# Table of Contents

What Is Amazon EventBridge? ............................................................................................................................................ 1  
How it works .................................................................................................................................................................. 1  

Setup and prerequisites ............................................................................................................................................ 3  
  Sign up for Amazon Web Services (AWS) .................................................................................................................. 3  
  Sign in to the Amazon EventBridge console ............................................................................................................... 3  
  Account credentials .......................................................................................................................................................... 3  
  Set up the AWS Command Line Interface ................................................................................................................... 4  
  Regional Endpoints .......................................................................................................................................................... 4  

Getting started ......................................................................................................................................................... 5  
  Create rule ..................................................................................................................................................................... 5  

Event buses ................................................................................................................................................................. 7  
  Permissions for event buses .......................................................................................................................................... 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: Deny sending events from specific Regions .......................................................................................... 13  
  Creating an event bus ..................................................................................................................................................... 15  

Events ........................................................................................................................................................................... 16  
  Event patterns ............................................................................................................................................................... 19  
    Create event patterns .................................................................................................................................................. 20  
    Example events and event patterns ............................................................................................................................. 21  
    Null values and empty strings ...................................................................................................................................... 23  
    Arrays ........................................................................................................................................................................... 24  
    Content-based filtering ............................................................................................................................................... 25  
  Adding events with PutEvents ....................................................................................................................................... 28  
    Handling failures with PutEvents ................................................................................................................................ 29  
    Sending events using the AWS CLI .................................................................................................................................. 30  
    Calculating PutEvent event entry sizes ........................................................................................................................ 31  

Rules ........................................................................................................................................................................... 32  
  Creating a rule that runs when events are received ........................................................................................................ 34  
  Creating a rule that runs on a schedule .......................................................................................................................... 36  
    Cron Expressions .......................................................................................................................................................... 37  
    Rate Expressions ........................................................................................................................................................ 39  
    Create rule .................................................................................................................................................................. 40  
  Disabling or deleting a rule .............................................................................................................................................. 42  
  Using dead-letter queues .................................................................................................................................................. 43  
    Considerations for using a dead-letter queue ................................................................................................................ 44  
    Granting permissions to the dead-letter queue ............................................................................................................. 44  
    How to resend events from a dead-letter queue ........................................................................................................... 45  
  Using AWS SAM templates .............................................................................................................................................. 45  
    Combined template ....................................................................................................................................................... 45  
    Separated template ......................................................................................................................................................... 46  

Targets ......................................................................................................................................................................... 48  
  Targets available in the EventBridge console ................................................................................................................ 48  
  Target parameters ........................................................................................................................................................... 49  
  Permissions ....................................................................................................................................................................... 50  
  Configure targets ............................................................................................................................................................. 51  
    API destinations ............................................................................................................................................................ 52  
    API Gateway ............................................................................................................................................................... 61  
    Cross-account events .................................................................................................................................................... 62  
    Cross-Region events .................................................................................................................................................... 64
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same account events</td>
<td>66</td>
</tr>
<tr>
<td>Transforming target input</td>
<td>68</td>
</tr>
<tr>
<td>Predefined variables</td>
<td>68</td>
</tr>
<tr>
<td>Input Transform Examples</td>
<td>68</td>
</tr>
<tr>
<td>Transforming input by using the EventBridge API</td>
<td>70</td>
</tr>
<tr>
<td>Transforming input by using AWS CloudFormation</td>
<td>70</td>
</tr>
<tr>
<td>Common Issues with Transforming Input</td>
<td>70</td>
</tr>
<tr>
<td>Archive and replay</td>
<td>71</td>
</tr>
<tr>
<td>Archiving events</td>
<td>72</td>
</tr>
<tr>
<td>Replaying archived events</td>
<td>74</td>
</tr>
<tr>
<td>Schemas</td>
<td>75</td>
</tr>
<tr>
<td>Finding a schema</td>
<td>76</td>
</tr>
<tr>
<td>Schema registries</td>
<td>77</td>
</tr>
<tr>
<td>Creating a schema</td>
<td>78</td>
</tr>
<tr>
<td>Create a schema by using a template</td>
<td>78</td>
</tr>
<tr>
<td>Edit a schema template directly in the console</td>
<td>79</td>
</tr>
<tr>
<td>Create a schema from the JSON of an event</td>
<td>80</td>
</tr>
<tr>
<td>Create a schema from events on an event bus</td>
<td>82</td>
</tr>
<tr>
<td>Code bindings</td>
<td>83</td>
</tr>
<tr>
<td>Events from AWS services</td>
<td>84</td>
</tr>
<tr>
<td>Related AWS services</td>
<td>88</td>
</tr>
<tr>
<td>Interface VPC Endpoints</td>
<td>89</td>
</tr>
<tr>
<td>Availability</td>
<td>89</td>
</tr>
<tr>
<td>Creating a VPC Endpoint for EventBridge</td>
<td>90</td>
</tr>
<tr>
<td>AWS X-Ray</td>
<td>91</td>
</tr>
<tr>
<td>Receiving events from a SaaS partner</td>
<td>93</td>
</tr>
<tr>
<td>Supported SaaS partner integrations</td>
<td>93</td>
</tr>
<tr>
<td>Configuring EventBridge</td>
<td>95</td>
</tr>
<tr>
<td>Create a rule for SaaS partner events</td>
<td>95</td>
</tr>
<tr>
<td>Receiving events from Salesforce</td>
<td>97</td>
</tr>
<tr>
<td>Step 1: Configure Amazon AppFlow to use Salesforce as a partner event source</td>
<td>97</td>
</tr>
<tr>
<td>Step 2: Configure EventBridge to receive Salesforce events</td>
<td>98</td>
</tr>
<tr>
<td>Tutorials</td>
<td>100</td>
</tr>
<tr>
<td>Archive and replay events</td>
<td>101</td>
</tr>
<tr>
<td>Step 1: Create a Lambda function</td>
<td>101</td>
</tr>
<tr>
<td>Step 2: Create archive</td>
<td>101</td>
</tr>
<tr>
<td>Step 3: Create rule</td>
<td>102</td>
</tr>
<tr>
<td>Step 4: Send test events</td>
<td>102</td>
</tr>
<tr>
<td>Step 5: Replay events</td>
<td>103</td>
</tr>
<tr>
<td>Step 6: Clean up your resources</td>
<td>103</td>
</tr>
<tr>
<td>Create rule based on CloudTrail API call</td>
<td>104</td>
</tr>
<tr>
<td>Step 1: Create a rule</td>
<td>104</td>
</tr>
<tr>
<td>Step 2: Confirm success</td>
<td>105</td>
</tr>
<tr>
<td>Step 3: Clean up your resources</td>
<td>103</td>
</tr>
<tr>
<td>Create a sample application</td>
<td>107</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>108</td>
</tr>
<tr>
<td>Step 1: Create application</td>
<td>108</td>
</tr>
<tr>
<td>Step 2: Run application</td>
<td>109</td>
</tr>
<tr>
<td>Step 3: Check the logs and verify the application works</td>
<td>109</td>
</tr>
<tr>
<td>Step 4: Clean up your resources</td>
<td>103</td>
</tr>
<tr>
<td>Download code bindings for events using the EventBridge schema registry</td>
<td>111</td>
</tr>
<tr>
<td>Log Auto Scaling group states</td>
<td>112</td>
</tr>
<tr>
<td>Step 1: Create a Lambda function</td>
<td>112</td>
</tr>
<tr>
<td>Step 2: Create a rule</td>
<td>112</td>
</tr>
<tr>
<td>Step 3: Test the rule</td>
<td>113</td>
</tr>
<tr>
<td>Step 4: Confirm success</td>
<td>105</td>
</tr>
<tr>
<td>Step 5: Clean up your resources</td>
<td>103</td>
</tr>
</tbody>
</table>
What Is Amazon EventBridge?

Amazon EventBridge is a serverless event bus service that you can use to connect your applications with data from a variety of sources. EventBridge delivers a stream of real-time data from your applications, software as a service (SaaS) applications, and AWS services to targets such as AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.

Note
EventBridge was formerly called Amazon CloudWatch Events. The default event bus and the rules you created in CloudWatch Events also display in the EventBridge console. EventBridge uses the same CloudWatch Events API, so your code that uses the CloudWatch Events API stays the same. New features added to EventBridge are not added to CloudWatch Events.

The following video explains more: What's the difference between CloudWatch Events and EventBridge?

How it works

EventBridge receives an event (p. 16), an indicator of a change in environment, and applies a rule (p. 32) to route the event to a target (p. 48). Rules match events to targets based on either the structure of the event, called an event pattern (p. 19), or on a schedule. For example, when an Amazon EC2 instance changes from pending to running, you can have a rule that sends the event to a Lambda function.

All events that come to EventBridge are associated with an event bus (p. 7). Rules are tied to a single event bus, so they can only be applied to events on that event bus. Your account has a default event
bus which receives events from AWS services, and you can create custom event buses to send or receive events from a different account or Region.

When an AWS Partner wants to send events to an AWS customer account, they set up a partner event source (p. 93). Then the customer must associate an event bus with the partner event source.

EventBridge API destinations (p. 52) are HTTP endpoints that you can set as the target of a rule, in the same way that you would send event data to an AWS service or resource. By using API destinations, you can use REST API calls to route events between AWS services, integrated SaaS applications, and your applications outside of AWS. 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.

To customize the text from an event before EventBridge passes it to a target, use the input transformer (p. 68) to edit the information before it goes to the target.

You can archive (p. 72), or save, events and then replay (p. 74) them at a later time from the archive. Archiving is useful for testing an application because you have a store of events to use rather than having to wait for new events.

When you build serverless applications that use EventBridge, it can be helpful to know the event pattern of typical events without having to generate the event. The event patterns are described in schemas (p. 75), which are available for all events generated by AWS services on EventBridge. You can also create or upload custom schemas for events that don't come from AWS services. Once you have a schema for an event, you can download code bindings for popular programming languages.

To organize AWS resources or to track costs in EventBridge, you can assign a custom label, or tag (p. 205), to AWS resources. Using tag-based policies (p. 147), you can control what resources can and can't do within EventBridge.

In addition to tag-based policies, EventBridge supports identity-based (p. 154) and resource-based (p. 162) policies to control access to EventBridge. Use identity-based policies to control the permissions of a group, role, or user. Use resource-based policies to give specific permissions to each resource, such as a Lambda function or Amazon SNS topic.
Amazon EventBridge setup and prerequisites

To use Amazon EventBridge, you need an AWS account. Your account allows you to use services such as Amazon EC2 to generate events that you can see in the EventBridge console. You can also install and configure the AWS Command Line Interface (AWS CLI) to use a command-line interface to see events.

Topics
- Sign up for Amazon Web Services (AWS) (p. 3)
- Sign in to the Amazon EventBridge console (p. 3)
- Account credentials (p. 3)
- Set up the AWS Command Line Interface (p. 4)
- Regional Endpoints (p. 4)

Sign up for Amazon Web Services (AWS)

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

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

To sign up for an 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 instead. If you're using an IAM account to access EventBridge, you must have the following permissions.
"Version": "2012-10-17",
"Statement": [
    {
        "Action": [
            "events:*",
            "iam:PassRole"
        ],
        "Effect": "Allow",
        "Resource": "*"
    }
]

For more information, see Authentication (p. 148).

Set up the AWS 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 the default regional endpoints to use EventBridge. For more information, see Activating and Deactivating AWS STS in an AWS Region in the IAM User Guide.
Getting started with Amazon EventBridge

The basis of EventBridge is to create rules (p. 32) that route events (p. 16) to a target (p. 48). In this section, you create a basic rule. For tutorials about specific scenarios and specific targets, see Amazon EventBridge tutorials (p. 100).

Create a rule in Amazon EventBridge

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

When an AWS service in your AWS account emits an event, it always goes to the default event bus (p. 7) for your account. To write a rule that matches events from AWS services in your account, you must associate it with the default event bus.

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.
   To customize the event pattern, choose Edit, make your changes, and then 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 match events that come from your 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 you want to act when EventBridge detects an event of the selected type.
12. The fields displayed vary depending on 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 earlier, 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 one 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. For more information, see **Granting permissions to the dead-letter queue** (p. 44).

16. (Optional) 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 tags** (p. 205).

18. Choose **Create**.
Amazon EventBridge event buses

An event bus is a pipeline that receives events (p. 16). Rules (p. 32) associated with the event bus evaluate events as they arrive. Each rule checks whether an event matches the rule’s criteria. You associate a rule with a specific event bus, so the rule only applies to events received by that event bus.

To manage permissions for an event bus, you can configure a resource-based policy (p. 162) for it. A resource-based policy specifies 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 sources such as a rule or an event bus in a different AWS account or AWS Region. By using policies, you can aggregate all events from your application or organization in a single account and Region.

You can configure up to 300 rules for each event bus. If you have more than 300 rules in your environment, you can create custom event buses in your account and then associate an additional 300 rules with each event bus. You can customize how events are received in your account by creating event buses with different permissions for different services.

The most common event buses are:

- The default event bus in each account receives events from AWS services.
- A custom event bus sends events to or receives events from a different account.
- A custom event bus sends events to or receives events from a different Region to aggregate events in a single location.
- A partner event bus receives events from a SaaS partner. For more information, see Receiving events from a SaaS partner with Amazon EventBridge (p. 93).
The following video describes what event buses are and explains some of the basics of them: What are event buses

The following video covers the different event buses and when to use them: The differences between event buses

Topics
- Permissions for Amazon EventBridge event buses (p. 9)
- Creating an Amazon EventBridge event bus (p. 15)
Permissions for Amazon EventBridge event buses

The default event bus (p. 7) in your AWS account only allows events (p. 16) from one account. You can grant additional permissions to an event bus by attaching a resource-based policy (p. 162) to it. With a resource-based policy, you can allow PutEvents, PutRule, and PutTargets API calls from another account. You can also use IAM conditions (p. 172) in the policy to grant permissions to an organization, apply tags (p. 205), or filter events to only those from a specific rule or account. You can set a resource-based policy for an event bus when you create it or afterward.

EventBridge APIs that accept an event bus Name parameter such as PutRule, PutTargets, DeleteRule, RemoveTargets, DisableRule, and EnableRule also accept the event bus ARN. Use these parameters to reference cross-account or cross-Region event buses through the APIs. For example, you can call PutRule to create a rule (p. 32) on an event bus in a different account without needing to assume a role.

You can use the example policies in this topic to grant permission to send events to a different account or Region. For information about creating a rule to send events to a different account or Region, see Sending and receiving Amazon EventBridge events between AWS accounts (p. 62).

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. 13)

Managing event bus permissions

Use the following procedure to modify the permissions for an existing event bus. For information about 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. In the left navigation pane, choose Event buses.
3. In Name, choose the name of the event bus to manage permissions for.

   If a resource policy is attached to the event bus, the policy displays.
4. Choose Manage permissions, and 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, and add additional actions that you authorize the principal in the policy to use.
5. Choose Update.

The template provides example policy statements that you can customize for your account and environment. The template isn't a valid policy. You can modify the template for your use case, or 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 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 but the policy contains an error, an error message indicates the specific issue in the policy.

```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",
            "Effect": "Allow",
            "Principal": {
                "AWS": "<ACCOUNT_ID>"
            },
            "Action": [
                "events:PutRule",
                "events:PutTargets",
                "events:DeleteRule",
                "events:RemoveTargets"
            ]
        }
    ]
}
```
Example policy: Send events to the default bus in a different account

The following example policy grants the account 111122223333 permission to use all API operations on the default event bus in the account 123456789012.

```
{
    "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 policy grants the account 111122223333 permission 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.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "WebStoreCrossAccountPublish",
            "Effect": "Allow",
            "Action": [
                "events:PutEvents"
            ],
            "Principal": {
                "AWS": "111112222333"
            },
            "Condition": {
                "StringEquals": {
                    "events:detail-type": "newOrderCreated",
                    "events:source": "com.exampleCorp.webStore"
                }
            }
        }
    ]
}
```
Example policy: Send events to the same account and restrict updates

The following example policy grants account 123456789012 permission 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 123456789012 can modify these rules and targets once they have been created.

```json
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Sid": "InvoiceProcessingRuleCreation",
     "Effect": "Allow",
     "Principal": {
       "AWS": "arn:aws:iam::123456789012: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 grants the account 111122223333 permission 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 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 account 111122223333. The Condition statement restricts events to only events that match the rules with the specified rule ARN.

```json
{
 "Version": "2012-10-17",
 "Statement": [
 }"
Example policy: Send events only from a specific Region to a different Region

The following example policy grants account 111122223333 permission to send all events that are generated in the Middle East (Bahrain) and US West (Oregon) Regions to the event bus named CrossRegionBus in account 123456789012 in the US East (N. Virginia) Region. Account 111122223333 doesn't have permission to send events that are generated in any other Region.

```
{  
  "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-2:111112222333:rule/CrossRegionBus/SendToUSE1AnotherAccount",  
            "arn:aws:events:me-south-1:111112222333:rule/CrossRegionBus/SendToUSE1AnotherAccount"  
          ]  
        }  
      }  
    }  
  ]  
}
```

Example policy: Deny sending events from specific Regions

The following example policy attached to an event bus named CrossRegionBus in account 123456789012 grants permission for the event bus to receive events from the account 111122223333, but not events that are generated in the US West (Oregon) Region.

```
{  
  "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: Deny sending events from specific Regions

```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:***"
          ]
        }
      }
    }
  ]
}
```
Creating an Amazon EventBridge event bus

You can create a custom event bus (p. 7) to receive events (p. 16) from your applications. Your applications can also send events to the default event bus. When you create an event bus, you can attach a resource-based policy (p. 162) to grant permissions to other accounts. Then other accounts can send events to the event bus in the current account.

The following video goes through creating event buses: Creating an event bus

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

An event indicates a change in an environment such as an AWS environment, a SaaS partner service or application, or one of your applications or services. The following are examples of events:

- Amazon EC2 generates an event when the state of an instance changes from pending to running.
- 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, including sample events from each service, see Events from AWS services (p. 84).

Events are represented as JSON objects and they all have a similar structure, and the same top-level fields.

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. 84).

Topics
- Amazon EventBridge event patterns (p. 19)
- Adding Amazon EventBridge events with PutEvents (p. 28)
The following video explains the basics of events: What is an event

The following video covers the ways events get to EventBridge: Where do events come from

The following fields appear in an event:

**version**

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

**id**

A Version 4 UUID that is generated for every event. You can use id to trace events as they move through rules to targets.

**detail-type**

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

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

**source**

Identifies the service that generated the event. All events that come from AWS services 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 condition keys table, select a service from the list, and look for the service prefix. 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 can report the start time, so this value might be before the time the event is received.

region

Identifies the AWS Region where the event originated.

resources

A JSON array that contains ARNs that identify resources that are involved in the event. The service generating the event determines whether to include these ARNs. 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 that contains information about the event. The service generating the event determines the content of this field. The detail content can be as simple as two fields. AWS API call events have detail objects with approximately 50 fields nested several levels deep.

The following event in Amazon EventBridge indicates an Amazon EC2 instance being terminated.

```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"
    }
}
```
Amazon EventBridge event patterns

Event patterns have the same structure as the events (p. 16) they match. Rules (p. 32) use event patterns to select events and send them to targets. An event pattern either matches an event or it doesn’t.

The following video goes over the basics of event patterns: How to filter events

Topics

- Create event patterns (p. 20)
- Example events and event patterns (p. 21)
- Matching null values and empty strings in Amazon EventBridge event patterns (p. 23)
- Arrays in Amazon EventBridge event patterns (p. 24)
- Content filtering in Amazon EventBridge event patterns (p. 25)

The following event shows a simple AWS event from Amazon EC2.

```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": {
```
Create event patterns

To create an event pattern, you specify the fields of an event that you want the event pattern to match. Only specify the fields that you use for matching. The previous event pattern example only provides 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 when applying the rule.

For an event pattern to match an event, the event must contain all the field names listed in the event pattern. The field names must also appear in the event with the same nesting structure.

EventBridge ignores the fields in the event that aren't included in the event pattern. The effect is that there is a `*:*` wildcard for fields that don't appear in the event pattern.

The values that event patterns match follow JSON rules. You can include strings enclosed in quotation marks (`"`), numbers, and the keywords `true`, `false`, and `null`.

For strings, EventBridge uses exact character-by-character matching without case-folding or any other string normalization.

For numbers, EventBridge uses string representation. For example, 300, 300.0, and 3.0e2 are not considered equal.

When you write event patterns to match events, you can use the `TestEventPattern` API or the `test-event-pattern` CLI command to test that your pattern matches the correct events. For more information, see `TestEventPattern`.

Here's a summary of all the comparison operators available in EventBridge:

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Example</th>
<th>Rule syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>UserID is null</td>
<td>&quot;UserID&quot;: [ null ]</td>
</tr>
<tr>
<td>Empty</td>
<td>LastName is empty</td>
<td>&quot;LastName&quot;: [&quot;&quot; ]</td>
</tr>
<tr>
<td>Equals</td>
<td>Name is &quot;Alice&quot;</td>
<td>&quot;Name&quot;: [ &quot;Alice&quot; ]</td>
</tr>
<tr>
<td>And</td>
<td>Location is &quot;New York&quot; and Day is &quot;Monday&quot;</td>
<td>&quot;Location&quot;: [ &quot;New York&quot; ], &quot;Day&quot;: [&quot;Monday&quot;]</td>
</tr>
<tr>
<td>Or</td>
<td>PaymentType is &quot;Credit&quot; or &quot;Debit&quot;</td>
<td>&quot;PaymentType&quot;: [ &quot;Credit&quot;, &quot;Debit&quot; ]</td>
</tr>
</tbody>
</table>
Comparison | Example | Rule syntax
---|---|---
Not | Weather is anything but “Raining” | "Weather": [{ “anything-but”: [ "Raining"] }
Numeric (equals) | Price is 100 | "Price": [{ “numeric": [ “=” , 100 ] }
Numeric (range) | Price is more than 10, and less than or equal to 20 | "Price": [{ “numeric": [ “>”, 10 , “≤”, 20 ] }
Exists | ProductName exists | "ProductName": [{ “exists”: true }]
Does not exist | ProductName does not exist | "ProductName": [{ “exists”: false }]
Begins with | Region is in the US | "Region": [{ “prefix”: “us-“ }]

Match Values

In an event pattern, the value to match is in a JSON array, surrounded by square brackets ("[", "]") so that you can provide multiple values. For example, to match events from Amazon EC2 or AWS Fargate, you could use the following pattern, which matches events where the value for the “source” field is either "aws.ec2" or "aws.fargate".

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

Example events and event patterns

You can use all of the JSON data types and values to match events. The following examples show events and the event patterns that match them.

Field matching

You can match on the value of a field. Consider the following Amazon EC2 Auto Scaling event.

```
{
    "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
    }
}
```

For the preceding event, you can use the “responseElements” field to match.
Example events and event patterns

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

### Value matching

Consider the following Amazon Macie Classic event, which is truncated.

```json
{
  "version": "0",
  "id": "3e355723-fca9-4de3-9fd7-154c289d6b59",
  "detail-type": "Macie Alert",
  "source": "aws.macie",
  "account": "123456789012",
  "time": "2017-04-24T22:28:49Z",
  "region": "us-east-1",
  "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": "2017-04-18T22:28:49Z"
  }
}
```

The following event pattern matches any event that has a risk score of 80 and a trigger risk of 8.

```json
{
  "source": ["aws.macie"],
  "detail-type": ["Macie Alert"],
  "detail": {
    "risk-score": [80],
    "trigger": {
      "risk": [8]
    }
  }
}
```
Matching null values and empty strings in Amazon EventBridge event patterns

You can create an event pattern (p. 19) that matches a field in an event (p. 16) that has a null value or is an empty string. 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 event pattern, which matches the preceding event.

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

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

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

Note
Null values and empty strings are not interchangeable in pattern matching. An event pattern that matches empty strings doesn't match values of null.
Arrays in Amazon EventBridge event patterns

The value of each field in an event pattern (p. 19) is an array containing one or more values. An event pattern matches the event (p. 16) 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 event pattern matches if the intersection of the event pattern array and the event array is non-empty.

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

```
"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"
]
```

The preceding event pattern matches an event that includes the following field because the first item in the event pattern array matches the second item in the event array.

```
"resources": [
    "arn:aws:ec2:us-east-1:123456789012:instance/i-b188560f"
]
```
Content filtering in Amazon EventBridge event patterns

Amazon EventBridge supports declarative content filtering using event patterns (p. 19). With content filtering, you can write complex event patterns that only match events under very specific conditions. For example, you can create an event pattern that matches an event when a field of the event (p. 16) is within a specific numeric range, if the event comes from a specific IP address, or only if a specific field doesn't exist in the event JSON.

Filter types

- Prefix matching (p. 25)
- Anything-but matching (p. 25)
- 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 an event depending on the prefix of a value in the event source. You can use prefix matching for string values.

For example, the following event pattern would match any event where the "time" field started with "2017-10-02" such as "time": "2017-10-02T18:43:48Z".

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

Anything-but matching

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

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

The following event pattern shows anything-but matching with strings and numbers.

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

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

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

```json
{
}
```
Content-based filtering

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

```
{ "detail": {
    "state": [ { "anything-but": [ 100, 200, 300 ] } ]
}
```

The following event pattern shows anything-but matching that matches any event that doesn't have the prefix "init" in the "state" field.

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

**Numeric matching**

Numeric matching works with values that are JSON numbers. It is limited to values between -1.0e9 and +1.0e9 inclusive, with 15 digits of precision, or six digits to the right of the decimal point.

The following shows numeric matching for an event pattern that only matches events that are true for all fields.

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

**IP address matching**

You can use IP address matching for IPv4 and IPv6 addresses. The following event pattern shows IP address matching to IP addresses that start with 10.0.0 and end with a number between 0 and 255.

```
{ "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.

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

The following event pattern matches any event that has a `detail.state` field.
Content-based filtering

The preceding event pattern matches the following event.

```
{
  "detail": {
    "state": [ { "exists": true } ]
  }
}
```

The preceding event pattern does NOT match the following event because it doesn't have a `detail.state` field.

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

Complex example with multiple matching

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

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

**Note**

When building event patterns, if you include a key more than once the last reference will be the one used to evaluate events. For example, for the following pattern:

```
{
  "detail": {
    "location": [ { "prefix": "us-" } ],
    "location": [ { "anything-but": "us-east" } ]
  }
}
```

only { "anything-but": "us-east" } will be taken into account when evaluating the location.
Adding Amazon EventBridge events with PutEvents

The PutEvents action sends multiple events (p. 16) to EventBridge in a single request. For more information, see PutEvents in the Amazon EventBridge API Reference and put-events in the AWS CLI Command Reference.

Each PutEvents request can support a limited number of entries. For more information, see Amazon EventBridge quotas (p. 200). The PutEvents operation attempts to process all entries in the natural order of the request. After you call PutEvents, EventBridge assigns each event a unique ID.

Topics
- Handling failures with PutEvents (p. 29)
- Sending events using the AWS CLI (p. 30)
- Calculating Amazon EventBridge PutEvents event entry size (p. 31)

The following example Java code sends two identical events to EventBridge.

AWS SDK for Java Version 2.x

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

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

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

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

PutEventsResponse result = eventBridgeClient.putEvents(eventsRequest);

for (PutEventResultEntry 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
EventBridgeClient eventBridgeClient = EventBridgeClient.builder().build();

PutEventsRequestEntry requestEntry = new PutEventsRequestEntry()
  .withTime(new Date());
```

28
Handling failures with \texttt{PutEvents}

By default, if an individual entry within a request fails, EventBridge continues processing the rest of the entries in the request. A response Entries array can include both successful and unsuccessful entries. You must detect unsuccessful entries and include them in a subsequent call.

Successful result entries include an \texttt{Id} value, and unsuccessful result entries include \texttt{ErrorCode} and \texttt{ErrorMessage} values. \texttt{ErrorCode} describes the type of error. \texttt{ErrorMessage} provides more information about the error. The following example has three result entries for a \texttt{PutEvents} request. The second entry is unsuccessful.

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

You can include entries that are unsuccessful in subsequent \texttt{PutEvents} requests. First, to find out if there are failed entries in the request, check the \texttt{FailedRecordCount} parameter in \texttt{PutEventsResult}. If it isn't zero, then you can add each \texttt{Entry} that has an \texttt{ErrorCode} that is not null to a subsequent request. The following example shows a failure handler.

```java
PutEventsRequestEntry requestEntry = new PutEventsRequestEntry()
    .withTime(new Date())
    .withSource("com.mycompany.myapp")
    .withDetailType("myDetailType")
```
Sending events using the AWS CLI

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

```java
List<PutEventsRequestEntry> putEventsRequestEntryList = new ArrayList<>();
for (int i = 0; i < 3; i++) {
    putEventsRequestEntryList.add(requestEntry);
}

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

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

You can use the AWS CLI to send custom events to EventBridge:

```bash
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 JSON file that contains custom events.

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

Then, to use the AWS CLI to read the entries from this file and send events, at a command prompt, type:

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

You can send custom events (p. 16) to EventBridge by using the `PutEvents` action. You can batch multiple event entries into one request for efficiency. The total entry size must be less than 256KB. You can calculate the entry size before you send the events.

**Note**
The size limit is imposed on the *entry*. Even if the entry is less than the size limit, the *event* in EventBridge 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).

EventBridge calculates the `PutEventsRequestEntry` size as follows:

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

**Note**
If the entry size is larger than 256KB, we recommend putting the event in an Amazon S3 object and including a link to that object in the `PutEvents` entry.
Amazon EventBridge rules

A rule matches incoming events (p. 16) and sends them to targets (p. 48) for processing. A single rule can send an event to multiple targets, which then run in parallel. Rules are based either on an event pattern (p. 19) or a schedule. An event pattern defines the event structure and the fields that a rule matches. Rules that are based on a schedule perform an action at regular intervals.

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 can also create an IAM policy (p. 148) that grants permission 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, but you should only delete them if you’re 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.

The following video goes over the basics of rules: What are rules

Topics
- Creating Amazon EventBridge rules that react to events (p. 34)
- Creating an Amazon EventBridge rule that runs on a schedule (p. 36)
- Disabling or deleting an Amazon EventBridge rule (p. 42)
- Event retry policy and using dead-letter queues (p. 43)
- Using Amazon EventBridge and AWS Serverless Application Model templates (p. 45)
Creating Amazon EventBridge rules that react to events

To create a rule (p. 32) for events (p. 16), you specify an action to take when EventBridge receives an event that matches the event pattern (p. 19) in the rule. When an event matches, EventBridge sends the event to the specified target (p. 48) and triggers the action defined in the rule.

The following video explores creating different kinds of rules and how to test them: Learning about rules

To create a rule that reacts to events

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.

   To customize the event pattern, choose Edit, make your changes, and then 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 match events that come from your 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 you want to act when EventBridge detects an event of the selected type.

12. The fields displayed vary depending on 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 earlier, 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 one 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. For more information, see Granting permissions to the dead-letter queue (p. 44).

16. (Optional) 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 tags (p. 205).

18. Choose **Create**.
Creating an Amazon EventBridge rule that runs on a schedule

A rule (p. 32) can run in response to an event (p. 16) or at certain time intervals. For example, to periodically run an AWS Lambda function, you can create a rule to run on a schedule. You can create rules that run on a schedule by using cron or rate expressions. All scheduled events use UTC+0 time zone, and the minimum precision for a schedule is one minute. Your scheduled rule runs within that minute, but not on the precise 0th second.

EventBridge supports cron expressions and rate expressions. Rate expressions are simpler to define and cron expressions offer the detailed schedule control. For example, with a cron expression, you can define a rule that runs at a specified time on a certain day of each week or month. In contrast, rate expressions run a rule at a regular rate, such as once every hour or once every day.

**Note**

EventBridge doesn't provide second-level precision in schedule expressions. The finest resolution using a cron expression is one minute. Due to the distributed nature of EventBridge and the target services, there can be a delay of several seconds between the time the scheduled rule is triggered and the time the target service runs the target resource.

The following video gives an overview of scheduling tasks: [Creating scheduled tasks with EventBridge](#)

**Formats**

- Cron Expressions (p. 37)
- Rate Expressions (p. 39)
- Create rule (p. 40)
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 includes January, February, and March.
- The - (dash) wildcard specifies ranges. In the Day field, 1-15 includes days 1 through 15 of the specified month.
- The * (asterisk) wildcard includes all values in the field. In the Hours field, * includes every hour. You can’t 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 / (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 any. In the Day-of-month field you could enter 7 and if any day of the week was acceptable, 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 * (asterisk) 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+0) 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+0) 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+0) 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+0) every 1st day of the month</td>
</tr>
<tr>
<td>0/15</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>Run every 15 minutes</td>
</tr>
<tr>
<td>0/10</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>MON-FRI</td>
<td>*</td>
<td>Run every 10 minutes Monday through Friday</td>
</tr>
<tr>
<td>0/5</td>
<td>8-17</td>
<td>?</td>
<td>*</td>
<td>MON-FRI</td>
<td>*</td>
<td>Run every 5 minutes Monday through Friday between 8:00 am and 5:55 pm (UTC+0)</td>
</tr>
<tr>
<td>0/30</td>
<td>20-2</td>
<td>?</td>
<td>*</td>
<td>MON-FRI</td>
<td>*</td>
<td>Run every 30 minutes Monday through Friday between 10:00 pm on the starting day to 2:00</td>
</tr>
</tbody>
</table>


Rate Expressions

A rate expression starts when you create the scheduled event rule, and then it runs on a defined schedule. Rate expressions have two required fields separated by white space.

Syntax

rate(value unit)

value

A positive number.

unit

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

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

Limitations

If the value is equal to 1, then the unit must be singular. If the value is greater than 1, the unit must be plural. For example, rate(1 hours) and rate(5 hour) aren’t 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.

```
aws events put-rule --schedule-expression "rate(1 minute)" --name MyRule2
```
The following steps walk you through how to create an EventBridge rule that triggers on a regular schedule.

**Note**
You can only create scheduled rules using the default event bus.

**To create a rule that runs on a regular schedule**

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
   A rule can't have the same name as another rule in the same Region and on the same event bus.
5. For Define pattern, choose Schedule.
6. Either choose Fixed rate of and specify how often the task is to run, or choose Cron expression and specify a cron expression that defines when the task is to be triggered.
7. For Select event bus, choose AWS default event bus. You can only create scheduled rules 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 permission to send events to the target. In these cases, EventBridge can create the IAM role needed for your rule to run. Do one of these things:
    - 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 it can't deliver them 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. 44).
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 tags (p. 205).
15. Choose Create.
Disabling or deleting an Amazon EventBridge rule

To stop a rule (p. 32) from processing events (p. 16) or running on a schedule, you can delete or disable the rule. The following steps walk you through how 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, 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.
   b. To temporarily disable a rule, select the button next to the rule and choose Disable, Disable.
      You can't disable a managed rule.
Event retry policy and using dead-letter queues

Sometimes an event isn’t successfully delivered to the target specified in a rule. This can happen when, for example, the target resource is unavailable, when EventBridge lacks permission to the target resource, or due to network conditions. When an event isn’t successfully delivered to a target, EventBridge retries sending the event. You set the length of time it tries, and number of retry attempts in 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, or randomized delay. If an event isn’t delivered after all retry attempts are exhausted, the event is dropped and EventBridge doesn’t 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 couldn’t 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 weren’t successfully delivered. Then you can resolve the issue that resulted in the failed event delivery and process the events at a later time.

Event errors are handled in different ways. 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 fail until an action is taken to resolve the underlying issue. Rather than retrying, EventBridge sends these events directly to the DLQ, if you have one.

When an event delivery fails, EventBridge publishes an event to Amazon CloudWatch metrics indicating that a target invocation failed. If you use a DLQ, additional metrics are sent to CloudWatch including InvocationsSentToDLQ and InvocationsFailedToBeSentToDLQ. For more information about EventBridge metrics, see Monitoring Amazon EventBridge.

The following video goes over settings up DLQs: Using dead-letter queues (DLQs)
Considerations for using a dead-letter queue

Consider the following when configuring a DLQ for EventBridge.

- Only standard queues are supported. You can't 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 (p. 162) 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. 44).
- Permissions for using Amazon SQS queues from a different AWS account.
  - If you create a rule from the console, queues from other accounts aren't 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. 44).
  - 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. 44).
- 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",
}
```
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. 165).

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.

Using Amazon EventBridge and AWS Serverless Application Model templates

You can build and test rules (p. 32) manually in the EventBridge console, which can help in the development process as you refine event patterns (p. 19). However, once you are ready to deploy your application, it’s easier to use a framework like AWS SAM to launch all your serverless resources consistently.

We’ll use this example application to look into the ways you can use AWS SAM templates to build EventBridge resources. The template.yaml file in this example is a AWS SAM template that defines four AWS Lambda functions and shows two different ways to integrate the Lambda functions with EventBridge.

For a walkthrough of this example application, see ??? (p. 107).

There are two approaches to using EventBridge and AWS SAM templates. For simple integrations where one Lambda function is invoked by one rule, the the Combined template approach is recommended. If you have complex routing logic, or you are connecting to resources outside of your AWS SAM template, the Separated template approach is the better choice.

Approaches:
- Combined template (p. 45)
- Separated template (p. 46)

Combined template

The first approach uses the Events property to configure the EventBridge rule. The following example code defines an event (p. 16) that invokes your Lambda function.

Note
This example automatically creates the rule on the default event bus (p. 7), which exists in every AWS account. To associate the rule with a custom event bus, you can add the EventBusName to the template.
atmConsumerCase3Fn:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: atmConsumer/
    Handler: handler.case3Handler
    Runtime: nodejs12.x
  Events:
    Trigger:
      Type: CloudWatchEvent
      Properties:
        Pattern:
          source:
          - custom.myATMapp
detail-type:
          - transaction
detail:
result:
  - "anything-but": "approved"

This YAML code is equivalent to an event pattern in the EventBridge console. In YAML, you only need to define the event pattern, and AWS SAM automatically creates an IAM role with the required permissions.

Separated template

In the second approach to defining an EventBridge configuration in AWS SAM, the resources are separated more clearly in the template.

1. First, you define the Lambda function:

   atmConsumerCase1Fn:
   Type: AWS::Serverless::Function
   Properties:
     CodeUri: atmConsumer/
     Handler: handler.case1Handler
     Runtime: nodejs12.x

2. Next, define the rule using an AWS::Events::Rule resource. The properties define the event pattern and can also specify targets (p. 48). You can explicitly define multiple targets.

   EventRuleCase1:
   Type: AWS::Events::Rule
   Properties:
     Description: "Approved transactions"
     EventPattern:
       source:
       - "custom.myATMapp"
       detail-type:
       - transaction
detail:
result:
  - "approved"
   State: "ENABLED"
   Targets:
     - Arn:
       Fn::GetAtt:
         - "atmConsumerCase1Fn"
         - "Arn"
       Id: "atmConsumerTarget1"
3. Finally, define an AWS::Lambda::Permission resource that grants permission to EventBridge to invoke the target.

```
PermissionForEventsToInvokeLambda:
  Type: AWS::Lambda::Permission
  Properties:
    FunctionName:Ref: "atmConsumerCase1Fn"
    Action:"lambda:InvokeFunction"
    Principal:"events.amazonaws.com"
    SourceArn:
      Fn::GetAtt:
        - "EventRuleCase1"
        - "Arn"
```
Amazon EventBridge targets

A target is a resource or endpoint that EventBridge sends an event (p. 16) to when the event matches the event pattern defined for a rule (p. 32). The rule processes the event (p. 16) data and sends the pertinent information to the target. To deliver event data to a target, EventBridge needs permission to access the target resource. You can define up to five targets for each rule.

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

The following video covers the basics of targets: What is a target

Targets available in the EventBridge console

You can configure the following targets for events in the EventBridge console:

- API destination (p. 52)
- API Gateway (p. 61)
- AWS Batch job queue
- Amazon CloudWatch Logs group
- AWS CodeBuild project
- AWS CodePipeline
- Amazon Elastic Compute Cloud (Amazon EC2) CreateSnapshot API call
• EC2 Image Builder
• Amazon EC2 RebootInstances API call
• Amazon EC2 StopInstances API call
• Amazon EC2 TerminateInstances API call
• Amazon ECS task
• Event bus in a different AWS account or AWS Region (p. 62)
• Firehose delivery stream (Amazon Kinesis Data Firehose)
• Incident Manager response plan
• Inspector assessment template (Amazon Kinesis Data Streams)
• Kinesis stream (Kinesis Data Streams)
• AWS Lambda function
• Amazon Redshift cluster (Data API statement execution)
• SageMaker Pipeline
• Amazon SNS topic
• Amazon SQS queue (including a FIFO queue)
• Amazon EC2 Systems Manager (SSM) Automation
• SSM OpsItem
• SSM Run Command
• AWS Step Functions state machine

Target parameters

These targets accept parameters.

• For a Kinesis data stream target, you can specify which shard the event goes to by using the KinesisParameters parameter.
• To invoke commands on multiple Amazon EC2 instances with one rule, you can use the RunCommandParameters field.
• To pass data to an API Gateway endpoint, you can use the HttpParameters field, including optional dynamic JSON path syntax. 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. For more information, see the section called “Transforming target input” (p. 68).
• To pass data to an Amazon Redshift cluster, you can use the RedshiftDataParameters field, including optional dynamic JSON path syntax.
• To pass data to a SageMaker pipeline, you can use the SageMakerPipelineParameters field, including optional dynamic JSON path syntax.

Input, InputPath, and InputTransformer are the mutually exclusive and optional parameters of a target. When a rule runs, the behavior is as follows:

• If you don’t specify any parameters for the target, then the entire event is passed to the target in JSON format. However, if the target is an Amazon EC2 Run command or an Amazon ECS task, the event isn’t passed to the target.
• If you specify Input in the form of valid JSON, then the matched event is overridden with this JSON.
• If you specify InputPath in the form of JSONPath, then only the part of the event that’s specified in the path is passed to the target. For example, if you specify $.detail, then only the detail part of the event is passed.
• If you specify `InputTransformer`, then 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.

Permissions

To make API calls on the resources that you own, EventBridge needs appropriate permission. For AWS Lambda and Amazon SNS resources, EventBridge uses resource-based policies (p. 162). For EC2 instances, Kinesis data streams, and Step Functions state machines, EventBridge uses IAM roles that you specify in the `RoleARN` parameter in `PutTargets`. You can invoke an API Gateway REST endpoint with configured IAM authorization, but the role is optional if you haven't configured authorization. For more information, see Amazon EventBridge and AWS Identity and Access Management (p. 148).

If another account is in the same Region and has granted you permission, then you can send events to that account. For more information, see Sending and receiving Amazon EventBridge events between AWS accounts (p. 62).
Configure targets

Learn how to configure settings for EventBridge targets.

Targets:
- API destinations (p. 52)
- Amazon EventBridge targets for Amazon API Gateway (p. 61)
- Sending and receiving Amazon EventBridge events between AWS accounts (p. 62)
- Sending and receiving Amazon EventBridge events between AWS Regions (p. 64)
- Sending and receiving Amazon EventBridge events between event buses in the same account and Region (p. 66)
API destinations

Amazon EventBridge API destinations are HTTP endpoints that you can invoke as the target (p. 48) of a rule (p. 32), similar to how you invoke an AWS service or resource as a target. Using API destinations, you can route events (p. 16) between AWS services, integrated software as a service (SaaS) applications, and your applications outside of AWS by 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 (p. 19) specified in the rule and then delivers the event information with the request. With EventBridge, you can use any HTTP method except CONNECT and TRACE for the request. The most common HTTP methods to use are PUT and POST. You can also use input transformers to customize the event to the parameters of a specific HTTP endpoint parameters. For more information, see Transforming Amazon EventBridge target input (p. 68).

Important
EventBridge requests to an API destination endpoint must have a maximum client execution timeout of 5 seconds. If the target endpoint takes longer than 5 seconds to respond, EventBridge times out the request. EventBridge retries timed out requests up to the maximums that are configured on your retry policy. By default the maximums are 24 hours and 185 times. After the maximum number of retries, events are sent to your dead-letter queue (p. 43) if you have one. Otherwise, the event is dropped.

The following video demonstrates the use of API destination: Using API destinations

In this topic:
- Connections for API destinations (p. 53)
- Create an API destination (p. 54)
- Service-linked role for API destinations (p. 55)
- Headers included in requests to API destinations (p. 55)
- API destination error codes (p. 56)
Connections for API destinations

When you create an API destination, you specify a connection to use for it. A connection specifies 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 required authorization headers for you. For OAuth authorization, EventBridge also exchanges your client ID and secret for an access token and then manages it securely. When you create a connection, you can also include the header, body, and query parameters that are required for authorization with an endpoint. You can use the same connection for more than one API destination if the authorization for the endpoint is the same.

When you create a connection and add authorization parameters, EventBridge creates a secret in AWS Secrets Manager. The cost of storing the Secrets Manager secret is included with the charge for using an API destination. To learn more about best practices for using secrets with API destinations, see AWS::Events::ApiDestination in the CloudFormation User Guide.

**Note**

To successfully create or update a connection, you must use an account that has permission to use Secrets Manager. The required permission is included in the AmazonEventBridgeFullAccess policy (p. 155). The same permission is granted to the service-linked role (p. 55) that's 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, and 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), and then enter the Username and Password to use to authorize with the HTTP endpoint.
   - Choose OAuth Client Credentials, and 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, add 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 API key, and then enter the API key name and associated Value to use for API Key authorization.

   Under Invocation Http Parameters, add 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, and then choose **Connections**.
2. In the **Connections** table, choose the connection to edit.
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. Removing authorization parameters removes the secret from the connection, so you can reuse it without having to create a new connection.

**Note**

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. In the **Connections** table, choose the connection.
2. On the **Connection details** page, choose **De-authorize**.
3. In the **Deauthorize connection?** dialog box, 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. Then the status changes to **De-authorized**. Now you can edit the connection to add new authorization parameters.

**Create an API destination**

Each API destination requires a connection. A **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, and 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 uppercase or lowercase 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 an **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. (Optional) For **Invocation rate limit per second** field, enter the maximum number of invocations per second to send to the API destination endpoint.
The rate limit you set may affect how EventBridge delivers events. For more information, see How invocation rate affects event delivery (p. 56).

9. For Connection, do one of the following:
   - Choose Use an existing connection, and then select the connection to use for this API destination.
   - Choose Create a new connection, and then enter the details for the connection to create. For more information, see Connections (p. 53).

10. Choose Create.

After you create an API destination, you can select it as the target of a rule (p. 32). To use an API destination as a target, you must provide an IAM role with the correct permissions. For more information, see ??? (p. 157)

### Service-linked role for API destinations

When you create a connection for an API destination, a service-linked role named AWSServiceRoleForAmazonEventBridgeApiDestinations to your account. EventBridge uses the service-linked role to create and store a secret in Secrets Manager. To grant the necessary permissions to the service-linked role, EventBridge attaches the AmazonEventBridgeApiDestinationsServiceRolePolicy policy 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 following policy is the AmazonEventBridgeApiDestinationsServiceRolePolicy.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "secretsmanager:CreateSecret",
            "secretsmanager:UpdateSecret",
            "secretsmanager:DescribeSecret",
            "secretsmanager:DeleteSecret",
            "secretsmanager:GetSecretValue",
            "secretsmanager:PutSecretValue"
         ],
         "Resource": "arn:aws:secretsmanager:*::*:secret:events:connection/*"
      }
   ]
}
```

For more information 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>
</tbody>
</table>
API destination error codes

When EventBridge tries to deliver an event to an API destination and an error occurs, EventBridge does the following:

- Events associated with error codes 429 and 5xx are retried.
- Events associated with error codes 1xx, 2xx, 3xx, and 4xx (excluding 429) aren't retried.

EventBridge API destinations read the standard HTTP response header `Retry-After` to find out how long to wait before making a follow-up request. EventBridge chooses the more conservative value between the defined retry policy and the `Retry-After` header. If `Retry-After` value is negative, 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. For more information, see Event retry policy and using dead-letter queues (p. 43).

API destination partners

Use the information provided by the following AWS Partners to configure an API destination and connection for their service or application.

**Datadog**

**API destination invocation endpoint URL**

For a full list of endpoints, see Datadog API Reference.

**Supported authorization types**

- API Key

**Additional authorization parameters required**

- None

---

<table>
<thead>
<tr>
<th>Header key</th>
<th>Header value</th>
</tr>
</thead>
<tbody>
<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 where the request is being sent.</td>
</tr>
</tbody>
</table>
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. For the correct URL for your organization, see documentation.

Freshworks

API destination invocation endpoint URL

For a list of endpoints, see 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 you can use Realm Webhooks to perform CRUD operations on collections in MongoDB Atlas from EventBridge API endpoints.

New Relic

API destination invocation endpoint URL

For more information, see Our EU and US region data centers.

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.eu.newrelic.com/metric/v1

Logs

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

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**

- For a full list of endpoints, see 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**

- Salesforce Connected App provides the client ID and client secret.
- The 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, you need an HEC token ID. For more information, 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

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
Commonly used API operations

POST https://your_Zendesk_subdomain/api/v2/tickets

Additional information

API requests EventBridge makes count against your Zendesk API limits. For information about Zendesk limits for your plan, see Usage limits.

To better safeguard your account and data, we recommend using an API key rather than basic username and password authentication.

Amazon EventBridge targets for Amazon API Gateway

You can use Amazon API Gateway 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 (p. 48), each event (p. 16) sent to the target maps to a request sent to the endpoint.

Important
EventBridge supports using API Gateway Edge-optimized and Regional endpoints as targets. To learn more about endpoints, see https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-api-endpoint-types.html.

You can use an API Gateway target for the following use cases:

• To invoke a customer-specified REST API hosted in API Gateway based on AWS or third-party events.
• To invoke an endpoint periodically on a schedule.

The EventBridge JSON event information is sent as the body of the HTTP request to your endpoint. You can specify the other request attributes in the target's HttpParameters field as follows:

• PathParameterValues lists the values that correspond sequentially to any path variables in your endpoint ARN, for example "arn:aws:execute-api:us-east-1:112233445566:myapi/*/POST/pets/*".
• QueryStringParameters represents the query string parameters that EventBridge appends to the invoked endpoint.
• HeaderParameters defines HTTP headers to add to the request.

Note
For security considerations, the following HTTP header keys aren't permitted:

• Anything prefixed with X-Amz or X-Amzn
• Authorization
• Connection
• Content-Encoding
• Content-Length
• Host
• Max-Forwards
• TE
• Transfer-Encoding
Dynamic Parameters

When invoking an API Gateway target, you can dynamically add data to events that are sent to the target. For more information, see the section called “Target parameters” (p. 49).

Invocation Retries

As with all targets, EventBridge retries some failed invocations. For API Gateway, EventBridge retries responses sent with a 5xx or 429 HTTP status code for up to 24 hours with exponential back off and jitter. After that, EventBridge publishes a FailedInvocations metric in Amazon CloudWatch. EventBridge doesn't retry other 4xx HTTP errors.

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 times out the request and then retries.

Sending and receiving Amazon EventBridge events between AWS accounts

You can configure EventBridge to send and receive events (p. 16) between AWS accounts. When you configure EventBridge to send or receive events between accounts, you can specify which AWS accounts can send events to or receive events from the event bus (p. 7) in your account. You can also allow or deny events from specific rules (p. 32) associated with the event bus, or events from specific sources. For more information, see Simplifying cross-account access with Amazon EventBridge resource policies.

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.

Note

If you're using an Incident Manager response plan as a target, all the response plans that are shared with your account are available by default.

You can send and receive events between AWS accounts within the same Region in all Regions.

The steps to configure EventBridge to send events to or receive events from a different account 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.

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.

The following video covers routing events between accounts: Routing events to buses in other AWS accounts

**Grant permissions to allow events from other AWS accounts**

To receive events from other accounts or organizations, you must first edit the permissions on the event bus you intend to receive events on. The default event bus accepts events from AWS services, other authorized AWS accounts, and PutEvents calls. The permissions for an 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 Amazon 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 (p. 19) 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 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 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. To learn more, see Permissions for Amazon 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. 43).
10. Choose Create to create the rule.

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

You can configure EventBridge to send and receive events (p. 16) between AWS Regions. You can also allow or deny events from specific Regions, specific rules (p. 32) associated with the event bus, or events
from specific sources. For more information, see Introducing cross-Region event routing with Amazon EventBridge.

The current list of supported destination Regions is:

- US East (N. Virginia)
- US West (Oregon)
- Europe (Ireland)

The following video covers routing events between Regions using the https://console.aws.amazon.com/events/, AWS CloudFormation, and AWS Serverless Application Model: Cross-Region event routing.

**To create a rule that sends events to a different AWS 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 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 Region. To learn more, see Permissions for Amazon 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. 43).

10. Choose **Create** to create the rule.

### Sending and receiving Amazon EventBridge events between event buses in the same account and Region

You can configure EventBridge to send and receive events (p. 16) between event buses (p. 7) in the same AWS account and Region.

When you configure EventBridge to send or receive events between event buses, you use IAM roles on the *sender* event bus to give the *sender* event bus permission to send events to the *receiver* event bus. You use Resource-based (p. 162) policies on the *receiver* event bus to give the *receiver* event bus permission to receive events from the *sender* event bus. You can also allow or deny events from certain event buses, specific rules (p. 32) associated with the event bus, or events from specific sources. For more information about event bus permissions, including example policies, see Permissions for Amazon EventBridge event buses (p. 9).

The steps to configure EventBridge to send events to or receive events between event buses in your account include the following:

• To use an existing IAM role, you need to give either the sender event bus permissions to the receiver event bus or the receiver event bus permissions to the sender event bus.

• On the *sender* event bus, set up one or more rules that have the receiver event bus as the target and create and IAM role.

• On the *receiver* event bus, edit the permissions to allow events to be passed from the other event bus.

• On the *receiver* event, set up one or more rules that match events that come from the sender event bus.

**Note**

EventBridge can’t route events received from a sender event bus to a third event bus.

Events sent from one event bus to another are charged as custom events. For more information, see Amazon EventBridge Pricing.

To send events to another event bus, you create a rule with an event bus as a target.

**To create a rule that sends events to a different event bus 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**, and then select the **Service provider** and **Service name** to filter events on. If you choose an AWS service, also choose the **Event type**. You might 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, and then choose **Save**.

6. For **Select event bus**, choose the event bus to associate with this rule.

7. Under **Select targets**, for **Target**, choose **Event bus in the same account and Region**, and then from the list of available event buses, select a different event bus than the one you selected in step 6.

8. 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.

9. (Optional) Configure a dead-letter queue for the rule. For more information, see Event retry policy and using dead-letter queues (p. 43).

10. Choose **Create** to create the rule.
Transforming Amazon EventBridge target input

You can customize the text from an event (p. 16) before EventBridge passes the event to the target (p. 48) of a rule (p. 32). Using the input transformer in the console or the API, you 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>`.

For a tutorial on using input transformer, see ??? (p. 143).

In this topic:
- Predefined variables (p. 68)
- Input Transform Examples (p. 68)
- Transforming input by using the EventBridge API (p. 70)
- Transforming input by using AWS CloudFormation (p. 70)
- Common Issues with Transforming Input (p. 70)

Predefined variables

There are pre-defined variables you can use without defining a JSON path. These variables are reserved, and you can't 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.

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

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

```
{
    "instance" : "$.detail.instance-id",
    "state" : "$.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>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 by using the EventBridge API

For information about using the EventBridge API to transform input, see [Use Input Transformer to extract data from an event and input that data to the target](#).

### Transforming input by using AWS CloudFormation

For information about using AWS CloudFormation to transform input, see [AWS::Events::Rule InputTransformer](#).

### 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 won't appear in the output.
- The JSON that is passed to the target is minified and escaped.
- EventBridge doesn't 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>{</td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>&quot;instance&quot;: &lt;instance&gt;,</td>
<td>&quot;instance&quot;: &quot;i-0123456789&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;state&quot;: &lt;state&gt;,</td>
<td>&quot;state&quot;: &quot;RUNNING&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;ruleArn&quot;: &lt;aws.events.rule-arn&gt;,</td>
<td>&quot;ruleArn&quot;: &quot;arn:aws:events:us-east-2:123456789012:rule/example&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;ruleName&quot;: &lt;aws.events.rule-name&gt;,</td>
<td>&quot;ruleName&quot;: &quot;example&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;originalEvent&quot;: &lt;aws.events.event&gt;</td>
<td>&quot;originalEvent&quot;: {}</td>
</tr>
<tr>
<td></td>
<td>}</td>
<td>// 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 archive and replay

In EventBridge, you can create an archive of events (p. 16) so that you can easily replay them at a later time. For example, you might want to replay events to recover from errors or to validate new functionality in your application.

The following video demonstrates the use of archive and replay: Creating archives and replays

Topics

- Archiving Amazon EventBridge events (p. 72)
- Replaying archived Amazon EventBridge events (p. 74)
Archiving Amazon EventBridge events

When you create an archive in EventBridge, you can determine which events (p. 16) are sent to the archive by specifying an event pattern (p. 19). EventBridge sends events that match the event pattern to the archive. You also set the 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. (Optional) 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, EventBridge deletes the events 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 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. (Optional) 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, EventBridge deletes the events 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, there are no events in the archive.
Replaying archived Amazon EventBridge events

After you create an archive, you can then replay events (p. 16) from the archive. 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 replay events, you can specify which archive to replay events from, the start and end time for the event to replay, the event bus (p. 7), or one or more rules (p. 32) to replay the events to.

Events aren’t necessarily replayed in the 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 on one 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. EventLastReplayedTime returns the time stamp of the last event replayed.

Events are replayed based on, but separate from, the PutEvents transactions per second limit for the AWS account. You can request an increase to the limit for PutEvents. For more information, see Amazon EventBridge Quotas.

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

To start an event replay

1. Open the Amazon EventBridge console at 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 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 events that occurred between the Start time and End time are replayed.

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 it as long as the status is Starting or Running.

To cancel a replay

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the left navigation pane, choose Replays.
3. Choose the replay to cancel.
4. Choose Cancel.
Amazon EventBridge schemas

A schema defines the structure of events (p. 16) that are sent to EventBridge. EventBridge provides schemas for all events that are generated by AWS services. You can also create or upload custom schemas (p. 78) or infer schemas (p. 82) directly from events on an event bus (p. 7). Once you have a schema for an event, you can download code bindings for popular programming languages and speed up development. You can work with code bindings for schemas and manage schemas from the EventBridge console, by using the API, or directly in your IDE by using the AWS toolkits. To build serverless apps that use events, use AWS Serverless Application Model.

EventBridge supports both OpenAPI 3 and JSONSchema Draft4 formats.

For AWS Toolkit for JetBrains and AWS Toolkit for VS Code, you can browse or search for schemas and download code bindings for schemas directly in your IDE.

The following video gives an overview of schemas and schema registries: Using the Schema Registry

Topics

- Finding an Amazon EventBridge schema (p. 76)
- Amazon EventBridge schema registries (p. 77)
- Creating an Amazon EventBridge schema (p. 78)
- Amazon EventBridge code bindings (p. 83)
Finding an Amazon EventBridge schema

EventBridge includes schemas (p. 75) for all AWS services that generate events. You can find these schemas in the EventBridge console, or you can find them by using the API action `SearchSchemas`.

To find schemas for AWS services in the EventBridge console

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.
   <result>
   The first page of available schemas is displayed.
   </result>
4. To find a schema, in Search AWS event schemas, enter a search term.
   A search returns matches for both the name and contents of the available schemas, and then displays which versions of the schema contain matches.
5. Open an event schema by selecting the name of the schema.
Amazon EventBridge schema registries

Schema registries are containers for schemas. Schema registries collect and organize schemas so that your schemas are in logical groups. The default schema registries are:

- **All schemas** – All the schemas from the AWS event, discovered, and custom schema registries.
- **AWS event schema registry** – The built-in schemas.
- **Discovered schema registry** – The schemas discovered by Schema discovery.

You can create custom registries to organize the schemas you create or upload.

**To create a custom registry**

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

To create a custom schema (p. 78) in your new registry, select Create custom schema. To add a schema to your registry, select that registry when you’re creating a new schema.

To create a registry by using the API, use CreateRegistry. For more information, see Amazon EventBridge Schema Registry API Reference.

For information about using the EventBridge schema registry through AWS CloudFormation, see EventSchemas Resource Type Reference in AWS CloudFormation.
Creating an Amazon EventBridge schema

You create schemas by using JSON files with either the OpenAPI Specification or the JSONSchema Draft4 specification. You can create or upload your own schemas in EventBridge by using a template or generating a schema based on the JSON of an event (p. 16). You can also infer the schema from events on an event bus (p. 7). To create a schema by using the EventBridge Schema Registry API, use the CreateSchema API action.

When you choose between OpenAPI 3 and JSONSchema Draft4 formats, consider the following differences:

- JSONSchema format supports additional keywords that aren't supported in OpenAPI, such as $schema, additionalItems.
- There are minor differences in how keywords are handled, such as type and format.
- OpenAPI doesn't 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.

We recommend using JSONSchema format to implement client-side validation so 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.

After you have a new schema, you can download code bindings (p. 83) to help create applications for events with that schema.

Topics
- Create a schema by using a template (p. 78)
- Edit a schema template directly in the console (p. 79)
- Create a schema from the JSON of an event (p. 80)
- Create a schema from events on an event bus (p. 82)

Create a schema by using a template

You can create a schema from a template or by editing a template directly in the EventBridge console. To get the template, you download it from the console. You can edit the template so that the schema matches your events. Then upload your new template through the console.

To download the schema 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.

Alternatively, you can copy the JSON template 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" 
      } 
    } 
  } 
}

To upload a schema template

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/. 
2. In the navigation pane, choose Schemas and then choose Create schema. 
3. (Optional) Select or create a schema registry. 
4. Under Schema details, enter a name for your schema. 
5. (Optional) Enter a description for your schema. 
6. For Schema type, choose either OpenAPI 3.0 or JSON Schema Draft 4. 
7. On the Create tab, in the text box, either drag your schema file to the text box, or paste the schema source. 
8. Select Create.

Edit a schema template directly in the console

To edit a schema 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 choose Create schema. 
3. (Optional) 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. (Optional) Enter a description for the schema to create.
Create a schema from the JSON of an event

If you have the JSON of an event, you can automatically create a schema for that type of event.

To create a schema based on the JSON of an event

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Schemas and then choose Create schema.
3. (Optional) Select or create a schema registry.
4. Under Schema details enter a name for your schema.
5. (Optional) Enter a description for the schema you created.
6. For Schema type, choose OpenAPI 3.0.

   You can't use JSONSchema when you create a schema from the JSON of an event.

7. Select Discover from JSON
8. In the text box under JSON, paste or drag the JSON source of an event.

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

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

10. EventBridge generates an OpenAPI schema for the event. For example, the following schema is generated for the preceding Step Functions event.

   ```json
   {
   "openapi": "3.0.0",
   "info": {
   ```
Create a schema from the JSON of an event

```json
"version": "1.0.0",
"title": "StepFunctionsExecutionStatusChange",
"paths": {},
"components": {
    "schemas": {
        "AWSEvent": {
            "type": "object",
            "required": ["detail-type", "resources", "detail", "id", "source", "time",
                          "region", "version", "account"],
            "x-amazon-events-detail-type": "Step Functions Execution Status Change",
            "x-amazon-events-source": "aws.states",
            "properties": {
                "detail": {
                    "$ref": "#/components/schemas/StepFunctionsExecutionStatusChange",
                },
                "account": {
                    "type": "string"
                },
                "detail-type": {
                    "type": "string"
                },
                "id": {
                    "type": "string"
                },
                "region": {
                    "type": "string"
                },
                "resources": {
                    "type": "array",
                    "items": {
                        "type": "string"
                    }
                },
                "source": {
                    "type": "string"
                },
                "time": {
                    "type": "string",
                    "format": "date-time"
                },
                "version": {
                    "type": "string"
                }
            }
        }
    },
    "StepFunctionsExecutionStatusChange": {
        "type": "object",
        "required": ["output", "input", "executionArn", "name", "stateMachineArn",
                      "startDate", "stopDate", "status"],
        "properties": {
            "executionArn": {
                "type": "string"
            },
            "input": {
                "type": "string"
            },
            "name": {
                "type": "string"
            },
            "output": {},
            "startDate": {
                "type": "integer",
                "format": "int64"
            },
            "stateMachineArn": {
```
"type": "string"
},
"status": {
  "type": "string"
},
"stopDate": {
  "type": "integer",
  "format": "int64"
}
}
}

11. After the schema has been generated, choose Create.

Create a schema from events on an event bus

EventBridge can infer schemas by discovering events. To infer schemas, you turn on event discovery on an event bus and every unique schema is added to the schema registry, including those for cross-account events. Schemas discovered by EventBridge appear in Discovered schemas registry on the Schemas page.

If the contents of events on the event bus change, EventBridge creates new versions of the related EventBridge schema.

Note
Enabling event discovery on an event bus can incur a cost. The first five million processed events in each month are free.

Note
EventBridge infers schemas from cross-account events by default but you can disable it by updating the cross-account property. For more information, see Discoverers in the EventBridge Schema Registry API Reference.

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. Do one of the following:
   - To enable discovery on the Default event bus, choose Start discovery.
   - To enable discovery on a Custom event bus, select the radio button for the custom event bus and then choose Start discovery.
Amazon EventBridge code bindings

You can generate code bindings for event schemas (p. 75) to speed up development in Java, Python, and TypeScript. Code bindings are available for AWS service events, schemas you create (p. 78), and for schemas you generate (p. 82) based on events (p. 16) on an event bus (p. 7). You can generate code bindings for a schema by using the EventBridge console, the EventBridge Schema Registry API, or in your IDE with an AWS toolkit.

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 want code bindings for, either by browsing the schema registries, or by searching for a schema.
4. Select the schema name.
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 downloaded file is a zip file of code bindings for the language you selected.
Events from AWS services

Many AWS services generate events (p. 16) that EventBridge receives. When an AWS service in your account emits an event, it goes to your account's default event bus.

AWS CloudTrail is a service that automatically records events such as AWS API calls. You can create EventBridge rules that use the information from CloudTrail. For more information about CloudTrail, see What is AWS CloudTrail?. For information about creating an EventBridge rule that uses CloudTrail, see Tutorial: Create an Amazon EventBridge rule for AWS CloudTrail API calls (p. 104).

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

Some occurrences in AWS services can be reported to EventBridge both by the service itself and by CloudTrail. For example, an Amazon EC2 API call that starts or stops an instance generates EventBridge events as well as events through CloudTrail.

Important
CloudTrail is extending event delivery through EventBridge to resource owners. Currently, CloudTrail supports both API callers and resource owners to receive events in their S3 buckets by creating trails, and delivers events to API callers through EventBridge. With this change, resource owners in addition to API callers will be able to monitor cross-account API calls through EventBridge. CloudTrail's integration with EventBridge provides a convenient way to set automated rules-based workflows in response to events.

The CloudTrail team began deploying this update on April 12, 2021 and expect to complete this update in all commercial regions early this month. As a result, some customers might experience an increase in the EventBridge events being delivered through EventBridge coinciding with this update. No customer action is required, but should you have any questions, please reach out to AWS Support.

You can't use AWS API call events that are larger than 256 KB in size as event patterns. For more information about the API calls that you can use, see Services Supported by CloudTrail Event History.

Note
Each AWS service that generates events sends them 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 might not be delivered. Guaranteed delivery means that all events from the service are successfully delivered to EventBridge.

This table includes a representation of the AWS services that send events to EventBridge, but it doesn't include every service.

The following table shows AWS services that generate events. Choose the service name to see more information about how that service and EventBridge work together.

<table>
<thead>
<tr>
<th>Service</th>
<th>Delivery type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon AppFlow</td>
<td>Best effort</td>
</tr>
<tr>
<td>Application Auto Scaling</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Application Cost Profiler</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon Athena</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Backup</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Batch</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>Amazon Braket</td>
<td>Guaranteed</td>
</tr>
</tbody>
</table>
## Amazon EventBridge User Guide

<table>
<thead>
<tr>
<th>Service</th>
<th>Delivery type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS Certificate Manager Private Certificate Authority</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon Chime</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS CloudTrail, events delivered by</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon CloudWatch</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>AWS CodeArtifact</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>AWS CodeBuild</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS CodeCommit</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS CodeDeploy</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Config</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon Connect</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Data Exchange</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon Data Lifecycle Manager</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS DataSync</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Elastic Beanstalk</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon Elastic Block Store</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon Elastic Block Store volume modifications</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon Elastic Compute Cloud (Amazon EC2)</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon EC2 Auto Scaling</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon EC2 Fleets</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon EC2 Spot Instance Interruption</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon Elastic Container Registry</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon Elastic Container Service</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>AWS Elemental MediaConvert</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>AWS Elemental MediaLive</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Elemental MediaPackage</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Elemental MediaStore</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>Amazon EMR</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon GameLift</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Glue</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Glue DataBrew</td>
<td>Best effort</td>
</tr>
<tr>
<td>Service</td>
<td>Delivery type</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>AWS Ground Station</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Health</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon Interactive Video Service</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS IoT Analytics</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>AWS IoT Greengrass V1</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS IoT Greengrass V2</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Key Management Service CMK deletion</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>AWS Key Management Service CMK rotation</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Key Management Service imported key material expiration</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon Location Service</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>Amazon Macie</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon Macie Classic</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon Managed Blockchain</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Managed Services</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS OpsWorks</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>AWS Proton</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon QLDB</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>Amazon Redshift</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>Amazon Relational Database Service</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Resource Access Manager</td>
<td>Best effort</td>
</tr>
<tr>
<td>Savings Plans</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Security Hub</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Signer</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>Amazon Simple Workflow Service</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Step Functions</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Storage Gateway</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>AWS Systems Manager</td>
<td>Best effort</td>
</tr>
<tr>
<td>Tag changes on resources</td>
<td>Best effort</td>
</tr>
<tr>
<td>AWS Transit Gateway</td>
<td>Best effort</td>
</tr>
<tr>
<td>Amazon Translate</td>
<td>Guaranteed</td>
</tr>
<tr>
<td>Service</td>
<td>Delivery type</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Amazon WorkSpaces</td>
<td>Best effort</td>
</tr>
</tbody>
</table>
Amazon EventBridge related services

Amazon EventBridge works with other AWS services to process events (p. 16) or invoke a resource as the target (p. 48) of a rule (p. 32). For more information about EventBridge integrations with other AWS services, see the following:

Topics
- Using Amazon EventBridge with Interface VPC Endpoints (p. 89)
- Amazon EventBridge integration with AWS X-Ray (p. 91)
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. Your resources on your VPC can use this connection to communicate with EventBridge.

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. 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, which enables private communication between AWS services using an elastic network interface with private IP addresses. For more information, see AWS PrivateLink and VPC endpoints.

When you use a private interface VPC endpoint, custom events (p. 16) your VPC sends to EventBridge use that endpoint. EventBridge then sends those events to other AWS services based on the rules (p. 32) and targets (p. 48) that you've configured. Once events are sent to another service you can receive them through 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 to receive messages from that queue in your VPC without using the public endpoint.

Availability

EventBridge currently supports VPC endpoints in the following Regions:

- US East (Ohio)
- US East (N. Virginia)
- US West (N. California)
- US West (Oregon)
- Africa (Cape Town)
- Asia Pacific (Mumbai)
- Asia Pacific (Hong Kong)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Asia Pacific (Osaka)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Milan)
- Europe (Paris)
- Europe (Stockholm)
- South America (São Paulo)
Creating a VPC Endpoint for EventBridge

To use EventBridge with your VPC, create an interface VPC endpoint for EventBridge and choose `com.amazonaws.Region.events` as the service name. For more information, see Creating an Interface Endpoint in the Amazon VPC User Guide.
Amazon EventBridge integration with AWS X-Ray

You can use AWS X-Ray to trace events (p. 16) that pass through EventBridge. EventBridge passes the original trace header to the target (p. 48) so that target services can track, analyze, and debug.

EventBridge can pass a trace header for an event only if the event came from a `PutEvents` request that passed the trace context. X-Ray doesn't trace events that originate from third-party partners, scheduled events, or AWS services (p. 84), and these event sources don't appear on your X-Ray service map.

X-Ray validates trace headers, and trace headers that aren't valid are dropped. However, the event is still processed.

**Important**
The trace header is not available on the event that's delivered to the invocation target.

- If you have an event archive (p. 72), the trace header isn't available on archived events. If you replay archived events, the trace header isn't included.
- If you have a dead-letter queue (DLQ) (p. 43), the trace header is included in the `SendMessage` request that sends the event to the DLQ. If you retrieve events (messages) from the DLQ by using `ReceiveMessage`, the trace header associated with the event is included on the Amazon SQS message attribute, but it isn't included in the event message.

For information about 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.

You can pass the following trace header information through EventBridge:

- **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 to a target. If you also use `PutEventsRequestEntry`, `PutEventsRequestEntry` overrides the HTTP trace header.

**Note**
The trace header doesn't count towards the `PutEventsRequestEntry` event size. For more information, see Calculating Amazon EventBridge `PutEvents` event entry size (p. 31).
The following video demonstrates the use of X-Ray and EventBridge together: [Using AWS X-Ray for tracing](#)
Receiving events from a SaaS partner with Amazon EventBridge

To receive events (p. 16) from SaaS partner applications and services, you need a partner event source from the partner. Then you can create a partner event bus (p. 7) and match it to the partner event source.

The following video covers SaaS integrations with EventBridge: Software as a service (SaaS) partners

Topics
• Supported SaaS partner integrations (p. 93)
• Configuring Amazon EventBridge to receive events from a SaaS integration (p. 95)
• Creating a rule that matches SaaS partner events (p. 95)
• Receiving events from Salesforce (p. 97)

Supported SaaS partner integrations

EventBridge supports the following SaaS partner integrations:
• Auth0
• Blitline
• BUIDLHub
• Buildkite
• Camunda
• CleverTap
• Datadog
• Epsagon
• Freshworks
• Genesys
• GS2
• Karte
• Kloudless
• Mackerel
• MongoDB
• New Relic
• OneLogin
• Opsgenie
• PagerDuty
• PayShield
• SailPoint
• Saviynt
• Segment
• Shopify
• SignalFx
• Site24x7
• 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>
</tbody>
</table>
Configuring 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 then choose Set up for that partner.
4. To copy your account ID to the clipboard, choose Copy.
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 using your account ID. The event source that you create is 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 then choose Associate with event bus.

The status of the event source changes from Pending to Active, and the name of the event bus updates to match the partner event source name. You can now start creating rules that match events from the partner event source. For more information, see Creating a rule that matches SaaS partner events (p. 95).

Creating a rule that matches SaaS partner events

Before you can create rules for events from SaaS partner applications and services, you need a partner event bus (p. 7). Then you can match the partner event bus to the partner event source. For more information, see Receiving events from a SaaS partner with Amazon EventBridge (p. 93).
To create a rule that matches 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 that come from this partner event source match the rule.
10. (Optional) To customize the event pattern, choose Edit, make your changes, and then choose Save.
11. For Service event bus, select the event bus that corresponds to this partner.
12. For Select targets, choose the AWS service that is to act when this type of event occurs.
13. In the other fields in this section, enter information specific to this target type, if any is needed.
14. 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
15. (Optional) Choose Add target to add another target for this rule.
16. (Optional) Enter one or more tags for the rule. For more information, see Amazon EventBridge tags (p. 205).
17. 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. Amazon AppFlow then sends Salesforce events to EventBridge by using a partner event bus.

Amazon AppFlow encapsulates events from Salesforce in an EventBridge event envelope. The following example shows a Salesforce event received by an EventBridge partner event bus.

```
{
    "version": "0",
    "id": "5c42b99e-e005-43b3-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"
        }
    }
}
```

**Step 1: Configure Amazon AppFlow to use Salesforce as a partner event source**

To send events to EventBridge, you first need to configure Amazon 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 Flow name enter a name for your flow.
3. (Optional) 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. In the Connect to Salesforce dialog box, 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.
7. In the Salesforce dialog box, do the following:
Step 2: Configure EventBridge to receive Salesforce events

- Enter your Salesforce username and password to log in to Salesforce.
- Select Salesforce events for the types of data for Amazon AppFlow to process.

8. In the **Choose Salesforce event** drop-down, select the type of event to send to EventBridge.
9. For a destination, select **Amazon EventBridge**.
10. Select **Create new partner event source**.
11. (Optional) Specify a unique suffix for the partner event source.
12. Choose **Generate partner event source**.
13. Choose an Amazon S3 bucket to store event payload files that are larger than 256 KB.
14. 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.
15. Choose **Next**.
16. For field mapping, select **Map all fields directly**. Alternatively, you can select the fields that are of interest from the **Source field name** list.

   For more information about field mapping, see **Map data fields**.
17. Choose **Next**.
18. (Optional) Configure filters for data fields in Amazon AppFlow.
19. Choose **Next**.
20. Review the settings and then 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.

**Step 2: 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 in Step 1.
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**.
7. Open the **Rules** page in the EventBridge console.
8. Choose **Create rule**.
9. Enter a unique name for the rule.
10. Choose **Event pattern** in the **Define pattern** section.
11. For **Event matching pattern**, select **Pre-defined pattern by service**.
12. For **Service provider** section, select **All Events**.
13. For **Select event bus**, choose **Custom or partner event bus**.
14. Select the event bus that you associated with the Amazon AppFlow partner event source.
15. For **Select targets**, choose the AWS service that is to act when the rule runs. One rule can have up to five targets.
16. Choose **Create**.

The target service receives all Salesforce events configured for your account. To filter the events or send some events to different targets, you can use content-based filtering with event patterns (p. 25).

**Note**
For events larger than 256KB, Amazon AppFlow doesn't send the full event to EventBridge. Instead, Amazon AppFlow puts the event into an S3 bucket in your account, and then sends an event to EventBridge with 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 (p. 32) for certain tasks and target (p. 48).

Tutorials:
- Archive and replay Amazon EventBridge events (p. 101)
- Tutorial: Create an Amazon EventBridge rule for AWS CloudTrail API calls (p. 104)
- Create an Amazon EventBridge sample application (p. 107)
- Tutorial: Download code bindings for events using the EventBridge schema registry (p. 111)
- Tutorial: Log the state of an Auto Scaling group using EventBridge (p. 112)
- Tutorial: Log AWS API calls using EventBridge (p. 115)
- Tutorial: Log the state of an Amazon EC2 instance using EventBridge (p. 118)
- Tutorial: Log Amazon EC2 instance state changes (p. 121)
- Tutorial: Log Amazon S3 object-level operations using EventBridge (p. 123)
- Tutorial: Send events to an Amazon Kinesis stream using EventBridge (p. 126)
- Tutorial: Use EventBridge to relay events to AWS Systems Manager Run Command (p. 129)
- Tutorial: Run an Amazon ECS task when a file is uploaded to an Amazon S3 bucket (p. 131)
- Tutorial: Schedule automated Amazon EBS snapshots using EventBridge (p. 133)
- Tutorial: Schedule automated builds using AWS CodeBuild (p. 135)
- Tutorial: Schedule AWS Lambda functions using EventBridge (p. 137)
- Tutorial: Set AWS Systems Manager automation as an EventBridge target (p. 141)
- Tutorial: Use input transformer to customize what EventBridge passes to the event target (p. 143)
Archive and replay Amazon EventBridge events

You can use EventBridge to route events (p. 16) to specific AWS Lambda functions using rules (p. 32).

In this tutorial, you'll create a function to use as the target for the EventBridge rule using the Lambda console. Then, you'll create an archive (p. 72) and a rule that'll archive test events using the EventBridge console. Once there are events in that archive, you'll replay (p. 74) them.

Steps:
- Step 1: Create a Lambda function (p. 101)
- Step 2: Create archive (p. 101)
- Step 3: Create rule (p. 102)
- Step 4: Send test events (p. 102)
- Step 5: Replay events (p. 103)
- Step 6: Clean up your resources (p. 103)

Step 1: Create a Lambda function

First, create a Lambda function to log the events.

To create a Lambda function:
1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. Choose Create function.
3. Choose Author from scratch.
4. Enter a name and description for the Lambda function. For example, name the function LogScheduledEvent.
5. Leave the rest of the options as the defaults and choose Create function.
6. On the Code tab of the function page, double-click index.js.
7. Replace the existing JavaScript code with the following code:

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

Step 2: Create archive

Next, create the archive that will hold all the test events.

To create an archive
1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Archives.
3. Choose Create archive.
4. Enter a name and description for the archive. For example, name the archive ArchiveTest.
5. Leave the rest of the options as the defaults and choose Next.
6. Choose Create archive.

**Step 3: Create rule**

Create a rule to archive events that are sent to the event bus.

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. For example, name the rule ARTestRule.
5. For Define pattern, do the following:
   a. Choose Event pattern.
   b. Choose Custom pattern and paste the following code into the Event pattern field:

   ```
   {
     "detail-type": [
       "customerCreated"
     ]
   }
   ```
   c. Choose Save.
6. For Select event bus, choose AWS default event bus.
7. For Target, choose Lambda function.
8. For Function, select the Lambda function that you created in the Step 1: Create a Lambda function section. In this example, select LogScheduledEvent.
9. Choose Create.

**Step 4: Send test events**

Now that you've set up the archive and the rule, we'll send test events to make sure the archive is working correctly.

**Note**
It can take some time for events to get to the archive.

To send test events (console)

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Event buses.
3. In the Default event bus tile, choose Actions, Send events.
4. Enter an event source. For example, TestEvent.
5. For Detail type, enter customerCreated.
6. For Event detail, enter {}.
7. Choose Send.
Step 5: Replay events

Once the test events are in the archive you can replay them.

To replay archived events (console)

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Replays.
3. Choose Start new replay.
4. Enter a name and description for the replay. For example, name the replay ReplayTest.
5. For Source, select the archive you created in the Step 2: Create archive section.
6. For Replay time frame, do the following.
   a. For Start time, select the date you sent test events and a time before you sent them. For example, 2021/08/11 and 08:00:00.
   b. For End time, select the current date and time. For example, 2021/08/11 and 09:15:00.
7. Choose Start Replay.

Step 6: Clean up your resources

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you are no longer using, you prevent unnecessary charges to your AWS account.

To delete the Lambda function(s)

1. Open the Functions page of the Lambda console.
2. Select the function(s) that you created.
3. Choose Actions, Delete.
4. Choose Delete.

To delete the EventBridge archives(s)

1. Open the Archives page of the EventBridge console.
2. Select the archive(s) you created.
3. Choose Delete.
4. Enter the archive name and choose Delete.

To delete the EventBridge rule(s)

1. Open the Rules page of the EventBridge console.
2. Select the rule(s) that you created.
3. Choose Delete.
4. Choose Delete.
Tutorial: Create an Amazon EventBridge rule for AWS CloudTrail API calls

To create a rule (p. 32) that triggers on an action by an AWS service that doesn't generate events (p. 16), you can match API calls made by that service instead. AWS CloudTrail records API calls for some services. For more information, see CloudTrail Supported Services and Integrations.

Rules in EventBridge only work 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 for each of those Regions, you must create a separate rule in each Region that you want to track.

All events that are delivered by 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 aren't processed by EventBridge, with the exception of events from the following AWS STS actions:

- GetFederationToken
- GetSessionToken

To record events with a detail-type value of AWS API Call via CloudTrail, a CloudTrail trail with logging enabled is required.

**Note**

It is possible to create a rule that leads to an infinite loop, where a rule runs repeatedly. To prevent infinite loops, rules must not run actions that match the same rule. For example, a rule that detects that ACLs have changed on an S3 bucket and then runs software to change them to a new state causes the same rule to run again. 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.

**Important**

CloudTrail is extending event delivery through EventBridge to resource owners. Currently, CloudTrail supports both API callers and resource owners to receive events in their S3 buckets by creating trails, and delivers events to API callers through EventBridge. With this change, resource owners in addition to API callers will be able to monitor cross-account API calls through EventBridge. CloudTrail's integration with EventBridge provides a convenient way to set automated rules-based workflows in response to events.

The CloudTrail team began deploying this update on April 12, 2021 and expect to complete this update in all commercial regions early this month. As a result, some customers might experience an increase in the EventBridge events being delivered through EventBridge coinciding with this update. No customer action is required, but should you have any questions, please reach out to AWS Support.

**Steps:**

- **Step 1:** Create a rule (p. 104)
- **Step 2:** Confirm success (p. 105)
- **Step 3:** Clean up your resources (p. 103)

**Step 1: Create a rule**

Create a rule that triggers on an API call via CloudTrail
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.

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 aren’t 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. 44).

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 tags (p. 205).
16. Choose Create.

Step 2: Confirm success

If you see the rule in the list of rules, you've successfully completed this tutorial.
Step 3: Clean up your resources

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you are no longer using, you prevent unnecessary charges to your AWS account.

To delete the EventBridge rule(s)

1. Open the Rules page of the EventBridge console.
2. Select the rule(s) that you created.
3. Choose Delete.
4. Choose Delete.
Create an Amazon EventBridge sample application

You can use EventBridge to route events (p. 16) to specific Lambda functions using rules (p. 32).

In this tutorial, you'll use the AWS CLI, Node.js, and the code in the GitHub repo to create the following:

- An AWS Lambda function that produces events for bank ATM transactions.
- Three Lambda functions to use as targets (p. 48) of an EventBridge rule.
- and the rule that routes the created events to the correct downstream function based on an event pattern (p. 19).

This example uses AWS SAM templates to define the EventBridge rules. To learn more about using AWS SAM templates with EventBridge see ?? (p. 45).

In the repo, the atmProducer subdirectory contains handler.js, which represents the ATM service producing events. This code is a Lambda handler written in Node.js, and publishes events to EventBridge via the AWS SDK using this line of JavaScript code.

```javascript
const result = await eventbridge.putEvents(params).promise()
```

This directory also contains events.js, listing several test transactions in an Entries array. A single event is defined in JavaScript as follows:

```javascript
exports.case1Handler = async (event) => {
  console.log('--- Approved transactions ---')
  console.log(JSON.stringify(event, null, 2))
}

exports.case2Handler = async (event) => {
  console.log('--- NY location transactions ---')
  console.log(JSON.stringify(event, null, 2))
}

exports.case3Handler = async (event) => {
  console.log('--- Unapproved transactions ---')
  console.log(JSON.stringify(event, null, 2))
}
```

The Detail section of the event specifies transaction attributes. These include the location of the ATM, the amount, the partner bank, and the result of the transaction.

The handler.js file in the atmConsumer subdirectory contains three functions:

```javascript
exports.case1Handler = async (event) => {
  console.log('--- Approved transactions ---')
  console.log(JSON.stringify(event, null, 2))
}

exports.case2Handler = async (event) => {
  console.log('--- NY location transactions ---')
  console.log(JSON.stringify(event, null, 2))
}

exports.case3Handler = async (event) => {
  console.log('--- Unapproved transactions ---')
  console.log(JSON.stringify(event, null, 2))
}
```
Prerequisites

Each function receives transaction events, which are logged via the `console.log` statements to Amazon CloudWatch Logs. The consumer functions operate independently of the producer and are unaware of the source of the events.

The routing logic is contained in the EventBridge rules that are deployed by the application's AWS SAM template. The rules evaluate the incoming stream of events, and route matching events to the target Lambda functions.

The rules use event patterns that are JSON objects with the same structure as the events they match. Here's the event pattern for the one of the rules.

```
{
    "detail-type": ["transaction"],
    "source": ["custom.myATMapp"],
    "detail": {
        "location": [{
            "prefix": "NY-
        }]
    }
}
```

Steps:

- Prerequisites (p. 108)
- Step 1: Create application (p. 108)
- Step 2: Run application (p. 109)
- Step 3: Check the logs and verify the application works (p. 109)
- Step 4: Clean up your resources (p. 103)

Prerequisites

To complete this tutorial, you'll need the following resources:

- An AWS account. Create an AWS account if you don't already have one.
- AWS CLI installed. To install the AWS CLI, see the Installing, updating, and uninstalling the AWS CLI version 2.
- Node.js 12.x installed. To install Node.js, see Downloads.

Step 1: Create application

To set up the example application, you'll use the AWS CLI and Git to create the AWS resources you'll need.

To create the application

1. Sign in to AWS.
2. Install Git and install the AWS Serverless Application Model CLI on your local machine.
3. Create a new directory, and then navigate to that directory in a terminal.
4. At the command line, enter `git clone https://github.com/aws-samples/amazon-eventbridge-producer-consumer-example`.
5. At the command line run the following command:
Step 2: Run application

Now that you've set up the resources, you'll use the console to test the functions.

To run the application

1. Open the Lambda console in the same Region where you deployed the AWS SAM application.
2. There are four Lambda functions with the prefix `atm-demo`. Select the `atmProducerFn` function, then choose Actions, Test.
3. Enter Test for the Name.
4. Choose Test.

Step 3: Check the logs and verify the application works

Now that you've run the application, you'll use the console to check the CloudWatch Logs.

To check the logs

1. Open the CloudWatch console in the same Region where you ran the AWS SAM application.
2. Choose Logs, and then choose Log groups.
3. Select the log group containing `atmConsumerCase1`. You see two streams representing the two transactions approved by the ATM. Choose a log stream to view the output.
4. Navigate back to the list of log groups, and then select the log group containing `atmConsumerCase2`. You'll see two streams representing the two transactions matching the New York location filter.
5. Navigate back to the list of log groups, and select the log group containing `atmConsumerCase3`. Open the stream to see the denied transactions.

Step 4: Clean up your resources

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you are no longer using, you prevent unnecessary charges to your AWS account.

To delete the EventBridge rule(s)

1. Open the Rules page of the EventBridge console.
2. Select the rule(s) that you created.
3. Choose Delete.
4. Choose Delete.

To delete the Lambda function(s)
1. Open the Functions page of the Lambda console.
2. Select the function(s) that you created.
3. Choose Actions, Delete.
4. Choose Delete.

To delete the CloudWatch Logs log group(s)
1. Open the Cloudwatch console.
2. Choose Logs, Log groups.
3. Select the log group(s) that were created in this tutorial.
4. Choose Actions, Delete log group(s).
5. Choose Delete.
Tutorial: Download code bindings for events using the EventBridge schema registry

You can generate code bindings (p. 83) for event schemas (p. 75) 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 (p. 16) on an event bus (p. 7). You can generate code bindings for a schema using one of the following:

- EventBridge console
- EventBridge schema registry API
- Your IDE with an AWS toolkit

In this tutorial you 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 the schema for the 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.
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: Log the state of an Auto Scaling group using EventBridge

You can run an AWS Lambda function that logs an events (p. 16) whenever an Auto Scaling group launches or terminates an Amazon EC2 instance that indicates whether an event was successful.

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

In this tutorial, you create a Lambda function, and you create a rule (p. 32) in the EventBridge console that calls that function when an Auto Scaling group launches or terminates an instance.

Steps:
- Step 1: Create a Lambda function (p. 112)
- Step 2: Create a rule (p. 112)
- Step 3: Test the rule (p. 113)
- Step 4: Confirm success (p. 105)
- Step 5: Clean up your resources (p. 103)

Step 1: Create a Lambda function

Create a Lambda function to log the scale-out and scale-in events for your Auto Scaling group.

To create a Lambda function

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. Choose Create function.
3. Choose Author from scratch.
4. Enter a name for the Lambda function. For example, name the function LogAutoScalingEvent.
5. Leave the rest of the options as the defaults and choose Create function.
6. On the Code tab of the function page, double-click index.js.
7. Replace the existing code with the following code.

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

Step 2: Create a rule

Create a rule to run the Lambda function you created in Step 1. The rule runs when your Auto Scaling group starts or stops 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 Auto Scaling group, choose Specific group name(s), and then 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 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. Wait a few minutes for the scale-out event to occur, and then 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 then 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.
Step 4: Confirm success

If you see the Lambda event in the CloudWatch logs, you've successfully completed this tutorial. If the event isn't in your CloudWatch logs, start troubleshooting by verifying the rule was created successfully and, if the rule looks correct, verify the code of your Lambda function is correct.

Step 5: Clean up your resources

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you are no longer using, you prevent unnecessary charges to your AWS account.

To delete the EventBridge rule(s)

1. Open the Rules page of the EventBridge console.
2. Select the rule(s) that you created.
3. Choose Delete.
4. Choose Delete.

To delete the Lambda function(s)

1. Open the Functions page of the Lambda console.
2. Select the function(s) that you created.
3. Choose Actions, Delete.
4. Choose Delete.
Tutorial: Log AWS API calls using EventBridge

You can use an AWS Lambda function to log AWS API calls. For example, you can create a rule (p. 32) to log any operation in Amazon EC2, or you can limit this rule to log only a specific API call.

In this tutorial, you create an AWS CloudTrail trail, a Lambda function, and a rule in the EventBridge console. The rule invokes the Lambda function when an Amazon EC2 instance is stopped.

Steps:
- **Step 1**: Create an AWS CloudTrail trail (p. 115)
- **Step 2**: Create an AWS Lambda function (p. 115)
- **Step 3**: Create a rule (p. 116)
- **Step 4**: Test the rule (p. 116)
- **Step 5**: Confirm success (p. 105)
- **Step 6**: Clean up your resources (p. 103)

**Step 1: Create an AWS CloudTrail trail**

If you already have a trail set up, skip to step 2.

**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.
5. For AWS KMS alias, type an alias for the KMS key.
6. Choose Next.
7. Choose Next.
8. Choose Create trail.

**Step 2: Create an AWS Lambda function**

Create a Lambda function to log the API call events.

**To create a Lambda function**

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. Choose Create function.
3. Choose Author from scratch.
4. Enter a name and description for the Lambda function. For example, name the function LogEC2StopInstance.
5. Leave the rest of the options as the defaults and choose Create function.
6. On the Code tab of the function page, double-click index.js.
7. Replace the existing code with the following code.

```javascript
'use strict';

exports.handler = (event, context, callback) => {
```
8. Choose **Deploy**.

## Step 3: Create a rule

Create a rule to run the Lambda function you created in step 2 whenever you stop an Amazon EC2 instance.

**To create a rule**

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose **Rules**.
3. Choose **Create rule**.
4. Enter a name and description for the rule.
5. For **Define pattern**, do the following:
   a. Choose **Event pattern**.
   b. Choose **Pre-defined pattern by service**.
   c. For **Service provider**, choose **AWS**.
   d. For **Service Name**, choose **EC2**.
   e. For **Event type**, choose **AWS API Call via CloudTrail**.
   f. Choose **Specific operations(s)** and enter **StopInstances** in the box.
6. For **Select event bus**, choose **AWS default event bus**. When an AWS service in your account emits an event, it 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 4: Test the rule

You can test your rule by stopping an Amazon EC2 instance using the Amazon EC2 console. Wait a few minutes for the instance to stop, and then check your AWS Lambda metrics on the CloudWatch console to verify that your function ran.

**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 then 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*.

### Step 5: Confirm success

If you see the Lambda event in the CloudWatch logs, you’ve successfully completed this tutorial. If the event isn’t in your CloudWatch logs, start troubleshooting by verifying the rule was created successfully and, if the rule looks correct, verify the code of your Lambda function is correct.

### Step 6: Clean up your resources

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you are no longer using, you prevent unnecessary charges to your AWS account.

**To delete the EventBridge rule(s)**

1. Open the **Rules page** of the EventBridge console.
2. Select the rule(s) that you created.
3. Choose **Delete**.
4. Choose **Delete**.

**To delete the Lambda function(s)**

1. Open the **Functions page** of the Lambda console.
2. Select the function(s) that you created.
3. Choose **Actions, Delete**.
4. Choose **Delete**.

**To delete the CloudTrail trail(s)**

1. Open the **Trails page** of the CloudTrail console.
2. Select the trail(s) that you created.
3. Choose **Delete**.
4. Choose **Delete**.
Tutorial: Log the state of an Amazon EC2 instance using EventBridge

You can create an AWS Lambda function that logs a state change for an Amazon EC2 instance. Then you can create a rule (p. 32) that runs your Lambda 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.

Steps:
- Step 1: Create an AWS Lambda function (p. 118)
- Step 2: Create a rule (p. 118)
- Step 3: Test the rule (p. 119)
- Step 4: Confirm success (p. 105)
- Step 5: Clean up your resources (p. 103)

Step 1: Create an AWS Lambda function

Create a Lambda function to log the state change events (p. 16). When you create your rule in Step 2, you specify this function.

To create a Lambda function
1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. Choose Create function.
3. Choose Author from scratch.
4. Enter a name and description for the Lambda function. For example, name the function LogEC2InstanceStateChange.
5. Leave the rest of the options as the defaults and choose Create function.
6. On the Code tab of the function page, double-click index.js.
7. Replace the existing code with the following code.

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

Step 2: Create a rule

Create a rule to run the Lambda function you created in Step 1. The rule runs when 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. Wait a few minutes for the instance to launch and initialize, and then verify that your Lambda function ran.

**To test your rule by launching 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*.
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 then 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, for example `/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*.

### Step 4: Confirm success

If you see the Lambda event in the CloudWatch logs, you’ve successfully completed this tutorial. If the event isn’t in your CloudWatch logs, start troubleshooting by verifying the rule was created successfully and, if the rule looks correct, verify the code of your Lambda function is correct.
Step 5: Clean up your resources

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you are no longer using, you prevent unnecessary charges to your AWS account.

To delete the EventBridge rule(s)

1. Open the Rules page of the EventBridge console.
2. Select the rule(s) that you created.
3. Choose Delete.
4. Choose Delete.

To delete the Lambda function(s)

1. Open the Functions page of the Lambda console.
2. Select the function(s) that you created.
3. Choose Actions, Delete.
4. Choose Delete.
Tutorial: Log Amazon EC2 instance state changes

In this tutorial, you create a rule (p. 32) causing state-change notifications in Amazon EC2 to be logged in Amazon CloudWatch Logs. Tracking state change notifications can help you troubleshoot issues with your applications, or warn you about things that need your attention.

Steps:
- Step 1: Create a rule (p. 121)
- Step 2: Confirm success (p. 105)
- Step 3: Clean up your resources (p. 103)

Step 1: Create a rule

Create a rule to log Amazon EC2 state-change notifications in CloudWatch Logs

To create a rule

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

Step 2: Confirm success

If you see the rule in the list of rules, you've successfully created the rule.

Step 3: Clean up your resources

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you are no longer using, you prevent unnecessary charges to your AWS account.

To delete the EventBridge rule(s)

1. Open the Rules page of the EventBridge console.
2. Select the rule(s) that you created.
3. Choose Delete.
4. Choose Delete.
Tutorial: Log Amazon S3 object-level operations using EventBridge

You can log the object-level API operations on your Amazon S3 buckets. Before Amazon EventBridge can match these events (p. 16), you must use AWS CloudTrail to set up and configure a trail to receive these events.

In this tutorial, you create CloudTrail trail, create a AWS Lambda function, and then create rule (p. 32) in the EventBridge console that invokes that function in response to an S3 data event.

Steps:
- Step 1: Configure your AWS CloudTrail trail (p. 123)
- Step 2: Create an AWS Lambda function (p. 123)
- Step 3: Create a Rule (p. 124)
- Step 4: Test the Rule (p. 124)
- Step 5: Confirm success (p. 105)
- Step 6: Clean up your resources (p. 103)

Step 1: Configure your AWS CloudTrail trail

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

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

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.
5. For AWS KMS alias, type an alias for the KMS key.
6. Choose Next.
7. For Event type, choose Data events
8. For Data events, do one of the following:
   - To log data events for all Amazon S3 objects in a bucket, specify an S3 bucket and an empty prefix. When an event occurs on an object in that bucket, the trail processes and logs the event.
   - To log data events for specific Amazon S3 objects in a bucket, specify an S3 bucket and 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.
9. For each resource, choose whether to log Read events, Write events, or both.
10. Choose Next.
11. Choose Create trail.

Step 2: Create an AWS Lambda function

Create a Lambda function to log data events for your S3 buckets.
To create a Lambda function

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. Choose Create function.
3. Choose Author from scratch.
4. Enter a name and description for the Lambda function. For example, name the function LogS3DataEvents.
5. Leave the rest of the options as the defaults and choose Create function.
6. On the Code tab of the function page, double-click index.js.
7. Replace the existing code with the following code.

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

### Step 3: Create a Rule

Create a rule to run the Lambda function you created in Step 2. This rule runs in response to an Amazon S3 data event.

To create a rule

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

Step 5: Confirm success

If you see the Lambda event in the CloudWatch logs, you've successfully completed this tutorial. If the event isn't in your CloudWatch logs, start troubleshooting by verifying the rule was created successfully and, if the rule looks correct, verify the code of your Lambda function is correct.

Step 6: Clean up your resources

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you are no longer using, you prevent unnecessary charges to your AWS account.

To delete the EventBridge rule(s)

1. Open the Rules page of the EventBridge console.
2. Select the rule(s) that you created.
3. Choose Delete.
4. Choose Delete.

To delete the Lambda function(s)

1. Open the Functions page of the Lambda console.
2. Select the function(s) that you created.
3. Choose Actions, Delete.
4. Choose Delete.

To delete the CloudTrail trail(s)

1. Open the Trails page of the CloudTrail console.
2. Select the trail(s) that you created.
3. Choose Delete.
4. Choose Delete.
Tutorial: Send events to an Amazon Kinesis stream using EventBridge

You can send AWS API call events (p. 16) in EventBridge to an Amazon Kinesis stream, create Kinesis Data Streams applications, and process large amounts of data. In this tutorial, you create a Kinesis stream, and then create a rule (p. 32) in the EventBridge console that sends events to that stream when an Amazon EC2 instance stops.

Steps:
- Prerequisites (p. 126)
- Step 1: Create an Amazon Kinesis stream (p. 126)
- Step 2: Create a rule (p. 126)
- Step 3: Test the rule (p. 127)
- Step 4: Verify that the event was sent (p. 127)
- Step 5: Clean up your resources (p. 103)

Prerequisites

In this tutorial you use the AWS CLI to work with Kinesis streams.

To install the AWS CLI, see the Installing, updating, and uninstalling the AWS CLI version 2.

Step 1: Create an Amazon Kinesis stream

To create a stream, at a command prompt, use the create-stream AWS CLI command.

```bash
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.

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

Step 2: Create a rule

Create a rule to send events to your stream when you stop an Amazon EC2 instance.

To create a rule

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Event pattern.
   b. Choose Pre-defined pattern by service.
   c. For Service provider, choose AWS.
d. For **Service Name**, choose **EC2**.
e. For **Event type**, choose **Instance State-change Notification**.
f. Choose **Specific state(s)**, **running**.

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

7. For **Targets**, choose **Kinesis stream**.

8. For **Stream**, select the stream that you created in step 1.

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. Wait a few minutes for the instance to stop, and then check your CloudWatch metrics to verify that your function ran.

**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.
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 was sent

You can use the AWS CLI to get the record from the stream to verify that the event was sent.

**To get the record**

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

```
aws kinesis get-shard-iterator --shard-id shardId-000000000000 --shard-iterator-type TRIM_HORIZON --stream-name test
```

The following is example output.

```
{
   "ShardIterator": "AAAAAAAAAHSywljv0zEgPX4NyKdZ5wryMzP9yALa8NeKbUjp1IXtZs1Sp+KEd9I6AJ9ZG4lNR1EMi+9md/mWvLYxp8hEzYvkT24D9DQVz/mBYNK66OTZkN9gd+efGN2aHFdkhi1rJ14BL9Wyrk+ghYg2g2T1da2EyNSH1+LaAI33gQwejJADBdyMwlo5r6PqcP2dzhe"
}
```

2. To get the record, use the following `get-records` command. Use the shard iterator from the output in the previous step.

```
aws kinesis get-records --shard-iterator AAAAAAAAAAAAHSywljv0zEgPX4NyKdZ5wryMzP9yALa8NeKbUjp1IXtZs1Sp+KEd9I6AJ9ZG4lNR1EMi
```
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`.

3. Records in Kinesis are encoded in Base64. Use a Base64 decoder to decode the data so that you can verify that it's the event that was sent to the stream in JSON form.

**Step 5: Clean up your resources**

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you are no longer using, you prevent unnecessary charges to your AWS account.

**To delete the EventBridge rule(s)**

1. Open the Rules page of the EventBridge console.
2. Select the rule(s) that you created.
3. Choose Delete.
4. Choose Delete.

**To delete the Kinesis stream(s)**

1. Open the Data streams page of the Kinesis console.
2. Select the stream(s) that you created.
3. Choose Actions, Delete.
4. Enter delete in the field and choose Delete.
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 specific events (p. 16) happen. In this tutorial, you 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.

**Note**
This tutorial assumes that you assigned a tag to the Amazon EC2 Auto Scaling group, with environment as the key and production as the value.

**Steps:**
- Step 1: Create a Rule (p. 129)
- Step 2: Confirm success (p. 105)
- Step 3: Clean up your resources (p. 103)

**Step 1: Create a Rule**
Create a rule to run shell commands and configure new Amazon EC2 instances.

**To create the EventBridge rule**

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Event pattern.
   b. Choose Pre-defined pattern by service.
   c. For Service provider, choose AWS.
   d. For Service Name, choose Auto Scaling.
   e. For Event type, choose Instance Launch and Terminate.
   f. Choose Specific instance event(s), EC2 Instance-launch Lifecycle Action.
   g. By default, the rule matches any Amazon EC2 Auto Scaling group in the Region. To make the rule match a specific group, choose Specific group name(s) and select one or more groups.
6. For Select event bus, choose AWS default event bus. When an AWS service in your account emits an event, it goes to your account’s default event bus.
7. For Target, select SSM Run Command.
8. For Document, choose AWS-RunShellScript.
9. For Target key, type tag:environment. For Target value(s), enter production and choose Add.
10. Under Configure automation parameter(s), do the following:
    a. Choose Constant.
    b. For Commands, enter a shell command and choose Add. Repeat this step for all commands you want to run when an instance launches.
    c. If necessary, enter the appropriate information in WorkingDirectory and ExecutionTimeout.
11. Do one of the following:
• To create an IAM role automatically, choose **Create a new role for this specific resource.**
  EventBridge creates the IAM role needed for your event to run.
• To use an IAM role that you created before, choose **Use existing role**

12. Choose **Create rule.**

**Step 2: Confirm success**

If you see the rule in the list of rules, you've successfully created the rule.

**Step 3: Clean up your resources**

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you are no longer using, you prevent unnecessary charges to your AWS account.

**To delete the EventBridge rule(s)**

1. Open the **Rules page** of the EventBridge console.
2. Select the rule(s) that you created.
3. Choose **Delete.**
4. Choose **Delete.**
Tutorial: Run an Amazon ECS task when a file is uploaded to an Amazon S3 bucket

AWS Systems Manager You can use EventBridge to run Amazon ECS tasks when specific AWS events (p. 16) occur.

In this tutorial, you set up an EventBridge rule (p. 32) that runs an Amazon ECS task when a file is uploaded to a specific Amazon S3 bucket by using the Amazon S3 PUT operation.

**Note**

This tutorial assumes that you have already created the task definition in Amazon ECS. For more information, see Amazon ECS task definitions.

**Steps:**

- Step 1: Create a Rule (p. 131)
- Step 2: Confirm success (p. 105)
- Step 3: Clean up your resources (p. 103)

**Step 1: Create a Rule**

Create a rule to run an Amazon ECS task when a file is uploaded to an S3 bucket using the PUT operation.

**To create a rule**

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Event pattern.
   b. Choose Pre-defined pattern by service.
   c. For Service provider, choose AWS.
   d. For Service Name, choose Simple Storage Service (S3).
   e. For Event type, choose Object Level Operations.
   f. Choose Specific operation(s), 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 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 appears 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, EventBridge uses the latest revision.
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. Do one of the following:

- 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**.

### Step 2: Confirm success

If you see the rule in the list of rules, you've successfully created the rule.

### Step 3: Clean up your resources

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you are no longer using, you prevent unnecessary charges to your AWS account.

**To delete the EventBridge rule(s)**

1. Open the **Rules page** of the EventBridge console.
2. Select the rule(s) that you created.
3. Choose **Delete**.
4. Choose **Delete**.
Tutorial: Schedule automated Amazon EBS snapshots using EventBridge

You can run EventBridge rules (p. 32) on a schedule. In this tutorial, you create a 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 create the snapshot at a specific time of day.

Important
To create rules with built-in targets (p. 48), you must use the AWS Management Console.

Steps:
• Step 1: Create the rule (p. 133)
• Step 2: Test the rule (p. 133)
• Step 3: Confirm success (p. 105)
• Step 4: Clean up your resources (p. 103)

Step 1: Create the 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 Creating an Amazon EventBridge rule that runs on a schedule (p. 36).

To create a rule
1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Schedule.
   b. Do one of the following:
      • Choose Fixed rate every and specify the schedule interval, for example, 5 minutes.
      • Choose Cron expression and specify a cron expression, for example, every 15 minutes, Monday through Friday, starting at the current time.
6. For Select event bus, choose AWS default event bus. You can only use the default event bus for scheduled rules.
7. For Target, choose select EC2 CreateSnapshot API call.
8. For Volume ID, enter the volume ID of the Amazon EBS volume.
9. Choose Create a new role for this specific resource. The new role grants the target permissions to access resources on your behalf.
10. Choose Create.

Step 2: Test the rule

You can verify your rule works 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.

Step 3: Confirm success

If you see the snapshot in the list, you’ve successfully completed this tutorial. If the snapshot isn’t in the list, start troubleshooting by verifying the rule was created successfully.

Step 4: Clean up your resources

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you are no longer using, you prevent unnecessary charges to your AWS account.

To delete the EventBridge rule(s)

1. Open the Rules page of the EventBridge console.
2. Select the rule(s) that you created.
3. Choose Delete.
4. Choose Delete.
Tutorial: Schedule automated builds using AWS CodeBuild

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

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, but you can specify the project using the ARN in Project ARN.

Steps:
- Step 1: Create a Rule (p. 135)
- Step 2: Confirm success (p. 105)
- Step 3: Clean up your resources (p. 103)

Step 1: Create a Rule

Create a rule to schedule a CodeBuild project build.

To create a rule
1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose Create rule.
4. Enter a name and description for the rule.
5. For Define pattern, do the following:
   a. Choose Schedule.
   b. Choose 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 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. (Optional) To pass a parameter to CodeBuild, do the following:
   a. Choose Configure input and then choose Constant (JSON text).
   b. In the box under Constant (JSON text), enter the following to set the timeout override to 30 minutes for these scheduled builds: { "timeoutInMinutesOverride": 30 }.
10. Do one of the following:
   - 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.
Step 2: Confirm success

If you see the rule in the list of rules, you've successfully created the rule.

Step 3: Clean up your resources

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you are no longer using, you prevent unnecessary charges to your AWS account.

To delete the EventBridge rule(s)

1. Open the Rules page of the EventBridge console.
2. Select the rule(s) that you created.
3. Choose Delete.
4. Choose Delete.
Tutorial: Schedule AWS Lambda functions using EventBridge

You can set up a rule (p. 32) 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 Installing, updating, and uninstalling the AWS CLI version 2.

For schedules, EventBridge doesn't provide second-level precision in schedule expressions (p. 36). The finest resolution using a cron expression is one minute. Due to the distributed nature of EventBridge and the target services, there can be a delay of several seconds between the time the scheduled rule is triggered and the time the target service runs the target resource.

Steps:
- Step 1: Create an AWS Lambda function (p. 101)
- Step 2: Create a Rule (p. 137)
- Step 3: Verify the rule (p. 139)
- Step 4: Confirm success (p. 105)
- Step 5: Clean up your resources (p. 103)

Step 1: Create an AWS Lambda function

Create a Lambda function to log the scheduled events.

To create a Lambda function

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. Choose Create function.
3. Choose Author from scratch.
4. Enter a name and description for the Lambda function. For example, name the function LogScheduledEvent.
5. Leave the rest of the options as the defaults and choose Create function.
6. On the Code tab of the function page, double-click index.js.
7. Replace the existing code with the following code.

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

Step 2: Create a Rule

Create a rule to run the Lambda function you created in step 1 on a schedule.

You can use either the console or the AWS CLI to create the rule. To use the AWS CLI, you first 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 (console)

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

To create a rule (AWS CLI)

1. To create a rule that runs on a schedule, use the put-rule command.

   ```bash
   aws events put-rule \
   --name my-scheduled-rule \
   --schedule-expression 'rate(5 minutes)'
   
   When this rule runs, it creates an event and then sends it to the targets. The following is an example event.
   
   ```json
   {
      "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. To grant the EventBridge service principal (events.amazonaws.com) permission to run the rule, use the add-permission command.

   ```bash
   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. To add the Lambda function that you created in step 1 to the rule, use the put-targets command.

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

   4. Create the file targets.json with the following contents.
Step 3: Verify the rule

Wait at least five minutes after completing step 2, and then you can verify that your Lambda function was invoked.

To test your rule

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, choose Rules.
3. Choose the name of the rule that you created in step 2, and then 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.

Step 4: Confirm success

If you see the Lambda event in the CloudWatch logs, you've successfully completed this tutorial. If the event isn't in your CloudWatch logs, start troubleshooting by verifying the rule was created successfully and, if the rule looks correct, verify the code of your Lambda function is correct.

Step 5: Clean up your resources

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you are no longer using, you prevent unnecessary charges to your AWS account.

To delete the EventBridge rule(s)

1. Open the Rules page of the EventBridge console.
2. Select the rule(s) that you created.
3. Choose Delete.
4. Choose Delete.

To delete the Lambda function(s)

1. Open the Functions page of the Lambda console.
2. Select the function(s) that you created.
3. Choose Actions, Delete.
4. Choose Delete.
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 specific events (p. 16) are detected. This tutorial assumes that you're invoking Systems Manager Automation based on specific events.

Steps:
- Step 1: Create a Rule (p. 141)
- Step 2: Confirm success (p. 105)
- Step 3: Clean up your resources (p. 103)

Step 1: Create a Rule

Create a rule to invoke Systems Manager Automation.

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. 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 goes to your account's default event bus.
7. For Target, choose SSM Automation.
8. For Document, choose the Systems Manager document to run.
9. (Optional) To specify a 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. Do one of the following:
   - To create an IAM role automatically, choose Create a new role for this specific resource. EventBridge creates the IAM role needed for your event to run.
   - To use an IAM role that you created before, choose Use existing role
12. Choose Create.

Step 2: Confirm success

If you see the rule in the list of rules, you've successfully created the rule.
Step 3: Clean up your resources

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you are no longer using, you prevent unnecessary charges to your AWS account.

To delete the EventBridge rule(s)

1. Open the Rules page of the EventBridge console.
2. Select the rule(s) that you created.
3. Choose Delete.
4. Choose Delete.
Tutorial: Use input transformer to customize what EventBridge passes to the event target

You can use the Input transformer (p. 68) in EventBridge to customize text from an event (p. 16) before you send it to the target of a rule (p. 32).

To do this, you define JSON paths from the event and assign their outputs to different variables. Then you can use those variables in the input template. The characters < and > can't be escaped. For more information, see Transforming Amazon EventBridge target input (p. 68)

Note
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, you create a rule that runs when any instance changes state. You extract the instance-id and state of an Amazon EC2 instance from the event. Then you use the input transformer to put that data into the message that goes to an Amazon SNS topic. The

For example, the rule in this tutorial matches the following Amazon EC2 instance state-change notification event.

```
{
  "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"}
}
```

The input transformer maps the instance variable to the $.detail.instance-id JSON path from the event, and the state variable to the $.detail.state JSON path. Then EventBridge puts the variables into the input template "The EC2 instance <instance> has changed state to <state>." The result is the following Amazon SNS message.

The EC2 instance i-1234567890abcdef0 has changed state to stopped.

Steps:
- Step 1: Create a rule (p. 143)
- Step 2: Confirm success (p. 105)
- Step 3: Clean up your resources (p. 103)

Step 1: Create a rule

Create a rule to use the input transformer to customize the instance state information that goes to a target.
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. 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 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.

Step 2: Confirm success

If you see the rule in the list of rules, you've successfully created the rule.

Step 3: Clean up your resources

You can now delete the resources that you created for this tutorial, unless you want to retain them. By deleting AWS resources that you are no longer using, you prevent unnecessary charges to your AWS account.

To delete the EventBridge rule(s)

1. Open the Rules page of the EventBridge console.
2. Select the rule(s) that you created.
3. Choose Delete.
4. Choose Delete.
Amazon EventBridge security

Amazon EventBridge uses AWS Identity and Access Management 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. 146)
- Tag-based policies (p. 147)
- Amazon EventBridge and AWS Identity and Access Management (p. 148)
- Log and monitor in Amazon EventBridge (p. 184)
- Compliance validation in Amazon EventBridge (p. 186)
- Amazon EventBridge resilience (p. 187)
- Infrastructure security in Amazon EventBridge (p. 188)
- Configuration and vulnerability analysis in Amazon EventBridge (p. 189)
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 confidential or sensitive information, such as your customers' email addresses, into tags or 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 tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Encryption at rest

EventBridge encrypts event metadata and message data that it stores. By default, EventBridge encrypts data using 256-bit Advanced Encryption Standard (AES-256) under an AWS owned key, which helps secure your data from unauthorized access. There is no additional charge for encrypting your data by using the AWS owned key.

Encryption in transit

EventBridge encrypts data that passes between EventBridge and other services by using Transport layer Security (TLS).
Tag-based policies

In Amazon EventBridge, you can use policies based on tags to control access to resources.

For example, you could restrict access to resources that include a tag with the key `environment` and the value `production`. The following example policy denies any resource with this tag the ability to create, delete, or modify tags, rules, or event buses for resources that have been tagged `environment/production`.

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

For more information about tagging, see the following.

- Amazon EventBridge tags (p. 205)
- Controlling Access Using IAM Tags
Amazon EventBridge and AWS Identity and Access Management

To access Amazon EventBridge, you need credentials that AWS can use to authenticate your requests. Your 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.

**Topics**
- Authentication (p. 148)
- Access control (p. 149)
- Managing access permissions to your Amazon EventBridge resources (p. 150)
- Using identity-based policies (IAM policies) for Amazon EventBridge (p. 154)
- Using resource-based policies for Amazon EventBridge (p. 162)
- Resource-based policies for Amazon EventBridge schemas (p. 167)
- Amazon EventBridge permissions reference (p. 170)
- Using IAM policy conditions for fine-grained access control (p. 172)

**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 permissions, for example, permission 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 to cryptographically sign your request, either through one of the SDKs or by using the AWS Command Line Interface (AWS CLI). If you don’t use AWS tools, you must sign the request yourself with 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. Using an IAM role, you can obtain temporary access keys 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 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 the user requests access 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 permission 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 an AWS service permission to access your account's resources. For example, you can create a role that allows Amazon Redshift to load data stored in an Amazon S3 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** – For Amazon EC2 applications that need access to EventBridge, you can either store access keys in the EC2 instance or you can use an IAM role to manage temporary credentials. To assign an AWS role to an EC2 instance, you create an instance profile that is attached to the instance. An instance profile contains the role, and it provides temporary credentials to applications running on the EC2 instance. For more information, see Using Roles for Applications on Amazon EC2 in the IAM User Guide.

## Access control

To create or access EventBridge resources, you need both valid credentials and permissions. For example, to invoke AWS Lambda, Amazon Simple Notification Service (Amazon SNS), and Amazon Simple Queue Service (Amazon SQS) targets, you must have permissions to those services.
Managing access permissions to your Amazon EventBridge resources

You manage access to EventBridge resources such as rules (p. 32) or events (p. 16) by using identity-based (p. 154) or resource-based (p. 162) policies.

EventBridge resources

EventBridge resources and subresources have unique Amazon Resource Names (ARNs) associated with them. You use ARNs in EventBridge to create event patterns. For more information about ARNs, see Amazon Resource Names (ARN) and AWS Service Namespaces in the Amazon Web Services General Reference.

For a list of operations EventBridge provides for working with resources, see Amazon EventBridge permissions reference (p. 170).

**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 (p. 19) 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.

The following table shows the resources in EventBridge.

<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</td>
<td>arn:aws:events:region:account:*</td>
</tr>
<tr>
<td>account in the specified Region</td>
<td></td>
</tr>
</tbody>
</table>

The following example shows how to indicate a specific rule (myRule) in your statement using its ARN.

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

To 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": "*"
To specify multiple resources or PutTargets in a single statement, separate their ARNs with commas as follows.

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

### Resource ownership

An account owns the resources in the account, no matter who creates the resources. The resource owner is the account of the principal entity, the account root user, an IAM user, or an IAM role that authenticates the request to create the resource. 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. In EventBridge, you can use both identity-based (IAM policies) and resource-based policies.

**Topics**

- Identity-based policies (IAM policies) (p. 151)
- Resource-based policies (IAM policies) (p. 152)

### 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 permission to view rules in the Amazon CloudWatch console, 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 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 permission 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 permission 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 Managing access permissions to your Amazon EventBridge resources (p. 150).

Resource-based policies (IAM policies)

When a rule runs in EventBridge, all of the targets (p. 48) associated with the rule are invoked, which means invoking the AWS Lambda functions, publishing to the Amazon SNS topics, or relaying the event to the Amazon Kinesis streams. To make API calls on the resources that you own, EventBridge needs the appropriate permission. For Lambda, Amazon SNS, and Amazon SQS resources, EventBridge uses resource-based policies. For Kinesis streams, EventBridge uses 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. 162).

Specifying policy elements: actions, effects, and principals

For each EventBridge resource, EventBridge 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 require permissions for more than one action to perform the API operation. For more information about resources and API operations, see EventBridge resources (p. 150) and Amazon EventBridge permissions reference (p. 170).

The following are the basic policy elements:

- **Resource** – Use an Amazon Resource Name (ARN) to identify the resource that the policy applies to. For more information, see EventBridge resources (p. 150).
- **Action** – Use keywords to identify resource operations that you want to allow or deny. For example, the events:Describe permission allows the user to perform the Describe operation.
- **Effect** – Specify either allow or deny. If you don’t explicitly grant access to (allow) a resource, access is denied. You can also explicitly deny access to a resource, which you 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).

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

For information about EventBridge API actions and the resources that they apply to, see Amazon EventBridge permissions reference (p. 170).

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 define conditions, you use condition keys. There are AWS condition keys and EventBridge specific keys that you can use as appropriate. For a complete list of AWS 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. 172).
Using identity-based policies (IAM policies) for Amazon EventBridge

Identity-based policies are permissions policies that you attach to IAM identities.

Topics
- Permissions required to use EventBridge (p. 154)
- AWS managed policies for EventBridge (p. 155)
- Permissions required for EventBridge to access targets using IAM roles (p. 157)
- Customer-managed policy examples (p. 159)

Permissions required to use EventBridge

For a user to work with the EventBridge console or API, that user must have a minimum set of permissions to access other AWS resources. Other AWS services can send events (p. 16) to EventBridge or be a target (p. 48) of an EventBridge rule (p. 32). The following list shows examples of AWS services and their corresponding minimum permissions:

- Automation
  - automation:CreateAction
  - automation:DescribeAction
  - automation:UpdateAction
- Amazon EC2 Auto Scaling
  - autoscaling:DescribeAutoScalingGroups
- AWS CloudTrail
  - cloudtrail:DescribeTrails
- Amazon EC2
  - ec2:DescribeInstances
  - ec2:DescribeVolumes
- EventBridge
  - events:DeleteRule
  - events:DescribeRule
  - events:DisableRule
  - events:EnableRule
  - events:ListRuleNamesByTarget
  - events:ListRules
  - events:ListTargetsByRule
  - events:PutEvents
  - events:PutRule
  - events:PutTargets
  - events:RemoveTargets
  - events:TestEventPattern
- IAM
  - iam:ListRoles
- Kinesis
  - kinesis:ListStreams
Using identity-based policies (IAM policies)

- Lambda
  - lambda:AddPermission
  - lambda:ListFunctions
  - lambda:RemovePermission
- Amazon SNS
  - sns:GetTopicAttributes
  - sns:ListTopics
  - sns:SetTopicAttributes
- Amazon SWF
  - swf:DescribeAction
  - swf:ReferenceAction
  - swf:RegisterAction
  - swf:RegisterDomain
  - swf:UpdateAction

If you create an IAM policy that is more restrictive than the minimum required permissions, the EventBridge console won't function as intended for users with that IAM policy. To ensure that those users can still use the EventBridge console, also attach the AmazonEventBridgeReadOnlyAccess managed policy to the user, as described in AWS managed policies for EventBridge (p. 155).

You don't need to allow minimum permissions for users that are making calls only to the AWS CLI.

**AWS managed policies for EventBridge**

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. Managed, or predefined, policies grant the necessary permissions for common use cases, so you don’t need to investigate what permissions are needed. For more information, see AWS managed policies in the IAM User Guide.

The following AWS managed policies that 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**

The AmazonEventBridgeFullAccess 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** – EventBridge requires this permission to create the service role in your account for API destinations. This permission grants only the IAM service permissions to create a role in your account specifically for API destinations.
- **iam:PassRole** – EventBridge requires this permission to pass an invocation role to EventBridge to invoke the target of a rule.
- **Secrets Manager permissions** – EventBridge requires these permissions to manage secrets in your account when you provide credentials through the connection resource to authorize API Destinations.
The following JSON shows 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",
        "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**

To relay events to targets, EventBridge needs an IAM role.

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

2. To create an IAM role, follow the steps in Creating a Role to Delegate Permissions to an AWS Service in the [IAM User Guide](https://docs.aws.amazon.com/IAM/latest/UserGuide/). As you follow the steps, 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](https://docs.aws.amazon.com/IAM/latest/UserGuide/Overview.html) in the [IAM User Guide](https://docs.aws.amazon.com/IAM/latest/UserGuide/).
Permissions required for EventBridge to access targets using IAM roles

For EventBridge to access targets that are an API destination, a Kinesis stream, a Systems Manager Run Command, an AWS Step Functions state machine, or an Amazon Elastic Container Service task, you must specify an IAM role for accessing that target, and the role must have a certain policy attached.

If the target is an API destination, the role that you specify must include the following policy.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "events:InvokeApiDestination" ],
      "Resource": [ "arn:aws:events:::api-destination/*" ]
    }
  ]
}
```

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 specify 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",
    }
  ]
}
```

If the target is Systems Manager run command, and you specify one or more tags for the command, the role that you specify must include the following policy.
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 Amazon 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" } }
  },
  {
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": [ "arn:aws:ec2:*:account-id:instance/*" ],
    "Condition": { "StringEquals": { "ec2:ResourceTag/": [ "[tagValues][]" ] } }
  },
  {
    "Action": "ssm:SendCommand",
    "Effect": "Allow",
    "Resource": [ "arn:aws:ssm:*:document/documentName" ]
  }]
}
```
Customer-managed policy examples

The following examples show user policies that grant permissions for EventBridge actions. These policies work when you use 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 that you need to replace to use these policies.

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

**Example 1: Access to Amazon EC2 targets**

The following policy allows built-in targets in EventBridge to perform Amazon EC2 actions on your behalf. You need to use the AWS Management Console to create rules with built-in targets.

```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 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 allows IAM users to use the EventBridge console.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ConsoleAccess",
      "Effect": "Allow",
      "Action": [
        "automation:CreateAction",
        "automation:DescribeAction",
        "automation:UpdateAction",
        "autoscaling:DescribeAutoScalingGroups",
        "cloudtrail:DescribeTrails",
        "ec2:DescribeInstances",
        "ec2:DescribeVolumes",
        "events:*",
        "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"
      ],
      "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 4: EventBridgeFullAccess

The following policy allows all AWS resources to perform actions against EventBridge through the AWS CLI and the SDK.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FullAccess",
      "Effect": "Allow",
      "Action": "events:*",
      "Resource": "*"
    }
  ]
}
```
Example 5: ReadOnlyAccess

The following policy allows all AWS resources to have read-only access to EventBridge.

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

Example 6: Using tagging to control access to rules

You can grant users access to specific EventBridge rules while preventing them from accessing other rules. To do so, you tag both sets of rules and then use IAM policies that refer to those tags. For more information about tagging EventBridge resources, see Amazon EventBridge tags (p. 205).

You can grant an IAM policy to a user to allow access to only the rules with a particular tag. You choose which rules to grant access to by tagging them with that particular tag. For example, the following policy grants a user access to rules with the value of Prod for the tag key Stack.

```json
{
  "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 (p. 32) runs in EventBridge, all of the targets (p. 48) associated with the rule are invoked. Rules can invoke AWS Lambda functions, publish to Amazon SNS topics, or relay the event to Kinesis streams. 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 uses resource-based policies. For Kinesis streams, EventBridge uses identity-based (p. 154) policies.

You 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
- Amazon API Gateway permissions (p. 162)
- CloudWatch Logs permissions (p. 162)
- AWS Lambda permissions (p. 163)
- Amazon SNS permissions (p. 164)
- Amazon SQS permissions (p. 165)

Amazon API Gateway permissions

To invoke your Amazon API Gateway endpoint by using an EventBridge rule, add the following permission to the policy of your API Gateway endpoint.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Action": "execute-api:Invoke",
      "Condition": {
        " ArnEquals": {
          "aws:SourceArn": "arn:aws:events:region:account-id:rule/rule-name"
        },
        " Resource": [
          "execute-api:/stage/GET/api"
        ]
      }
    }
  ]
}
```

CloudWatch Logs permissions

When CloudWatch Logs is the target of a rule, EventBridge creates log streams, and CloudWatch Logs stores the text from the 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, the resource-based policy is created automatically. If you use the AWS CLI to add the target, and the policy doesn’t already exist, you must create it.
The following example allows EventBridge to write to all log groups that have names that start with /aws/events/. If you use a different naming policy for these types of logs, adjust the example accordingly.

```json
{
    "Statement": [
        {
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": ["events.amazonaws.com", "delivery.logs.amazonaws.com"]
            },
            "Resource": "arn:aws:logs:region:account:log-group:/aws/events/*:*",
            "Sid": "TrustEventsToStoreLogEvent"
        }
    ],
    "Version": "2012-10-17"
}
```

For more information, see `PutResourcePolicy` in the *CloudWatch Logs API Reference guide*.

### AWS Lambda permissions

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

```json
{
    "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 the above permissions that enable EventBridge to invoke Lambda functions using the AWS CLI

- 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 the `aws sns set-topic-attributes` commands.

Note
You can't use of `Condition` blocks in Amazon SNS topic policies for EventBridge.

To add permissions that enable EventBridge to publish SNS topics

1. To list the attributes of an SNS topic, use the following command.

```
```

The following example shows the result of a new 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",
        "Policy": {
            "Version": "2012-10-17",
            "Id": "__default_policy_ID",
            "Statement": [{
                "Sid": "__default_statement_ID",
                "Effect": "Allow",
                "Principal": {
                    "Service": "events.amazonaws.com"
                },
                "Condition": {
                    "StringEquals": {
                        "AWS:SourceOwner": "account-id"
                    }
                }
            }]
        },
        "SubscriptionsPending": "0"
    }
}
```

2. Use a JSON to string converter to convert the following statement to a string.

```
{
    "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 looks like the following example.

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

3. Add the string you created in the previous step to the "Statement" collection inside the "Policy" attribute.

4. Use the `aws sns set-topic-attributes` command to set the new policy.
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
4. Create a file called `set-queue-attributes.json` with the following content.

```json
{
}
```

5. Set the policy attribute by using the `set-queue-attributes.json` file you just created as the input, as shown in the following command.

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

For more information, see Amazon SQS Policy Examples in the Amazon Simple Queue Service Developer Guide.
Resource-based policies for Amazon EventBridge schemas

The EventBridge schema registry (p. 77) supports resource-based policies (p. 162). A resource-based 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 about 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

You can use the following APIs 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, you must always attach a resource-based policy to a registry. To grant access to a schema, you specify the schema ARN and 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, replacing the "Principal" with 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": [
```
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 use resource-based policies with the EventBridge schema registry to grant access to an organization. For more information, see the AWS Organizations User Guide. The following example grants organization with an ID of o-a1b2c3d4e5 access to the schema registry.

```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*"
            ]
        }
    ]
}
```
"arn:aws:schemas:us-east-1:012345678901:schema/default*"
],
"Condition": {
  "StringEquals": {
    "aws:PrincipalOrgID": [
      "o-a1b2c3d4e5"
    ]
  }
}
}
Amazon EventBridge permissions reference

To specify an action in an EventBridge policy, use the `events:` prefix followed by the API operation name, as shown in the following example.

```
"Action": "events:PutRule"
```

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

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

To specify multiple actions, you can also use wildcards. For example, you can specify all actions that begin 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 EventBridge API operations and corresponding actions that you can specify in an IAM policy.

<table>
<thead>
<tr>
<th>EventBridge API operation</th>
<th>Required permissions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeleteRule</td>
<td>events:DeleteRule</td>
<td>Required to delete a rule.</td>
</tr>
<tr>
<td>DescribeEventBus</td>
<td>events:DescribeEventBus</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>
<td>Required to list the details about a rule.</td>
</tr>
<tr>
<td>DisableRule</td>
<td>events:DisableRule</td>
<td>Required to disable a rule.</td>
</tr>
<tr>
<td>EnableRule</td>
<td>events:EnableRule</td>
<td>Required to enable a rule.</td>
</tr>
<tr>
<td>ListRuleNamesByTarget</td>
<td>events:ListRuleNamesByTarget</td>
<td>Required to list rules associated with a target.</td>
</tr>
<tr>
<td>ListRules</td>
<td>events:ListRules</td>
<td>Required to list all rules in your account.</td>
</tr>
<tr>
<td>ListTagsForResource</td>
<td>events:ListTagsForResource</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>
<td>Required to list all targets associated with a rule.</td>
</tr>
<tr>
<td>PutEvents</td>
<td>events:PutEvents</td>
<td>Required to add custom events that can be matched to rules.</td>
</tr>
<tr>
<td>EventBridge API operation</td>
<td>Required permissions</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>PutPermission</td>
<td>events:PutPermission</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>
<td>Required to create or update a rule.</td>
</tr>
<tr>
<td>PutTargets</td>
<td>events:PutTargets</td>
<td>Required to add targets to a rule.</td>
</tr>
<tr>
<td>RemovePermission</td>
<td>events:RemovePermission</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>
<td>Required to remove a target from a rule.</td>
</tr>
<tr>
<td>TestEventPattern</td>
<td>events:TestEventPattern</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

To grant permissions, you use the IAM policy language in a policy statement to specify the conditions when a policy should take effect. For example, you can have a policy that is applied only after a specific date.

A condition in a policy consists of key-value pairs. Condition keys aren’t case sensitive.

If you specify multiple conditions or keys in a single condition, all conditions and keys must be met for EventBridge to grant permission. If you specify a single condition with multiple values for one key, EventBridge grants permission if one of the values is met.

You can use placeholders or policy variables 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 (p. 16) in your account. To access events, a user must be authorized for the PutRule API action. If an IAM user or role is authorized for the events:PutRule action, they can create a rule (p. 32) that matches certain events. However, for the rule to be useful, the user must also have permissions for the events:PutTargets action because, if you want the rule to do more than publish a CloudWatch metric, you must also add a target (p. 48) to a rule.

You can provide a condition in the policy statement of an IAM user or role that allows the user or role to create a rule that only matches a specific set of sources and event types. To grant access to specific sources and types of events, use the events:source and events:detail-type condition keys.

Similarly, you can provide a condition in the policy statement of an IAM user or role that allows the user or role to create a rule that only matches a specific resource in your accounts. To grant access to a specific resource, use the events:TargetArn condition key.

The following example is a policy that allows users to access all events except Amazon EC2 events in EventBridge using a deny statement on the PutRule API action.

```
{
    "Version" : "2012-10-17",
    "Statement" : [ 
       {
           "Sid" : "DenyPutRuleForAllEC2Events",
           "Effect" : "Deny",
           "Action" : "events:PutRule",
           "Resource" : "*",
           "Condition" : { 
               "StringEquals" : { 
                   "events:source" : "aws.ec2"
               }
           }
       }
    ]
}
```

The following table shows the condition keys and key and value pairs that you can use in a policy in EventBridge.

<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>
</tbody>
</table>

172
<table>
<thead>
<tr>
<th>Condition key</th>
<th>Key value pair</th>
<th>Evaluation types</th>
</tr>
</thead>
<tbody>
<tr>
<td>creatorAccount</td>
<td>For creatorAccount, use 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;boolean&quot;</td>
<td>eventBusInvocation, Null</td>
</tr>
<tr>
<td></td>
<td>For boolean, use true when a rule sends an event to a target that is an event bus in another account. Use false when when a PutEvents API call is used.</td>
<td></td>
</tr>
<tr>
<td>events:source</td>
<td>&quot;events:source&quot;: &quot;source&quot;</td>
<td>Source, Null</td>
</tr>
<tr>
<td></td>
<td>Use source for the literal string for the source field of the event such as &quot;aws.ec2&quot; or &quot;aws.s3&quot;. For more possible values for source, see the example events in Events from AWS services (p. 84).</td>
<td></td>
</tr>
<tr>
<td>events:detail-type</td>
<td>&quot;events:detail-type&quot;: &quot;detail-type&quot;</td>
<td>Detail Type, Null</td>
</tr>
<tr>
<td></td>
<td>Where detail-type is the literal string for the detail-type field of the event such as &quot;AWS API Call via CloudTrail&quot; and &quot;EC2 Instance State-change Notification&quot;.</td>
<td></td>
</tr>
<tr>
<td>events: detail.userIdentity.principalId</td>
<td>&quot;events:detail.userIdentity.principalId&quot;: &quot;principal-id&quot;</td>
<td>Principal Id, Null</td>
</tr>
<tr>
<td></td>
<td>For principal-id, use the literal string for the detail.userIdentity.principalId field of the event with detail-type &quot;AWS API Call via CloudTrail&quot; such as &quot;AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName.&quot;.</td>
<td></td>
</tr>
<tr>
<td>events: detail.service</td>
<td>&quot;events:detail.service&quot;: &quot;service&quot;, Null</td>
<td>service, Null</td>
</tr>
<tr>
<td></td>
<td>For service, use the literal string for the detail.service field of the event, such as &quot;ABUSE&quot;.</td>
<td></td>
</tr>
<tr>
<td>events: detail.eventTypeCode</td>
<td>&quot;events:detail.eventTypeCode&quot;: &quot;eventTypeCode&quot;, Null</td>
<td>eventTypeCode, Null</td>
</tr>
<tr>
<td></td>
<td>For eventTypeCode, use the literal string for the detail.eventTypeCode field of the event, such as &quot;AWS_ABUSE_DOS_REPORT&quot;.</td>
<td></td>
</tr>
</tbody>
</table>
### Condition key

<table>
<thead>
<tr>
<th>Condition key</th>
<th>Key value pair</th>
<th>Evaluation types</th>
</tr>
</thead>
<tbody>
<tr>
<td>events:TargetArn</td>
<td>&quot;events:TargetArn&quot;:&quot;target-arn&quot;</td>
<td>ARN, Null</td>
</tr>
<tr>
<td></td>
<td>For target-arn, use the ARN of the target for the rule, for example &quot;arn:aws:lambda:<em>:</em>:function:*&quot;.</td>
<td></td>
</tr>
<tr>
<td>events:ManagedBy</td>
<td>Used internally by AWS services. For a rule 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 [Managing access permissions to your Amazon EventBridge resources (p. 150)](https://docs.aws.amazon.com/AmazonEventBridge/latest/UserGuide/EventBridge-IAM.html).

### Example: Using the creatorAccount condition

The following example policy statement shows how to use the creatorAccount condition in a policy to only allow rules to be created 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": {
```
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. The following example indicates an invocation from a cross-account target.

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

Example: Limiting 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**

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

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

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

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

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

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

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

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

See the following table for a comparison of Policy A and Policy B.

<table>
<thead>
<tr>
<th>Event Pattern</th>
<th>Allowed by Policy A</th>
<th>Allowed by Policy B</th>
</tr>
</thead>
<tbody>
<tr>
<td><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 isn't 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>
Event Pattern | Allowed by Policy A | Allowed by Policy B
--- | --- | ---
| } | | |

**Example: Defining multiple sources that can be used in an event pattern individually**

The following policy allows an IAM user or role to create a rule where the source in the EventPattern is either Amazon EC2 or Amazon ECS.

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

The following table shows some examples of event patterns that are 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: Defining 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"
                }
            }
        }
    ]
}
```

The following table shows some examples of event patterns that are 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: Ensuring that the source is defined in the event pattern

The following policy allows users to only create rules with EventPatterns that have the source field. With this policy, an IAM user or role can't create a rule with an EventPattern that doesn't provide a specific source.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutRuleIfSourceIsSpecified",
            "Effect": "Allow",
            "Action": "events:PutRule",
            "Resource": "*",
            "Condition": {
                "Null": {
                    "events:source": "false"
                }
            }
        }
    ]
}
```

The following table shows some examples of event patterns that are allowed or denied by this policy.

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

Example: Defining a list of allowed sources in an event pattern with multiple sources

The following policy allows users to create rules with EventPatterns that have multiple sources in them. Each source in the event pattern must be a member of the list provided in the condition. When you use the ForAllValues condition, make sure that at least one of the items in the condition list is defined.

```json
{

```
The following table shows some examples of event patterns that are 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 isn't necessarily the name of an AWS service.

This policy condition is helpful when you work 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.
"Version": "2012-10-17",
"Statement": [
{
"Sid": "AllowPutRuleEventsWithDetailServiceEC2",
"Effect": "Allow",
"Action": "events:PutRule",
"Resource": "*",
"Condition": {
"StringEquals": {
"events:detail.service": "ABUSE"
}
}
}
]

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

For example, the following policy allows the creation of rules only for events where the value of `events:details.eventTypeCode` is `AWS_ABUSE_DOS_REPORT`.

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

Example: Ensuring that only AWS CloudTrail events for API calls from a certain `PrincipalId` are allowed

All AWS CloudTrail events have the PrincipalId of the user who made the API call in the `detail.userIdentity.principalId` path of an event. Using 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.

```json
"Version": "2012-10-17",
"Statement": [
{
"Sid": "AllowPutRuleOnlyForCloudTrailEventsWhereUserIsASpecificIAMUser",
"Effect": "Allow",
"Action": "events:PutRule",
"Resource": "*",
"Condition": {
"StringEquals": {
"events:detail.userIdentity.principalId": "$PrincipalId"
}
}
]
```
IAM policy conditions

```
"Resource": "*",
"Condition": {
  "StringEquals": {
    "events:detail-type": [ "AWS API Call via CloudTrail" ],
    "events:detail.userIdentity.principalId": [ "AIDAJ45Q7YFFAREXAMPLE" ]
  }
}
```

The following table shows some examples of event patterns that are 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. The following policy limits users to adding targets to only a specific rule: `MyRule` under account `123456789012`.

```
{ "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. You can limit targets to only Lambda functions, as in the following example.
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutTargetsOnASpecificRuleAndOnlyLambdaFunctions",
            "Effect": "Allow",
            "Action": "events:PutTargets",
            "Condition": {
                "ArnLike": {
                    "events:TargetArn": "arn:aws:lambda:*:*:function:*"
                }
            }
        }
    ]
}
Log and monitor in Amazon EventBridge

Amazon EventBridge works with AWS CloudTrail, a service that records actions from AWS services. CloudTrail captures API calls made by or on behalf of your AWS account from the EventBridge console and to EventBridge API operations.

Using the information collected by CloudTrail, you can determine what request was made to EventBridge, the IP address from which the request was made, who made the request, when it was made, and more.

For more information about CloudTrail, see the AWS CloudTrail User Guide.

**Topics**
- EventBridge information in CloudTrail (p. 184)
- Example: EventBridge log file entries (p. 185)

**EventBridge information in CloudTrail**

CloudTrail is enabled on your AWS account when you create your account. When an event occurs in EventBridge, CloudTrail records the event 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 a record of events in your AWS account, including events for EventBridge, create a trail. A **trail** is a configuration that CloudTrail uses to deliver log files to an Amazon S3 bucket. By default, the trail logs events from all Regions in the AWS partition and then delivers the log files to an S3 bucket. You can configure other AWS services to analyze and act on 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

You can log the following EventBridge actions as events in CloudTrail log files:

- DeleteRule
- DescribeEventBus
- DescribeRule
- DisableRule
- EnableRule
- ListRuleNamesByTarget
- ListRules
- ListTargetsByRule
- PutPermission
- PutRule
- PutTargets
- RemoveTargets
- TestEventPattern
Every event and log entry contains information about who generated the entry. You can use this information to 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 CloudTrail uses to deliver events as log files to an Amazon S3 bucket. CloudTrail log files contain log entries. An event represents a log entry, and it includes information about the requested action, the date and time of the action, and request parameters.

**Note**
CloudTrail log files 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-b0fcdcaeb98",
    "eventType":"AwsApiCall",
    "apiVersion":"2015-10-07",
    "recipientAccountId":"123456789012"
}
```

185
Compliance validation in Amazon EventBridge

Third-party auditors such as SOC, PCI, FedRAMP, and HIPAA assess the security and compliance of AWS services as part of multiple AWS compliance programs.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using EventBridge is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- **Security and Compliance Quick Start Guides** – Architectural considerations and steps for deploying security- and compliance-focused baseline environments on AWS.
- **Architecting for HIPAA Security and Compliance Whitepaper** – How companies can use AWS to create HIPAA-compliant applications.
- **AWS Compliance Resources** – A collection of workbooks and guides.
- **Evaluating Resources with Rules** in the AWS Config Developer Guide – Information about how AWS Config assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- **AWS Security Hub** – 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.
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, and you can use resource-based access policies (p. 162) in EventBridge, 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.
Monitoring Amazon EventBridge

EventBridge sends metrics to Amazon CloudWatch every minute for everything from the number of matched events (p. 16) to the number of times a target (p. 48) is invoked by a rule (p. 32).

The following video reviews monitoring and auditing EventBridge behavior through CloudWatch:
Monitoring and auditing events

Topics

- EventBridge metrics (p. 190)
- Dimensions for EventBridge metrics (p. 192)

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>The number of times a rule's target isn't invoked in response to an event. This includes invocations that would result in running the same rule again, causing an infinite loop.</td>
</tr>
<tr>
<td></td>
<td>Valid Dimensions: RuleName</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FailedInvocations</td>
<td>The number of invocations that failed permanently. This doesn't include invocations that are retried or invocations that succeeded after a retry attempt. It also doesn't count failed invocations that are counted in DeadLetterInvocations. Valid Dimensions: RuleName Units: Count</td>
</tr>
</tbody>
</table>
| Invocations                  | The number of times a target is invoked by a rule in response to an event. This includes successful and failed invocations, but doesn't include throttled or retried attempts until they fail permanently. It doesn't include DeadLetterInvocations.  
  **Note**  
  EventBridge only sends this metric to CloudWatch if it isn't zero. Valid Dimensions: RuleName Units: Count |
| InvocationsFailedToBeSentToDlq| The number of invocations that couldn't be moved to a dead-letter queue. Dead-letter queue errors occur due to permissions errors, unavailable resources, or size limits.  
  **Note**  
  EventBridge only sends this metric to CloudWatch if it isn't zero. Valid Dimensions: RuleName Units: Count |
| InvocationsSentToDlq         | The number of invocations that are moved to a dead-letter queue.  
  **Note**  
  EventBridge only sends this metric to CloudWatch if it isn't zero. Valid Dimensions: RuleName Units: Count |
| ThrottledRules               | The number of rules that have tried to run but are being throttled. Valid Dimensions: RuleName Units: Count |
| TriggeredRules               | The number of rules that have run and matched with any event. You won't see this metric in CloudWatch until a rule is triggered. Valid Dimensions: RuleName Units: Count |
| MatchedEvents                | The number of events that matched with any rule. Valid Dimensions: None Units: Count |
Dimensions for EventBridge metrics

EventBridge metrics have one *dimension*, or sortable attribute, 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>
My rule ran but my Lambda function wasn't invoked

One reason your Lambda function might not run is if you don't have the right permissions.

To check your permissions for your Lambda function

1. Using the AWS CLI, run the following command with your function and your AWS Region:

   ```bash
tools/lambda get-policy --function-name MyFunction --region us-east-1
   ```

You should see the following output.

   ```json
   {
     "Policy": "{
       "Version": "2012-10-17",
       "Statement": [ {
         "Condition": {"ArnLike": {"AWS:SourceArn": "arn:aws:events:us-east-1:123456789012:rule/MyRule"}}},
         "Action": "lambda:InvokeFunction",
   ```
I just created or modified a rule, but it didn't match a test event

When you make a change to a rule (p. 32) or to its targets (p. 48), incoming events (p. 16) might not immediately start or stop matching to new or updated rules. Allow a short period of time for changes to take effect.

If events still don't match after a short period of time, check the CloudWatch metrics TriggeredRules, Invocations, and FailedInvocations for your rule. For more information about these metrics, see Monitoring Amazon EventBridge (p. 190).

If the rule is intended to match an event from an AWS service, use the TestEventPattern action to test the event pattern of your rule matches a test event. For more information, see TestEventPattern in the Amazon EventBridge API Reference.
My rule didn't run at the time I specified in the ScheduleExpression

Make sure you have set the schedule for the rule (p. 32) in the UTC+0 time zone. If the ScheduleExpression is correct, then follow the steps under I just created or modified a rule, but it didn't match a test event (p. 194).

My rule didn't run at the time that I expected

EventBridge runs rules (p. 32) within one minute of the start time you set. The count down to run time begins as soon as you create the rule.

You can use a cron expression to invoke targets (p. 48) at a specified time. To create a rule that runs every four hours on the 0th minute, you do one of the following:

- In the EventBridge console, you use the cron expression 0 0/4 * * ? *.
- Using the AWS CLI, you use the expression cron(0 0/4 * * ? *).

For example, to create a rule named TestRule that runs every 4 hours by using the AWS CLI, you use the following command.

```
aws events put-rule --name TestRule --schedule-expression 'cron(0 0/4 * * ? *)'
```

To run a rule every five minutes, you use the following cron expression.

```
aws events put-rule --name TestRule --schedule-expression 'cron(0/5 * * * ? *)'
```

The finest resolution for an EventBridge rule that uses a cron expression is one minute. Your scheduled rule runs within that minute but not on the precise 0th second.

Because EventBridge and target services are distributed, there can be a delay of several seconds between the time the scheduled rule runs and the time the target service performs the action on the target resource.

My rule matches IAM API calls but it didn't run

The IAM service is only available in the US East (N. Virginia) Region, so events from AWS API calls from IAM are only available in that region. For more information, see Events from AWS services (p. 84).

The IAM role associated with my rule is being ignored when the rule runs

EventBridge only uses IAM roles for rules (p. 32) that send events (p. 16) to Kinesis streams. For rules that invoke Lambda functions or Amazon SNS topics, you need to provide resource-based permissions (p. 162).

Make sure your regional AWS STS endpoints are enabled, so that EventBridge can use them when assuming the IAM role you provided. For more information, see Activating and Deactivating AWS STS in an AWS Region in the IAM User Guide.
My rule has an event pattern that is supposed to match a resource, but no events match

Most services in AWS treat a colon (:) or slash (/) as the same character in Amazon Resource Names (ARNs), but EventBridge uses an exact match in event patterns (p. 19) and rules (p. 32). Be sure to use the correct ARN characters when creating event patterns so that they match the ARN syntax in the event (p. 16) to match.

Some events, such as AWS API call events from CloudTrail, don't have anything in the resources field.

My event's delivery to the target was delayed

EventBridge tries to deliver an event (p. 16) to a target (p. 48) 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. If the target service is having problems, EventBridge automatically reschedules another delivery. If 24 hours has passed since the arrival of event, EventBridge stops trying to deliver the event and publishes the FailedInvocations metric in CloudWatch. We recommend that you create a CloudWatch alarm on the FailedInvocations metric.

Some events were never delivered to my target

If the target (p. 48) of an EventBridge rule (p. 32) is constrained for a prolonged time, EventBridge might not retry delivery. For example, if the target is not provisioned to handle the incoming event (p. 16) traffic and the target service is throttling requests that EventBridge makes on your behalf, then EventBridge might not retry delivery.

My rule ran more than once in response to one event

In rare cases, the same rule (p. 32) can run more than once for a single event (p. 16) or scheduled time, or the same target (p. 48) can be invoked more than once for a given triggered rule.

Preventing infinite loops

In EventBridge, it is possible to create a rule (p. 32) that leads to infinite loops, where the rule runs repeatedly. If you have a rule that causes an infinite loop, rewrite it so that the actions that the rule takes don't match the same rule.

For example, a rule that detects that ACLs have changed on an Amazon S3 bucket and then runs software to change them to a new state causes an infinite loop. One way to resolve it is to rewrite the rule so that it only matches ACLs that are in a bad state.

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

If your Amazon SQS queue is encrypted, you must include the following section in your KMS key policy.

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

My rule runs, but I don't see any messages published into my Amazon SNS topic

Scenario 1

You need permission for messages to be published into your Amazon SNS topic. Use the following command using the AWS CLI, replacing us-east-1 with your Region and using your topic ARN.

```
```

To have the correct permission, your policy attributes similar to the following.

```json
```

If you don't see `events.amazonaws.com` with `Publish` permission in your policy, first copy the current policy and add the following statement to the list of statements.
My Amazon SNS topic still has permissions for EventBridge even after I deleted the rule associated with the Amazon SNS topic

```json
```

Then set the topic attributes by using the AWS CLI, use the following command.

```
```

**Note**

If the policy is incorrect, you can also edit the rule (p. 32) in the EventBridge console by removing and then adding it back to the rule. EventBridge sets the correct permissions on the target (p. 48).

**Scenario 2**

If your SNS topic is encrypted, you must include the following section in your KMS key policy.

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

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 (p. 32) with Amazon SNS as the target (p. 48), EventBridge adds permission to your Amazon SNS topic on your behalf. If you delete the rule shortly after you create it, EventBridge might not remove the permission from your Amazon SNS topic. If this happens, you can remove the permission from the topic by using the `aws sns set-topic-attributes` command. For information about resource-based permissions for sending events, see Using resource-based policies for Amazon EventBridge (p. 162).

**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. 172).
How can I tell when EventBridge rules are broken?

You can use the following alarm to notify you when your EventBridge rules (p. 32) 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 aren't 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 quotas

There are quotas for most aspects of EventBridge.

**Topics**
- EventBridge quotas (p. 200)
- PutEvents quotas by Region (p. 202)
- PutPartnerEvents quotas by Region (p. 203)
- Invocation quotas by Region (p. 203)

## EventBridge quotas

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 PutEvents quotas by Region (p. 202).</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 APIs 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>API destinations</td>
<td>Default: 3,000</td>
</tr>
<tr>
<td></td>
<td>The maximum number of API destinations (p. 52) per account per Region. API destinations are HTTP invocation endpoints that you can use as the target for a rule.</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>Rate of invocations per API destination</td>
<td>Default: 300 TPS.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> This soft limit prevents your endpoint from being overloaded with event traffic. If your endpoint can</td>
</tr>
</tbody>
</table>
## EventBridge quotas

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default limit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Resource</strong></td>
<td><strong>Default limit</strong></td>
</tr>
<tr>
<td></td>
<td>handle more event traffic, use the <a href="#">Service Quota console</a> to raise this limit.</td>
</tr>
<tr>
<td>Connections</td>
<td>Default: 3,000.</td>
</tr>
<tr>
<td></td>
<td>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.</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>Maximum 100 event buses per account.</td>
</tr>
<tr>
<td>Event buses - other quotas</td>
<td>There's no restriction on the rate of <a href="#">events</a> that can be received from AWS services or other AWS accounts. If you send custom events to your <a href="#">event bus</a> using the <a href="#">PutEvents API</a>, the <a href="#">PutEvents API quotas</a> apply. Any events that are sent 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 <a href="#">DescribeEventBus API</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>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 to the rule's targets. Quotas vary by region. See <a href="#">Invocation 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>ListRuleNamesByTarget</td>
<td>Maximum 100 results per page for requests.</td>
</tr>
<tr>
<td>ListRules</td>
<td>Maximum 100 results per page for requests.</td>
</tr>
<tr>
<td>ListTargetsByRule</td>
<td>Maximum 100 results per page for requests.</td>
</tr>
<tr>
<td>PutEvents entry size</td>
<td>Maximum 256KB</td>
</tr>
<tr>
<td>PutTargets</td>
<td>10 entries per request. Up to 5 targets per rule.</td>
</tr>
<tr>
<td>RemoveTargets</td>
<td>10 entries per request.</td>
</tr>
</tbody>
</table>
## Amazon EventBridge User Guide
### PutEvents quotas by Region

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rules</td>
<td>300 per event bus. 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. Before increasing quotas, examine your rules. If you have multiple rules that match very specific events, consider broadening their scope by using fewer identifiers in your Amazon EventBridge events (p. 16). In addition, consider adding more targets to your rules.</td>
</tr>
<tr>
<td>Systems Manager Run Command target</td>
<td>1 target key and 1 target value Systems Manager Run Command doesn't currently support multiple target values.</td>
</tr>
<tr>
<td>Targets</td>
<td>Maximum 5 targets per rule.</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 host a call to best support you.

**Note**

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><strong>PutEvents</strong> has a soft limit of 10,000 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• US West (Oregon)</td>
<td><strong>PutEvents</strong> has a soft limit of 2,400 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• Europe (Ireland)</td>
<td><strong>PutEvents</strong> has a soft limit of 1,200 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• US East (Ohio)</td>
<td><strong>PutEvents</strong> has a soft limit of 2,400 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• Europe (Frankfurt)</td>
<td><strong>PutEvents</strong> has a soft limit of 1,200 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• US West (N. California)</td>
<td><strong>PutEvents</strong> has a soft limit of 600 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• Europe (London)</td>
<td><strong>PutEvents</strong> has a soft limit of 600 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• Asia Pacific (Sydney)</td>
<td><strong>PutEvents</strong> has a soft limit of 600 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• Asia Pacific (Tokyo)</td>
<td><strong>PutEvents</strong> has a soft limit of 600 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• Asia Pacific (Singapore)</td>
<td><strong>PutEvents</strong> has a soft limit of 600 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• Canada (Central)</td>
<td><strong>PutEvents</strong> has a soft limit of 600 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• Europe (Paris)</td>
<td><strong>PutEvents</strong> has a soft limit of 600 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• Europe (Stockholm)</td>
<td><strong>PutEvents</strong> has a soft limit of 600 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• South America (São Paulo)</td>
<td><strong>PutEvents</strong> has a soft limit of 600 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• Asia Pacific (Seoul)</td>
<td><strong>PutEvents</strong> has a soft limit of 600 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• Asia Pacific (Mumbai)</td>
<td><strong>PutEvents</strong> has a soft limit of 600 requests per second by default in these Regions.</td>
</tr>
</tbody>
</table>
PutPartnerEvents quotas by Region

To request quota increases, please contact support.

<table>
<thead>
<tr>
<th>Regions</th>
<th>Transactions per second</th>
</tr>
</thead>
<tbody>
<tr>
<td>• US East (N. Virginia)</td>
<td></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></td>
</tr>
<tr>
<td>• Europe (Frankfurt)</td>
<td></td>
</tr>
<tr>
<td>• US West (N. California)</td>
<td></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></td>
</tr>
<tr>
<td>• Europe (Paris)</td>
<td></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>
<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></td>
</tr>
<tr>
<td>• Africa (Cape Town)</td>
<td></td>
</tr>
<tr>
<td>• Europe (Milan)</td>
<td></td>
</tr>
<tr>
<td><strong>PutPartnerEvents</strong></td>
<td>has a soft limit of 1,400 throughput requests per second and 3,600 burst requests per second by default in all Regions.</td>
</tr>
</tbody>
</table>

Invocation quotas by Region

An invocation is an event matching a rule and then being sent to the rule’s targets. If the invocation of a target fails due to a problem with the target service or account throttling, EventBridge reattempts for up to 24 hours.
If you receive events from another account, each event that matches a rule in your account and is sent to the rule's targets counts against your account's quota of invocations per second.

After you reach the invocation quota in your region, EventBridge throttles invocations. They still happen but are delayed.

The Service Quotas console provides information about EventBridge quotas. You can also use the Service Quotas console to request quota increases for adjustable quotas. For quotas above 100,000 TPS, our service team will set up a call to best support you.

<table>
<thead>
<tr>
<th>Regions</th>
<th>Invocations per second</th>
</tr>
</thead>
<tbody>
<tr>
<td>• US East (N. Virginia)       • US West (Oregon)       • Europe (Ireland)</td>
<td>The invocations quota has a soft limit of 18,750 requests per second by default in these Region.</td>
</tr>
<tr>
<td>• US East (Ohio)            • Europe (Frankfurt)</td>
<td>The invocations quota has a soft limit of 4,500 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• US West (N. California)   • Europe (London)         • Asia Pacific (Sydney) • Asia Pacific (Tokyo) • Asia Pacific (Singapore)</td>
<td>The invocations quota has a soft limit of 2,250 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• Canada (Central)          • South America (São Paulo)  • Europe (Paris) • Europe (Stockholm) • Asia Pacific (Seoul) • Asia Pacific (Mumbai) • Asia Pacific (Hong Kong) • Middle East (Bahrain)</td>
<td>The invocations quota has a soft limit of 1,100 requests per second by default in these Regions.</td>
</tr>
<tr>
<td>• China (Ningxia)           • China (Beijing)         • Asia Pacific (Osaka) • Africa (Cape Town) • Europe (Milan)</td>
<td>The invocations quota has a soft limit of 750 requests per second by default in these Regions.</td>
</tr>
</tbody>
</table>
Amazon EventBridge tags

A *tag* is a custom attribute label that you or AWS assigns to an AWS resource. In EventBridge, you can assign tags to [rule](#) (p. 32) and [event buses](#) (p. 7). Each resource can have a maximum of 50 tags.

You use tags to 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 an EventBridge rule that you assign to an EC2 instance.

You also use tags to 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 then 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](#).

A tag has two parts:

- **A tag key**, for example, `CostCenter`, `Environment`, or `Project`.
  - Tag keys are case sensitive.
  - The maximum tag key length is 128 Unicode characters in UTF-8.
  - For each resource, each tag key must be unique.
  - Allowed characters are letters, numbers, spaces representable in UTF-8, and the following characters: `. : + = @ _ / - (hyphen)`.  
  - 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.
  - An optional tag value field, for example, `111122223333` or `Production`.
    - Each tag key can have only one value.
    - Tag values are case sensitive.
    - Omitting the tag value is the same as using an empty string.
    - 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)`.

**Tip**

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 then use the same convention for all tags.

You can use the EventBridge console, the EventBridge API, or the AWS CLI to add, edit, or delete tags. For more information, 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](#)
# 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:  
  - Archiving Amazon EventBridge events (p. 72).  
  - 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. 43).  
  - 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. 75)  
  - 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. 167)  
  - 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 tags (p. 205)  
  - Tag-based policies (p. 147)  
  - 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>
</tbody>
</table>
| | • Amazon EventBridge quotas (p. 200)  
| | • PutEvents quotas by Region (p. 202)  
| | • Invocation quotas by Region (p. 203)  
| | • Request a Quota Increase | |
| Added a new topic on transforming target input, and added a link to Application Auto Scaling Events. | Improved documentation on the input transformer. | December 20, 2019 |
| | • Transforming Amazon EventBridge target input (p. 68)  
| | • Use Input Transformer to extract data from an event and input that data to the target  
| | • Tutorial: Use input transformer to customize what EventBridge passes to the event target (p. 143) | |
| | Added a link to Application Auto Scaling Events. | |
| | • Application Auto Scaling Events and EventBridge  
| | • Events from AWS services (p. 84) | |
| Content-based filtering | | December 19, 2019 |
| Added links to Amazon Augmented AI event examples. | 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. | December 13, 2019 |
| | • Use Events in Amazon Augmented AI  
| | • Events from AWS services (p. 84) | |
| Added links to Amazon Chime event examples. | Added a link to the Amazon Chime topic that provides example events for that service. For more information, see the following. | December 12, 2019 |
| | • Automating Amazon Chime with EventBridge  
| | • Events from AWS services (p. 84) | |
| Amazon EventBridge Schemas | You can now manage schemas and generate code bindings for events in Amazon EventBridge. For more information, see the following. | December 1, 2019 |
| | • Amazon EventBridge schemas (p. 75)  
| | • EventBridge Schemas API Reference  
<p>| | • EventSchemas Resource Type Reference in AWS CloudFormation |</p>
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<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>