send events to or receive events from a different AWS account.
- A custom event bus to send events to or receive events from a different Region to aggregate events in a single location.
- A partner event bus to receive events from a SaaS partner. To learn more, see Receive events from a SaaS partner with Amazon EventBridge (p. 128).

Topics

- Permissions for EventBridge event buses (p. 9)
- Create Amazon EventBridge event buses (p. 15)
Permissions for EventBridge event buses

The default event bus in your account restricts permissions to only allow events from the same account. You can grant additional permissions to the default bus and any additional buses you create by attaching a resource-based policy to the bus. With a resource-based policy, you can reference an event bus in another account when granting permissions to allow PutEvents, PutRule, and PutTargets API calls. You can also use conditions in the policy to grant permissions to an AWS Organization, apply Tags, or filter events to only those from a specific rule or account. You can set a resource-based policy for an event bus either when you create it or after it is created.

EventBridge APIs that accept an event bus Name parameter (including PutRule, PutTargets, DeleteRule, RemoveTargets, DisableRule, and EnableRule) also support providing the event bus ARN. This allows you to target cross-account or cross-Region event buses through the APIs. For example, you can call PutRule to create a rule on an event bus in a different account, without needing to assume a role.

You can use the example policies in this topic to enable sending events to a different account or Region. To learn how to create a rule to send events to a different AWS account or Region, see Sending and receiving Amazon EventBridge events between AWS accounts and Regions (p. 55).

Topics
- Managing event bus permissions (p. 9)
- Example policy: Send events to the default bus in a different account (p. 11)
- Example policy: Send events to a custom bus in a different account (p. 11)
- Example policy: Send events to the same account and restrict updates (p. 12)
- Example policy: Send events only from a specific rule to the bus in a different Region (p. 12)
- Example policy: Send events only from a specific Region to a different Region (p. 13)
- Example policy: Deny sending events from specific Regions (p. 14)

Managing event bus permissions

Use the following procedure to modify the permissions for an existing event bus. To learn how to use AWS CloudFormation to create an event bus policy, see AWS::Events::EventBusPolicy.

To manage permissions for an existing event bus

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. Choose Event buses in the left navigation pane.
3. Choose the name of the bus to manage permissions for.
   - If a resource policy is attached to the event bus, the policy is displayed.
4. Choose Manage permissions, then do one of the following:
   - Enter the policy that includes the permissions to grant for the event bus. You can paste in a policy from another source, or enter the JSON for the policy.
   - To use a template for the policy, choose Load template. Modify the policy as appropriate for your environment, including adding additional actions that you authorize the principal in the policy to use.
5. Choose Update.

The template provides some example policy statements that you can customize for your account and environment. The template is not a valid policy. If you are comfortable making policies, you can modify
the template as appropriate for your use case. If not, you can copy one of the example policies and customize it.

The template loads policies that include an example of how to grant permissions to an account to use the PutEvents action, how to grant permissions to an AWS Organizations organization, and how to grant permissions to the account to manage rules in the account. You can customize the template for your specific account, and then delete the other sections from the template. More example policies are included later in this topic.

If you try to update the permissions for the bus and the policy contains an error, an error message indicates the specific issue in the policy.

The policy template that loads:

```json
### Choose which sections to include in the policy to match your use case. ###
### Be sure to remove all lines that start with ###, including the ### at the end of the line. ###

### The policy must include the following: ###
{
    "Version": "2012-10-17",
    "Statement": [
        ### To grant permissions for an account to use the PutEvents action, include the following, otherwise delete this section: ###
        {
            "Sid": "allow_account_to_put_events",
            "Effect": "Allow",
            "Principal": {
                "AWS": "<ACCOUNT_ID>"
            },
            "Action": "events:PutEvents",
            "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/default"
        },

        ### Include the following section to grant permissions to all members of your AWS Organizations to use the PutEvents action ###
        {
            "Sid": "allow_all_accounts_from_organization_to_put_events",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "events:PutEvents",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "o-yourOrgID"
                }
            }
        },

        ### Include the following section to grant permissions to the account to manage the rules created in the account ###
        {
            "Sid": "allow_account_to_manage_rules_they_created",
            "Version": "2012-10-17",
            "Effect": "Allow",
            "Principal": {
                "AWS": "<ACCOUNT_ID>"
            },
            "Action": "events:DeleteRule",
        }
    ]
}
```
Example policy: Send events to the default bus in a different account

Here is an example policy that grants permission for the AWS account 111112222333 to use all API operations against the default event bus in the AWS account 123456789012:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "sid1",
      "Effect": "Allow",
      "Principal": {"AWS":"arn:aws:iam::111112222333:root"},
      "Action": "events:*",
      "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/default"
    }
  ]
}
```

Example policy: Send events to a custom bus in a different account

The following example demonstrates a policy that grants AWS account 111112222333 permissions to publish events to the central-event-bus in account 123456789012, but only for events with a source value set to `com.exampleCorp.webStore` and a detail-type set to `newOrderCreated`.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "WebStoreCrossAccountPublish",
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Condition": {
        "StringEqualsIfExists": {
          "events:creatorAccount": "<ACCOUNT_ID>"
        }
      }
    }
  ]
}
```
Example policy: Send events to the same account and restrict updates

This example policy demonstrates how to grant permissions to allow AWS account 123412341234 to create, delete, update, disable and enable rules, and add or remove targets. It limits these rules that match against events with a source of com.exampleCorp.webStore, and it uses the "events:creatorAccount": "${aws:PrincipalAccount}" to ensure that only account 123412341234 can modify these rules and targets once they have been created.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "InvoiceProcessingRuleCreation",  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::123412341234:root"  
      },  
      "Action": [  
        "events:PutRule",  
        "events:DeleteRule",  
        "events:DescribeRule",  
        "events:DisableRule",  
        "events:EnableRule",  
        "events:PutTargets",  
        "events:RemoveTargets"  
      ],  
      "Condition": {  
        "StringEqualsIfExists": {  
          "events:creatorAccount": "${aws:PrincipalAccount}"  
        }  
      }  
    }  
  ]  
}
```

Example policy: Send events only from a specific rule to the bus in a different Region

The following example policy shows how to grant permission to allow the AWS account 111112222333 to send events that match a rule named SendToUSE1AnotherAccount in the Middle East (Bahrain) and US West (Oregon) Regions to an event bus named CrossRegionBus in the US East (N. Virginia) in AWS.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "InvoiceProcessingRuleCreation",  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::123412341234:root"  
      },  
      "Action": [  
        "events:PutRule",  
        "events:DeleteRule",  
        "events:DescribeRule",  
        "events:DisableRule",  
        "events:EnableRule",  
        "events:PutTargets",  
        "events:RemoveTargets"  
      ],  
      "Condition": {  
        "StringEqualsIfExists": {  
          "events:creatorAccount": "${aws:PrincipalAccount}"  
        }  
      }  
    }  
  ]  
}
```
account 123456789012. The example policy is added to the event bus named CrossRegionBus in account 123456789012. The policy allows events only if they match a rule specified for the event bus in AWS account 111112222333. The Condition statement restricts events to only events that match the rules with the specified rule ARN.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "allow_specific_rules_as_cross_region_source",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111112222333:root"
      },
      "Action": "events:PutEvents",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:events:us-west-2:111112222333:rule/CrossRegionBus/SendToUSE1AnotherAccount",
            "arn:aws:events:me-south-1:111112222333:rule/CrossRegionBus/SendToUSE1AnotherAccount"
          ]
        }
      }
    }
  ]
}
```

**Example policy: Send events only from a specific Region to a different Region**

This example shows how to grant permission to send all events from AWS account 111112222333 that are generated in the Middle East (Bahrain) and US West (Oregon) Regions to the event bus named CrossRegionBus in AWS account 123456789012 in the US East (N. Virginia) Region. Events that are generated in AWS account 111112222333 in any other Region are not allowed.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "allow_cross_region_events_from_us-west-2_and_me-south-1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111112222333:root"
      },
      "Action": "events:PutEvents",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:events:us-west-2::*:*",
            "arn:aws:events:me-south-1::*:*"
          ]
        }
      }
    }
  ]
}
```
Example policy: Deny sending events from specific Regions

The following example policy attached to an event bus named CrossRegionBus in AWS account 123456789012 shows how to allow events from the AWS account 111112222333, but deny events if they are generated in the US West (Oregon) Region.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "1_allow_any_events_from_account_111112222333",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111112222333:root"
            },
            "Action": "events:PutEvents",
            "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/CrossRegionBus"
        },
        {
            "Sid": "2_deny-all-cross-region-us-west-2-events",
            "Effect": "Deny",
            "Principal": {
                "AWS": "*
            },
            "Action": "events:PutEvents",
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": [
                        "arn:aws:events:us-west-2:*:*
                    ]
                }
            }
        }
    ]
}
```
Create Amazon EventBridge event buses

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. When you create an event bus, you can attach a resource-based policy to grant permissions to other accounts to enable cross-account targets. This lets you authorized other accounts to send events to the event bus in the current account.

To create a custom event bus

1. Open the Amazon EventBridge console at 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. Do one of the following:
   - Enter the policy that includes the permissions to grant for the event bus. You can paste in a policy from another source, or enter the JSON for the policy. You can use one of the example policies and modify it for your environment.
   - To use a template for the policy, choose Load template. Modify the policy as appropriate for your environment, including adding additional actions that you authorize the principal in the policy to use.
6. Choose Create.
Amazon EventBridge events

Topics

• Amazon EventBridge event patterns (p. 18)
• Adding Amazon EventBridge events with PutEvents (p. 28)

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.

The following is an example event in Amazon EventBridge.

```
{  
  "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 Events from AWS services (p. 74).

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. 85).

**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 Events from AWS services (p. 74).
Amazon EventBridge 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.

```
{
  "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 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.

```
{
  "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.

```
{
  "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.
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.

```
{  
  "version": "0",
  "id": "3e3c153a-8339-4e30-8c35-687ebebf85fe",
  "detail-type": "EC2 Instance Launch Successful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2015-11-11T21:31:47Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "eventVersion": "",
    "responseElements": null
  }
}
```

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 Classic event (truncated for brevity).

```
{  
  "version": "0",
  "id": "5e355723-fca9-4de3-9fd7-154c289d6b59",
  "detail-type": "Macie Alert",
  "source": "aws.macie",
  "account": "123456789012",
  "time": "2017-04-24T22:28:49Z",
  "region": "us-east-1",
  "resources": [{
  "detail": {
    "notification-type": "ALERT_CREATED",
    "name": "Scanning bucket policies",
    "tags": [
      "Custom_Alert",
      "Insider"
    ],
    "url": "https://lb00.us-east-1.macie.aws.amazon.com/111122223333/posts/alert_id",
```
If you want to match anything that has a risk score of 80 and a trigger risk of 8, do the following.

```json
{
    "source": ["aws.macie"],
    "detail-type": ["Macie Alert"],
    "detail": {
        "risk-score": [80],
        "trigger": {
            "risk": [8]
        }
    }
}
```

**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.

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.

```json
  ]
}
```

```json
{  "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.

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

```json
{  "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 Amazon 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:

```json
{
    "version": "0",
    "id": "3e3c153a-8339-4e30-8c35-687ebef853fe",
    "detail-type": "EC2 Instance Launch Successful",
    "source": "aws.autoscaling",
    "account": "123456789012",
    "time": "2015-11-11T21:31:47Z",
    "region": "us-east-1",
    "resources": [
    ],
    "detail": {
        "eventVersion": ",",
        "responseElements": null
    }
}
```

To match events where the value of `eventVersion` is an empty string, use the following pattern, which would match the event example.

```json
{
    "detail": {
        "eventVersion": ","
    }
}
```

To match events where the value of `responseElements` is null, use the following pattern, which would match the event example.

```json
{
    "detail": {
        "responseElements": [null]
    }
}
```

Null values and empty strings are not interchangeable in pattern matching. A pattern that is written to detect empty strings will not catch values of `null`.
Arrays in Amazon EventBridge 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:ec2:us-east-1:123456789012:instance/i-b188560f"
]
```
Content-based filtering in Amazon EventBridge 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. 24)
- Anything-but Matching (p. 24)
- Numeric Matching (p. 26)
- IP Address Matching (p. 26)
- Exists Matching (p. 26)
- Complex Example with Multiple Matching (p. 27)

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".

```json
{
    "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.

```json
{
    "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.
The following shows anything-but matching with a list of strings.

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

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

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

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."

```json
{
  "source": [ { "anything-but": { "prefix": "aws." } } ]
}
```

## Numeric Matching

The following shows numeric matching for an event pattern.

```json
{
  "detail": {
    "c-count": [ { "numeric": [ ">", 0, "<=" , 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.

```json
{
  "detail": {
    "sourceIPAddress": [ { "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.

```json
{
  "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.

```json
{
  "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.

```json
{
   "detail-type": [ "EC2 Instance State-change Notification" ],
   "resources": [ "arn:aws:ec2:us-east-1:123456789012:instance/i-02ebd4584a2ebd341" ],
   "detail": {
      "state": [ "initializing", "running" ]
      "c-count": {
         "c1": 100
      }
   }
}
```

**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 ] } ]
   }
}
```
Adding Amazon EventBridge events with PutEvents

Topics

- Handling Failures When Using PutEvents (p. 29)
- Sending Events Using the AWS CLI (p. 30)
- Calculating Amazon EventBridge PutEvents event entry size (p. 32)

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 SaaS quotas (p. 211). 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 lt;
PutEventsRequestEntry > requestEntries = new ArrayList & lt;
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")
```
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

```json
{
   "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

```java
PutEventsRequestEntry requestEntry = new PutEventsRequestEntry()
```
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:

```
aws events put-events \
  --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:

```
aws events put-events --entries file:///entries.json
```
Calculating Amazon EventBridge PutEvents event entry size

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 Amazon EventBridge events (p. 16).

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;
}
```
Amazon EventBridge rules

Topics

- Create Amazon EventBridge rule that triggers on a schedule (p. 34)
- Event retry policy and using dead-letter queues (p. 35)
- Disable or delete Amazon EventBridge rules (p. 37)
- Managed Amazon EventBridge rules (p. 38)
- Schedule expressions for Amazon EventBridge rules (p. 39)

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.
Create Amazon 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.
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 Amazon EventBridge rules (p. 39).
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. For Retry policy and dead-letter queue, under Retry policy:
    a. For Maximum age of event, enter a value between 1 minute (00:01) and 24 hours (24:00).
    b. For Retry attempts, enter a number between 0 and 185.
12. For Dead-letter queue, choose whether to use a standard Amazon SQS queue as a dead-letter queue. EventBridge sends events that match this rule to the dead-letter queue if they are not successfully delivered to the target. Do one of the following:
    - Choose None to not use a dead-letter queue.
    - Choose Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue and then select the queue to use from the drop-down list.
    - Choose Select an Amazon SQS queue in another AWS account as a dead-letter queue and then enter the ARN of the queue to use. You must attach a resource-based policy to the queue that grants EventBridge permission to send messages to it. To learn more, see Granting permissions to the dead-letter queue (p. 36).
13. (Optional) Choose Add target to add another target for this rule.
14. (Optional) Enter one or more tags for the rule. For more information, see Amazon EventBridge SaaS tagging (p. 216).
15. Choose Create.
Event retry policy and using dead-letter queues

Sometimes an event is not successfully delivered to the target specified in a rule. This can happen when, for example, the target resource is unavailable, when EventBridge lacks permissions to access the target resource, or due to network conditions. When an event is not successfully delivered to a target, EventBridge retries sending the event. The length of time and number of retry attempts is determined by the Retry policy settings for the target. By default, EventBridge retries sending the event for 24 hours and up to 185 times with an exponential back off and jitter (randomized delay). If an event is not delivered before all retry attempts are exhausted, the event is dropped and EventBridge does not continue to process it. To avoid losing events after they fail to be delivered to a target, you can configure a dead-letter queue (DLQ) and send all failed events to it for processing later.

EventBridge DLQs are standard Amazon SQS queues that EventBridge uses to store events that could not successfully be delivered to a target. When you create a rule and add a target, you can choose whether or not to use a DLQ. When you configure a DLQ, you can retain any events that were not successfully delivered. This lets you resolve the issue that resulted in the failed invocation and then process the events at a later time.

Not all event errors are handled in the same way. Some events are dropped or sent to a DLQ without any retry attempts. For example, for errors that result from missing permissions to a target, or a target resource that no longer exists, all retry attempts will fail until an action is taken to resolve the underlying issue. EventBridge sends these events directly to the DLQ, if configured, upon delivery failure.

When an event fails to successfully be delivered to a target, EventBridge publishes an event to Amazon CloudWatch metrics, including indicating that a target invocation failed. If you use a DLQ, additional metrics are sent to CloudWatch including InvocationsSentToDLQ and InvocationsFailedToBeSentToDLQ. To learn more about EventBridge metrics, see Amazon EventBridge monitoring (p. 202).

Considerations for using a dead-letter queue

Consider the following when configuring a DLQ for EventBridge.

- Only standard queues are supported. You cannot use a FIFO queue for a DLQ in EventBridge.
- EventBridge includes event metadata and message attributes in the message, including: the Error Code, Error Message, the Exhausted Retry Condition, Rule ARN, Retry Attempts, and the Target ARN. You can use these values to identify an event and the cause of the failure.
- Permissions for DLQs in the same account:
  - If you add a target to a rule using the console, and you choose an Amazon SQS queue in the same account, a resource-based policy that grants EventBridge access to the queue is attached to the queue for you.
  - If you use the PutTargets operation of the EventBridge API to add or update a target for a rule, and you choose an Amazon SQS queue in the same account, you must manually grant permissions to the queue selected. To learn more, see Granting permissions to the dead-letter queue (p. 36).
- Permissions for using Amazon SQS queues from a different AWS account.
  - If you create a rule from the console, queues from other accounts are not displayed for you to select. You must provide the ARN for the queue in the other account, and then manually attach a resource-based policy to grant permission to the queue. To learn more, see Granting permissions to the dead-letter queue (p. 36).
  - If you create a rule using the API, you must manually attach a resource-based policy to the SQS queues in another account that is used as the dead-letter queue. To learn more, see Granting permissions to the dead-letter queue (p. 36).
  - The Amazon SQS queue you use must be in the same Region in which you create the rule.
Granting permissions to the dead-letter queue

When you configure a DLQ for a target of a rule, EventBridge sends the events with failed invocations to the Amazon SQS queue selected. To successfully deliver events to the queue, EventBridge must have permission to do so. When you configure a target for a rule and select a DLQ using the EventBridge console, the permissions are automatically added. If you create a rule using the API, or use a queue that is in a different AWS account, you must manually create a resource-based policy that grants the required permissions and then attach it to the queue.

The following resource-based policy demonstrates how to grant the required permissions for EventBridge to send event messages to an Amazon SQS queue. The policy example grants the EventBridge service permissions to use the `SendMessage` operation to send messages to a queue named "MyEventDLQ". The queue must be in the us-west-2 Region in AWS account 123456789012. The `Condition` statement allows only requests that come from a rule named "MyTestRule" that is created in the us-west-2 Region in the AWS account 123456789012.

```json
{
    "Sid": "Dead-letter queue permissions",
    "Effect": "Allow",
    "Principal": {
        "Service": "events.amazonaws.com"
    },
    "Action": "sqs:SendMessage",
    "Condition": {
        " ArnEquals": {
            " aws:SourceArn": "arn:aws:events:us-west-2:123456789012:rule/MyTestRule"
        }
    }
}
```

To attach the policy to the queue, use the Amazon SQS console, open the queue, then choose the Access policy and edit the policy. You can also use the AWS CLI, to learn more see Amazon SQS Permissions (p. 176).

How to resend events from a dead-letter queue

You can move messages out of a DLQ in two ways:

- Avoid writing Amazon SQS consumer logic – Set your DLQ as an event source to the Lambda function to drain your DLQ.
- Write Amazon SQS consumer logic – Use the Amazon SQS API, AWS SDK, or AWS CLI to write custom consumer logic for polling, processing, and deleting the messages in the DLQ.
Disable or delete Amazon EventBridge rules

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 Managed Amazon EventBridge rules (p. 38).
   b. To temporarily disable a rule, select the button next to the rule and choose Disable, Disable.
      You can't disable a managed rule.
Managed Amazon EventBridge 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.
Schedule expressions for Amazon EventBridge 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. 39)
- Rate Expressions (p. 41)

## 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>, - */</td>
</tr>
<tr>
<td>Hours</td>
<td>0-23</td>
<td>, - */</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.

Note
If you use a '#' character, you can define only one expression in the day-of-week field. For example, "3#1, 6#3" is not valid because it is interpreted as two expressions.

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>
</tbody>
</table>
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.

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.

```bash
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.

```bash
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.

```bash
```
Rate Expressions

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
```
Amazon EventBridge targets

A target is a resource or endpoint that EventBridge sends an event to when the event matches the filter pattern defined for a rule. You can define up to 5 targets for each rule. EventBridge supports the following targets:

- API destination (p. 46)
- Amazon API Gateway REST API endpoints
- API Gateway
- AWS Batch job queue
- CloudWatch Logs group
- CodeBuild project
- CodePipeline
- Amazon EC2 CreateSnapshot API call
- Amazon EC2 RebootInstances API call
- Amazon EC2 StopInstances API call
- Amazon EC2 TerminateInstances API call
- Amazon ECS tasks
- Event bus in a different account or Region (p. 55)
- Firehose delivery stream (Kinesis Data Firehose)
- Inspector assessment template (Amazon Inspector)
- Kinesis stream (Kinesis Data Streams)
- AWS Lambda function
- Amazon Redshift clusters (Data API statement execution)
- Amazon SNS topic
- Amazon SQS queues (includes FIFO queues
- SSM Automation
- SSM OpsItem
- SSM Run Command
- Step Functions state machines

Topics

- Target-specific Parameters (p. 43)
- Permissions (p. 44)
- Configure targets (p. 45)
- Transforming Amazon EventBridge target input (p. 58)

Target-specific Parameters

Some target types accept parameters. For example, if you are using a Kinesis data stream target, you can optionally specify which shard the event goes to by using the KinesisParameters argument. To invoke a command on multiple EC2 instances with one rule, you can use the RunCommandParameters field. To pass data to API Gateway endpoints, you can use the HttpParameters field, including optional dynamic JSON path syntax (see the section called “API Gateway” (p. 54)).
Permissions

In order to make API calls against the resources that you own, EventBridge needs appropriate permissions. For AWS Lambda and Amazon SNS resources, EventBridge relies on resource-based policies. For EC2 instances, Kinesis Data Streams, and AWS Step Functions state machines, EventBridge relies on IAM roles that you specify in the RoleARN argument in PutTargets. API Gateway REST endpoints with configured IAM authorization can be invoked via IAM roles, but the role is optional if no Authorization is configured. For more information, see Amazon EventBridge Identity and Access Management (p. 160).

If another AWS account is in the same Region and has granted you permission (using PutPermission), you can send events to that account. Set that account's event bus as a target of the rules in your account. To send the matched events to the other account, specify that account's event bus as the Arn value when you run PutTargets. If your account sends events to another account, your account is charged for each sent event. Each event sent to another account is charged as a custom event. The account receiving the event is not charged. For more information, see Amazon CloudWatch Pricing.

Note

Input, InputPath, and InputTransformer are not available with PutTarget if the target is an event bus of a different AWS account.

If you are setting the event bus of another account as the target, and that account granted permission to your account through an organization instead of directly by the account ID, you must specify a RoleARN with proper permissions in the Target structure. For more information, see Sending and receiving Amazon EventBridge events between AWS accounts and Regions (p. 55). For more information about enabling cross-account events, see PutPermission in the Amazon EventBridge API Reference.

Input, InputPath, and InputTransformer are mutually exclusive and optional parameters of a target. When a rule is triggered due to a matched event, the behavior is as follows:

- If none of the following arguments are specified for a target, the entire event is passed to the target in JSON format. However, if the target is Amazon EC2 Run command or Amazon ECS task, nothing from the event is passed to the target.
- If Input is specified in the form of valid JSON, the matched event is overridden with this constant.
- If InputPath is specified in the form of JSONPath (for example, $.detail), only the part of the event specified in the path is passed to the target (for example, only the detail part of the event is passed).
- If InputTransformer is specified, one or more specified JSONPaths are extracted from the event and used as values in a template that you specify as the input to the target.

When you specify InputPath or InputTransformer, you must use JSON dot notation, not bracket notation.

When you add targets to a rule and the associated rule triggers soon after, new or updated targets might not be immediately invoked. Allow a short period of time for changes to take effect.

This action can partially fail if too many requests are made at the same time. If that happens, FailedEntryCount is non-zero in the response and each entry in FailedEntries provides the ID of the failed target and the error code.
Configure targets

Learn how to configure settings for EventBridge targets.

Targets:
- API destinations (p. 46)
- Amazon EventBridge targets for Amazon API Gateway (p. 54)
- Sending and receiving Amazon EventBridge events between AWS accounts and Regions (p. 55)
API destinations

EventBridge API destinations are HTTP endpoints that you can invoke as the target of a rule similar to invoking an AWS service or resource as a target. This lets you route events between AWS services, integrated SaaS applications, and your own applications outside of AWS using REST API calls. When you specify an API destination as the target of a rule, EventBridge invokes the HTTP endpoint for any event that matches the event pattern specified in the rule and delivers the event payload with the request. EventBridge supports using any HTTP method (except CONNECT and TRACE) for the request, such as PUT or POST. You can also use input transformers to customize the event payload to the parameters of a specific HTTP endpoint parameters. To learn more, see Transforming Amazon EventBridge target input (p. 58).

Important
EventBridge requests to the API destination endpoint must have a maximum client execution timeout of 5 seconds. If the target invocation endpoint takes longer than 5 seconds to respond, EventBridge times out the request. Timed out requests are retried up to the maximum configured on your optional retry policy (default 24 hours, 185 times). After the maximum number of retries, events are sent to your dead-letter queue (p. 35) if you configured one. If not, the event is dropped.

In this topic:
- Connections for API destinations (p. 46)
- Create an API destination (p. 48)
- Service-linked role for API destinations (p. 48)
- Headers included in requests to API destinations (p. 49)
- API destination error codes (p. 49)
- How invocation rate affects event delivery (p. 49)
- API destination partners (p. 50)

Connections for API destinations

When you create an API destination, you specify a connection to use for it. Each connection includes the details about the authorization type and parameters to use to authorize with the API destination endpoint. You can choose an existing connection from your account, or create a connection when you create an API destination. EventBridge supports Basic, OAuth, and API Key authorization. For Basic and API Key authorization, EventBridge populates the requisite authorization headers for your invocation. For OAuth authorization, EventBridge also exchanges your client ID and secret for an access token, and manages it securely. When you create a connection you can also include additional header, body, and query parameters required for authorization with an endpoint. You can use the same connection for more than one API destination if the authorization for the endpoints is the same. When you create a connection and add authorization parameters for the selected type, the data is used to create a secret in AWS Secrets Manager. The cost of storing the Secrets Manager secret is included with the charge for using an API destination.

Note
To successfully create or update a connection, you must use an account that has permissions to use Secrets Manager. The required permissions are included in the AmazonEventBridgeFullAccess policy (p. 167). The same permissions are granted to the service-linked role (p. 48) created in your account for the connection.

To create a connection

1. Log in to AWS using an account that has permissions to manage EventBridge and open the EventBridge console.
2. In the left navigation pane, choose **API destinations**.
3. Scroll down to the **API destinations** table, then choose the **Connections** tab.
4. Choose **Create connection**.
5. On the **Create connection** page, enter a **Connection name** for the connection.
6. Enter a **Description** for the connection.
7. For **Authorization type**, select the type of authorization to use to authorize connections to the HTTP endpoint specified for the API destination that uses this connection. Do one of the following:
   - Choose **Basic (Username/Password)**, then enter the **Username** and **Password** to use to authorize with the HTTP endpoint.
   - Choose **OAuth Client Credentials**, then enter the **Authorization endpoint**, **HTTP method**, **Client ID**, and **Client secret** to use to authorize with the endpoint.

Under **OAuth Http Parameters**, add any additional parameters to include for authorization with the authorization endpoint. Select a **Parameter** from the drop-down list, then enter a **Key** and **Value**. To include an additional parameter, choose **Add parameter**.

Under **Invocation Http Parameters**, you can include any additional parameters to include in the authorization request. To add a parameter, select a **Parameter** from the drop-down list, then enter a **Key** and **Value**. To include an additional parameter, choose **Add parameter**.

   - Choose **API key**, and then enter the **API key name** and associated **Value** to use for API Key authorization.

   Under **Invocation Http Parameters**, you can include any additional parameters to include in the authorization request. To add a parameter, select a **Parameter** from the drop-down list, then enter a **Key** and **Value**. To include an additional parameter, choose **Add parameter**.

8. Choose **Create**.

**To edit a connection**

1. Open the **API destinations** page then choose **Connections**.
2. Choose the connection to edit from the **Connections** table.
3. On the **Connection details** page, choose **Edit**.
4. Update the values for the connection and then choose **Update**.

**De-authorizing connections**

When you de-authorize a connection, it removes all authorization parameters. This lets you remove the secret from the connection so you can reuse it without having to create a new connection. You must update any API destinations that use the de-authorized connection to use a different connection to successfully send requests to the API destination endpoint.

**To de-authorize a connection**

1. Choose the connection from the Connections table.
2. On the **Connection details** page, choose **De-authorize**.
3. On the **Deauthorize connection?** dialog, enter the name of the connection and then choose **De-authorize**.

The status of the connection changes to **De-authorizing** until the process is complete and the status changes to **De-authorized**. You can then edit the connection to add new authorization parameters.
Create an API destination

Each API destination requires a connection. The connection specifies the authorization type and credentials to use to authorize with the API destination endpoint. You can choose an existing connection, or create a connection at the same time that you create the API destination.

To create an API destination

1. Log in to AWS using an account that has permissions to manage EventBridge and open the EventBridge console.
2. In the left navigation pane, choose API destinations.
3. Scroll down to the API destinations table, then choose Create API destination.
4. On the Create API destination page, enter a Name for the API destination. You can use up to 64 upper or lower-case letters, numbers, dot (.), dash (-), or underscore (_) characters. The name must be unique to your account in the current Region.
5. Enter a Description for the API destination.
6. Enter a API destination endpoint for the API destination. The API destination endpoint is an HTTP invocation endpoint target for events. The authorization information you include in the connection used for this API destination is used to authorize against this endpoint. The URL must use HTTPS.
7. Enter the HTTP method to use to connect to the API destination endpoint.
8. For Invocation rate limit per second field, optionally enter the maximum number of invocations per second to send to the API destination endpoint.
   The rate limit you set may impact event delivery. To learn more, see How invocation rate affects event delivery (p. 49).
9. For Connection, do one of the following:
   • Choose Use an existing connection, then select the connection to use for this API destination.
   • Choose Create a new connection, then enter the details for the connection to create. To learn more, see Connections (p. 46).
10. Choose Create.

After you create an API destination you can select it as the target of a rule.

Service-linked role for API destinations

When you create a connection for an API destination, a service-linked role named AWSServiceRoleForAmazonEventBridgeApiDestinations is added to your account. The service-linked role is used to create and store a secret in Secrets Manager. To grant the necessary permissions to the service-linked role, the AmazonEventBridgeApiDestinationsServiceRolePolicy policy is attached to the role. The policy limits the permissions granted to only those necessary for the role to interact with the secret for the connection. No other permissions are included, and the role can interact only with the connections in your account to manage the secret.

The AmazonEventBridgeApiDestinationsServiceRolePolicy policy:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "secretsmanager:CreateSecret",
            "secretsmanager:UpdateSecret",
            ...other actions...
         ]
      }
   ]
}
```
API destinations

```json
{
    "secretsmanager:DescribeSecret",
    "secretsmanager:DeleteSecret",
    "secretsmanager:GetSecretValue",
    "secretsmanager:PutSecretValue"
}],
"Resource": "arn:aws:secretsmanager:*:*:secret:events!connection/*"
}
}
```

To learn more about service-linked roles, see Using service-linked roles in the IAM documentation.

Headers included in requests to API destinations

In addition to the authorization headers defined for the connection used for an API destination, EventBridge includes the following headers in each request.

<table>
<thead>
<tr>
<th>Header key</th>
<th>Header value</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-Agent</td>
<td>Amazon/EventBridge/ApiDestinations</td>
</tr>
<tr>
<td>Content-Type</td>
<td>application/json; charset=utf-8</td>
</tr>
<tr>
<td>Range</td>
<td>bytes=0-1048575</td>
</tr>
<tr>
<td>Accept-Encoding</td>
<td>gzip, deflate</td>
</tr>
<tr>
<td>Connection</td>
<td>close</td>
</tr>
<tr>
<td>Content-Length</td>
<td>An entity header that indicates the size of the entity-body, in bytes, sent to the recipient.</td>
</tr>
<tr>
<td>Host</td>
<td>A request header that specifies the host and port number of the server to which the request is being sent.</td>
</tr>
</tbody>
</table>

API destination error codes

When an error occurs with event delivery to an API destination, events are retried as follows:

Events associated with error codes 429 and 5xx are retried.

Events associated with error codes 1xx, 2xx, 3xx, and 4xx (excluding 429) are not retried.

EventBridge API destinations respects the standard HTTP response header Retry-After, which indicates how long the user agent should wait before making a follow-up request. EventBridge chooses the more conservative value between the defined retry policy and the Retry-After header. If a negative Retry-After value exists EventBridge stops retrying delivery for that event.

How invocation rate affects event delivery

If you set the invocation rate per second to a value much lower than the number of invocations generated, events may not be delivered within the 24 hour retry time for events. For example, if you set the invocation rate to 10 invocations per second, but thousands of events per second are generated, you will quickly have a backlog of events to deliver that exceeds 24 hours. To ensure that no events are lost, set up a dead-letter queue to send events with failed invocations to so you can process the events at a later time. To learn more, see Event retry policy and using dead-letter queues (p. 35).
API destination partners

Use the information provided by the partners in this section to configure an API destination and connection for their service or application.

API destination partners:
- Datadog (p. 50)
- Freshworks (p. 50)
- MongoDB (p. 51)
- New Relic (p. 51)
- Salesforce (p. 52)
- Splunk (p. 53)
- TriggerMesh (p. 53)
- Zendesk (p. 54)

Datadog

API destination invocation endpoint URL:
View a full list of endpoints, see Datadog API Reference.

Supported authorization types:
- API Key

Additional authorization parameters required:
- None

Datadog documentation:
- Authentication

Commonly used API operations:
- POST https://api.datadoghq.com/api/v1/events
- POST https://http-intake.logs.datadoghq.com/v1/input

Additional information:
- Endpoint URLs differ depending on the location of your Datadog organization. View the documentation to retrieve the correct URL for your organization.

Freshworks

API destination invocation endpoint URL:
View endpoints at https://developers.freshworks.com/documentation/

Supported authorization types:
- Basic, API Key

Additional authorization parameters required:
- Not applicable

Freshworks documentation:
- Authentication
Commonly used API operations:
- https://developers.freshdesk.com/api/#create_ticket
- https://developers.freshdesk.com/api/#update_ticket
- https://developer.freshsales.io/api/#create_lead
- https://developer.freshsales.io/api/#update_lead

Additional information:
None

**MongoDB**

API destination invocation endpoint URL:

https://webhooks.mongodb-realm.com/api/client/v2.0/app/Realm_App_ID/service/HTTP_Service_Name/incoming_webhook/Webhook_Name

Supported authorization types:
- API Key

Additional authorization parameters required:
None

MongoDB documentation:
- API Key
- Authentication Providers

Commonly used API operations:
- POST https://webhooks.mongodb-realm.com/api/client/v2.0/app/Realm_App_ID/service/HTTP_Service_Name/incoming_webhook/Webhook_Name

Additional information:
- EventbridgeAtlas demonstrates how Realm Webhooks can be invoked to perform CRUD operations on collections residing in MongoDB Atlas from EventBridge API Endpoints option.

**New Relic**

API destination invocation endpoint URL:

For details, see Our EU and US region data centers.

Events:
- US: https://insights-collector.newrelic.com/v1/accounts/YOUR_NEW_RELIC_ACCOUNT_ID/events
- EU: https://insights-collector.eu01.nr-data.net/v1/accounts/YOUR_NEW_RELIC_ACCOUNT_ID/events

Metrics:
- US: https://metric-api.newrelic.com/metric/v1
- EU: https://metric-api.eu01.nr-data.net/metric/v1

Logs:
**API destinations**

**US:** https://log-api.newrelic.com/log/v1

**EU:** https://log-api.eu.newrelic.com/log/v1

**Traces:**

**US:** https://trace-api.newrelic.com/trace/v1

**EU:** https://trace-api.eu.newrelic.com/trace/v1

**Supported authorization types:**

- API Key

**Additional authorization parameters required:**

- List additional parameters

**New Relic documentation:**

- Metric API
- Event API
- Log API
- Trace API

**Commonly used API operations:**

- Metric API
- Event API
- Log API
- Trace API

**Additional information:**

- Metric API limits
- Event API limits
- Log API limits
- Trace API limits

**Salesforce**

**API destination invocation endpoint URL:**

- View a full list of endpoints in the Salesforce API Reference

**Supported authorization types:**

- OAuth Client Credentials

**Additional authorization parameters required:**

- grant_type
- username
- password

**Salesforce documentation:**

- REST API Developer Guide
Commonly used API operations:

- Working with Object Metadata
- Working with Records

Additional information:

Client ID and Client Secret are provided by Salesforce Connected App

grant_type value is “password”

username and password values are from Salesforce user credentials

Splunk

API destination invocation endpoint URL:

https://SPLUNK_HEC_ENDPOINT:optional_port/services/collector/raw

Supported authorization types:

- Basic, API Key

Additional authorization parameters required:

None

Splunk documentation:

For both authorization types, an HEC token ID is required. To learn more, see Set up and use HTTP Event Collector in Splunk Web.

Commonly used API operations:

POST https://SPLUNK_HEC_ENDPOINT:optional_port/services/collector/raw

Additional information:

API Key – When configuring the endpoint for EventBridge, the API key name is “Authorization” and value is the “Splunk HEC token ID”

Basic (Username/Password) – When configuring the endpoint for EventBridge, the username is “Splunk” and the password is “HEC tokenID”

TriggerMesh

API destination invocation endpoint URL:

Use the information in the Event Source for HTTP topic to formulate the endpoint URL. An endpoint URL includes the event source name and user namespace in the following format:

https://source-name.user-namespace.cloud.triggermesh.io

Include the Basic authorization parameters in the request to the endpoint.

Supported authorization types:

- Basic

Additional authorization parameters required:

None

TriggerMesh documentation:

Event Source for HTTP
Commonly used API operations:
Not applicable
Additional information:
None

Zendesk
API destination invocation endpoint URL:
https://developer.zendesk.com/rest_api/docs/support/tickets
Supported authorization types:
Basic, API Key
Additional authorization parameters required:
None
Zendesk documentation:
Security and Authentication
Commonly used API operations:
POST https://your_zendesk_subdomain/api/v2/tickets
Additional information:
API requests made by EventBridge will count against your Zendesk API limits. You can read more about Zendesk limits for your plan type here (https://developer.zendesk.com/rest_api/docs/support/usage_limits). To better safeguard your account and data, we recommend using an API key rather than basic (username/password) authentication.

Amazon EventBridge targets for Amazon API Gateway
API Gateway enables customers to create, publish, maintain, and monitor REST APIs. Amazon EventBridge supports sending events to an API Gateway REST endpoint. When you specify an API Gateway endpoint as a target, each event sent to the target maps to a request sent to the endpoint.

Important
EventBridge does not support using API Gateway private endpoints as targets. To learn more about private endpoints, see Introducing Amazon API Gateway Private Endpoints.

You can use the API Gateway target for the following use cases:
• Invoke a customer-specified REST API hosted in API Gateway in an event-driven manner based on AWS or third-party events.
• Invoke an endpoint periodically based on a schedule.

The EventBridge JSON event payload is sent as the body of the HTTP request to your endpoint. You can specify the other request attributes in the rule Target's HttpParameters field as follows:
• The PathParameterValues list values that correspond sequentially to any path variables in your endpoint ARN (for example "arn:aws:execute-api:us-east-1:1122345566:myapi/*/POST/pets/*").
• The QueryStringParameters map represents keys/values of query string parameters that are appended to the invoked endpoint.
• The HeaderParameters map enables you to specify HTTP headers to add to the request.

**Note**
For security considerations, the following HTTP header keys are not currently permitted:

- Anything prefixed with `X-Amz` or `X-Amzn`
- Authorization
- Connection
- Content-Encoding
- Content-Length
- Host
- Max-Forwards
- TE
- Transfer-Encoding
- Trailer
- Upgrade
- Via
- WWW-Authenticate
- X-Forwarded-For

**Dynamic Parameters**

Any HttpParameters field value (except HeaderParameters map keys) can contain JSON paths instead of static values (for example `.detail.state`). Such paths are replaced dynamically at runtime with data from the event payload itself at the specified path. The supported syntax for dynamic parameter JSON paths is the same as for Target InputTransformers.

**Note**
The dynamic parameter syntax is only supported for API Gateway target invocations.

**Invocation Retries**

As with all targets, EventBridge will retry some failed invocations. For API Gateway, EventBridge will retry responses sent with a 5xx or 429 HTTP status code for up to 24 hours with exponential backoff and jitter. After that, a FailedInvocations metric is published in Amazon CloudWatch. Other 4xx errors are not retried.

**Timeout**

EventBridge API Gateway requests must have a maximum client execution timeout of 5 seconds. If API Gateway takes longer than 5 seconds to respond, EventBridge will timeout the request and then retry.

**Sending and receiving Amazon EventBridge events between AWS accounts and Regions**

You can configure EventBridge to send and receive events between AWS accounts and Regions. When you configure EventBridge to send or receive events between accounts or Regions, you can specify which AWS accounts can send events to or receive events from the event bus in your account. You can also
allow or deny events from specific Regions, specific rules associated with the event bus, or events from specific sources.

**Note**
If you use AWS Organizations, 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 steps to configure EventBridge to send events to or receive events from a different account or Region include the following:

- **On the receiver account**, edit the permissions on an 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.

- **Important**
  If the sender account inherits permissions to send events because from an AWS Organizations organization, 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 event bus in the receiver account, the role is created 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.

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.

### Grant permissions to allow 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. The permissions for the event bus are granted or denied using a resource-based policy attached to the event bus. In the policy, you can grant permissions to other AWS accounts using the account ID, or to an AWS Organizations organization the organization ID. To learn more about event bus permissions, including example policies, see Permissions for EventBridge event buses (p. 9).

- **Important**
  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 Amazon EventBridge events (p. 16).

### Rules for events between AWS accounts

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 create a rule that sends events to a different AWS account or Region using the console

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules, then on the Rules page choose Create Rule.
3. Enter a name, and optionally a description for the rule.
4. For Event Source, choose Event Pattern.
5. Under Event matching pattern, do one of the following:
   - Choose Pre-defined pattern by service, then select the Service provider and Service name to filter events on. If you choose an AWS service also choose the Event type. You may need to specify additional settings depending on the service and event type you choose.
   - Choose Custom pattern, enter the Event pattern to use for the rule, then choose Save.
6. For Select event bus, choose the event bus that you want to associate with this rule.
7. Under Select targets, for Target choose Event bus in a different account or Region, then enter the ARN of the event bus in a different account or Region to send events to.
8. Do one of the following:
   - Choose Create a new role for this specific resource to have EventBridge create a new IAM role that has permissions to send events to the specified event bus.
   - Choose Use existing role and then select the role to use. If you choose to use an existing role, the role must have permissions to send events to the event bus in a different account or Region. To learn more, see Permissions for EventBridge event buses (p. 9).
9. Optionally, configure a dead-letter queue for the rule. To learn more, see Event retry policy and using dead-letter queues (p. 35).
10. Choose Create to create the rule.
Transforming Amazon EventBridge 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 up to 100 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.
- `aws.events.event.ingestion-time` — The time at which the event was received by EventBridge. This variable is generated by EventBridge and can't be overwritten.
- `aws.events.event.json` — The exact payload of an event as a string.

Input Transform Examples

The following is an example Amazon EC2 event.

```
{
    "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
{
  "instance" : "$.detail.instance",
  "state" : "$.detail.state"
}
```

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></td>
<td>Note that this is the behavior in the EventBridge console. The AWS CLI escapes the slash characters and the result is &quot;instance &quot;i-0123456789&quot; is in RUNNING&quot;.</td>
<td></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 strings and variables</td>
<td>{ &quot;instance&quot; : &lt;instance&gt;, &quot;state&quot;: &quot;&lt;state&gt;&quot;, &quot;instanceStatus&quot;: &quot;instance &quot;&lt;instance&gt;&quot; is in &lt;state&gt;&quot; }</td>
<td>{ &quot;instance&quot; : &quot;i-0123456789&quot;, &quot;state&quot;: &quot;RUNNING&quot;, &quot;instanceStatus&quot;: &quot;instance &quot;i-0123456789&quot; is in 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>
</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.

---

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

Topics
- Archive Amazon EventBridge events (p. 62)
- Replay archived Amazon EventBridge event (p. 64)

In EventBridge, you can create an archive of events so that you can easily replay them at a later time by starting an event replay. You might want to replay events to, for example, recover from errors or validate new functionality in your application.
Archive Amazon EventBridge events

When you create an archive in EventBridge, you can determine which events are sent to the archive by specifying a filter pattern to match events. Any events that match the filter criteria you specify are sent to the archive. You also set a retention period to store events in the archive before they are discarded.

By default, EventBridge encrypts event data in an archive using 256-bit Advanced Encryption Standard (AES-256) under an AWS owned CMK, which helps secure your data from unauthorized access.

To create an archive for all events

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the left navigation pane, choose Archives.
3. Choose Create archive.
4. Under Archive detail, enter a Name for the archive. The name must be unique to your account in the selected Region.
   
   You can't change the name after you create the archive.
5. Optionally, enter a Description for the archive.
6. For Source, select the event bus that emits the events to send to the archive.
7. For Retention period, do one of the following:
   • Choose Indefinite to retain the events in the archive and not ever delete them.
   • Enter the number of days to retain the events. After the number of days specified events are deleted from the archive.
8. Choose Next.
9. Under Event pattern, choose No event filtering.
10. Choose Create archive.

To create an archive with an event filter pattern

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the left navigation pane, choose Archives.
3. Choose Create archive.
4. Under Archive detail, enter a Name for the archive. The name must be unique to your account in the selected Region.
   
   You can't change the name after you create the archive.
5. Optionally, enter a Description for the archive.
6. For Source, select the event bus that emits the events to send to the archive.
7. For Retention period, do one of the following:
   • Choose Indefinite to retain the events in the archive and not ever delete them.
   • Enter the number of days to retain the events. After the number of days specified events are deleted from the archive.
8. Choose Next.
10. Do one of the following:
   • Select Pattern builder, then choose the Service provider. If you choose AWS, also select the AWS service name and Event type to use in the pattern.
   • Select JSON editor to create a pattern manually. You can also copy the pattern from a rule and then paste it into the JSON editor.
11. Choose **Create archive**.

To confirm that events are successfully sent to the archive, you can use the **DescribeArchive** operation of the EventBridge API to retrieve details for the archive. The value returned for **EventCount** reflects the number of events in the archive. If it is 0, no events have been sent to the archive.
Replay archived Amazon EventBridge event

After you create an archive, you can then replay events from the archive at a later time. For example, if you update an application with additional functionality, you can replay historical events to ensure that the events are reprocessed to keep the application consistent. You can also use an archive to replay events for new functionality. When you create a replay, you can specify the archive to replay events from, the start and end time for the event to replay, and the target event bus or one or more rules to replay the events to.

Events are not necessarily replayed in the exact same order that they were added to the archive. A replay processes events to replay based on the time in the event, and replays them using 1 minute intervals. If you specify an event start time and an event end time that covers a 20 minute time range, the events are replayed from the first minute of that 20 minute range first. Then the events from the second minute are replayed. You can use the DescribeReplay operation of the EventBridge API to determine the progress of a replay. The value returned for EventLastReplayedTime indicates the time within the specified time range associated with the last event replayed.

Events are replayed based on, but separate from, the PutEvents transactions per second (TPS) limit for the account. You can request an increase to the limit for PutEvents. To learn more, see Amazon EventBridge Quotas.

**Note**
You can have a maximum of 10 active concurrent replays per account per Region.

**To start an event replay**

1. Open the Amazon EventBridge console at [https://console.aws.amazon.com/events/](https://console.aws.amazon.com/events/).
2. In the left navigation pane, choose **Replays**.
3. Choose **Start new replay**.
4. Enter a **Name** for the replay and, optionally, a **Description**.
5. For **Source**, select the archive to replay events from.
6. For destination, you can currently replay events only to the same event bus that emitted the events.
7. For **Specify rules**, do one of the following:
   - Choose **All rules** to replay events to all rules.
   - Choose **Specify rules**, and then select the rule or rules to replay the events to.
8. Under **Replay time frame**, specify the **Date**, **Time**, and **Time zone** for the **Start time** and the **End time**. Only events that occurred between the **Start time** and **End time** are replayed.
9. Choose **Start replay**.

When the events from the archived are replayed, the status of the replay is **Completed**.

If you start a replay and then want to interrupt it, you can cancel any replay with a status of **Starting** or **Running**.

**To cancel a replay**

1. Open the Amazon EventBridge console at [https://console.aws.amazon.com/events/](https://console.aws.amazon.com/events/).
2. In the left navigation pane, choose **Replays**.
3. Select the radio button next to the replay to cancel.
4. Choose **Cancel**.
Amazon EventBridge schemas

Topics

• Amazon EventBridge schema discovery (p. 66)
• Amazon EventBridge schema registry (p. 67)
• Create a Amazon EventBridge schema (p. 68)
• Amazon EventBridge code bindings (p. 73)

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.

EventBridge supports both OpenAPI 3 and JSONSchema Draft4 formats. Consider the additional following differences when determining the format to use.

• JSONSchema format supports additional keywords that are not supported in OpenAPI, such as $schema, additionalItems.
• There are minor differences in how keywords are handled, such as type and format.
• OpenAPI does not support JSONSchema Hyper-Schema hyperlinks in JSON documents.
• Tools for OpenAPI tend to focus on build-time, whereas tools for JSONSchema tend to focus on run-time operations, such as client tools for schema validation.

For a practical use case, consider using JSONSchema format to implement client-side validation so that you can ensure that events sent to EventBridge conform to the schema. You can use JSONSchema to define a contract for valid JSON documents, and then use a JSON schema validator before sending the associated events.
Amazon EventBridge schema discovery

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.

4. 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.
Amazon EventBridge schema registry

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

<result>

The custom registry is created. You can now see it on the main Schemas page.
</result>

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.

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

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

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
Create a Amazon EventBridge schema

Schemas are defined using JSON files using either the OpenAPI Specification or the JSON Schema Draft4 specification. You can create or upload your own event schemas in EventBridge using either of these specifications. You can download a template, or you can edit a template directly in the EventBridge console.

Topics
- Generate a Schema from Event JSON (p. 69)
- Generate a Schema Based on Events on an Event Bus (p. 71)

To create an OpenAPI 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 copy the JSON from the following code example.

```json
{
  "openapi": "3.0.0",
  "info": {
    "version": "1.0.0",
    "title": "Event"
  },
  "paths": {},
  "components": {
    "schemas": {
      "Event": {
        "type": "object",
        "properties": {
          "ordinal": {
            "type": "number",
            "format": "int64"
          },
          "name": {
            "type": "string"
          },
          "price": {
            "type": "number",
            "format": "double"
          },
          "address": {
            "type": "string"
          },
          "comments": {
            "type": "array",
            "items": {
              "type": "string"
            }
          },
          "created_at": {
            "type": "string",
            "format": "date-time"
          }
        }
      }
    }
  }
}
```
5. Edit the template so that the schema matches your events. For more information on events, see Amazon EventBridge events (p. 16).

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 to create.
6. For Schema type, choose either OpenAPI 3.0 or JSON Schema Draft 4.
7. With the Create tab selected, either drag your schema file to the text box, or paste the schema source.
8. 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. For Schema type, choose either OpenAPI 3.0 or JSON Schema Draft 4.
6. Optionally, enter a description for the schema to create.
7. For Schema type, choose either OpenAPI 3.0 or JSON Schema Draft 4.
8. With the Create tab selected, choose Load template.
9. Edit the template so that the schema matches your events. For more information on events, see Amazon EventBridge events (p. 16).
10. 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. For Schema type, choose OpenAPI 3.0.

You cannot use Discover from JSON to create a JSONSchema schema.

7. Select Discover from JSON
8. 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.

```
{
  "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
  }
}
```


10. EventBridge generates an OpenAPI schema for the event. For example, the following is the generated schema for the event you pasted in.

```
{
  "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"
          },
          "id": {
            "type": "string"
          },
          "detail-type": {
            "type": "string"
          },
          "region": {
            "type": "string"
          }
        }
      }
    }
  }
}
```
11. Once the schema has been generated, select **Create**.

### 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.
Amazon EventBridge code bindings

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.
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.
Events from AWS services

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

Amazon EventBridge processes all received events at least once. Each service that integrates with EventBridge offers event delivery to EventBridge as either "best effort" or "guaranteed" delivery. Best effort delivery means that the service attempts to send all events to EventBridge, but in some rare cases an event may not be delivered. Guaranteed delivery means that all events from the service are successfully delivered to 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. 85).

AWS service events

- Amazon AppFlow Events (p. 75)
- Application Auto Scaling Events (p. 76)
- Amazon Athena Events (p. 76)
- Amazon Augmented AI Events (p. 76)
- AWS Batch Events (p. 76)
- Amazon Braket Events (p. 76)
- AWS Certificate Manager Private Certificate Authority Events (p. 76)
- Amazon Chime Events (p. 76)
- AWS CodeArtifact Events (p. 76)
- AWS CodeBuild Events (p. 77)
- AWS CodeCommit Events (p. 77)
- AWS CodeDeploy Events (p. 77)
- AWS CodePipeline Events (p. 78)
- AWS Config Events (p. 79)
- Amazon Connect Events (p. 79)
- AWS Data Exchange Events (p. 79)
- Amazon Data Lifecycle Manager Events (p. 80)
- AWS DataSync Events (p. 80)
- Amazon EBS Events (p. 80)
- Amazon EBS volume modifications Events (p. 80)
- Amazon EC2 Auto Scaling Events (p. 80)
- Amazon EC2 Events (p. 80)
- Amazon EC2 Fleets Events (p. 80)
- Amazon EC2 Instance Rebalance Recommendation Events (p. 80)
- Amazon EC2 Spot Instance Interruption Events (p. 81)
- Amazon EC2 State Change Events (p. 80)
- AWS Elastic Beanstalk Events (p. 81)
- Amazon Elastic Container Registry (Amazon ECR) Events (p. 81)
- Amazon Elastic Container Service (Amazon ECS) Events (p. 82)
- AWS Elemental MediaConvert Events (p. 82)
- AWS Elemental MediaLive Events (p. 82)
Amazon AppFlow Events

Amazon AppFlow offers event delivery as best effort delivery.
Application Auto Scaling Events

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

Application Auto Scaling offers event delivery as best effort delivery.

Amazon Athena Events

Amazon Athena offers event delivery as best effort delivery.

Amazon Augmented AI Events

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

AWS Batch Events

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

AWS Batch offers event delivery as guaranteed delivery.

Amazon Braket Events

Amazon Braket offers event delivery as guaranteed delivery. To learn more see Events and automated actions for Amazon Braket with Amazon EventBridge

AWS Certificate Manager Private Certificate Authority Events

AWS Certificate Manager Private Certificate Authority offers event delivery as best effort delivery.

Amazon Chime Events

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

Amazon Chime offers event delivery as best effort delivery.

AWS CodeArtifact Events

AWS CodeArtifact offers event delivery as guaranteed delivery.
AWS CodeBuild Events

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

CodeBuild offers event delivery as best effort delivery.

AWS CodeCommit Events

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

CodeCommit offers event delivery as best effort delivery.

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 offers event delivery as best effort delivery.

**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": "c071bf6b-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",
}
```
AWS 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",
```
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.

AWS Config Events

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

AWS Config offers event delivery as best effort delivery.

Amazon Connect Events

Amazon Connect offers event delivery as best effort delivery.

AWS Data Exchange Events

AWS Data Exchange offers event delivery as best effort delivery.
Amazon Data Lifecycle Manager Events

Amazon Data Lifecycle Manager offers event delivery as best effort delivery.

AWS DataSync Events

AWS DataSync offers event delivery as best effort delivery.

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 EBS offers event delivery as best effort delivery.

Amazon EBS volume modifications Events

Amazon EBS volume modifications offers event delivery as best effort delivery.

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.

Auto Scaling offers event delivery as best effort delivery.

Amazon EC2 Events

Amazon EC2 offers event delivery as best effort delivery.

Amazon EC2 Fleets Events

Amazon EC2 Fleets offers event delivery as best effort delivery.

Amazon EC2 Instance Rebalance Recommendation Events

For information about the events for EC2 instance rebalance recommendations, see Monitoring rebalance recommendation signals in the Amazon EC2 User Guide for Linux Instances.
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.

Spot Instance Interruption offers event delivery as best effort delivery.

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"
    }
}
```

AWS Elastic Beanstalk Events

For information about the Elastic Beanstalk events, see Using Elastic Beanstalk with Amazon EventBridge in the AWS Elastic Beanstalk Developer Guide.

Elastic Beanstalk offers event delivery as best effort delivery.

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

Amazon ECR offers event delivery as best effort delivery.
Amazon Elastic Container Service (Amazon ECS) Events

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

Amazon ECS offers event delivery as guaranteed delivery.

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.

MediaConvert offers event delivery as guaranteed delivery.

AWS Elemental MediaLive Events

AWS Elemental MediaLive offers event delivery as best effort delivery.

AWS Elemental MediaPackage Events

For MediaPackage sample events, see Monitoring AWS Elemental MediaPackage with Amazon CloudWatch Events in the AWS Elemental MediaPackage User Guide.

MediaPackage offers event delivery as best effort delivery.

AWS Elemental MediaStore Events

For MediaStore sample events, see Automating AWS Elemental MediaStore with CloudWatch Events in the AWS Elemental MediaStore User Guide.

MediaStore offers event delivery as guaranteed delivery.

Amazon EMR Events

Events reported by Amazon EMR have aws.emr as the value for Source, while Amazon EMR API events reported via CloudTrail have aws.elasticmapreduce as the value for Source.

Amazon EMR offers event delivery as best effort delivery.

The following are examples of events reported by Amazon EMR.

Amazon EMR Auto Scaling Policy State Change

{}
Amazon EventBridge User Guide
Amazon EMR Events

Amazon EMR Cluster State Change – Starting

{
  "version": "0",
  "id": "999ccccaa-eeaa-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-123456789ABCD",
    "state": "STARTING",
    "message": "Amazon EMR cluster j-123456789ABCD (Development Cluster) was requested at 2016-12-16 20:43 UTC and is being created."
  }
}

Amazon EMR Cluster State Change – Terminated

{
  "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
Amazon EventBridge Scheduled Events

The following is an example of a scheduled event:

```json
{
  "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" ]
}
```
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 Tutorial: Create a Amazon EventBridge rule for AWS CloudTrail API calls (p. 154).

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": "9D7678CC3B4E7487",
"eventID": "24ba271e-d595-4e66-a7fd-9c16cbf8abae",
"eventType": "AwsApiCall"
}
```

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.

Events from CloudWatch

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

Events from CloudWatch are guaranteed delivery.

Amazon GameLift Event

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

Amazon GameLift offers event delivery as best effort delivery.

Matchmaking Searching

```json
{
  "version": "0",
  "id": "cc3d3ebe-1d90-48f8-b268-c96655b8f013",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-08T21:15:36.421Z",
  "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"
          }
        ],
        "estimatedWaitMillis": "NOT_AVAILABLE",
        "type": "MatchmakingSearching",
        "gameSessionInfo": {
          "players": [
            {
              "playerId": "player-1"
            }
          ]
        }
      }
    ],
    "type": "Potential Match Created"
  }
}
```
"version": "0",
"id": "fcee863f-aea3-45bc-aeba-99d639cad2d4",
"detail-type": "GameLift Matchmaking Event",
"source": "aws.gamelift",
"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": "b3f76666-c8e5-416a-aa4c-aa127813edc",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-09T20:04:42.660Z",
  "region": "us-west-2",
  "resources": [
  ],
  "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
Amazon EventBridge User Guide

Amazon GameLift Event

Matchmaking Succeeded

```json
{
"version": "0",
"id": "b1990d3d-f737-4d6c-b150-af5ace8c35d3",
"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"
    }
}
}```
Matchmaking Timed Out

{
  "version": "0",
  "id": "fe528a7d-46ad-4bdc-96cb-b094b5f6bf56",
  "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": [

```json
],
  "detail": {
    "tickets": [
    {
      "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 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",
            }
        ]
    }
}
```
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"
                    }
                ]
            }
        ]
    }
}```
AWS Glue DataBrew Events

AWS Glue DataBrew offers event delivery as best effort delivery.

AWS Glue Events

AWS Glue offers event delivery as best effort delivery.

The following is the format for AWS Glue events.

**Successful Job Run**

```json
{
  "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_abcd00123456789abcdef0123456789abcdef0123456789abcdef0123456789",
    "message": "Job run succeeded"
  }
}
```

**Failed Job Run**

```json
{
  "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_0123456789abcdef0123456789abcdef0123456789abcdef",
  "message":"JobName:MyJob and
  JobRunId:jr_0123456789abcdef0123456789abcdef0123456789abcdef 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_abcd0123456789abcdef0123456789abcdef0123456789abcdef",
    "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_abcd0123456789abcdef0123456789abcdef0123456789abcdef",
    "message":"Job run stopped"
  }
}

Crawler Started

{
  "version":"0",
  "id":"05e8a2-c309-6884-a41b-3508bcd9695",
  "detail-type":"Glue Crawler State Change",
  "source":"aws.glue",
  "account":"561226563745",
  "message":"Crawler started"
}
Crawler Succeeded

{  "version": "0",  "id": "3d675db5-59b9-6388-b8e8-e0a9b6d567a9",  "detail-type": "Glue Crawler State Change",  "source": "aws.glue",  "account": "561226563745",  "time": "2017-11-11T01:25:00Z",  "region": "us-east-1",  "resources": [    ],  "detail": {    "accountId": "561226563745",    "crawlerName": "SchedulerTestCrawler51fb3a8b-1015-49f0-a969-ca126680b94b",    "startTime": "2017-11-11T01:25:00Z",    "state": "Succeeded",    "message": "Crawler Succeeded",    "partitionsCreated": "0",    "partitionsUpdated": "0",    "partitionsDeleted": "0",    "runningTime (sec)": "7",    "tablesCreated": "0",    "tablesUpdated": "0",    "tablesDeleted": "0",    "cloudWatchLogLink": "https://console.aws.amazon.com/cloudwatch/home?region=us-east-1#logEventViewer:group=/aws-glue/crawlers;stream=SchedulerTestCrawler51fb3a8b-1015-49f0-a969-ca126680b94b"  } }

Crawler Failed

{  "version": "0",  "id": "f7965b59-470f-2e06-bb89-a8cebaabefac",  "detail-type": "Glue Crawler State Change",  "source": "aws.glue",  "account": "782104008917",  "time": "2017-10-20T05:10:08Z",  "region": "us-east-1",  "resources": [    ],  "detail": {    "crawlerName": "test-crawler-notification",    "errorMessage": "Internal Service Exception",    "errorType": "Internal Server Error",    "reason": "Internal Service Exception",    "cloudWatchLogLink": "https://console.aws.amazon.com/cloudwatch/home?region=us-east-1#logEventViewer:group=/aws-glue/crawlers;stream=test-crawler-notification"  } }
Job Run is in Starting 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":"STARTING",
    "jobRunId":"jr_6aa58e7a3aa44e2e4c7db2c50e2f7396cb57901729e4b702dcb2cfbbeb3f7a86",
    "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_6aa58e7a3aa44e2e4c7db2c50e2f7396cb57901729e4b702dcb2cfbbeb3f7a86",
    "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_6aa58e7a3aa44e2e4c7db2c50e2f7396cb57901729e4b702dcb2cfbbeb3f7a86",
    "message":"Job is in STOPPING state",
    "startedOn":"2018-04-24T20:55:47.941Z"
  }
}
```
"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_6aa58e7a3aa44e2e4c7db2c50e2f7396cb57901739e4b702dcb2cfbb6b3f7a86",
  "message":"Job is in STOPPING state",
  "startedOn":"2018-04-24T20:55:47.941Z"
}

AWS Glue Data Catalog Table State Change

{
  "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.

{
  "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 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.

AWS Ground Station offers event delivery as best effort delivery.

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 offers event delivery as best effort delivery.

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

```
{
  "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"
    }]
  }
}
```
Amazon Interactive Video Service (IVS) Events

For information about example Amazon IVS events, see Using Amazon EventBridge with Amazon IVS in the Amazon IVS User Guide.

Amazon IVS offers event delivery as best effort delivery.

AWS IoT Analytics Events

AWS IoT Analytics offers event delivery as guaranteed delivery.

AWS IoT Greengrass Events

AWS IoT Greengrass offers event delivery as best effort delivery.

For information about AWS IoT Greengrass V1 events, see:

- Get Deployment Notifications in the AWS IoT Greengrass V1 Developer Guide.
- Gathering system health telemetry data from AWS IoT Greengrass core devices in the AWS IoT Greengrass V1 Developer Guide.

For information about AWS IoT Greengrass V2 events, see:

- Gather system health telemetry data from AWS IoT Greengrass core devices in the AWS IoT Greengrass V2 Developer Guide.

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.

AWS KMS CMK Rotation offers event delivery as best effort delivery.

AWS KMS Imported Key Material Expiration offers event delivery as best effort delivery.

AWS KMS CMK Deletion offers event delivery as guaranteed delivery.

KMS CMK Rotation

AWS KMS automatically rotated a CMK's key material.
Amazon Location Service events

For information about example Amazon Location Service events, see Reacting to Amazon Location Service events with Amazon EventBridge.
Amazon Location Services offers event delivery as guaranteed delivery.

Amazon Macie Classic Events

Macie Classic offers event delivery as best effort delivery.

The following are examples of Amazon Macie Classic events.

**Alert Created**

```json
{
    "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": "2017-04-18T20:21:12.059000",
        "actor": "555566667777:assumed-role:superawesome:aroaidpldctnsefnheji",
        "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
  }
},
"GetBucketPolicy": {
  "count": 37,
  "ISP": {
    "Amazon": 37
  },
  "Error Code": {
    "NoSuchBucketPolicy": 22
  }
},
"GetBucketAcl": {
  "count": 37,
  "ISP": {
    "Amazon": 37
  }
},
"ListBuckets": {
  "count": 1,
  "ISP": {
    "Amazon": 1
  }
},
"recipientAccountId": {
  "123456789012": 125
}

{
  "version": "0",
  "id": "CWE-event-id",
  "detail-type": "Macie Alert",
  "source": "aws.macie",
  "account": "123456789012",
  "time": "2017-04-18T18:15:41Z",
  "region": "us-east-1",
  "resources": ["arn:aws:macie:us-east-1:123456789012:trigger/trigger_id/alert/alert_id",
                "arn:aws:macie:us-east-1:123456789012:trigger/trigger_id"
  ],
  "detail": {
    "notification-type": "ALERT_CREATED",
    "name": "Bucket is writable by all authenticated users",
    "tags": [
      "Custom_Alert",
      "Audit"
    ],
    "url": "https://lb00.us-east-1.macie.aws.amazon.com/1112223333/posts/alert_id",
    "risk-score": 70,
    "trigger": 
  }
}
"alert-type": "basic",
"created-at": "2017-04-08 00:21:30.749000",
"description": "Bucket is writable by all authenticated users",
"risk": 7
},
"created-at": "2017-04-18T18:16:17.046454",
"actor": "444455556666",
"summary": {
  "Description": "Bucket is writable by all authenticated users",
  "Bucket": {
    "secret-bucket-name": 1
  },
  "Record Count": 1,
  "ACL": {
    "secret-bucket-name": [
      {
        "Owner": {
          "DisplayName": "bucket_owner",
          "ID": "089d28d42f4b392f5c561f073bd2e4a37b3bb2e62659318c6960e8981648a17e"
        },
        "Grants": [
          {
            "Grantee": {
              "Type": "Group",
              "URI": "http://acs.amazonaws.com/groups/global/AuthenticatedUsers"
            },
            "Permission": "WRITE"
          }
        ]
      }
    ],
    "Event Count": 1,
    "Timestamps": {
      "2017-01-10T22:48:06.784937": 1
    }
  }
}

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/11112223333/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": "2017-04-08T00:36:26.270000",
"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/111122223333/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": "Information collection activity suggesting the start of a reconnaissance or exfiltration campaign",
"anomalous": true,
"multiplier": 8.420560747663552,
"excession_times": [
"2017-04-21T18:00:00Z"
],
"risk": 1
},
"ListRolePolicies": {
"name": "ListRolePolicies",
"description": "Lists the names of the inline policies that are embedded in the specified IAM role",
"narrative": "Information collection activity suggesting the start of a reconnaissance or exfiltration campaign",
"anomalous": true,
"multiplier": 12.017441860465116,
"excession_times": [
"2017-04-21T18:00:00Z"
],
"risk": 2
}
}
}

Amazon Macie Events

For examples of events generated by Amazon Macie, see Event schema for Amazon Macie findings. Macie offers event delivery as best effort delivery.

Amazon Managed Blockchain Events

Amazon Managed Blockchain offers event delivery as best effort delivery.

AWS Managed Services Events

AWS Managed Services offers event delivery as best effort delivery.

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:

```
{
    "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",
        "source IPAddress": "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"
        },
        "additionalEventData": {
            "LoginTo": "https://console.aws.amazon.com/console/home?state=hashArgs%23isauthcode=true",
            "MobileVersion": "No",
            "MFAUsed": "No"
        },
        "eventId": "324731c0-64b3-4421-b552-dfc3c27df4f6",
        "eventType": "AwsConsoleSignIn"
    }
}
```

AWS OpsWorks Stacks Events

AWS OpsWorks offers event delivery as guaranteed delivery.

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

{  
  "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-a50z3e4z500z"
  ],
  "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-a50z3e4z500z",
    "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.
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

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.
Amazon QLDB Events

Amazon QLDB offers event delivery as guaranteed delivery.

AWS RAM Events

AWS RAM offers event delivery as best effort delivery.

Amazon RDS Events

Amazon RDS offers event delivery as best effort delivery.

Amazon Redshift Events

Amazon Redshift offers event delivery as best effort delivery.

Amazon SageMaker Events

For information about example SageMaker events, see Automating Amazon SageMaker with Amazon EventBridge in the Amazon SageMaker Developer Guide.

Savings Plans Events

For examples of events generated by Savings Plans, see Automating Savings Plans with Amazon EventBridge in the Savings Plans User Guide.

Savings Plans offers event delivery as best effort delivery.

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.

Security Hub offers event delivery as best effort delivery.
AWS Server Migration Service Events

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

**Deleted replication job notification**

```json
{
    "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**

```json
{
    "version": "0",
    "id": "3f9c59cc-f941-522a-be6d-f08e44ff1715",
    "detail-type": "Server Migration Job State Change",
    "source": "aws.sms",
    "account": "123456789012",
    "time": "2018-02-07T22:54:00Z",
    "region": "us-west-1",
    "resources": [
    ],
    "detail": {
        "state": "Completed",
        "replication-run-id": "sms-run-e1a64388",
        "replication-job-id": "sms-job-2ea64347",
        "ami-id": "ami-746d6314",
        "version": "1.0"
    }
}
```

**AWS Signer Events**

AWS Signer offers event delivery as guaranteed delivery.

**AWS Step Functions Events**

For Step Functions sample events, see [Step Functions Event Examples](#) in the AWS Step Functions Developer Guide.
Step Functions offers event delivery as best effort delivery.

AWS Storage Gateway Events

AWS Storage Gateway offers event delivery as guaranteed delivery.

Amazon SWF Events

Amazon SWF offers event delivery as best effort delivery.

AWS Systems Manager Events

The following are examples of the events for AWS Systems Manager. For more information, see Monitoring Systems Manager events with Amazon EventBridge in the AWS Systems Manager User Guide.

Systems Manager offers event delivery as best effort delivery.

Topics

- AWS Systems Manager Automation Events (p. 112)
- AWS Systems Manager Change Calendar Events (p. 113)
- AWS Systems Manager Compliance Events (p. 114)
- AWS Systems Manager Maintenance Windows Events (p. 115)
- AWS Systems Manager Parameter Store Events (p. 117)
- AWS Systems Manager Run Command Events (p. 118)
- AWS Systems Manager State Manager Events (p. 119)

AWS Systems Manager Automation Events

Automation Step Status-change Notification

```json
{
  "version": "0",
  "id": "eeca120b-a321-433e-9635-dab369006a6b",
  "detail-type": "EC2 Automation Step 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",
                "arn:aws:ssm:us-east-1:123456789012:automation-definition/runcommand1:1"],
  "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",
  }
}```
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",
                "arn:aws:ssm:us-east-1:123456789012:automation-definition/runcommand1:1"],
  "detail": {
    "ExecutionId": "333ba70b-2333-48db-b17e-a5e69c6f4d1c",
    "Definition": "runcommand1",
    "DefinitionVersion": 1.0,
    "Status": "Success",
    "StartTime": "Nov 29, 2016 7:43:20 PM",
    "EndTime": "Nov 29, 2016 7:43:26 PM",
    "Time": 5753.0,
    "ExecutedBy": "arn:aws:iam::123456789012:user/userName"
  }
}
```

AWS Systems Manager Change Calendar Events

The following are examples of the events for AWS Systems Manager Change Calendar.

**Calendar OPEN**

```
{
  "version": "0",
  "id": "47a3f03a-f30d-1011-ac9a-du3bdEXAMPLE",
  "detail-type": "Calendar State Change",
  "source": "aws.ssm",
  "account": "111222333444",
  "time": "2020-09-19T18:00:07Z",
  "region": "us-east-2",
  "resources": ["arn:aws:ssm:us-east-2:111222333444:document/MyCalendar"],
  "detail": {
    "state": "OPEN",
    "atTime": "2020-09-19T18:00:07Z",
    "nextTransitionTime": "2020-10-11T18:00:07Z"
  }
}
```

**Calendar CLOSED**

```
{
  "version": "0",
  "id": "f30df03a-1011-ac9a-47a3-f761eEXAMPLE",
  "detail-type": "Calendar State Change",
  "source": "aws.ssm",
  "account": "111222333444",
```
AWS Systems Manager Compliance Events

The following are examples of the events for AWS Systems Manager Compliance.

**Association 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": "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": "2017-07-17T19:01:26Z",
  "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": "non_compliant",
    "resource-type": "managed-instance",
    "resource-id": "i-01234567890abcdef",
    "compliance-type": "Association"
  }
}
```

**Patch Compliant**

```
```

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

**Register a Target**

The other valid status value is **DEREGISTERED**.

```json
{
    "version": "0",
    "id": "01234567-0123-0123-0123-012345678901",
    "detail-type": "Maintenance Window Target Registration Notification",
    "source": "aws.ssm",
    "account": "012345678901",
    "time": "2016-11-16T00:58:37Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:ssm:us-east-1:012345678901:managed-instance/i-01234567890abcdef"
    ],
    "detail": {
        "resource-type": "managed-instance",
        "resource-id": "i-01234567890abcdef",
        "compliance-status": "non_compliant",
        "compliance-type": "Patch",
        "patch-baseline-id": "PB789",
        "severity": "critical"
    }
}
```
Window Execution Type

The other valid status values are PENDING, IN_PROGRESS, SUCCESS, FAILED, TIMED_OUT, and SKIPPED_OVERLAPPING.

The other valid status values are IN_PROGRESS, SUCCESS, FAILED, and TIMED_OUT.

Task Target Processed
The other valid status values are IN_PROGRESS, SUCCESS, FAILED, and TIMED_OUT.

```
{
  "version":"0",
  "id":"01234567-0123-0123-0123-0123456789ab",
  "detail-type":"Maintenance Window Task Target Invocation State-change Notification",
  "source":"aws.ssm",
  "account":"012345678901",
  "time":"2016-11-16T01:00:57Z",
  "region":"us-east-1",
  "resources":[
  ],
  "detail":{
    "start-time":"2016-11-16T01:00:56.427Z",
    "end-time":"2016-11-16T01:00:57.070Z",
    "window-id":"mw-0ed7251d3fcf6e0c2",
    "window-execution-id":"b60fb56e-776c-4e5c-84ee-123456789012",
    "task-execution-id":"e7265f13-3cc5-4f2f-97a9-123456789012",
    "window-target-id":"e7265f13-3cc5-4f2f-97a9-123456789012",
    "status":"TIMED_OUT",
    "owner-information":"Owner"
  }
}
```

**Window State Change**

The valid status values 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 Systems Manager Parameter Store.

**Create Parameter**

```
{
  "version": "0",
  "id": "6a7e4f6b-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": "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 Systems Manager Run Command Events

Run Command Status-change Notification

{
  "version": "0",
  "id": "51c0891d-0e34-45b1-83d6-95db273d1602",
  "detail-type": "EC2 Command Status-change Notification",
}
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",
        "expire-after": "2016-07-14T22:01:30.049Z",
        "parameters": {
            "executionTimeout": ["3600"],
            "commands": ["date"]
        },
        "requested-date-time": "2016-07-10T21:51:30.049Z",
        "status": "Success"
    }
}
Tag Change Events on AWS Resources

Tag Change offers event delivery as best effort delivery.

The following is an example of a tag event.

```json
{
  "version": "0",
  "id": "f7fba9fe-32f8-ef66-c85c-11111111111",
  "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-0000000aaaaaaaaaa"
  ],
  "detail": {
    "changed-tag-keys": [
      "key2",
      "key3"
    ]
  }
}
```
AWS Transit Gateway Events

AWS Transit Gateway offers event delivery as best effort delivery.

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-123a-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"
        }
    }
}
```
Load Balancer Optimization

{
  "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": "Load Balancer Optimization",
    "check-item-detail": {
      "Instances in Zone a": "1",
      "Status": "Yellow",
      "Instances in Zone b": "0",
      "# of Zones": "2",
      "Region": "eu-central-1",
      "Load Balancer Name": "my-load-balance",
      "Instances in Zone e": null,
      "Instances in Zone c": null,
      "Reason": "Single AZ",
      "Instances in Zone d": null
    },
    "status": "WARN",
    "uuid": "aa12345f-55c7-498e-b7ac-123456789012"
  }
}

Exposed Access Keys

{
  "version": "0",
  "id": "1234abcd-ab12-123a-123a-1234567890ab",
  "detail-type": "Trusted Advisor Check Item Refresh Notification",
  "source": "aws.trustedadvisor",
  "account": "123456789012",
  "time": "2018-01-12T19:38:24Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "check-name": "Exposed Access Keys",
    "check-item-detail": {
      "Case ID": "12345678-1234-1234-abcd-1234567890ab",
      "Usage (USD per Day)": "0",
      "User Name (IAM or Root)": "my-username",
      "Deadline": "144053299248",
      "Access Key ID": "AKIAIOSFODNN7EXAMPLE",
      "Time Updated": "1440021299248",
      "Fraud Type": "Exposed",
      "Location": "www.example.com"
    },
    "status": "ERROR",
    "resource_id": "",
  }
}
Amazon WorkSpaces Events

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

Amazon WorkSpaces offers event delivery as best effort delivery.

AWS X-Ray Events

AWS X-Ray offers event delivery as best effort delivery.
Amazon EventBridge related services

You can use EventBridge with other AWS services:

**Topics**
- Using Amazon EventBridge with Interface VPC Endpoints (p. 125)
- Amazon EventBridge integration with AWS X-Ray (p. 127)
Using Amazon 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.

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.

The following steps are for users of Amazon VPC. For more information, see Getting Started in the Amazon VPC User Guide.

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.

When you use a private interface VPC endpoint, custom events sent from your VPC to EventBridge use that endpoint. EventBridge then sends those events to other AWS services based on the rules and
targets that you configure. Once events are sent to another service you can receive them via either the public endpoint or a VPC endpoint for that service. For example, if you create a rule to send events to an Amazon SQS queue, you can configure an interface VPC endpoint for Amazon SQS in order to receive messages from that queue in your VPC without using the public endpoint. If you do not configure a VPC endpoint for Amazon SQS, you cannot use the EventBridge endpoint for receiving those events.
Amazon EventBridge integration with AWS X-Ray

AWS X-Ray integrates with Amazon EventBridge to trace events that are passed through EventBridge. If a service traces requests by using the X-Ray SDK, the trace header is passed through EventBridge and the original trace header propagates to the invocation targets with consistent trace context. Trace continuity enables users to track, analyze, and debug throughout downstream services.

Trace headers are validated against X-Ray’s validation logic. Trace headers that don’t satisfy X-Ray’s validation logic will be dropped and not propagated. However, the events will still be processed. The trace header will not be available on the event that’s delivered to the invocation target. This means:

- If you have an event archive, the trace header won’t be available on archived events. If you replay archived events the trace header won’t be included.
- If you have configured a dead-letter queue (DLQ) to send events with failed invocations, the trace header is included in the SendMessage request that sends the event to the DLQ. If you retrieve events (messages) from the DLQ using ReceiveMessage, the trace header associated with the event is included on the Amazon SQS message attribute but is not included in the event payload (message body).

To see how an EventBridge event node connects source and target services, see Viewing source and targets in the X-Ray service map in the AWS X-Ray Developer Guide.

EventBridge supports the following trace header information:

- **Default HTTP Header** – The X-Ray SDK automatically populates the trace header as the X-Amzn-Trace-Id HTTP header for all invocation targets. To learn more about the default HTTP header, see Tracing header in the AWS X-Ray Developer Guide.
- **TraceHeader System Attribute** – TraceHeader is a PutEventsRequestEntry attribute reserved by EventBridge to carry the X-Ray trace header with events to invocation targets. When both header instrumentations are set, the PutEventsRequestEntry attribute overrides the HTTP trace header.

**Note**

The trace header won’t count towards the PutEventsRequestEntry event size calculation, regardless of whether it is defined as an HTTP header or as a PutEventsRequestEntry attribute. To learn more, see Calculating Amazon EventBridge PutEvents event entry size (p. 32).
Receive events from a SaaS partner with Amazon EventBridge

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.

**Topics**
- Supported SaaS partner integrations (p. 128)
- Configure Amazon EventBridge to receive events from a SaaS integration (p. 130)
- Receiving events from SalesForce (p. 132)

**Supported SaaS partner integrations**

EventBridge supports the following SaaS partner integrations:

- Auth0
- Blitline
- BUIDLHub
- Buildkite
- Camunda
- CleverTap
- Datadog
- Epsagon
- Freshworks
- GS2
- Genesys
- Karte
- Kloudless
- Mackerel
- MongoDB
- New Relic
- OneLogin
- Opsgenie
- PagerDuty
- Payshield
- SailPoint
- Saviynt
- Segment
- Shopify
- SignalFx
- Site24x7
Supported SaaS partner integrations

- Stax
- SugarCRM
- Symantec
- Thundra
- TriggerMesh
- Whispir
- Zendesk
- Amazon Seller Partner API

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>
<tr>
<td>eu-north-1</td>
<td>Europe (Stockholm)</td>
</tr>
<tr>
<td>eu-south-1</td>
<td>Europe (Milan)</td>
</tr>
<tr>
<td>af-south-1</td>
<td>Africa (Cape Town)</td>
</tr>
<tr>
<td>ap-east-1</td>
<td>Asia Pacific (Hong Kong)</td>
</tr>
<tr>
<td>ap-northeast-1</td>
<td>Asia Pacific (Tokyo)</td>
</tr>
<tr>
<td>ap-northeast-2</td>
<td>Asia Pacific (Seoul)</td>
</tr>
<tr>
<td>ap-southeast-1</td>
<td>Asia Pacific (Singapore)</td>
</tr>
<tr>
<td>ap-southeast-2</td>
<td>Asia Pacific (Sydney)</td>
</tr>
<tr>
<td>ap-south-1</td>
<td>Asia Pacific (Mumbai)</td>
</tr>
<tr>
<td>sa-east-1</td>
<td>South America (São Paulo)</td>
</tr>
</tbody>
</table>
Configure Amazon EventBridge to receive events from a SaaS integration

The following procedure describes how to configure EventBridge to receive events from a SaaS integration partner.

**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 SaaS partner events (p. 130).

Creating a rule that triggers on SaaS partner events

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 Receive events from a SaaS partner with Amazon EventBridge (p. 128).

**To create a rule that triggers on an event from a 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 Amazon EventBridge SaaS tagging (p. 216).

16. Choose **Create**.
Receiving events from Salesforce

You can use Amazon EventBridge to receive events from Salesforce by configuring a flow in Amazon AppFlow that uses Salesforce as a data source. AppFlow then sends Salesforce events EventBridge using a partner event bus (p. 128).

Events received from Salesforce via Amazon AppFlow are encapsulated in a standard EventBridge event envelope. The following example displays a Salesforce event received by an EventBridge partner event bus.

```json
{
  "version": "0",
  "id": "5c42b99e-e005-49b3-c744-07990c50d2cc",
  "detail-type": "AccountChangeEvent",
  "source": "aws.partner/appflow.test/salesforce.com/364228160620/CustomSF-Source-Final",
  "account": "000000000",
  "time": "2020-08-20T18:25:51Z",
  "region": "us-west-2",
  "resources": [],
  "detail": {
    "ChangeEventHeader": {
      "commitNumber": 248197218874,
      "commitUser": "0056g000003XW7AAW",
      "sequenceNumber": 1,
      "entityName": "Account",
      "changeType": "UPDATE",
      "changedFields": [
        "LastModifiedDate",
        "Region__c"
      ],
      "changeOrigin": "com/salesforce/api/soap/49.0;client=SfdcInternalAPI/",
      "transactionKey": "000035af-b239-0581-9f14-461e4187de11",
      "commitTimestamp": 1597947935000,
      "recordIds": [
        "0016g00000MLhLeAAL"
      ],
      "LastModifiedDate": "2020-08-20T18:25:35.000Z",
      "Region__c": "America"
    }
  }
}
```

Configure AppFlow to use Salesforce as a partner event source

To send events to EventBridge, you first need to configure AppFlow to use Salesforce as a partner event source.

1. In the Amazon AppFlow console, choose Create flow.
2. In the Flow details section, in the Flow name field, specify a name for your flow.
3. Optionally, enter a description for the flow, and then choose Next.
4. Under Source details, choose Salesforce from the Source name drop-down, and then choose Connect to create a new connection.
5. On the Connect to Salesforce dialog, choose either Production or Sandbox for the Salesforce environment.
6. In the Connection name field, enter a unique name for the connection, and then choose Continue.
A Salesforce log in dialog box displays.

7. In the Salesforce dialog box, enter your Salesforce username and password to log in to Salesforce.
8. Select Salesforce events for the types of data for AppFlow to process.
9. In the Choose Salesforce event drop-down, select the type of event to send to EventBridge.
10. Select Amazon EventBridge as a destination.
11. Select Create new partner event source from the drop-down.
12. (Optional) Specify a unique suffix for the partner event source.
13. Choose Generate partner event source.
14. Choose an Amazon S3 bucket to store event payload files that are larger than 256 KB.
15. In the Flow trigger section, ensure that Run flow on event is selected. This setting ensures that the flow is executed when a new Salesforce event occurs.
16. Choose Next.
17. For field mapping, select Map all fields directly. Alternatively, you can select the fields that are of interest from the Source field name list.
   To learn more about field mapping, see Map data fields.
18. Choose Next.
20. Choose Next.
21. Review the settings and choose Create flow.

With the flow configured, Amazon AppFlow creates a new partner event source that you then need to associate with a partner event bus in your account.

Configure EventBridge to receive Salesforce events

Ensure that the Amazon AppFlow flow that is triggered from Salesforce events with EventBridge as a destination is configured before following instructions in this section.

To configure EventBridge to receive Salesforce events

1. Open the Partner event sources page in the EventBridge console.
2. Select the Salesforce partner event source that you created using the previous procedure in this topic.
3. Choose Associate with event bus.
4. Validate the name of the partner event bus.
5. Choose Associate.
6. In the Amazon AppFlow console, open the flow you created and choose Activate flow.
8. Choose Create rule.
9. Enter a unique name for the rule.
10. Choose Event pattern in the Define pattern section.
11. In the Event matching pattern section select Pre-defined pattern by service.
12. In the Service provider section select All Events. This option ensures that all events sent to the partner event bus are included.
13. In the Select event bus section, choose Custom or partner event bus.
14. Select the newly-created partner event bus, that was associated with the Amazon AppFlow partner event source.
15. Select a target for the rule. This is the service that is invoked when the rule is triggered. One rule can have up to five targets.

16. Choose Create.

With the active rule, the destination service will receive all Salesforce events configured for your account. If you need to filter the kinds of events that you want to process, or send different events to different targets, you can use .

**Note**
For events larger than 256KB, Amazon AppFlow does not send the full event to EventBridge. Instead, full event is put into an S3 bucket in your account, and the event payload sent to EventBridge includes a pointer to the Amazon S3 bucket. You can use the pointer to get the full event from the bucket.
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. 135)
- Tutorial: Log the State of an Amazon EC2 Instance Using EventBridge (p. 136)
- Tutorial: Log the State of an Auto Scaling Group Using EventBridge (p. 138)
- Tutorial: Log Amazon S3 Object-Level Operations Using EventBridge (p. 140)
- Tutorial: Use Input Transformer to Customize What Is Passed to the Event Target (p. 142)
- Tutorial: Log AWS API Calls Using EventBridge (p. 143)
- Tutorial: Schedule Automated Amazon EBS Snapshots Using EventBridge (p. 145)
- Tutorial: Schedule AWS Lambda Functions Using EventBridge (p. 146)
- Tutorial: Set AWS Systems Manager Automation as an EventBridge Target (p. 149)
- Tutorial: Relay Events to an Amazon Kinesis Stream Using EventBridge (p. 149)
- Tutorial: Run an Amazon ECS Task When a File Is Uploaded to an Amazon S3 Bucket (p. 151)
- Tutorial: Schedule Automated Builds Using AWS CodeBuild (p. 152)
- Tutorial: Log State Changes of Amazon EC2 Instances (p. 153)
- Tutorial: Download Code Bindings for Events using the EventBridge Schema Registry (p. 154)
- Tutorial: Create a Amazon EventBridge rule for AWS CloudTrail API calls (p. 154)

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.
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

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 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.
   a. 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.
   b. 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/](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 Amazon EventBridge target input (p. 58)
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:
   - Enter a name and description for the Lambda function. For example, name the function LogEC2StopInstance.
   - 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');
     }
     ```
   - For Role, choose Choose an existing role. For Existing role, select your basic execution role. Otherwise, create a basic execution role.
   - 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:
   - Choose Event pattern.
   - Choose Pre-defined pattern by service.
   - For Service provider, choose AWS.
   - For Service Name, choose EC2.
   - For Event type, choose AWS API Call via CloudTrail.
   - Choose Specific operations(s) and enter StopInstances in the box.
   - 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 Amazon EventBridge rules (p. 39).

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:
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.
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 `LogScheduledEvent`.
   b. Edit the sample code for the Lambda function. For example:
      ```javascript
      'use strict';
      exports.handler = (event, context, callback) => {
        console.log('LogScheduledEvent');
        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 on a schedule.

#### To create a rule using the console

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 **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 `put-rule` command.

   ```bash
   aws events put-rule \
   --name my-scheduled-rule \
   --schedule-expression 'rate(5 minutes)'
   ```

---

147
When this rule triggers, it generates an event that serves as input to the targets of this rule. The following is an example event.

```
{
   "version": "0",
   "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/my-scheduled-rule"
   ],
   "detail": {}
}
```

2. Use the `add-permission` command to trust the EventBridge service principal (`events.amazonaws.com`) and scope permissions to the rule with the specified Amazon Resource Name (ARN).

   ```
   aws lambda add-permission \
   --function-name LogScheduledEvent \
   --statement-id my-scheduled-event \
   --action 'lambda:InvokeFunction' \
   --principal events.amazonaws.com \
   --source-arn arn:aws:events:us-east-1:123456789012:rule/my-scheduled-rule
   ```

3. Use the `put-targets` command to add the Lambda function that you created to this rule so that it runs every 5 minutes.

   ```
   aws events put-targets --rule my-scheduled-rule --targets file://targets.json
   ```

Create the file `targets.json` with the following contents.

```
[
   {
      "Id": "1",
      "Arn": "arn:aws:lambda:us-east-1:123456789012:function:LogScheduledEvent"
   }
]
```

**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/](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:
   
   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 `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 `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/](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/](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](https://docs.aws.amazon.com/AmazonEC2/latest/UserGuide/Launch_Instances_Linux.html).
3. Open the Amazon EventBridge console at [https://console.aws.amazon.com/events/](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. (Optional) When you're finished, terminate the instance. For more information, see Terminate Your Instance in the Amazon EC2 User Guide for Linux Instances.

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 `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": "AAAAAAAAAAHSywljv0zEgPX4NyKd25wryMr9yALs8NeKbUjp1IxtZs1Sp +KEd9I6AJ9ZG4lNR1EMi+9Md/nHvtLyxpfeZyvKZ4D9DQve/mBYWR06OTZKnnW9gd +efGNzahFdkH1rjL4B19Wyrk+ghYG22D2tDa2EyNSH1+LABK33gQweTJADBdyMwlo5r6PqcP2dzhg="
}
```

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 AAAAAAAAAHHSyw1jv0zEgPX4NyKd25wryMr9yALs8NeKbUjp1IxtZs1Sp+KEd9I6AJ9ZG4lNR1EMi +9Md/nHvtLyxpfeZyvKZ4D9DQve/mBYWR06OTZKnnW9gd+efGNzahFdkH1rjL4B19Wyrk +ghYG22D2tDa2EyNSH1+LABK33gQweTJADBdyMwlo5r6PqcP2dzhg=
```

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/.
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.
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.
9. Unzip the downloaded file and add it to your project.

   The downloaded package contains a README file that explains how to configure the package's dependencies in various frameworks.

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

**Tutorial: Create a Amazon EventBridge rule for AWS CloudTrail API calls**

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 AWS API Call via CloudTrail as the value for detail-type. Events from API actions that start with the keywords List, Get, or Describe are not processed by EventBridge, with the exception of events from the following AWS STS actions:

- GetFederationToken
- GetSessionToken

**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 Managing Your Costs with Budgets in the AWS Billing and Cost Management User Guide.

**To create a rule that triggers on an API call via CloudTrail**

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 CloudTrail.
9. For Event type, choose AWS API Call via CloudTrail.
   If you want to customize the event pattern, choose Edit, make your changes, and choose Save.
10. 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.
11. For Target, select up to 5 targets to send AWS API Call via CloudTrail to. Configure any additional settings for the selected target. The additional settings vary by target.

   For many target types, such as a Step Functions state machine, EventBridge needs permissions to send events to the target resource. 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
12. For Retry policy and dead-letter queue, under Retry policy:
   a. For Maximum age of event, enter a value between 1 minute (00:01) and 24 hours (24:00).
   b. For Retry attempts, enter a number between 0 and 185.
13. For Dead-letter queue, choose whether to use a standard Amazon SQS queue as a dead-letter queue. EventBridge sends events that match this rule to the dead-letter queue if they are not successfully delivered to the target. Do one of the following:
   - Choose None to not use a dead-letter queue.
Choose **Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue** and then select the queue to use from the drop-down list.

Choose **Select an Amazon SQS queue in another AWS account as a dead-letter queue** and then enter the ARN of the queue to use. You must attach a resource-based policy to the queue that grants EventBridge permission to send messages to it. To learn more, see [Granting permissions to the dead-letter queue](p. 36).

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 [Amazon EventBridge SaaS tagging](p. 216).

16. Choose **Create**.
Amazon EventBridge security

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.

Topics
- Data protection in Amazon EventBridge (p. 158)
- Tag-based policies (p. 159)
- Amazon EventBridge Identity and Access Management (p. 160)
- Logging and monitoring in Amazon EventBridge (p. 195)
- Compliance validation in Amazon EventBridge (p. 198)
- Amazon EventBridge resilience (p. 199)
- Infrastructure security in Amazon EventBridge (p. 200)
- Configuration and vulnerability analysis in Amazon EventBridge (p. 201)
Data protection in Amazon EventBridge

The AWS shared responsibility model applies to data protection in Amazon EventBridge. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the AWS Security Blog.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM). That way each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- 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.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put 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.

Encryption at Rest

Event metadata and payload data stored in EventBridge is fully encrypted at rest. By default, EventBridge encrypts data using 256-bit Advanced Encryption Standard (AES-256) under an AWS owned CMK, which helps secure your data from unauthorized access. Encryption at rest using the AWS owned CMK is provided at no additional charge.

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"
         ],
         "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.

- Amazon EventBridge SaaS tagging (p. 216)
- Controlling Access Using IAM Tags
Amazon EventBridge Identity and Access Management

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. 160)
- Access Control (p. 161)

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 Amazon EventBridge Resources (p. 162)
• Using identity-based policies (IAM policies) for Amazon EventBridge (p. 166)
• Using resource-based policies for Amazon EventBridge (p. 174)
• Resource-based policies for Amazon EventBridge schemas (p. 178)
• Amazon EventBridge permissions reference (p. 181)
Overview of Managing Access Permissions to Your Amazon EventBridge Resources

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.

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>ARN Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archive</td>
<td>arn:aws:events:region:account:archive/archive-name</td>
</tr>
<tr>
<td>Replay</td>
<td>arn:aws:events:region:account:replay/replay-name</td>
</tr>
<tr>
<td>Rule</td>
<td>arn:aws:events:region:account:rule/[event-bus-name]/rule-name</td>
</tr>
<tr>
<td>Event bus</td>
<td>arn:aws:events:region:account:event-bus/event-bus-name</td>
</tr>
<tr>
<td>All EventBridge resources</td>
<td>arn:aws:events:*</td>
</tr>
<tr>
<td>All EventBridge resources owned by the specified account in the specified Region</td>
<td>arn:aws:events:region:account:*</td>
</tr>
</tbody>
</table>

**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.

"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.

"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.

"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.

"Resource": ["arn1", "arn2"]
EventBridge provides a set of operations to work with the EventBridge resources. For a list of available operations, see Amazon EventBridge permissions reference (p. 181).

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. 163)
- Resource-Based Policies (IAM Policies) (p. 164)

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 Amazon EventBridge Resources (p. 162).

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 Amazon EventBridge (p. 174).

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. 162) and Amazon EventBridge permissions reference (p. 181).

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. 162).
- **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 Amazon EventBridge permissions reference (p. 181).

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 IAM policy conditions for fine-grained access control (p. 183).
Using identity-based policies (IAM policies) for Amazon 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 sections in this topic cover the following:

Topics

- Permissions required to use the EventBridge console (p. 166)
- AWS Managed (Predefined) Policies for EventBridge (p. 167)
- Permissions Required for EventBridge to Access Certain Targets (p. 169)
- Customer Managed Policy Examples (p. 170)

Permissions required to use the EventBridge console

For a user to work with EventBridge in the 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 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 AmazonEventBridgeReadOnlyAccess managed policy to the user, as described in AWS Managed (Predefined) Policies for EventBridge (p. 167).

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

166
• 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:

• **AmazonEventBridgeFullAccess** – Grants full access to EventBridge
• **AmazonEventBridgeReadOnlyAccess** – Grants read-only access to EventBridge

AmazonEventBridgeFullAccess policy

This policy was updated on March 4th, 2021 to include `iam:CreateServiceLinkedRole` and AWS Secrets Manager permissions necessary for using API destinations.

The AmazonEventBridgeFullAccess policy grants permissions to use all EventBridge actions, as well as the following permissions:

• `iam:CreateServiceLinkedRole` - This permissions is used to create the service role in your account used for API destinations. This permissions grants only the IAM service permissions to create a role in your account specifically for API destinations.
• `iam:PassRole` - This permission is used so that you can pass to EventBridge an Invocation role to Invoke Rule Targets.
- **Secrets Manager permissions** - These permissions are used to manage secrets in your account when you provide credentials via the Connection resource to authorize Api Destinations

The AmazonEventBridgeFullAccess policy:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "events:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/AmazonEventBridgeApiDestinationsServiceRolePolicy",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "apidestinations.events.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:CreateSecret",
                "secretsmanager:UpdateSecret",
                "secretsmanager:DeleteSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager::*:secret:events:*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "events.amazonaws.com"
                }
            }
        }
    ]
}
```

**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**

1. Open the IAM console at https://console.aws.amazon.com/iam/.
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.
   - 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 AmazonEventBridgeFullAccess.
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": [
        "arn:aws:ec2:{{region}}:{{accountId}}:instance/[[instanceIds]]",
        "arn:aws:ssm:{{region}}:*:document/{{documentName}}"
      ]
    }
  ]
}
```

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.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "ssm:SendCommand",
      "Effect": "Allow",
      "Resource": [
        "arn:aws:ec2:{{region}}:{{accountId}}:instance/*"
      ],
      "Condition": {
        "StringEquals": {
          "ec2:ResourceTag/*": [
            "[[tagValues]]"
          ]
        }
      }
    }
  ]
}
```
Use identity-based policies (IAM policies)

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

```json
{
  "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.

```json
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [ "ecs:RunTask" ],
    "Resource": [ "arn:aws:ecs:*:{account-id}:task-definition/{{task-definition-name}}" ],
    "Condition": {
      "ArnLike": {
        "ecs:cluster": "arn:aws:ecs:*:{account-id}:cluster/{{cluster-name}}"
      }
    }
  }
}
```

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: Access to Amazon EC2 targets (p. 171)
Example 1: Access to Amazon EC2 targets

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": "TargetInvocationAccess",
      "Effect": "Allow",
      "Action": [ 
        "ec2:Describe*",
        "ec2:RebootInstances",
        "ec2:StopInstances",
        "ec2:TerminateInstances",
        "ec2:CreateSnapshot"
      ],
      "Resource": "*"
    }
  ]
}
```

Example 2: Kinesis

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

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

Example 3: Console access

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

```json
{
  "Version": "2012-10-17",
  "Statement": [
    { 
      "Sid": "ConsoleAccess",
    }
  ]
}
```
Use identity-based policies (IAM policies)

Example 4: EventBridgeFullAccess

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

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

Example 5: ReadOnlyAccess

The following policy provides read-only access to EventBridge.

```json
{
  "Version": "2012-10-17",
  "Statement": [  
    {
      "Sid": "FullAccess",
      "Effect": "Allow",
      "Action": "events:*",
      "Resource": "*
    },
    {
      "Sid": "IAMPassRole",
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::*:role/AWS_Events_Invoke_Targets"
    }
  ]
}
```
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 Amazon EventBridge SaaS tagging (p. 216).

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.

```
{
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "events:*",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "aws:ResourceTag/Stack": "Prod"
            }
         }
      }
   ]
}
```

For more information about using IAM policy statements, see Controlling Access Using Policies in the IAM User Guide.
Using resource-based policies for Amazon EventBridge

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. 174)
- Amazon SNS Permissions (p. 175)
- Amazon SQS Permissions (p. 176)
- CloudWatch Logs Permissions (p. 177)

**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": "InvokeLambdaFunction"
}
```

To add permissions that enable EventBridge to invoke Lambda functions

- At a command prompt, enter the following command.

```
aws lambda add-permission --statement-id "InvokeLambdaFunction" \
 --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 `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.

   ```bash
   ```

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

   ```json
   {
     "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",
       "SubscriptionsPending": "0"
     }
   }
   ```

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

   ```json
   {"Sid": "PublishEventsToMyTopic",
   "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": "PublishEventsToMyTopic", "Effect": "Allow", "Principal": {
   ```

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.
Use resource-based policies

```bash
aws sns set-topic-attributes --topic-arn "arn:aws:sns:region:account-id:topic-name" \
--attribute-name Policy \
"
```

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

**Amazon SQS Permissions**

To allow a 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": "EventsToMyQueue",  
   "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
   ```

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

```
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.

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

For more information, see Controlling Access to Resources in the IAM User Guide.
Resource-based policies for Amazon EventBridge schemas

The EventBridge Schema Registry supports resource-based policies. A resource policy is a policy that is attached to a resource rather than to an IAM identity. For example, in Amazon Simple Storage Service (Amazon S3), a resource policy is attached to an Amazon S3 bucket.

For more information on EventBridge Schemas and resource-based policies, see the following.

- Amazon EventBridge Schemas REST API Reference
- Identity-Based Policies and Resource-Based Policies in the IAM User Guide

Supported APIs for Resource-based Policies

The following APIs can be used with resource-based policies for the EventBridge Schema Registry.

- DescribeRegistry
- UpdateRegistry
- DeleteRegistry
- ListSchemas
- SearchSchemas
- DescribeSchema
- CreateSchema
- DeleteSchema
- UpdateSchema
- ListSchemaVersions
- DeleteSchemaVersion
- DescribeCodeBinding
- GetCodeBindingSource
- PutCodeBinding

Example Policy Granting All Supported actions to an AWS Account

For the EventBridge Schema Registry, a resource-based policy is always attached to a registry. To enable access to a schema, you must specify the schema ARN as well as the registry ARN in the policy.

To grant a user access to all available APIs for EventBridge Schemas, use a policy similar to the following, where the "Principal" is the account ID of the account you want to grant access.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Test",
      "Effect": "Allow",
      "Action": [
        "schemas:*"
      ],
      "Principal": {
        "AWS": [
        ...
```
Resource-based policies for EventBridge schemas

Example Policy Granting Read-only Actions to an AWS Account

The following example grants access to an account for only the read-only APIs for EventBridge Schemas.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Test",
            "Effect": "Allow",
            "Action": [
                "schemas:DescribeRegistry",
                "schemas:ListSchemas",
                "schemas:SearchSchemas",
                "schemas:DescribeSchema",
                "schemas:ListSchemaVersions",
                "schemas:DescribeCodeBinding",
                "schemas:GetCodeBindingSource"
            ],
            "Principal": {
                "AWS": [
                    "109876543210"
                ]
            },
            "Resource": [
                "arn:aws:schemas:us-east-1:012345678901:registry/default",
                "arn:aws:schemas:us-east-1:012345678901:schema/default*"
            ]
        }
    ]
}
```

Example Policy Granting All Actions to an Organization

You can also use resource-based policies with the EventBridge Schema Registry to grant access to an organization ID. For more information, see the AWS Organizations User Guide.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Test",
            "Effect": "Allow",
            "Action": [
                "schemas:*"
            ],
            "Principal": "*",
            "Resource": [
                "arn:aws:schemas:us-east-1:012345678901:registry/default",
                "arn:aws:schemas:us-east-1:012345678901:schema/default*"
            ]
        }
    ]
}
```
"Condition": {
  "StringEquals": {
    "aws:PrincipalOrgID": [
      "o-a1b2c3d4e5"
    ]
  }
}
]
Amazon EventBridge permissions reference

When you are setting up Access Control (p. 161) 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>
<tr>
<td>ListRuleNamesByTarget</td>
<td>events:ListRuleNamesByTarget</td>
</tr>
<tr>
<td></td>
<td>Required to list rules associated with a target.</td>
</tr>
<tr>
<td>EventBridge API Operations</td>
<td>Required Permissions (API Actions)</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>ListRules</td>
<td>events:ListRules</td>
</tr>
<tr>
<td></td>
<td>Required to list all rules in your account.</td>
</tr>
<tr>
<td>ListTagsForResource</td>
<td>events:ListTagsForResource</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>Required to list all targets associated with a rule.</td>
</tr>
<tr>
<td>PutEvents</td>
<td>events:PutEvents</td>
</tr>
<tr>
<td></td>
<td>Required to add custom events that can be matched to rules.</td>
</tr>
<tr>
<td>PutPermission</td>
<td>events:PutPermission</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>Required to create or update a rule.</td>
</tr>
<tr>
<td>PutTargets</td>
<td>events:PutTargets</td>
</tr>
<tr>
<td></td>
<td>Required to add targets to a rule.</td>
</tr>
<tr>
<td>RemovePermission</td>
<td>events:RemovePermission</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>Required to remove a target from a rule.</td>
</tr>
<tr>
<td>TestEventPattern</td>
<td>events:TestEventPattern</td>
</tr>
<tr>
<td></td>
<td>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 PutRule API action. If you allow an IAM user or role for the 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 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 events:source and 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 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 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 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>events:creatorAccount</td>
<td>&quot;events:creatorAccount&quot;: &quot;creatorAccount&quot;</td>
<td>creatorAccount, Null</td>
</tr>
<tr>
<td></td>
<td>Where creatorAccount is the account ID for the account that the rule was created in. Use this condition to authorize API calls on rules from a specific account.</td>
<td></td>
</tr>
<tr>
<td>events:eventBusInvocation</td>
<td>&quot;events:eventBusInvocation&quot;: &quot;true&quot;</td>
<td>eventBusInvocation, Null</td>
</tr>
<tr>
<td></td>
<td>Where boolean is true when a rule sends an event to a target that is an event bus in another account. The value is false when a PutEvents API call is used.</td>
<td></td>
</tr>
</tbody>
</table>
## Use conditions

<table>
<thead>
<tr>
<th>Condition Key</th>
<th>Key/Value Pair</th>
<th>Evaluation Types</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>events:source</code></td>
<td>&quot;events:source&quot;: &quot;source&quot;</td>
<td>Source, Null</td>
</tr>
<tr>
<td></td>
<td>Where <em>source</em> is the literal string for the <code>source</code> field of the event such as &quot;aws.ec2&quot; and &quot;aws.s3&quot;. To see more possible values for <em>source</em>, see the example events in <a href="#">Events from AWS services (p. 74)</a>.</td>
<td></td>
</tr>
<tr>
<td><code>events:detail-type</code></td>
<td>&quot;events:detail-type&quot;: &quot;detail-type&quot;</td>
<td>Detail Type, Null</td>
</tr>
<tr>
<td></td>
<td>Where <em>detail-type</em> is the literal string for the <code>detail-type</code> field of the event such as &quot;AWS API Call via CloudTrail&quot; and &quot;EC2 Instance State-change Notification&quot;. To see more possible values for <em>detail-type</em>, see the example events in <a href="#">Events from AWS services (p. 74)</a>.</td>
<td></td>
</tr>
<tr>
<td><code>events:detail.userIdentity.principalId</code></td>
<td>&quot;events:detail.userIdentity.principalId&quot;: &quot;principal-id&quot;</td>
<td>Principal Id, Null</td>
</tr>
<tr>
<td></td>
<td>Where <em>principal-id</em> is the literal string for the <code>detail.userIdentity.principalId</code> field of the event with <code>detail-type</code> &quot;AWS API Call via CloudTrail&quot; such as &quot;AROAIDPPEZS3SZEXAMPLE:AssumedRoleSessionName&quot;.</td>
<td></td>
</tr>
<tr>
<td><code>events:detail.service</code></td>
<td>&quot;events:detail.service&quot;: &quot;service&quot;</td>
<td>service, Null</td>
</tr>
<tr>
<td></td>
<td>Where <em>service</em> is the literal string for the <code>detail.service</code> field of the event, such as &quot;ABUSE&quot;.</td>
<td></td>
</tr>
<tr>
<td><code>events:detail.eventTypeCode</code></td>
<td>&quot;events:detail.eventTypeCode&quot;: &quot;eventTypeCode&quot;</td>
<td><code>eventTypeCode</code>, Null</td>
</tr>
<tr>
<td></td>
<td>Where <em>eventTypeCode</em> is the literal string for the <code>detail.eventTypeCode</code> field of the event, such as &quot;AWS_ABUSE_DOS_REPORT&quot;.</td>
<td></td>
</tr>
<tr>
<td><code>events:TargetArn</code></td>
<td>&quot;events:TargetArn&quot;: &quot;target-arn&quot;</td>
<td>ARN, Null</td>
</tr>
<tr>
<td></td>
<td>Where <em>target-arn</em> is the ARN of the target that can be put to a rule such as &quot;arn:aws:lambda::<em>:</em>:function:*&quot;.</td>
<td></td>
</tr>
</tbody>
</table>
Use conditions

<table>
<thead>
<tr>
<th>Condition Key</th>
<th>Key/Value Pair</th>
<th>Evaluation Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>events:ManagedBy</td>
<td>Used internally by AWS services. If a rule is created by an AWS service on your behalf, the value is the principal name of the service that created the rule.</td>
<td>Not intended for use in customer policies.</td>
</tr>
<tr>
<td>aws:SourceArn</td>
<td>The ARN of the rule that is sending the event.</td>
<td>ARN, Null</td>
</tr>
<tr>
<td>aws:SourceAccount</td>
<td>The account in which the rule specified by aws:SourceArn exists.</td>
<td>Account Id, Null</td>
</tr>
</tbody>
</table>

For example policy statements for EventBridge, see Overview of Managing Access Permissions to Your Amazon EventBridge Resources (p. 162).

Topics

- Example: Using the creatorAccount condition (p. 185)
- Example: Using the eventBusInvocation condition (p. 186)
- Example: Limit Access to a Specific Source (p. 186)
- Example: Define Multiple Sources That Can Be Used in an Event Pattern Individually (p. 188)
- Example: Define a Source and a DetailType That Can Be Used in an Event Pattern (p. 189)
- Example: Ensure That the Source Is Defined in the Event Pattern (p. 190)
- Example: Define a List of Allowed Sources in an Event Pattern with Multiple Sources (p. 190)
- Example: Limiting PutRule Access by detail.service (p. 191)
- Example: Limiting PutRule Access by detail.eventTypeCode (p. 192)
- Example: Ensure That AWS CloudTrail Events for API Calls from a Certain PrincipalId Are Consumed (p. 192)
- Example: Limiting Access to Targets (p. 193)

Example: Using the creatorAccount condition

The following example policy statement demonstrates how to use the creatorAccount condition in a policy to filter API actions on rules to allow requests only if the account specified as the creatorAccount is the account that created the rule.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPutRuleForOwnedRules",
      "Effect": "Allow",
      "Action": "events:PutRule",
      "Resource": "*",
      "Condition": {
        "StringEqualsIfExists": {
          "events:creatorAccount": "${aws:PrincipalAccount}" 
        }
      }
    }
  ]
}
```
Example: Using the `eventBusInvocation` condition

The `eventBusInvocation` indicates whether the invocation originates from a cross-account target or a `PutEvents` API request. The value is `true` when the invocation results from a rule that include a cross-account target, such as when the target is an event bus in another account. The value is `false` when the invocation results from a `PutEvents` API request.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCrossAccountInvocationEventsOnly",
            "Effect": "Allow",
            "Action": "events:PutEvents",
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "events:eventBusInvocation": "true"
                }
            }
        }
    ]
}
```

Example: 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 all events**

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

**Policy B:—allow events only from Amazon EC2**

```json
{
    "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.

```json
{
    "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".

```json
{
    "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><code>{ &quot;source&quot;: [ &quot;aws.ec2&quot; ] }</code></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><code>{ &quot;source&quot;: [ &quot;aws.ec2&quot;, &quot;aws.s3&quot; ] }</code></td>
<td>Yes</td>
<td>No (Source aws.s3 is not allowed)</td>
</tr>
<tr>
<td><code>{ &quot;source&quot;: [ &quot;aws.ec2&quot; ], &quot;detail-type&quot;: [ &quot;EC2 Instance State-change Notification&quot; ] }</code></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><code>{ &quot;detail-type&quot;: [ &quot;EC2 Instance State-change Notification&quot; ] }</code></td>
<td>Yes</td>
<td>No (Source must be specified)</td>
</tr>
</tbody>
</table>
Example: 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.

<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;source&quot;: [ &quot;aws.ecs&quot; ] }</td>
<td>Yes</td>
</tr>
<tr>
<td>{ &quot;source&quot;: [ &quot;aws.s3&quot; ] }</td>
<td>No</td>
</tr>
<tr>
<td>{ &quot;source&quot;: [ &quot;aws.ec2&quot;, &quot;aws.ecs&quot; ] }</td>
<td>No</td>
</tr>
<tr>
<td>{ &quot;detail-type&quot;: [ &quot;AWS API Call via CloudTrail&quot; ] }</td>
<td>No</td>
</tr>
</tbody>
</table>
Example: 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 DetailType 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.

<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>No</td>
</tr>
<tr>
<td>`{ &quot;source&quot;: [ &quot;aws.ecs&quot; ] }</td>
<td>No</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>
</tr>
<tr>
<td>`{ &quot;source&quot;: [ &quot;aws.ec2&quot; ], &quot;detail-type&quot;: [ &quot;EC2 Instance Health Failed&quot; ] }</td>
<td>No</td>
</tr>
<tr>
<td>`{ &quot;detail-type&quot;: [ &quot;EC2 Instance State-change Notification&quot; ] }</td>
<td>No</td>
</tr>
</tbody>
</table>
Example: 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.

```
{
  "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.

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

Example: 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.

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

```json
{
  "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>`{ &quot;source&quot;: [ &quot;aws.ec2&quot;, &quot;aws.s3&quot; ] }</td>
<td>Yes</td>
</tr>
<tr>
<td>`{ &quot;source&quot;: [ &quot;aws.ec2&quot;, &quot;aws.autoscaling&quot; ] }</td>
<td>No</td>
</tr>
<tr>
<td>`{ &quot;detail-type&quot;: [ &quot;EC2 Instance State-change Notification&quot; ] }</td>
<td>No</td>
</tr>
</tbody>
</table>

**Example: 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": [  
```
Use conditions

Example: 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`.

```
{  
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPutRuleEventsWithDetailServiceEC2",
      "Effect": "Allow",
      "Action": "events:PutRule",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "events:detail.service": "ABUSE"
        }
      }
    }
  ]
}
```

Example: 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.

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "AllowPutRuleOnlyForCloudTrailEventsWhereUserIsASpecificIAMUser",
    "Effect": "Allow",
    "Action": "events:PutRule",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "events:detail.eventTypeCode": "AWS_ABUSE_DOS_REPORT"
      }
    }
  }
]
```
"events:detail.userIdentity.principalId": [ "AIDAJ45Q7YFFAREXAMPLE" ]

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><code>{ &quot;detail-type&quot;: [ &quot;AWS API Call via CloudTrail&quot; ] }</code></td>
<td>No</td>
</tr>
</tbody>
</table>
| `{ "detail-type": [ "AWS API Call via CloudTrail" ],
  "detail.userIdentity.principalId": [ "AIDAJ45Q7YFFAREXAMPLE" ] }`         | Yes                   |
| `{ "detail-type": [ "AWS API Call via CloudTrail" ],
  "detail.userIdentity.principalId": [ "AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName" ] }` | No                    |

**Example: 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": [
```


```json
{
  "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. 195)
- Example: EventBridge Log File Entries (p. 196)

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
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":"
    },
    "responseElements":{
        "ruleArn":"arn:aws:events:us-east-1:123456789012:rule/cttest2"
    },
    "requestID":"e9caf887-8d88-11e5-a331-3332aa445952",
    "eventID":"49d14f36-6450-44a5-a501-b0f2dcdaeb98",
    "eventType":"AwsApiCall",
    "apiVersion":"2015-10-07",
```
"recipientAccountId":"123456789012"
Compliance validation in Amazon EventBridge

Third-party auditors assess the security and compliance of 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 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.
Amazon EventBridge resilience

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.
Amazon EventBridge monitoring

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. Valid Dimensions: RuleName 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. Valid Dimensions: RuleName 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. Note: EventBridge only sends this metric to CloudWatch if it has a non-zero value. Valid Dimensions: RuleName Units: Count</td>
</tr>
<tr>
<td>InvocationsFailedToBeSentToDlq</td>
<td>Measures the number of invocations that couldn’t be moved to a dead-letter queue. Dead-letter errors can occur due to permissions errors, unavailable resources, or size limits. Note: EventBridge only sends this metric to CloudWatch if it has a non-zero value. Valid Dimensions: RuleName Units: Count</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InvocationsSentToDlq</td>
<td>Measures the number of invocations that are moved to a dead-letter queue.</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>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>
<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>
</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 troubleshooting

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. 204)
- I have just created/modified a rule but it did not match a test event (p. 205)
- My rule did not self-trigger at the time specified in the ScheduleExpression (p. 206)
- My rule did not trigger at the time that I expected (p. 206)
- My rule matches IAM API calls but my rule was not triggered (p. 206)
- My rule is not working because the IAM role associated with the rule is ignored when the rule is triggered (p. 206)
- 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. 207)
- My event's delivery to the target experienced a delay (p. 207)
- Some events were never delivered to my target (p. 207)
- 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. 207)
- Preventing Infinite Loops (p. 208)
- My events are not delivered to the target Amazon SQS queue (p. 208)
- My rule is being triggered but I don't see any messages published into my Amazon SNS topic (p. 208)
- My Amazon SNS topic still has permissions for EventBridge even after I deleted the rule associated with the Amazon SNS topic (p. 210)
- Which IAM condition keys can I use with EventBridge (p. 210)
- How can I tell when EventBridge rules are broken (p. 210)

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"}},
    "Action":"lambda:InvokeFunction",
```

204
I have just created/modified a rule but it did not match a test event

```json
{"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-1:123456789012: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 rule is correctly set. For more information, see `TestEventPattern` in the Amazon CloudWatch Events API Reference.
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. 205).

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 Events from AWS services (p. 74).

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.
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.

```
{
  "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):

```
```

You should see policy attributes similar to the following:

```
{""Version":"2012-10-17",""Id":"__default_policy_ID",""Statement":[{""Sid":"__default_statement_ID",""Effect":"Allow"}],""Resource":"
```
My rule is being triggered but I don’t see any messages published into my Amazon SNS topic

```json
{"Principal":{"AWS":"*"},
"Condition": {"StringEquals":{"AWS:SourceOwner": "123456789012"}},
}```

If you see a policy similar to the following, you have only the default policy set:

```json
{"Version": "2008-10-17", 
"Id": "__default_policy_ID",
"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:

```json
```

The new policy should look like the one described earlier.

Set topic attributes with the AWS CLI:

```
```

**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. EventBridge sets the correct permissions on the target.
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 Amazon EventBridge (p. 174).

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. 183).

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.
# Amazon EventBridge SaaS quotas

**Topics**
- [PutEvents Quotas by Region](#)
- [Invocation Quotas by Region](#)

EventBridge has the following quotas.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Publishing API requests</td>
<td>PutEvents operations are limited based on AWS Region. See <a href="#">PutEvents Quotas by Region</a>.</td>
</tr>
<tr>
<td></td>
<td>The Service Quotas console provides information about EventBridge quotas. Along with viewing the default quotas, you can use the Service Quotas console to request quota increases for adjustable quotas.</td>
</tr>
<tr>
<td>All other API requests</td>
<td>All EventBridge API other than PutEvents are limited to 50 requests per second by default.</td>
</tr>
<tr>
<td></td>
<td>The Service Quotas console provides information about EventBridge quotas. Along with viewing the default quotas, you can use the Service Quotas console to request quota increases for adjustable quotas.</td>
</tr>
<tr>
<td>Event buses</td>
<td>Up to 100 event buses per account.</td>
</tr>
<tr>
<td>Event buses - other quotas</td>
<td>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. 213) apply. Any events that are sent on to the targets of the rules in your account count against your invocations quota. This includes cross-account and cross-Region event delivery.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>The Service Quotas console provides information about EventBridge quotas. Along with viewing the default quotas, you can use the Service Quotas console to request quota increases for adjustable quotas.</td>
</tr>
<tr>
<td>Event pattern</td>
<td>2048 characters maximum.</td>
</tr>
<tr>
<td>Invocations</td>
<td>An invocation is an event matching a rule and being sent on to the rule's targets. Quotas vary by region.</td>
</tr>
<tr>
<td></td>
<td>See <a href="#">Invocation Quotas by Region</a>.</td>
</tr>
</tbody>
</table>
|                                   | The Service Quotas console provides information about EventBridge quotas. Along with viewing the default quotas,
<table>
<thead>
<tr>
<th>Resource</th>
<th>Default Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rate of invocations per API destination</strong></td>
<td>Default: 300 TPS. The maximum number of invocations per second to send to each API destination endpoint per account per Region. Once the quota is met, future invocations to that API endpoint are throttled. The invocations will still occur, but are delayed. The Service Quotas console provides information about EventBridge quotas. Along with viewing the default quotas, you can use the Service Quotas console to request quota increases for adjustable quotas.</td>
</tr>
<tr>
<td><strong>Connections</strong></td>
<td>Default: 3,000 The maximum number of connections per account per Region. Connections are used with API destination to define the authorization parameters to use to connect to the destination HTTP endpoint. The Service Quotas console provides information about EventBridge quotas. Along with viewing the default quotas, you can use the Service Quotas console to request quota increases for adjustable quotas.</td>
</tr>
<tr>
<td><strong>API destinations</strong></td>
<td>Default: 3,000 The maximum number of API destinations per account per Region. API destinations are HTTP invocation endpoints you can use as the target for a rule. The Service Quotas console provides information about EventBridge quotas. Along with viewing the default quotas, you can use the Service Quotas console to request quota increases for adjustable quotas.</td>
</tr>
<tr>
<td><strong>ListRuleNamesByTarget</strong></td>
<td>Up to 100 results per page for requests.</td>
</tr>
<tr>
<td><strong>ListRules</strong></td>
<td>Up to 100 results per page for requests.</td>
</tr>
<tr>
<td><strong>ListTargetsByRule</strong></td>
<td>Up to 100 results per page for requests.</td>
</tr>
<tr>
<td><strong>PutTargets</strong></td>
<td>10 entries per request. Up to 5 targets per rule.</td>
</tr>
<tr>
<td><strong>RemoveTargets</strong></td>
<td>10 entries per request.</td>
</tr>
</tbody>
</table>
## PutEvents Quotas by Region

The Service Quotas console provides information about EventBridge quotas. Along with viewing the default quotas, you can use the Service Quotas console to request quota increases for adjustable quotas.

For quotas above 100,000 TPS, our service team will setup a call to best support your use case. Note that events sent to a different Region using PutEvents count towards the PutEvents quota of the destination Region for the account that owns the role used to send the events.

<table>
<thead>
<tr>
<th>Regions</th>
<th>Transactions per Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>• US East (N. Virginia)</td>
<td>PutEvents has a soft limit of 10,000 requests per second by</td>
</tr>
<tr>
<td>• US West (Oregon)</td>
<td>default in this Region.</td>
</tr>
<tr>
<td>• Europe (Ireland)</td>
<td></td>
</tr>
<tr>
<td>• US East (Ohio)</td>
<td>PutEvents has a soft limit of 2,400 requests per second by</td>
</tr>
<tr>
<td>• Europe (Frankfurt)</td>
<td>default in these Regions.</td>
</tr>
<tr>
<td>• US West (N. California)</td>
<td>PutEvents has a soft limit of 1,200 requests per second by</td>
</tr>
<tr>
<td>• Europe (London)</td>
<td>default in these Regions.</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>PutEvents has a soft limit of 600 requests per second by</td>
</tr>
<tr>
<td>• Europe (Paris)</td>
<td>default in these Regions.</td>
</tr>
<tr>
<td>• Europe (Stockholm)</td>
<td></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>
</tbody>
</table>

### Resource

<table>
<thead>
<tr>
<th>Default Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rules 300 per event bus.</td>
</tr>
<tr>
<td>The Service Quotas console provides information about EventBridge quotas.</td>
</tr>
<tr>
<td>Along with viewing the default quotas, you can use the Service Quotas console</td>
</tr>
<tr>
<td>Before increasing quotas, examine your rules. You may have multiple rules</td>
</tr>
<tr>
<td>each matching to very specific events. Consider broadening their scope by</td>
</tr>
<tr>
<td>using fewer identifiers in your Amazon EventBridge events (p. 16). In</td>
</tr>
<tr>
<td>addition, a rule can invoke several targets each time it matches an event.</td>
</tr>
<tr>
<td>Consider adding more targets to your rules.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Systems Manager Run Command target</th>
<th>1 target key and 1 target value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Systems Manager Run Command does not currently support multiple target values.</td>
<td></td>
</tr>
</tbody>
</table>

### Targets

Up to 5 targets per rule.
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.

The Service Quotas console provides information about EventBridge quotas. Along with viewing the default quotas, you can use the Service Quotas console to request quota increases for adjustable quotas. For quotas above 100,000 TPS, our service team will setup a call to best support your use case.

<table>
<thead>
<tr>
<th>Regions</th>
<th>Transactions per Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Middle East (Bahrain)</td>
<td></td>
</tr>
<tr>
<td>• China (Ningxia)</td>
<td></td>
</tr>
<tr>
<td>• China (Beijing)</td>
<td></td>
</tr>
<tr>
<td>• Asia Pacific (Osaka)</td>
<td>PutEvents has a soft limit of 400 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• Africa (Cape Town)</td>
<td></td>
</tr>
<tr>
<td>• Europe (Milan)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Regions</th>
<th>Invocations per Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>• US East (N. Virginia)</td>
<td>The invocations quota has a soft limit of 18,750 requests per second by default in this Region.</td>
</tr>
<tr>
<td>• US West (Oregon)</td>
<td></td>
</tr>
<tr>
<td>• Europe (Ireland)</td>
<td></td>
</tr>
<tr>
<td>• US East (Ohio)</td>
<td>The invocations quota has a soft limit of 4,500 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• Europe (Frankfurt)</td>
<td></td>
</tr>
<tr>
<td>• US West (N. California)</td>
<td>The invocations quota has a soft limit of 2,250 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></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 invocations quota has a soft limit of 1,100 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• South America (São Paulo)</td>
<td></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>
</tbody>
</table>
## Regions

<table>
<thead>
<tr>
<th>Regions</th>
<th>Invocations per Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>China (Ningxia)</td>
<td></td>
</tr>
<tr>
<td>China (Beijing)</td>
<td></td>
</tr>
<tr>
<td>Asia Pacific (Osaka)</td>
<td></td>
</tr>
<tr>
<td>Africa (Cape Town)</td>
<td></td>
</tr>
<tr>
<td>Europe (Milan)</td>
<td></td>
</tr>
</tbody>
</table>

The invocations quota has a soft limit of 750 requests per second by default in these Regions.
Amazon EventBridge SaaS tagging

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.

**Topics**

- Supported Resources in EventBridge (p. 216)
- Managing Tags (p. 216)
- Tag Naming and Usage Conventions (p. 217)

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. 216).

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.
## 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>
</table>
| Added support for archives and event replays. | Amazon EventBridge now supports using archives to store events, and event replays to replay the events from an archive. To learn more, see the following:  
  - Archive Amazon EventBridge events (p. 62).  
  - CreateArchive  
  - StartReplay | November 05, 2020 |
| Added support for dead-letter queues and retry policy for targets. | Amazon EventBridge now supports using dead-letter queues and defining a retry policy for targets. To learn more, see the following:  
  - Event retry policy and using dead-letter queues (p. 35).  
  - PutTargets | October 12, 2020 |
| Added support for JSONSchema Draft4 format schemas. | Amazon EventBridge now supports schemas in JSONSchema Draft 4 format. You can also now export schemas using the EventBridge API. To learn more, see the following.  
  - Amazon EventBridge schemas (p. 65)  
  - Export in the EventBridge Schema Registry API Reference. | September 28, 2020 |
| Resource-based policies for the EventBridge Schema Registry | The Amazon EventBridge Schema Registry now supports resource-based policies. For more information, see the following.  
  - Resource-based policies for Amazon EventBridge schemas (p. 178)  
  - Policy in the EventBridge Schema Registry API Reference  
  - RegistryPolicy Resource Type in the AWS CloudFormation User Guide | April 30, 2020 |
| Tags for Event Buses | 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.  
  - Amazon EventBridge SaaS tagging (p. 216)  
  - Tag-based policies (p. 159)  
  - TagResource  
  - UntagResource  
  - ListTagsForResource | February 24, 2020 |
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increased service quotas</td>
<td>Amazon EventBridge has increased quotas for invocations and for PutEvents. Quotas vary by region, and can be increased if necessary.</td>
<td>February 11, 2020</td>
</tr>
<tr>
<td></td>
<td>• Amazon EventBridge SaaS quotas (p. 211)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PutEvents Quotas by Region (p. 213)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Invocation Quotas by Region (p. 214)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Request a Quota Increase</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.</td>
<td>December 20, 2019</td>
</tr>
<tr>
<td></td>
<td>• Transforming Amazon EventBridge target input (p. 58)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Use Input Transformer to extract data from an event and input that data to the target</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Tutorial: Use Input Transformer to Customize What Is Passed to the Event Target (p. 142)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Added a link to Application Auto Scaling Events.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Application Auto Scaling Events and EventBridge</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Events from AWS services (p. 74)</td>
<td></td>
</tr>
<tr>
<td>Content-based filtering</td>
<td></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></td>
<td>• Use Events in Amazon Augmented AI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Events from AWS services (p. 74)</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>
<tr>
<td></td>
<td>• Automating Amazon Chime with EventBridge</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Events from AWS services (p. 74)</td>
<td></td>
</tr>
<tr>
<td>Amazon EventBridge Schemas</td>
<td>You can now manage schemas and generate code bindings for events in Amazon EventBridge. For more information, see the following.</td>
<td>December 1, 2019</td>
</tr>
<tr>
<td></td>
<td>• Amazon EventBridge schemas (p. 65)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• EventBridge Schemas API Reference</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• EventSchemas Resource Type Reference in AWS CloudFormation</td>
<td></td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Release Date</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>AWS CloudFormation support for Event Buses</td>
<td>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.</td>
<td>October 7, 2019</td>
</tr>
<tr>
<td>New service</td>
<td>Initial release of Amazon EventBridge.</td>
<td>July 11, 2019</td>
</tr>
</tbody>
</table>