aws

用户指南

Amazon EventBridge

Amazon EventBridge: 用户指南

Table of Contents

什么是 Amazon EventBridge？

EventBridge 是无服务器服务，使用事件将应用程序组件连接在一起，可让您更轻松地构建可扩展的事件驱动型应用程序。事件驱动型架构是一种构建松耦合软件系统的风格，这些系统通过发出和响应事件来协同工作。事件驱动型架构可以帮助您提高敏捷性，并构建可靠、可扩展的应用程序。

可以使用 EventBridge 将事件从本地应用程序、AWS 服务和第三方软件等来源路由到您的组织中的使用者应用程序。EventBridge 提供了简单且一致的方式，用于摄取、筛选、转换和交付事件，使您可以快速构建应用程序。

以下视频简要介绍了 Amazon EventBridge 的功能：

EventBridge 包括两种处理事件的方式：事件总线和管道。

- 事件总线是接收事件，并将其传送到零个或多个目标的路由器。事件总线非常适合将事件从多个源路由到多个目标，在将事件传送到目标之前可以选择转换事件。

  以下视频简要概述了事件总线：

- 管道 EventBridge Pipes 适用于点对点集成；每个管道接收来自单一来源的事件，并处理和传送到单一目标。Pipes 还支持在传送到目标之前对事件进行高级转换和富集。

管道和事件总线经常配合使用。一个常见的使用场景是创建一个管道，它以事件总线为目标；该管道将事件发送到此事件总线，然后事件总线会将这些事件发送到多个目标。例如，您可以创建一个管道，将 DynamoDB 流作为源，将事件总线作为目标。管道接收来自 DynamoDB 流的事件，并将它们发送到事件总线，然后事件总线根据您在事件总线中指定的规则，将它们发送到多个目标。

EventBridge 是 Amazon CloudWatch Events 的演进版本

EventBridge 以前称为 Amazon CloudWatch Events。您在 CloudWatch Events 中创建的默认事件总线和规则也会显示在 EventBridge 控制台中。EventBridge 使用相同的 CloudWatch Events API，因此使用 CloudWatch Events API 的代码可保持不变。

EventBridge 依托 CloudWatch Events 的功能构建，提供合作伙伴事件、架构注册表和 EventBridge Pipes 等功能。添加到 EventBridge 的新功能不会添加到 CloudWatch Events。有关更多信息，请参阅???。

您在 CloudWatch Events 中会用到的所有功能也都包含在 EventBridge 中，包括：

- ???
- ???
- ???
- ???

EventBridge 以事件功能为基础并进行扩展的功能包括：

- ???
- ???
- ???
- ???

Amazon EventBridge 设置和先决条件

要使用 Amazon EventBridge，您需要一个 AWS 账户。您可通过此账户使用 Amazon EC2 等服务，生成可在 EventBridge 控制台中显示的事件。您也可以安装和配置 AWS Command Line Interface (AWS CLI)，使用命令行界面查看事件。

主题

注册 AWS 账户

如果您还没有 AWS 账户，请完成以下步骤来创建一个。

注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册 AWS 账户时，系统将会创建一个 AWS 账户根用户。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为管理用户分配管理访问权限，并且只使用根用户执行需要根用户访问权限的任务。

注册过程完成后，AWS 会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建管理用户

注册 AWS 账户后，保护您的 AWS 账户根用户，启用 AWS IAM Identity Center，并创建一个管理用户，以避免使用根用户执行日常任务。

保护您的 AWS 账户根用户

1. 选择根用户并输入您的 AWS 账户电子邮件地址，以账户拥有者身份登录 AWS Management Console。在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。

2. 对您的根用户启用多重身份验证（MFA）。

   有关说明，请参阅《IAM 用户指南》中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建管理用户

1. 启用 IAM Identity Center

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，为管理用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《AWS IAM Identity Center 用户指南》中的使用默认 IAM Identity Center 目录 配置用户访问权限。

作为管理用户登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  要获取使用 IAM Identity Center 用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的登录 AWS 访问门户。

登录 Amazon EventBridge 控制台

登录 Amazon EventBridge 控制台

- 登录 AWS Management Console并打开 Amazon EventBridge 控制台 (https://console.aws.amazon.com/events/)。

账户凭证

虽然可以使用根用户凭证访问 EventBridge，但是建议您使用 AWS Identity and Access Management (IAM) 账户。如果您使用 IAM 账户访问 EventBridge，则必须拥有以下权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "events:*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:events:*:*:*"
    },
    {
      "Action": [
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "events.amazonaws.com"
        }
      }
    }
  ]
}
```

有关更多信息，请参阅身份验证。

设置 AWS Command Line Interface

您可以使用 AWS CLI 执行 EventBridge 操作。

有关如何安装和配置 AWS CLI 的信息，请参阅 AWS Command Line Interface 用户指南中的使用 AWS Command Line Interface 进行设置。

区域终端节点

必须启用默认区域端点，才能使用 EventBridge。有关更多信息，请参阅 IAM 用户指南 中的在 AWS 区域中激活和停用 AWS STS。

开始使用亚马逊 EventBridge

的基础 EventBridge 是创建将事件路由到目标的规则。在这个部分，您将创建一条基本规则。有关针对特定场景和特定目标的教程，请参阅 Amazon EventBridge 教程。

在 Amazon 中创建规则 EventBridge

要为事件创建规则，您需要指定在 EventBridge 收到与规则中的事件模式相匹配的事件时要采取的操作。当事件匹配时， EventBridge 将该事件发送到指定的目标并触发规则中定义的操作。

当你 AWS 账户中的某项 AWS 服务发出事件时，它总是会进入你账户的默认事件总线。要编写匹配您账户中 AWS 服务的事件的规则，必须将其与默认事件总线相关联。

为 AWS 服务创建规则

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择规则。
3. 选择创建规则。
4. 为规则输入名称和描述。

   规则不能与同一区域中的另一个规则和同一事件总线上的名称相同。
5. 对于事件总线，请选择要与此规则关联的事件总线。如果您希望此规则对来自您自己的账户的匹配事件触发，请选择 AWS 默认事件总线。当您账户中的某个 AWS 服务发出一个事件时，它始终会发送到您账户的默认事件总线。
6. 对于规则类型，选择具有事件模式的规则。
7. 选择下一步。
8. 对于事件源，选择AWS 服务。
9. （可选）对于示例事件，请选择事件的类型。
10. 对于事件模式，执行以下操作之一：
    - 要使用模板创建您的事件模式，请选择事件模式表，然后选择事件源和事件类型。如果您选择 All Events 作为事件类型，则此 AWS 服务发出的所有事件都将匹配规则。

      要自定义模板，请选择 Custom pattern (JSON editor)（自定义模式（JSON 编辑器））进行您的更改。

- 要使用自定义事件模式，请选择 Custom pattern (JSON editor)（自定义模式（JSON 编辑器）），然后创建您的事件模式。

11. 选择 Next（下一步）。
12. 对于目标类型，选择AWS 服务。
13. 在 “选择目标” 中，选择在 EventBridge 检测到与事件模式匹配的事件时要向其发送信息的 AWS 服务。
14. 显示的字段因您选择的服务而异。根据需要输入此目标类型的特定信息。
15. 对于许多目标类型， EventBridge 需要向目标发送事件的权限。在这些情况下， EventBridge 可以创建规则运行所需的 IAM 角色。请执行以下操作之一：

    - 要自动创建 IAM 角色，请选择为此特定资源创建新角色。
    - 要使用您之前创建的 IAM 角色，请选择使用现有角色，然后从下拉列表中选择现有角色。

16. （可选）对于 Additional settings（其他设置），执行以下操作：

    a. 对于 Maximum age of event（事件的最大时长），输入一分钟（00:01）与 24 小时（24:00）之间的值。

    b. 对于重试尝试，输入 0 到 185 之间的数字。

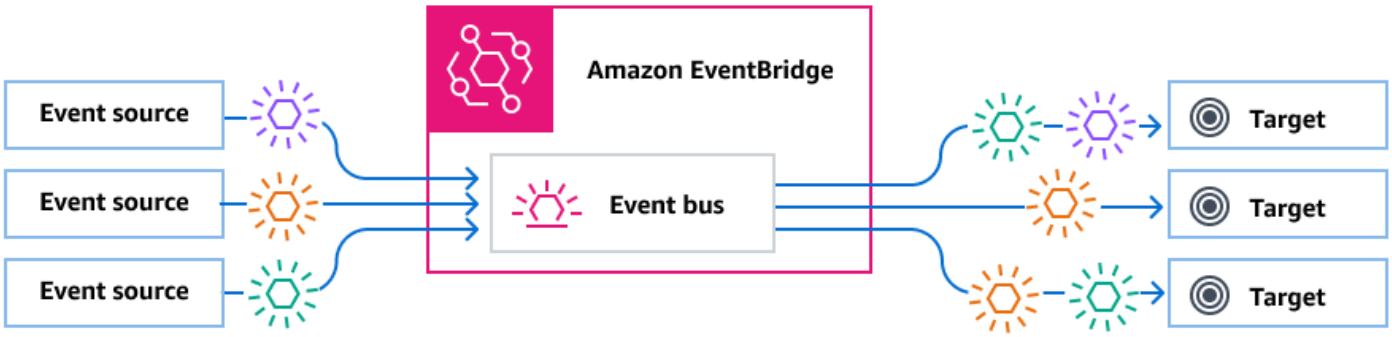
    c. 对于死信队列，选择是否使用标准的 Amazon SQS 队列作为死信队列。 EventBridge 如果匹配此规则的事件未成功传送到目标，则将其发送到死信队列。请执行以下操作之一：

       - 选择无不使用死信队列。
       - 在当前 AWS 帐户中选择选择一个Amazon SQS队列用作死信队列，然后从下拉列表中选择要使用的队列。
       - 选择选择其他 AWS 账户中的 Amazon SQS 队列作为死信队列，然后输入要使用的队列的 ARN。您必须将基于资源的策略附加到队列，以授予向该队列发送消息的 EventBridge 权限。有关更多信息，请参阅 为死信队列授予权限。

17. （可选）选择 Add another target（添加其他目标），以为此规则添加其他目标。
18. 选择 Next（下一步）。
19. （可选）为规则输入一个或多个标签。有关更多信息，请参阅 亚马逊 EventBridge 标签。
20. 选择 下一步。
21. 查看规则详细信息并选择创建规则。

亚马逊 EventBridge 活动巴士

事件总线是接收事件并将其传送到零个或多个目的地或目标的路由器。事件总线非常适合将事件从多个源路由到多个目标，在将事件传送到目标之前可以选择转换事件。



与事件总线关联的规则会在事件到达时评估事件。每条规则都会检查事件是否与规则的模式相匹配。如果事件确实匹配，则 EventBridge 发送事件

您可以将规则与特定的事件总线相关联，使该规则仅适用于该事件总线接收的事件。

> **Note**
>
> 您也可以使用 Pip EventBridge es 处理事件。 EventBridge 管道用于 point-to-point 集成；每个管道都接收来自单一来源的事件，用于处理和传送到单个目标。Pipes 还支持在传送到目标之前对事件进行高级转换和富集。有关更多信息，请参阅???。

主题

事件总线的工作原理

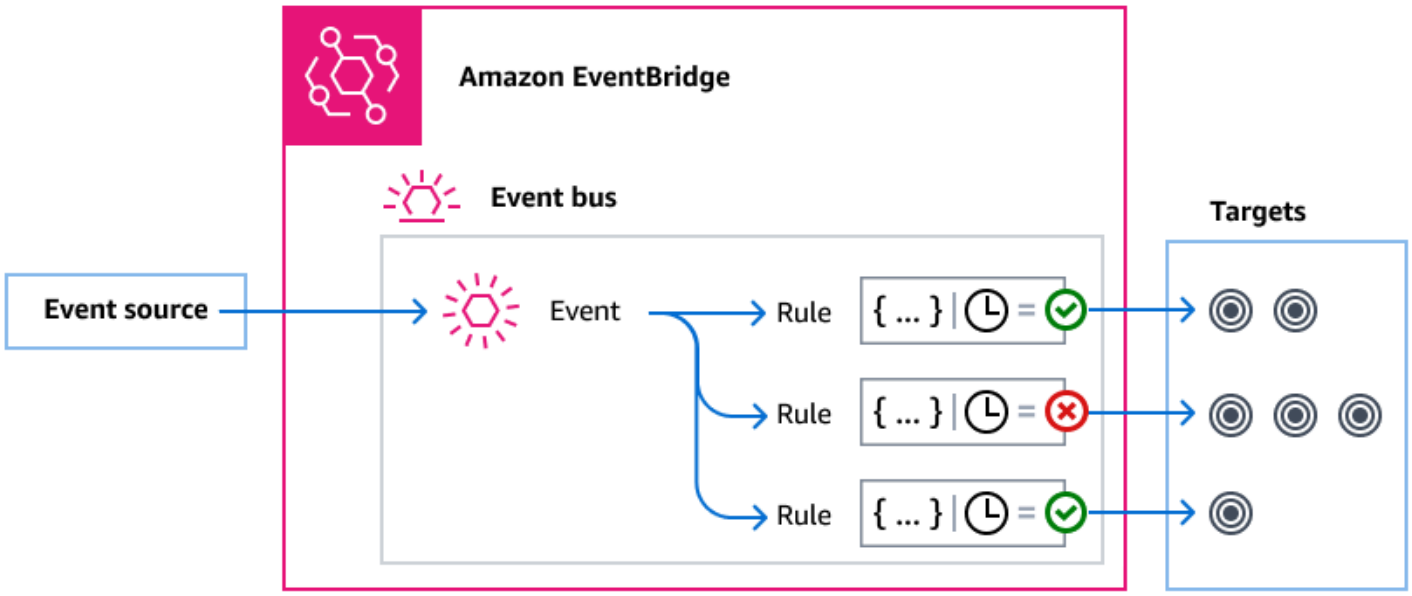
事件总线使您能够将来自多个源的事件路由到多个目的地，也称为目标。

它的工作方式可概括如下：

1. 事件源（可以是 AWS 服务、您自己的自定义应用程序或 SaaS 提供者）将事件发送到事件总线。
2. EventBridge 然后根据为该事件总线定义的每条规则评估事件。

   对于每个与规则匹配的事件，EventBridge 然后将该事件发送到为该规则指定的目标。或者，作为规则的一部分，您还可以指定在将事件发送到目标之前 EventBridge 应如何转换事件。

   一个事件可能匹配多个规则，每个规则最多可以指定五个目标。（事件可能不符合任何规则，在这种情况下，不 EventBridge 执行任何操作。）



举一个使用 EventBridge 默认事件总线的示例，它会自动接收来自 AWS 服务的事件：

1. 您在默认事件总线中为 `EC2 Instance State-change Notification` 事件创建一条规则：
   - 您指定该规则匹配的事件是 Amazon EC2 实例的 `state` 更改为 `running`。

     您可以通过指定 JSON 来实现此目的，JSON 定义触发规则的事件必须匹配的属性和值，称为事件模式。

```
{
```

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

- 您可以将规则的目标指定为给定的 Lambda 函数。

2. 每当 Amazon EC2 实例更改状态时，Amazon EC2（事件源）就会自动将该事件发送到默认事件总线。
3. EventBridge 根据您创建的规则评估发送到默认事件总线的所有事件。

   如果事件符合您的规则（也就是说，如果事件是状态更改为 Amazon EC2 实例`running`），则 EventBridge 会将该事件发送到指定的目标。在本例中，目标就是 Lambda 函数。

以下视频描述了什么是事件总线及其用途：什么是事件总线

以下视频介绍了不同的事件总线以及何时使用它们：事件总线之间的差别

Amazon EventBridge 事件总线概念

以下是基于事件总线构建的事件驱动型架构主要组件的详细介绍。

事件总线

事件总线是接收事件并将其传送到零个或多个目的地或目标的路由器。如果您需要将事件从多个来源路由到多个目标，可使用事件总线，在将事件传送到目标之前还可以选择转换事件。

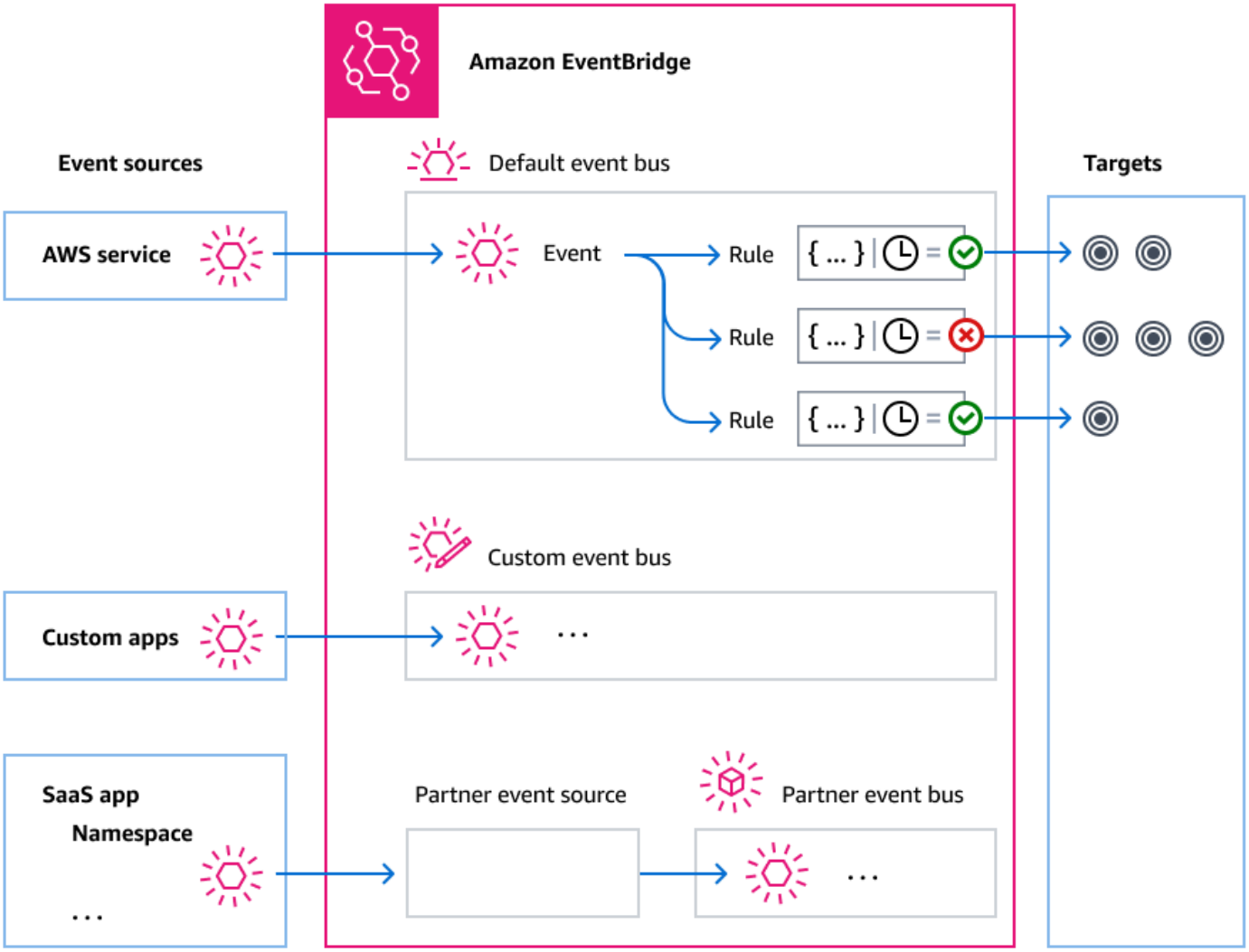
您的账户包含一个默认事件总线，该总线可自动接收来自 AWS 服务的事件。您也可以：

- 创建其他事件总线，称为自定义事件总线，并指定它们接收哪些事件。
- 创建合作伙伴事件总线，用于接收来自 SaaS 合作伙伴的事件。

活动总线的常见使用场景包括：

- 使用事件总线作为不同工作负载、服务或系统之间的代理。

- 在应用程序中使用多条事件总线来分配事件流量。例如，创建一条总线来处理包含个人身份信息 (PII) 的事件，创建另一条总线来处理其他事件。
- 将事件从多个事件总线发送到集中式事件总线来聚合事件。该集中式总线可以与其他总线位于同一账户中，也可以位于不同的账户或区域中。



事件

简单来说， EventBridge 事件是发送到事件总线或管道的 JSON 对象。

在事件驱动型架构 (EDA) 的背景下，事件通常代表资源或环境中发生变化的指标。

有关更多信息，请参阅???。

事件来源

EventBridge 可以接收来自事件源的事件，包括：

- AWS 服务
- 自定义应用程序
- 软件即服务 (SaaS) 合作伙伴

规则

规则接收传入事件，并将其发送到适当的目标进行处理。您可以指定每条规则如何调用其目标，其依据是：

- 事件模式，包含一个或多个用于匹配事件的筛选器。事件模式可以包括筛选器，匹配以下内容：
  - 事件元数据 - 有关 事件的数据，例如事件源或事件来自的账户或区域。
  - 事件数据 - 事件本身的属性。这些属性因事件而异。
  - 事件内容 - 事件数据的实际属性值。
- 定期调用目标的计划。

  您可以在中指定计划规则 EventBridge，也可以使用计划EventBridge 程序来指定预定规则。

> **Note**
>
> EventBridge 提供 Amazon S EventBridge cheduler，这是一款无服务器计划程序，允许您通过一个中央托管服务创建、运行和管理任务。 EventBridge Scheduler 具有高度可定制性，与 EventBridge计划规则相比具有更高的可扩展性，具有更广泛的目标 API 操作和 AWS 服务。
> 我们建议您使用 EventBridge 调度器按计划调用目标。有关更多信息，请参阅???。

每条规则都是针对特定事件总线定义的，并且仅适用于该事件总线上的事件。

一条规则最多可以向五个目标发送事件。

默认情况下，每个事件总线最多可以配置 300 条规则。可以在服务限额控制台中将此配额提高到数千条规则。由于规则限制适用于每条总线，如果您需要更多规则，则可以在您的账户中创建其他自定义事件总线。

您在创建事件总线时可以为不同的服务授予不同权限，自定义账户中各事件的接收方式。

要在将事件 EventBridge 传递到目标之前对其结构或日期进行自定义，请在信息到达目标之前使用输入转换器对其进行编辑。

有关更多信息，请参阅???。

目标

目标是一种资源或端点，当事件与为规则定义的事件模式相匹配时，它会向其 EventBridge 发送事件。

一个目标可以从多个事件总线接收多个事件。

有关更多信息，请参阅???。

事件总线的高级功能

EventBridge 包括以下功能，可帮助您开发、管理和使用事件总线。

使用 API 目标在服务之间启用 REST API 调用

EventBridge API 目的地是您可以设置为规则目标的 HTTP 端点，就像向 AWS 服务或资源发送事件数据一样。使用 API 目标，您可以通过 API 调用在 AWS 服务、集成的 SaaS 应用程序和 AWS外部的应用程序之间路由事件。创建 API 目标时，要指定用于该目标的连接。每个连接都包括向 API 目标端点授权的授权类型和要使用参数的详细信息。

存档和重放事件，协助开发和灾难恢复

您可以存档 或保存事件，稍后再从存档中重放这些事件。存档可用于：

- 测试应用程序，因为您有存储的事件可供使用，不必等待新事件。
- 新服务首次上线时为其注入数据。
- 增加您的事件驱动型应用程序的持久性。

使用架构注册表快速开始创建事件模式

在构建使用的无服务器应用程序时 EventBridge，了解典型事件的结构而不必生成事件会很有帮助。在架构中描述了事件结构，这些架构适用于上 AWS EventBridge服务生成的所有事件。

对于不是来自 AWS 服务的活动，您可以：

- 创建或上传自定义架构。

- 使用 Schema Discovery EventBridge 自动为发送到事件总线的事件创建架构。

一旦您有了事件的架构，就可以下载常用编程语言的代码绑定。

使用策略管理资源和访问权限

要组织 AWS 资源或跟踪成本 EventBridge，您可以为 AWS 资源分配自定义标签或标签。使用基于标签的策略，您可以控制哪些资源可以在其中做什么和不能做什么。 EventBridge

除了基于标签的策略外，还 EventBridge 支持基于身份和基于资源的策略来控制访问权限。EventBridge使用基于身份的策略可控制组、角色或用户的权限。使用基于资源的策略可为每种资源授予特定权限，例如 Lambda 函数或 Amazon SNS 主题。

创建 Amazon EventBridge 事件总线

您可以创建自定义事件总线，从您的应用程序接收事件。您的应用程序还可以将事件发送到默认事件总线。创建事件总线时，可以附加基于资源的策略，向其他账户授予权限。然后，其他账户就可以向当前账户中的事件总线发送事件。

以下视频介绍了如何创建事件总线：创建事件总线

创建自定义事件总线

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择 Event Buses (事件总线)。
3. 选择 Create event bus (创建事件总线)。
4. 输入新事件总线的名称。
5. 配置事件总线：

   - 通过执行以下任一操作来指定基于资源的策略：
     - 输入策略，其中包含要为事件总线授予的权限。您可以粘贴其他来源的策略，也可以为此策略输入 JSON。您可以使用其中一个示例策略并根据您的环境对其进行修改。
     - 要使用策略模板，请选择加载模板。根据您的环境修改策略，包括添加其他操作，授权策略中的主体使用。

     有关通过基于资源的策略向事件总线授予权限的更多信息，请参阅???。
   - 启用存档（可选）

您可以创建事件档案，以便日后可以轻松地重播这些事件。例如，您可能希望重放事件来从错误中恢复，或验证应用程序中的新功能。有关更多信息，请参阅 ???。

a. 在 “存档” 下，选择 “启用”。

b. 为档案指定名称和描述。

- 启用架构发现（可选）

  启用架构发现功能，可以直接从在此事件总线上运行的事件中 EventBridge 自动推断架构。有关更多信息，请参阅 ???。

  a. 在 “架构发现” 下，选择 “启用”。

- 指定标签（可选）

  标签是您分配给 AWS 资源的自定义属性标签。使用标签来识别和整理您的 AWS 资源。许多 AWS 服务都支持标记，因此您可以为来自不同服务的资源分配相同的标签，以表明这些资源是相关的。有关更多信息，请参阅 ???。

  a. 在 Tags（标签）下，选择 Add new tag（添加新标签）。

  b. 为新标签指定密钥和值（可选）。

6. 选择创建。

更新 Amazon EventBridge 事件总线

创建事件总线后，您可以更新其配置。这包括默认事件总线，它 EventBridge 会在您的账户中自动创建。

主题

更新事件总线上的权限

您可以为事件总线附加基于资源的策略，向其授予其他权限。有关更新给定事件总线的权限的详细说明，请参阅管理事件总线权限。

在事件总线上添加或移除档案

存档使您能够捕获事件，以便日后可以轻松地重播这些事件。例如，您可能希望重放事件来从错误中恢复，或验证应用程序中的新功能。有关更多信息，请参阅EventBridge 存档和重播。

使用 EventBridge 控制台向事件总线添加或删除存档

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择 Event Buses (事件总线)。
3. 选择要更新的事件总线。
4. 在活动总线详细信息页面上，选择存档选项卡。
5. 请执行以下操作之一：
   - 要添加档案，请执行以下操作：
     a. 选择创建存档。
     b. 为档案指定属性。
     c. 选择下一步。
     d. 选择要应用于存档事件的事件模式。
     e. 选择创建存档。
   - 要删除档案，请执行以下操作：
     a. 对于要删除的标签，请选择删除。
     b. 输入档案的名称，然后选择删除。

        存档已永久删除。您无法撤消此操作。

要使用创建或删除事件总线的存档 AWS CLI

- 要创建档案，请使用创建存档。

  要永久删除档案，请使用删除存档。

在事件总线上启动或停止架构发现

有关架构发现的更多信息，请参阅EventBridge 架构。

使用 EventBridge 控制台在事件总线上启动或停止架构发现

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择 Event Buses (事件总线)。
3. 选择要更新的事件总线。
4. 请执行以下操作之一：
   - 要启动架构发现，请选择开始发现。
   - 要停止架构发现，请选择删除发现。

要在事件总线上启动或停止架构发现，请使用 AWS CLI

- 要启动架构发现，请使用 create- discoverer。

  要停止架构发现，请使用 delete-d iscoverer。

在事件总线上添加或移除标签

标签是您或 AWS 分配给 AWS 资源的自定义属性标签。使用标签来识别和整理您的 AWS 资源。有关更多信息，请参阅EventBridge 标签。

使用 EventBridge 控制台向事件总线添加或删除标签

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择 Event Buses (事件总线)。
3. 选择要更新的事件总线。
4. 在事件总线详细信息页面上，选择标签选项卡，然后选择管理标签。
5. 请执行以下操作之一：
   - 要添加标签，请执行以下操作：
     a. 选择添加新标签。
     b. 指定标签的键和值
     c. 选择更新。
   - 要移除标签，请执行以下操作：
     a. 对于要删除的标签，请选择移除。

b. 选择更新。

要使用事件总线添加或移除标签 AWS CLI

- 要添加标签，请使用标签资源。

  要移除标签，请使用 untag-resource。

删除 Amazon EventBridge 事件总线

您可以删除自定义事件总线或合作伙伴事件总线。您无法删除默认的事件总线。删除事件总线会删除与该事件总线关联的规则。

使用 EventBridge 控制台删除事件总线

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择 Event Buses (事件总线)。
3. 选择要删除的事件总线。
4. 请执行以下操作之一：
   - 选择 Delete （删除)。
   - 选择事件总线的名称。

     在活动总线详细信息页面上，选择删除。

Amazon EventBridge 事件总线的权限

您 AWS 账户中的默认事件总线仅允许来自一个账户的事件。您可以为事件总线附加基于资源的策略，向其授予其他权限。使用基于资源的策略，您可以允许来自其他账户的 `PutEvents`、`PutRule` 和 `PutTargets` API 调用。您还可以在策略中使用 IAM 条件向组织授予权限、应用标签或仅筛选来自特定规则或账户的事件。您可以在创建事件总线时为其设置基于资源的策略，也可以稍后设置。

接受事件总线 `Name` 参数的 EventBridge API，例如 `PutRule`、`PutTargets`、`DeleteRule`、`RemoveTargets`、`DisableRule` 和 `EnableRule`，也接受事件总线 ARN。使用这些参数通过 API 引用跨账户或跨区域的事件总线。例如，您可以调用 `PutRule`，在其他账户的事件总线上创建规则，而无需担任角色。

您可以将本主题中的示例策略附加到 IAM 角色，授予向其他账户或区域发送事件的权限。使用 IAM 角色设置组织控制策略和边界，确定谁可以将事件从您的账户发送到其他账户。如果规则的目标是事件总线，我们建议始终使用 IAM 角色。您可以使用 `PutTarget` 调用来附加 IAM 角色。有关如何创建规则并向其他账户或区域发送事件的的信息，请参阅在 AWS 账户之间发送和接收 Amazon EventBridge 事件。

主题

管理事件总线权限

使用以下过程修改现有事件总线的权限。有关如何使用 AWS CloudFormation 创建事件总线策略的信息，请参阅 AWS::Events::EventBusPolicy。

管理现有事件总线的权限

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在左侧导航窗格中，选择事件总线。
3. 在名称中，选择要管理权限的事件总线的名称。

   如果有某个资源策略附加到此事件总线，则会显示该策略。
4. 选择管理权限，然后执行以下操作之一：
   - 输入策略，其中包含要为事件总线授予的权限。您可以粘贴其他来源的策略，也可以为此策略输入 JSON。
   - 要使用策略模板，请选择加载模板。根据您的环境修改策略，并添加其他操作，授权策略中的主体使用。
5. 选择更新。

该模板提供了示例策略语句，您可以根据自己的账户和环境对其进行自定义。模板不是有效策略。您可以根据您的使用场景修改模板，也可以复制其中一个示例策略并对其进行自定义。

该模板加载的策略包括一个示例，展示如何向账户授予使用 PutEvents 操作的权限、如何向组织授予权限以及如何向该账户授予权限，以管理账户中的规则。您可以为特定账户自定义模板，然后从模板中删除其他部分。本主题的后续部分会包含更多策略示例。

如果您尝试更新总线的权限，但策略中包含错误，则会显示一条错误消息，指明策略中的具体问题。

```
 ### 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": "AllowAccountToPutEvents",
       "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": "AllowAllAccountsFromOrganizationToPutEvents",
       "Effect": "Allow",
       "Principal": "*",
       "Action": "events:PutEvents",
       "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/default",
```

```
        "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": "AllowAccountToManageRulesTheyCreated",
        "Effect": "Allow",
        "Principal": {
          "AWS": "<ACCOUNT_ID>"
        },
        "Action": [
          "events:PutRule",
          "events:PutTargets",
          "events:DeleteRule",
          "events:RemoveTargets",
          "events:DisableRule",
          "events:EnableRule",
          "events:TagResource",
          "events:UntagResource",
          "events:DescribeRule",
          "events:ListTargetsByRule",
          "events:ListTagsForResource"],
        "Resource": "arn:aws:events:us-east-1:123456789012:rule/default",
        "Condition": {
          "StringEqualsIfExists": {
            "events:creatorAccount": "<ACCOUNT_ID>"
          }
        }
    }]
  }
```

策略示例：向另一账户中的默认总线发送事件

以下策略示例授予账户 111122223333 向账户 123456789012 中的默认事件总线发布事件的权限。

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

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

策略示例：向另一账户中的自定义总线发送事件

以下策略示例向账户 111122223333 授予权限，向账户 123456789012 中的 central-event-bus 发布事件，但仅适用于源值设置为 com.exampleCorp.webStore、detail-type 设置为 newOrderCreated 的事件。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "WebStoreCrossAccountPublish",
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Principal": {
        "AWS": "arn:aws:iam::111112222333:root"
      },
      "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/central-event-bus",
      "Condition": {
        "StringEquals": {
          "events:detail-type": "newOrderCreated",
          "events:source": "com.exampleCorp.webStore"
        }
      }
    }
  ]
}
```

策略示例：将事件发送到同一账户中的事件总线

以下策略示例附加到名为 CustomBus1 的事件总线，允许该事件总线接收来自同一账户和区域的事件。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "arn:aws:events:us-east-1:123456789:event-bus/CustomBus1"
      ]
    }
  ]
}
```

策略示例：向同一账户发送事件并限制更新

以下策略示例向账户 123456789012 授予权限，以创建、删除、更新、禁用和启用规则，并添加或删除目标。它限制这些规则，与来源为 com.exampleCorp.webStore 的事件匹配，并使用 "events:creatorAccount": "${aws:PrincipalAccount}" 来确保只有账户 123456789012 才能在创建这些规则和目标后对其进行修改。

```
{
  "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"
      ],
      "Resource": "arn:aws:events:us-east-1:123456789012:rule/central-event-bus/*",
      "Condition": {
        "StringEqualsIfExists": {
          "events:creatorAccount": "${aws:PrincipalAccount}",
          "events:source": "com.exampleCorp.webStore"
        }
      }
    }
  ]
}
```

策略示例：仅将事件从特定规则发送到不同区域中的总线

以下策略示例授予账户 111122223333 权限，将与中东（巴林）和美国西部（俄勒冈州）区域中名为 SendToUSE1AnotherAccount 的规则匹配的事件发送到账户 123456789012 中名为 CrossRegionBus 的事件总线，该账户位于美国东部（弗吉尼亚州北部）。策略示例已添加到账户 123456789012 中名为 CrossRegionBus 的事件总线。只有当事件与账户 111122223333 中为此事件总线指定的规则相匹配时，该策略才允许这些事件。Condition 语句将事件限制为仅与具有指定规则 ARN 的规则匹配的事件。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSpecificRulesAsCrossRegionSource",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111112222333:root"
      },
      "Action": "events:PutEvents",
      "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/CrossRegionBus",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:events:us-west-2:111112222333:rule/CrossRegionBus/
SendToUSE1AnotherAccount",
            "arn:aws:events:me-south-1:111112222333:rule/CrossRegionBus/
SendToUSE1AnotherAccount"
          ]
```

```
        }
      }
    }
  ]
}
```

策略示例：仅将事件从特定区域发送到另一区域

以下策略示例授予账户 111122223333 权限，将在中东（巴林）和美国西部（俄勒冈州）区域中生成的所有事件发送到账户 123456789012 中名为 CrossRegionBus 的事件总线，该账户位于美国东部（弗吉尼亚州北部）区域。账户 111122223333 无权发送在任何其他区域生成的事件。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCrossRegionEventsFromUSWest2AndMESouth1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111112222333:root"
      },
      "Action": "events:PutEvents",
      "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/CrossRegionBus",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:events:us-west-2:*:*",
            "arn:aws:events:me-south-1:*:*"
          ]
        }
      }
    }
  ]
}
```

策略示例：拒绝从特定区域发送事件

以下策略示例附加到账户 123456789012 中名为 CrossRegionBus 的事件总线，允许此事件总线接收来自账户 111122223333 的事件，但不允许接收在美国西部（俄勒冈州）区域生成的事件。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "1AllowAnyEventsFromAccount111112222333",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111112222333:root"
      },
      "Action": "events:PutEvents",
      "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/CrossRegionBus"
    },
    {
      "Sid": "2DenyAllCrossRegionUSWest2Events",
      "Effect": "Deny",
      "Principal": {
        "AWS": "*"
      },
      "Action": "events:PutEvents",
      "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/CrossRegionBus",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:events:us-west-2:*:*"
          ]
        }
      }
    }
  ]
}
```

从 Amazon EventBridge 事件总线生成 AWS CloudFormation 模板

AWS CloudFormation 可将基础设施视为代码，使您能够以集中且可重复的方式跨账户和区域配置和管理 AWS 资源。CloudFormation 实现这一功能的方式是允许您创建模板，它定义了您要预置和管理的资源。

EventBridge 允许您从账户中的现有事件总线生成模板，以此来帮助您快速开始开发 CloudFormation 模板。此外，EventBridge 还提供选项，可在模板中包含与该事件总线关联的规则。然后，您可以使用这些模板作为基础创建资源堆栈，实现 CloudFormation 管理。

有关 CloudFormation 的更多信息，请参阅《AWS CloudFormation 用户指南》。

> **Note**
>
> EventBridge 在生成的模板中不包含托管规则。

您也可以根据所选事件总线中包含的一条或多条规则生成模板。

从事件总线生成 CloudFormation 模板

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Event Buses (事件总线)。
3. 选择要生成 CloudFormation 模板的事件总线。
4. 从操作菜单中选择 CloudFormation 模板，然后选择您希望 EventBridge 生成模板的格式：JSON 或 YAML。

   EventBridge 将显示以所选格式生成的模板。默认情况下，与该事件总线关联的所有规则都包含在模板中。

   - 要生成不包含规则的模板，请取消选择包含此 EventBus 上的规则。
5. EventBridge 支持您选择下载模板文件或将模板复制到剪贴板。
   - 选择下载，下载模板文件。
   - 要将此模板复制到剪贴板，请选择复制。
6. 要退出模板，请选择取消。

根据使用场景的需要自定义 AWS CloudFormation 模板后，即可使用它在 CloudFormation 中创建堆栈。

使用 Amazon EventBridge 生成的 CloudFormation 模板时的注意事项

使用从事件总线生成的 CloudFormation 模板时，请考虑以下因素：

- EventBridge 在生成的模板中不包含任何密码。

  您可以编辑此模板，以包含模板参数，使用户能够在使用模板创建或更新 CloudFormation 堆栈时指定密码或其他敏感信息。

此外，用户可以使用 Secrets Manager 在所需区域创建密钥，然后编辑生成的模板以使用动态参数。

- 生成的模板中的目标，与原始事件总线中指定的目标完全相同。如果在使用模板在其他区域创建堆栈之前，未对模板进行适当的编辑，可能会导致跨区域问题。

  此外，生成的模板不会自动创建下游目标。

亚马逊 EventBridge 活动

事件 表示环境中发生的改变，这些环境包括 AWS 环境，SaaS 合作伙伴服务或应用程序，或您的某个应用程序或服务等。以下是事件的示例：

- 当实例状态从待处理变为正在运行时，Amazon EC2 将生成事件。
- Amazon EC2 Auto Scaling 会在启动或终止实例时生成事件。
- AWS CloudTrail 在您进行 API 调用时发布事件。

您还可以设置定期生成的计划事件。

有关生成事件的服务的列表，包括来自每项服务的示例事件，请参阅 来自 AWS 服务的事件并查看表中的链接。

事件表示为 JSON 对象，都具有相似的结构和相同的顶级字段。

detail 顶级字段的内容因生成事件的服务以及所生成的事件而异。source 字段和 detail-type 字段的组合用于标识在 detail 字段中找到的字段和值。有关 AWS 服务生成的事件的示例，请参阅来自 AWS 服务的事件。

主题

- 事件结构参考
- 使用 Amazon EventBridge 活动添加 PutEvents
- 来自 AWS 服务的事件
- 通过 Amazon 接收来自 SaaS 合作伙伴的事件 EventBridge
- 调试 Amazon EventBridge 事件传送

以下视频解释了事件的基础知识：什么是事件

以下视频介绍了事件的到达方式 EventBridge：事件来自哪里

事件结构参考

事件中会出现以下字段：

```
{
  "version": "0",
  "id": "UUID",
  "detail-type": "event name",
  "source": "event source",
  "account": "ARN",
  "time": "timestamp",
  "region": "region",
  "resources": [
    "ARN"
  ],
  "detail": {
    JSON object
  }
}
```

版本

默认情况下，在所有事件中设置为 0 (零)。

id

为每个事件生成的版本 4 UUID。事件通过各种规则移动到目标时，可以使用 `id` 跟踪事件。

detail-type

与 source 字段组合起来标识显示在 detail 字段中的字段和值。

由交付的事件 CloudTrail `AWS API Call via CloudTrail`具有的价值`detail-type`。

源

标识生成事件的服务。所有来自 AWS 服务的事件都以“aws”开头。客户生成的事件可具有任意值，前提是它不以“aws.”开头。建议使用 Java 包名样式反向域名字符串。

要查找 AWS 服务的正确值，请参阅条件键表，从列表中选择一个服务，然后查找服务前缀。`source`例如，Amazon 的`source`值 CloudFront 为`aws.cloudfront`。

account

用于标识 AWS 账户的 12 位数字。

time

事件时间戳，可由发起事件的服务指定。如果事件跨时间间隔，则服务可报告开始时间，因此该值可能早于接收事件的时间。

region

标识事件起源的地 AWS 区。

资源

一个 JSON 数组，包含用于标识事件中所涉及资源的 ARN。生成事件的服务决定是否包含这些 ARN。例如，Amazon EC2 实例状态更改包含 Amazon EC2 实例 ARN，Auto Scaling 事件包含实例和 Auto Scaling 组的 ARN，而对 AWS CloudTrail 的 API 调用不包含资源 ARN。

detail

包含关于事件信息的 JSON 对象。生成事件的服务决定该字段的内容。详细内容可以像两个字段一样简单。 AWS API 调用事件具有细节对象，其中大约 50 个字段嵌套在几个层次深处。

Example 示例：Amazon EC2 实例状态更改通知

亚马逊中的以下事件 EventBridge 表示 Amazon EC2 实例已终止。

```
{
  "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"
  }
}
```

有效的自定义事件所需的最少信息

创建自定义事件时，必须包括以下字段：

```
{
  "detail-type": "event name",
```

```
  "source": "event source",
  "detail": {
  }
}
```

- `detail` - 包含有关事件信息的 JSON 对象。可以是 `"{}"`。

> **Note**
>
> PutEvents接受 JSON 格式的数据。对于 JSON 数字（整数）数据类型，限制条件为：最小值为 -9,223,372,036,854,775,808，最大值为 9,223,372,036,854,775,807。

- `detail-type` - 标识事件类型的字符串。
- `source` - 标识事件源的字符串。客户生成的事件可具有任意值，前提是它不以“aws.”开头。建议使用 Java 包名样式反向域名字符串。

使用 Amazon EventBridge 活动添加 **PutEvents**

该`PutEvents`操作 EventBridge 在单个请求中向发送多个事件。有关更多信息，请参阅 PutEventsAmazon EventBridge API 参考和AWS CLI 命令参考中的 put-events。

每个 `PutEvents` 请求可支持有限数目的条目。有关更多信息，请参阅 Amazon EventBridge 配额。`PutEvents` 操作将尝试按请求的自然顺序处理所有条目。在您致电后`PutEvents`，为每个事件 EventBridge 分配一个唯一的 ID。

主题

- 使用 PutEvents 处理故障
- 使用发送事件 AWS CLI
- 计算 Amazon EventBridge PutEvents 活动条目大小

以下示例 Java 代码向发送了两个相同的事件 EventBridge。

AWS SDK for Java Version 2.x

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

PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
```

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

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

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

PutEventsResponse result = eventBridgeClient.putEvents(eventsRequest);

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

AWS SDK for Java Version 1.0

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

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

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

PutEventsResult result = awsEventsClient.putEvents(request);
```

```
for (PutEventsResultEntry resultEntry : result.getEntries()) {
    if (resultEntry.getEventId() != null) {
        System.out.println("Event Id: " + resultEntry.getEventId());
    } else {
        System.out.println("Injection failed with Error Code: " +
 resultEntry.getErrorCode());
    }
}
```

运行此代码后，PutEvents 结果将包含一组响应条目。响应数组中的每个条目对应请求数组中的一个条目，顺序按请求和响应从开始到结束的顺序。响应 Entries 数组包含的条目数量始终与请求数组相同。

使用 PutEvents 处理故障

默认情况下，如果请求中的单个条目失败，则会 EventBridge 继续处理请求中的其余条目。响应 Entries 数组可以包括成功条目和不成功条目。您必须删除不成功的条目，并在后续调用中包括它们。

成功的结果条目包含一个 Id 值，不成功的结果条目包括 ErrorCode 和 ErrorMessage 值。ErrorCode 描述错误的类型。ErrorMessage 提供有关错误的更多信息。以下示例包含 PutEvents 请求的三个结果条目。第二个条目不成功。

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

> **Note**
>
> 如果您使用将事件发布PutEvents到不存在的事件总线，则 EventBridge 事件匹配将找不到相应的规则，因此会删除该事件。尽管 EventBridge 会发送200响应，但它不会使请求失败，也不会将事件包含在请求响应的FailedEntryCount值中。

不成功的条目可包含在后续 PutEvents 请求中。首先，要查找请求中是否存在失败的条目，请检查 PutEventsResult 中的 FailedRecordCount 参数。如果该参数不为零，则可以将每个具有非空值 ErrorCode 的 Entry 添加到后续请求中。以下示例展示了一个故障处理程序。

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

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

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

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

```
    putEventsResult = awsEventsClient.putEvents(putEventsRequest);
    }
```

使用发送事件 AWS CLI

您可以使用 AWS CLI 向发送自定义事件， EventBridge 以便对其进行处理。以下示例将一个自定义事件放入 EventBridge：

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

您还可以创建包含自定义事件的 JSON 文件。

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

然后，要使用从该 AWS CLI 文件中读取条目并发送事件，请在命令提示符下键入：

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

计算 Amazon EventBridge PutEvents 活动条目大小

您可以使用PutEvents操作向发送自定义事件。 EventBridge 可将多个事件条目批量注入一个请求中，以提高效率。条目总大小必须小于 256KB。您可以在发送事件之前计算条目大小。

> **Note**
>
> 已对条目 施加大小限制。即使条目小于大小限制，由于事件 EventBridge 的 JSON 表示形式需要使用必要的字符和键，因此中的事件也始终大于条目大小。有关更多信息，请参阅 亚马逊 EventBridge 活动。

EventBridge 按如下方式计算PutEventsRequestEntry大小：

- 如果指定，则 Time 参数为 14 字节。
- Source 和 DetailType 参数为其 UTF-8 编码形式的字节数。
- 如果指定，则 Detail 参数为其 UTF-8 编码形式的字节数。
- 如果指定，则 Resources 参数的每个条目为其 UTF-8 编码形式的字节数。

以下示例 Java 代码计算给定 PutEventsRequestEntry 对象的大小。

```
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**
>
> 如果条目大小大于 256KB，我们建议将事件上传到 Amazon S3 桶，并在 `PutEvents` 条目中加入 `Object URL`。

来自 AWS 服务的事件

许多 AWS 服务都会生成 EventBridge 接收到的事件。当你账户中的某项 AWS 服务发出事件时，它会进入你账户的默认事件总线。

通过 AWS 服务交付活动

生成事件的每项 AWS 服务都会将事件 EventBridge 作为最大努力或持久交付尝试发送到。

- 尽力交付意味着服务会尝试将所有事件发送到 EventBridge，但在极少数情况下，事件可能无法传送。
- 持久交付意味着该服务将成功尝试 EventBridge 至少将事件传送到一次。

  EventBridge 将在正常条件下接受所有有效事件。如果由于服务中断而无法交付事件， EventBridge 服务部门稍后将再次重试这些事件，最长可达 24 小时。 AWS

将事件传送到后 EventBridge，将其与规则进行 EventBridge 匹配，然后遵循重试策略和为事件目标指定的任何死信队列。

有关生成事件的 AWS 服务的列表，请参阅???。

通过以下方式访问 AWS 服务事件 AWS CloudTrail

AWS CloudTrail 是一项自动记录 AWS API 调用等事件的服务。您可以创建使用来自的信息的 EventBridge 规则 CloudTrail。有关的更多信息 CloudTrail，请参阅什么是 AWS CloudTrail？ 。

由传递的所有事件`AWS API Call via CloudTrail`的值 CloudTrail 都是`detail-type`。

要记录`detail-type`值为的事件`AWS API Call via CloudTrail`，需要启用日志记录的 CloudTrail 跟踪。

CloudTrail 与 Amazon S3 一起使用时，您需要进行配置 CloudTrail 以记录数据事件。有关更多信息，请参阅为 S3 存储桶和对象启用 CloudTrail 事件记录。

AWS 服务中发生的某些事件 EventBridge 既可以由服务本身报告，也可以由报告给。 CloudTrail例如，启动或停止实例的 Amazon EC2 API 调用会生成 EventBridge 事件以及通过的事件 CloudTrail。

CloudTrail 支持 API 调用者和资源所有者通过创建跟踪在其 Amazon S3 存储桶中接收事件，并通过向 API 调用者传送事件。 EventBridge除了 API 调用者之外，资源所有者还可以通过监控跨账户 API 调用。 EventBridge CloudTrail与的集成 EventBridge 提供了一种便捷的方法来设置基于规则的自动化工作流程以响应事件。

您不能使用大小大于 256 KB 的 AWS Put*Events API 调用事件作为事件模式，因为任何 Put*Events 请求的最大大小为 256 KB。有关您可以使用的 API 调用的更多信息，请参阅CloudTrail 支持的服务和集成。

接收来自 AWS 服务的只读管理事件

您可以在默认或自定义事件总线上设置规则，通过接收来自 AWS 服务的只读管理事件 CloudTrail。管理事件可让您了解对 AWS 账户中的资源执行的管理操作。这些也称为控制层面操作。有关更多信息，请参阅《CloudTrail 用户指南》中的记录管理事件。

对于针对默认或自定义事件总线的每条规则，您可以设置规则状态，以控制要接收的事件类型：

- 禁用该规则，这样就 EventBridge 不会将事件与规则相匹配。
- 启用该规则，以便将事件 EventBridge 与规则进行匹配，但通过传送的只读 AWS 管理事件除外 CloudTrail。
- 启用该规则，以便将所有事件与该规则进行 EventBridge 匹配，包括通过传送的只读管理事件 CloudTrail。

合作伙伴事件总线不接收 AWS 事件。

在决定是否接收只读管理事件时需要考虑以下几点：

- 某些只读管理事件（例如 AWS Key Management Service `GetKeyPolicy`和`DescribeKey`或 IAM `GetPolicy` 和`GetRole`事件）的发生量远高于典型的变更事件。
- 如果只读管理事件不是以 `Describe`、`Get` 或 `List` 开头，您可能已经在接收这些事件了。例如，来自以下 AWS STS API 的事件是变更事件，即使它们以动词开头也是如此`Get`：
  - `GetFederationToken`

- GetSessionToken

有关 AWS 服务不遵守Describe、Get或List命名约定的只读管理事件的列表，请参阅???。

使用 AWS CLI 创建接收只读管理事件的规则

- 使用 put-rule 命令创建或更新规则，同时使用参数执行以下操作：
  - 指定规则属于默认事件总线或特定的自定义事件总线
  - 将规则状态设置为 ENABLED_WITH_ALL_CLOUDTRAIL_MANAGEMENT_EVENTS

```
aws events put-rule --name "ruleForManagementEvents" --event-bus-name
"default" --state "ENABLED_WITH_ALL_CLOUDTRAIL_MANAGEMENT_EVENTS"
```

> **Note**
>
> 仅支持通过 AWS CLI 和 AWS CloudFormation 模板为 CloudWatch 管理事件启用规则。

Example

以下示例说明了如何匹配特定事件。最佳实践是定义一个用于匹配特定事件的专用规则，这样既清晰又便于编辑。

在这种情况下，专用规则与来自的AssumeRole管理事件相匹配 AWS Security Token Service。

```
{
    "source" : [ "aws.sts" ],
    "detail-type": ["AWS API Call via CloudTrail"],
    "detail" : {
        "eventName" : ["AssumeRole"]
    }
}
```

AWS 生成事件的服务

下表显示了生成事件的 AWS 服务。选择服务名称以查看有关该服务和如何协同 EventBridge 工作的更多信息。

生成事件的每项 AWS 服务都会将事件 EventBridge 作为最大努力或持久交付尝试发送到。有关更多信息，请参阅 ???。

此表列出了向其发送事件的 AWS 服务 EventBridge，但不包括所有服务。对于未列出的向其发送事件的服务 EventBridge，请假设已尽力交付。

服务	尝试类型
Alexa for Business	最大努力
AWS Account Management	最大努力
Amazon API Gateway	最大努力
AWS AppConfig	最大努力
Amazon AppFlow	最大努力
Application Auto Scaling	最大努力
AWS 应用程序成本剖析器	最大努力
AWS Application Migration Service	最大努力
Amazon Athena	最大努力
AWS Backup	最大努力
AWS Batch	持久
Amazon Braket	持久
AWS Certificate Manager	最大努力
Amazon Chime	最大努力
Amazon Cloud Directory	最大努力
AWS CloudFormation	持久
Amazon CloudFront	最大努力

服务	尝试类型
AWS CloudHSM	最大努力
Amazon CloudSearch	最大努力
AWS CloudShell	最大努力
活动来自 AWS CloudTrail	最大努力
Amazon CloudWatch	持久
Amazon CloudWatch 应用程序洞察	最大努力
Amazon CloudWatch 互联网监视器	最大努力
Amazon CloudWatch 日志	最大努力
Amazon S CloudWatch ynthetics	最大努力
AWS CodeArtifact	持久
AWS CodeBuild	最大努力
AWS CodeCommit	最大努力
AWS CodeDeploy	最大努力
Amazon P CodeGuru rofiler	最大努力
AWS CodePipeline	最大努力
AWS CodeStar	最大努力
CodeConnections	最大努力
Amazon Cognito Identity	最大努力
Amazon Cognito 用户群体	最大努力
Amazon Cognito Sync	最大努力

服务	尝试类型
AWS Config	最大努力
Amazon Connect	最大努力
Amazon Connect Voice ID	最大努力
AWS Control Tower	最大努力
AWS Database Migration Service	最大努力
AWS Data Exchange	最大努力
Amazon Data Lifecycle Manager	最大努力
AWS Data Pipeline	最大努力
AWS DataSync	最大努力
AWS Device Farm	最大努力
Amazon DevOps Guru	最大努力
AWS Direct Connect	最大努力
AWS Directory Service	最大努力
Amazon DynamoDB	最大努力
AWS Elastic Beanstalk	最大努力
Amazon Elastic Block Store	最大努力
Amazon Elastic Block Store 卷修改	最大努力
Amazon ElastiCache	最大努力
Amazon Elastic Compute Cloud (Amazon EC2)	最大努力
Amazon EC2 Auto Scaling	最大努力

服务	尝试类型
Amazon EC2 Fleet	最大努力
Amazon EC2 竞价型实例中断	最大努力
Amazon Elastic Container Registry	最大努力
Amazon Elastic Container Service	持久
AWS Elastic Disaster Recovery	最大努力
Amazon Elastic File System	最大努力
Amazon Elastic Kubernetes Service	最大努力
Elastic Load Balancing	最大努力
亚马逊弹性 MapReduce	最大努力
Amazon Elastic Transcoder	最大努力
AWS Elemental MediaConnect	最大努力
AWS Elemental MediaConvert	持久
AWS Elemental MediaLive	最大努力
AWS Elemental MediaPackage	最大努力
AWS Elemental MediaStore	持久
Amazon EMR	最大努力
Amazon EMR on EKS	最大努力
Amazon EMR Serverless	最大努力
亚马逊 EventBridge 预定规则	持久
亚马逊 EventBridge 架构	最大努力

服务	尝试类型
AWS Fault Injection Service	最大努力
预测	最大努力
Amazon GameLift	最大努力
AWS Glue	最大努力
AWS Glue DataBrew	最大努力
AWS Ground Station	最大努力
Amazon GuardDuty	最大努力
AWS Health	持久
AWS HealthLake	持久
AWS Identity and Access Management (IAM)	最大努力
IAM Access Analyzer	最大努力
Amazon Inspector Classic	最大努力
Amazon Inspector	最大努力
AWS IoT	最大努力
AWS IoT Analytics	持久
AWS IoT Greengrass V1	最大努力
AWS IoT Greengrass V2	最大努力
Amazon Interactive Video Service	最大努力
Amazon Kinesis	最大努力
Amazon Data Firehose	最大努力

服务	尝试类型
AWS Key Management Service 删除密钥	持久
AWS Key Management Service 密钥轮换	最大努力
AWS Key Management Service 导入的密钥材料到期	最大努力
AWS Lambda	最大努力
Amazon Location Service	持久
Amazon Machine Learning	最大努力
Amazon Macie	最大努力
Amazon Managed Blockchain	最大努力
AWS Managed Services	最大努力
AWS Management Console 登录	最大努力
AWS 计量市场	最大努力
AWS Migration Hub	最大努力
AWS Migration Hub Refactor Spaces	最大努力
AWS 监控	最大努力
AWS Network Manager	最大努力
亚马逊 OpenSearch 服务	最大努力
AWS OpsWorks	持久
AWS OpsWorks CM	最大努力
AWS Organizations	最大努力
Amazon Polly	最大努力

服务	尝试类型
AWS Private Certificate Authority	最大努力
AWS Proton	最大努力
Amazon QLDB	持久
Amazon QuickSight	最大努力
Amazon RDS	最大努力
AWS 回收站	最大努力
Amazon Redshift	持久
Amazon Redshift 数据 API	最大努力
Amazon Redshift Serverless	最大努力
AWS Resource Access Manager	最大努力
AWS Resource Groups	最大努力
AWS Resource Groups Tagging API	最大努力
Amazon Route 53	最大努力
Amazon Route 53 Recovery 就绪性	最大努力
Amazon SageMaker	最大努力
Savings Plans	最大努力
AWS Secrets Manager	最大努力
AWS Security Hub	持久
AWS Security Token Service	最大努力
AWS Server Migration Service	最大努力

服务	尝试类型
AWS Service Catalog	最大努力
AWS Signer	持久
Amazon Simple Email Service	最大努力
Amazon Simple Storage Service (Amazon S3)	持久
Amazon S3 Glacier	最大努力
Amazon S3 on Outposts	最大努力
Amazon Simple Queue Service	最大努力
Amazon Simple Notification Service	最大努力
Amazon Simple Workflow Service	最大努力
AWS Step Functions	最大努力
AWS Storage Gateway	持久
AWS Support	最大努力
AWS Systems Manager	最大努力
Amazon Transcribe	最大努力
AWS Transfer Family	最大努力
AWS Transit Gateway	最大努力
Amazon Translate	持久
AWS Trusted Advisor	最大努力
AWS WAF	最大努力
AWS WAF 区域性	最大努力

服务	尝试类型
AWS Well-Architected Tool	最大努力
Amazon WorkDocs	最大努力
Amazon WorkSpaces	最大努力
AWS X-Ray	最大努力

AWS 服务生成的管理事件

通常，生成管理（或只读）事件的 API 以动词 `Describe`、`Get` 或 `List` 开头。下表列出了不遵循此命名约定的 AWS 服务及其生成的管理事件。有关管理事件的更多信息，请参阅???。

不以 **`Describe`**、**`Get`** 或 **`List`** 开头的管理事件

下表列出了不遵循以、或开头的典型命名惯例的 AWS 服务及其生成的管理事件`List`。`Describe` `Get`

服务	事件名称	事件类型
Alexa for Business	ResolveRoom	API 调用
Alexa for Business	SearchAddressBooks	API 调用
Alexa for Business	SearchContacts	API 调用
Alexa for Business	SearchDevices	API 调用
Alexa for Business	SearchProfiles	API 调用
Alexa for Business	SearchRooms	API 调用
Alexa for Business	SearchSkillGroups	API 调用
Alexa for Business	SearchUsers	API 调用
IAM Access Analyzer	ValidatePolicy	API 调用

服务	事件名称	事件类型
AWS AdSpace 干净的房间	BatchGetSchema	API 调用
AWS Amplify 用户界面生成器	ExportComponents	API 调用
AWS Amplify 用户界面生成器	ExportForms	API 调用
AWS Amplify 用户界面生成器	ExportThemes	API 调用
亚马逊 OpenSearch 服务	BatchGetCollection	API 调用
Amazon API Gateway	ExportApi	API 调用
AWS AppConfig	ValidateConfiguration	API 调用
Amazon AppFlow	RetrieveConnectorData	API 调用
Amazon CloudWatch 应用程序洞察	UpdateApplicationDashboardConfiguration	API 调用
Amazon Athena	BatchGetNamedQuery	API 调用
Amazon Athena	BatchGetPreparedStatement	API 调用
Amazon Athena	BatchGetQueryExecution	API 调用
Amazon Athena	CheckQueryCompatibility	API 调用
Amazon Athena	ExportNotebook	API 调用
AWS Auto Scaling	AreScalableTargetsRegistered	API 调用
AWS Auto Scaling	测试	API 调用
AWS Marketplace	SearchAgreements	API 调用
AWS Backup	CreateLegalHold	API 调用
AWS Backup	ExportBackupPlanTemplate	API 调用
AWS Backup gateway	TestHypervisorConfiguration	API 调用

服务	事件名称	事件类型
AWS Billing and Cost Management	AWSPaymentInstrumentGateway.Get	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.DescribeMakePaymentPage	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.DescribePaymentsDashboard	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetAccountPreferences	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetAdvancePaySummary	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetAsoBulkDownload	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetBillingContactAddress	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetDocuments	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetEligiblePaymentInstruments	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetEntitiesByIds	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetFundingDocuments	控制台操作

服务	事件名称	事件类型
AWS Billing and Cost Management	AWSPaymentPortalService.GetKybcValidationStatus	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetOneTimePasswordStatus	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentHistory	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentProfileByArn	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentProfileCurrencies	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentProfiles	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentProfileServiceProviders	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentsDue	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetRemittanceInformation	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetTaxInvoiceMetadata	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetTermsAndConditionsForProgramGroup	控制台操作

服务	事件名称	事件类型
AWS Billing and Cost Management	AWSPaymentPortalService.GetTransactionsHistory	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetUnappliedFunds	控制台操作
AWS Billing and Cost Management	AWSPaymentPortalService.GetUnpaidInvoices	控制台操作
AWS Billing and Cost Management	AWSPaymentPreferenceGateway.Get	控制台操作
AWS Billing and Cost Management	CancelBulkDownload	控制台操作
AWS Billing and Cost Management	DownloadCommercialInvoice	控制台操作
AWS Billing and Cost Management	DownloadCsv	控制台操作
AWS Billing and Cost Management	DownloadDoc	控制台操作
AWS Billing and Cost Management	下载 CSV ForBillingPeriod	控制台操作
AWS Billing and Cost Management	DownloadPaymentHistory	控制台操作
AWS Billing and Cost Management	DownloadRegistrationDocument	控制台操作
AWS Billing and Cost Management	DownloadTaxInvoice	控制台操作
AWS Billing and Cost Management	FindBankRedirectPaymentInstruments	控制台操作

服务	事件名称	事件类型
AWS Billing and Cost Management	FindecsV ForBillingPeriod	控制台操作
AWS Billing and Cost Management	ValidateReportDestination	控制台操作
AWS Billing and Cost Management	VerifyChinaPaymentEligibility	控制台操作
Amazon Braket	SearchCompilations	API 调用
Amazon Braket	SearchDevices	API 调用
Amazon Braket	SearchQuantumTasks	API 调用
Amazon Connect Cases	BatchGetField	API 调用
Amazon Connect Cases	SearchCases	API 调用
Amazon Connect Cases	SearchRelatedItems	API 调用
Amazon Chime	RetrieveDataExports	API 调用
Amazon Chime	SearchChannels	API 调用
Amazon Chime SDK 身份	DeleteProfile	服务事件
Amazon Chime SDK 身份	DeleteWorkTalkAccount	服务事件
AWS 干净的房间	BatchGetSchema	API 调用
Amazon Cloud Directory	BatchRead	API 调用
Amazon Cloud Directory	LookupPolicy	API 调用
AWS CloudFormation	DetectStackDrift	API 调用
AWS CloudFormation	DetectStackResourceDrift	API 调用
AWS CloudFormation	DetectStackSetDrift	API 调用

服务	事件名称	事件类型
AWS CloudFormation	EstimateTemplateCost	API 调用
AWS CloudFormation	ValidateTemplate	API 调用
AWS CloudShell	RedeemCode	API 调用
AWS CloudTrail	LookupEvents	API 调用
AWS CodeArtifact	ReadFromRepository	API 调用
AWS CodeArtifact	SearchPackages	API 调用
AWS CodeArtifact	VerifyResourcesExistForTagris	API 调用
AWS CodeBuild	BatchGetBuildBatches	API 调用
AWS CodeBuild	BatchGetBuilds	API 调用
AWS CodeBuild	BatchGetProjects	API 调用
AWS CodeBuild	BatchGetReportGroups	API 调用
AWS CodeBuild	BatchGetReports	API 调用
AWS CodeBuild	BatchPutCodeCoverages	API 调用
AWS CodeBuild	BatchPutTestCases	API 调用
AWS CodeBuild	RequestBadge	服务事件
AWS CodeCommit	BatchDescribeMergeConflicts	API 调用
AWS CodeCommit	BatchGetCommits	API 调用
AWS CodeCommit	BatchGetPullRequests	API 调用
AWS CodeCommit	BatchGetRepositories	API 调用

服务	事件名称	事件类型
AWS CodeCommit	EvaluatePullRequestApprovalRules	API 调用
AWS CodeCommit	GitPull	API 调用
AWS CodeDeploy	BatchGetApplicationRevisions	API 调用
AWS CodeDeploy	BatchGetApplications	API 调用
AWS CodeDeploy	BatchGetDeploymentGroups	API 调用
AWS CodeDeploy	BatchGetDeploymentInstances	API 调用
AWS CodeDeploy	BatchGetDeployments	API 调用
AWS CodeDeploy	BatchGetDeploymentTargets	API 调用
AWS CodeDeploy	BatchGetOnPremisesInstances	API 调用
Amazon P CodeGuru rofiler	BatchGetFrameMetricData	API 调用
Amazon P CodeGuru rofiler	SubmitFeedback	API 调用
AWS CodePipeline	PollForJobs	API 调用
AWS CodePipeline	PollForThirdPartyJobs	API 调用
CodeConnections	StartAppRegistrationHandshake	API 调用
CodeConnections	StarTo AuthHandshake	API 调用
CodeConnections	ValidateHostWebhook	API 调用
Amazon CodeWhisperer	CreateCodeScan	API 调用
Amazon CodeWhisperer	CreateProfile	API 调用

服务	事件名称	事件类型
Amazon CodeWhisperer	CreateUploadUrl	API 调用
Amazon CodeWhisperer	GenerateRecommendations	API 调用
Amazon CodeWhisperer	UpdateProfile	API 调用
Amazon Cognito Identity	LookupDeveloperIdentity	API 调用
Amazon Cognito 用户群体	AdminGetDevice	API 调用
Amazon Cognito 用户群体	AdminGetUser	API 调用
Amazon Cognito 用户群体	AdminListDevices	API 调用
Amazon Cognito 用户群体	AdminListGroupsForUser	API 调用
Amazon Cognito 用户群体	AdminListUserAuthEvents	API 调用
Amazon Cognito 用户群体	Beta_Authorize_GET	服务事件
Amazon Cognito 用户群体	Confirm_GET	服务事件
Amazon Cognito 用户群体	ConfirmForgotPassword_GET	服务事件
Amazon Cognito 用户群体	Error_GET	服务事件
Amazon Cognito 用户群体	ForgotPassword_GET	服务事件
Amazon Cognito 用户群体	IntrospectToken	API 调用
Amazon Cognito 用户群体	Login_Error_POST	服务事件
Amazon Cognito 用户群体	Login_GET	服务事件
Amazon Cognito 用户群体	Mfa_GET	服务事件
Amazon Cognito 用户群体	MfaOption_GET	服务事件
Amazon Cognito 用户群体	ResetPassword_GET	服务事件

服务	事件名称	事件类型
Amazon Cognito 用户群体	Signup_GET	服务事件
Amazon Cognito 用户群体	UserInfo_GET	服务事件
Amazon Cognito 用户群体	UserInfo_POST	服务事件
Amazon Cognito Sync	BulkPublish	API 调用
Amazon Comprehend	BatchContainsPiiEntities	API 调用
Amazon Comprehend	BatchDetectDominantLanguage	API 调用
Amazon Comprehend	BatchDetectEntities	API 调用
Amazon Comprehend	BatchDetectKeyPhrases	API 调用
Amazon Comprehend	BatchDetectPiiEntities	API 调用
Amazon Comprehend	BatchDetectSentiment	API 调用
Amazon Comprehend	BatchDetectSyntax	API 调用
Amazon Comprehend	BatchDetectTargetedSentiment	API 调用
Amazon Comprehend	ClassifyDocument	API 调用
Amazon Comprehend	ContainsPiiEntities	API 调用
Amazon Comprehend	DetectDominantLanguage	API 调用
Amazon Comprehend	DetectEntities	API 调用
Amazon Comprehend	DetectKeyPhrases	API 调用
Amazon Comprehend	DetectPiiEntities	API 调用
Amazon Comprehend	DetectSentiment	API 调用

服务	事件名称	事件类型
Amazon Comprehend	DetectSyntax	API 调用
Amazon Comprehend	DetectTargetedSentiment	API 调用
Amazon Comprehend	DetectToxicContent	API 调用
AWS Compute Optimizer	ExportAutoScalingGroupRecommendations	API 调用
AWS Compute Optimizer	exporteB VolumeRecommendations	API 调用
AWS Compute Optimizer	exportec InstanceRecommendations	API 调用
AWS Compute Optimizer	Exportecs ServiceRecommendations	API 调用
AWS Compute Optimizer	ExportLambdaFunctionRecommendations	API 调用
AWS Compute Optimizer	导出 RDS InstanceRecommendations	API 调用
AWS Config	BatchGetAggregateResourceConfig	API 调用
AWS Config	BatchGetResourceConfig	API 调用
AWS Config	SelectAggregateResourceConfig	API 调用
AWS Config	SelectResourceConfig	API 调用
Amazon Connect	AdminGetEmergencyAccessToken	API 调用
Amazon Connect	SearchQueues	API 调用

服务	事件名称	事件类型
Amazon Connect	SearchRoutingProfiles	API 调用
Amazon Connect	SearchSecurityProfiles	API 调用
Amazon Connect	SearchUsers	API 调用
AWS Glue DataBrew	SendProjectSessionAction	API 调用
AWS Data Pipeline	EvaluateExpression	API 调用
AWS Data Pipeline	QueryObjects	API 调用
AWS Data Pipeline	ValidatePipelineDefinition	API 调用
AWS DataSync	VerifyResourcesExistForTagris	API 调用
AWS DeepLens	BatchGetDevice	API 调用
AWS DeepLens	BatchGetModel	API 调用
AWS DeepLens	BatchGetProject	API 调用
AWS DeepLens	CreateDeviceCertificates	API 调用
AWS DeepRacer	AdminGetAccountConfig	API 调用
AWS DeepRacer	AdminListAssociatedUsers	API 调用
AWS DeepRacer	TestRewardFunction	API 调用
AWS DeepRacer	VerifyResourcesExistForTagris	API 调用
Amazon Detective	BatchGetGraphMemberDatasources	API 调用
Amazon Detective	BatchGetMembershipDatasources	API 调用

服务	事件名称	事件类型
Amazon Detective	SearchGraph	API 调用
Amazon DevOps Guru	SearchInsights	API 调用
Amazon DevOps Guru	SearchOrganizationInsights	API 调用
AWS Database Migration Service	BatchStartRecommendations	API 调用
AWS Database Migration Service	ModifyRecommendation	API 调用
AWS Database Migration Service	StartRecommendations	API 调用
AWS Database Migration Service	VerifyResourcesExistForTagris	API 调用
AWS Directory Service	VerifyTrust	API 调用
Amazon Elastic Compute Cloud	ConfirmProductInstance	API 调用
Amazon Elastic Compute Cloud	ReportInstanceStatus	API 调用
Amazon Elastic Container Registry	BatchCheckLayerAvailability	API 调用
Amazon Elastic Container Registry	BatchGetImage	API 调用
Amazon Elastic Container Registry	BatchGetImageReferrer	API 调用
Amazon Elastic Container Registry	BatchGetRepositoryScanningConfiguration	API 调用

服务	事件名称	事件类型
Amazon Elastic Container Registry	DryRunEvent	服务事件
Amazon Elastic Container Registry	PolicyExecutionEvent	服务事件
Amazon Elastic Container Registry Public	BatchCheckLayerAvailability	API 调用
Amazon Elastic Container Service	DiscoverPollEndpoint	API 调用
Amazon Elastic Container Service	FindSubfleetRoute	API 调用
Amazon Elastic Container Service	ValidateResources	API 调用
Amazon Elastic Container Service	VerifyTaskSetsExist	API 调用
Amazon Elastic Kubernetes Service	AccessKubernetesApi	API 调用
AWS Elastic Beanstalk	CheckDNSAvailability	API 调用
AWS Elastic Beanstalk	RequestEnvironmentInfo	API 调用
AWS Elastic Beanstalk	RetrieveEnvironmentInfo	API 调用
AWS Elastic Beanstalk	ValidateConfigurationSettings	API 调用
Amazon Elastic File System	NewClientConnection	服务事件
Amazon Elastic File System	UpdateClientConnection	服务事件
Amazon Elastic Transcoder	ReadJob	API 调用
Amazon Elastic Transcoder	ReadPipeline	API 调用

服务	事件名称	事件类型
Amazon Elastic Transcoder	ReadPreset	API 调用
Amazon EventBridge	TestEventPattern	API 调用
Amazon EventBridge	TestScheduleExpression	API 调用
Amazon FinSpace API	BatchListCatalogNodesByDataset	API 调用
Amazon FinSpace API	BatchListNodesByDataset	API 调用
Amazon FinSpace API	BatchValidateAccess	API 调用
Amazon FinSpace API	CreateAuditRecordsQuery	API 调用
Amazon FinSpace API	SearchDatasets	API 调用
Amazon FinSpace API	SearchDatasetsV	API 调用
Amazon FinSpace API	ValidateIdToken	API 调用
AWS Firewall Manager	DisassociateAdminAccount	API 调用
Amazon Forecast	InvokeForecastEndpoint	API 调用
Amazon Forecast	QueryFeature	API 调用
Amazon Forecast	QueryForecast	API 调用
Amazon Forecast	QueryWhatIfForecast	API 调用
Amazon Forecast	VerifyResourcesExistForTagris	API 调用
Amazon Fraud Detector	BatchGetVariable	API 调用
Amazon Fraud Detector	VerifyResourcesExistForTagris	API 调用
FreeRTOS	VerifyEmailAddress	API 调用

服务	事件名称	事件类型
Amazon GameLift	RequestUploadCredentials	API 调用
Amazon GameLift	ResolveAlias	API 调用
Amazon GameLift	SearchGameSessions	API 调用
Amazon GameLift	ValidateMatchmakingRuleSet	API 调用
Amazon GameSparks	ExportSnapshot	API 调用
Amazon Location Service	BatchGetDevicePosition	API 调用
Amazon Location Service	CalculateRoute	API 调用
Amazon Location Service	CalculateRouteMatrix	API 调用
Amazon Location Service	SearchPlaceIndexForPosition	API 调用
Amazon Location Service	SearchPlaceIndexForSuggestions	API 调用
Amazon Location Service	SearchPlaceIndexForText	API 调用
Amazon S3 Glacier	InitiateJob	API 调用
AWS Glue	BatchGetBlueprints	API 调用
AWS Glue	BatchGetColumnStatisticsForTable	API 调用
AWS Glue	BatchGetCrawlers	API 调用
AWS Glue	BatchGetCustomEntityTypes	API 调用
AWS Glue	BatchGetDataQualityResult	API 调用
AWS Glue	BatchGetDevEndpoints	API 调用
AWS Glue	BatchGetJobs	API 调用

服务	事件名称	事件类型
AWS Glue	BatchGetmlTran	API 调用
AWS Glue	BatchGetPartition	API 调用
AWS Glue	BatchGetTriggers	API 调用
AWS Glue	BatchGetWorkflows	API 调用
AWS Glue	QueryJobRuns	API 调用
AWS Glue	QueryJobRunsAggregated	API 调用
AWS Glue	QueryJobs	API 调用
AWS Glue	QuerySchemaVersion Metadata	API 调用
AWS Glue	SearchTables	API 调用
AWS HealthLake	ReadResource	API 调用
AWS HealthLake	SearchWithGet	API 调用
AWS HealthLake	SearchWithPost	API 调用
AWS Identity and Access Management	GenerateCredentialReport	API 调用
AWS Identity and Access Management	GenerateOrganizationsAccess Report	API 调用
AWS Identity and Access Management	GenerateServiceLas tAccessedDetails	API 调用
AWS Identity and Access Management	SimulateCustomPolicy	API 调用
AWS Identity and Access Management	SimulatePrincipalPolicy	API 调用

服务	事件名称	事件类型
AWS 身份存储	IsMemberInGroups	API 调用
AWS 身份存储验证	BatchGetSession	API 调用
Amazon Inspector Classic	PreviewAgents	API 调用
Amazon Inspector Classic	BatchGetAccountStatus	API 调用
Amazon Inspector Classic	BatchGetFreeTrialInfo	API 调用
Amazon Inspector Classic	BatchGetMember	API 调用
AWS 开票	ValidateDocumentDeliveryS3LocationInfo	API 调用
AWS IoT	SearchIndex	API 调用
AWS IoT	TestAuthorization	API 调用
AWS IoT	TestInvokeAuthorizer	API 调用
AWS IoT	ValidateSecurityProfileBehaviors	API 调用
AWS IoT Analytics	SampleChannelData	API 调用
AWS IoT SiteWise	GatewaysVerifyResourcesExistForTagrisInternal	API 调用
AWS IoT Things Graph	SearchEntities	API 调用
AWS IoT Things Graph	SearchFlowExecutions	API 调用
AWS IoT Things Graph	SearchFlowTemplates	API 调用
AWS IoT Things Graph	SearchSystemInstances	API 调用
AWS IoT Things Graph	SearchSystemTemplates	API 调用
AWS IoT Things Graph	SearchThings	API 调用

服务	事件名称	事件类型
AWS IoT TwinMaker	ExecuteQuery	API 调用
AWS IoT Wireless	CreateNetworkAnalyzerConfiguration	API 调用
AWS IoT Wireless	DeleteNetworkAnalyzerConfiguration	API 调用
AWS IoT Wireless	DeregisterWirelessDevice	API 调用
Amazon Interactive Video Service	BatchGetChannel	API 调用
Amazon Interactive Video Service	BatchGetStreamKey	API 调用
Amazon Kendra	BatchGetDocumentStatus	API 调用
Amazon Kendra	查询	API 调用
适用于 Apache Flink 的亚马逊托管服务	DiscoverInputSchema	API 调用
AWS Key Management Service	Decrypt	API 调用
AWS Key Management Service	Encrypt	API 调用
AWS Key Management Service	GenerateDataKey	API 调用
AWS Key Management Service	GenerateDataKeyPair	API 调用
AWS Key Management Service	GenerateDataKeyPairWithoutPlaintext	API 调用

服务	事件名称	事件类型
AWS Key Management Service	GenerateDataKeyWithoutPlaintext	API 调用
AWS Key Management Service	GenerateMac	API 调用
AWS Key Management Service	GenerateRandom	API 调用
AWS Key Management Service	ReEncrypt	API 调用
AWS Key Management Service	Sign	API 调用
AWS Key Management Service	验证	API 调用
AWS Key Management Service	VerifyMac	API 调用
AWS Lake Formation	SearchDatabasesByLFTags	API 调用
AWS Lake Formation	SearchTablesByLFTags	API 调用
AWS Lake Formation	StartQueryPlanning	API 调用
Amazon Lex	BatchCreateCustomVocabularyItem	API 调用
Amazon Lex	BatchDeleteCustomVocabularyItem	API 调用
Amazon Lex	BatchUpdateCustomVocabularyItem	API 调用
Amazon Lex	DeleteCustomVocabulary	API 调用

服务	事件名称	事件类型
Amazon Lex	SearchAssociatedTranscripts	API 调用
Amazon Lightsail	创建 GUI SessionAccessDetails	API 调用
Amazon Lightsail	DownloadDefaultKeyPair	API 调用
Amazon Lightsail	IsVpcPeered	API 调用
Amazon CloudWatch 日志	FilterLogEvents	API 调用
Amazon Macie	BatchGetCustomDataIdentifiers	API 调用
Amazon Macie	UpdateFindingsFilter	API 调用
AWS Elemental MediaConnect	ManagedDescribeFlow	API 调用
AWS Elemental MediaConnect	PrivateDescribeFlowMeta	API 调用
AWS Application Migration Service	OperationalDescribeJobLogItems	API 调用
AWS Application Migration Service	OperationalDescribeJobs	API 调用
AWS Application Migration Service	OperationalDescribeReplicationConfigurationTemplates	API 调用
AWS Application Migration Service	OperationalDescribeSourceServer	API 调用
AWS Application Migration Service	OperationalGetLaunchConfiguration	API 调用

服务	事件名称	事件类型
AWS Application Migration Service	OperationalListSourceServers	API 调用
AWS Application Migration Service	VerifyClientRoleForMgn	API 调用
AWS HealthOmics	VerifyResourceExists	API 调用
AWS HealthOmics	VerifyResourcesExistForTagris	API 调用
Amazon Polly	SynthesizeLongSpeech	API 调用
Amazon Polly	SynthesizeSpeech	API 调用
Amazon Polly	SynthesizeSpeechGet	API 调用
AWS 提供托管专用网络的服务	Ping	API 调用
AWS Proton	DeleteEnvironmentTemplateVersion	API 调用
AWS Proton	DeleteServiceTemplateVersion	API 调用
Amazon QLDB	ShowCatalog	API 调用
Amazon QuickSight	GenerateEmbedUrlForAnonymousUser	API 调用
Amazon QuickSight	GenerateEmbedUrlForRegisteredUser	API 调用
Amazon QuickSight	QueryDatabase	服务事件
Amazon QuickSight	SearchAnalyses	API 调用
Amazon QuickSight	SearchDashboards	API 调用

服务	事件名称	事件类型
Amazon QuickSight	SearchDataSets	API 调用
Amazon QuickSight	SearchDataSources	API 调用
Amazon QuickSight	SearchFolders	API 调用
Amazon QuickSight	SearchGroups	API 调用
Amazon QuickSight	SearchUsers	API 调用
Amazon Relational Database Service	DownloadComplete数据库 LogFile	API 调用
Amazon Relational Database Service	下载DB LogFilePortion	API 调用
Amazon Rekognition	CompareFaces	API 调用
Amazon Rekognition	DetectCustomLabels	API 调用
Amazon Rekognition	DetectFaces	API 调用
Amazon Rekognition	DetectLabels	API 调用
Amazon Rekognition	DetectModerationLabels	API 调用
Amazon Rekognition	DetectProtectiveEquipment	API 调用
Amazon Rekognition	DetectText	API 调用
Amazon Rekognition	RecognizeCelebrities	API 调用
Amazon Rekognition	SearchFaces	API 调用
Amazon Rekognition	SearchFacesByImage	API 调用
Amazon Rekognition	SearchUsers	API 调用
Amazon Rekognition	SearchUsersByImage	API 调用

服务	事件名称	事件类型
AWS 资源探索器	BatchGetView	API 调用
AWS 资源探索器	Search	API 调用
AWS Resource Groups	SearchResources	API 调用
AWS Resource Groups	ValidateResourceSharing	API 调用
AWS RoboMaker	BatchDescribeSimulationJob	API 调用
Amazon Route 53	TestDNSAnswer	API 调用
Amazon Route 53 Domains	checkAvailabilities	API 调用
Amazon Route 53 Domains	CheckDomainAvailability	API 调用
Amazon Route 53 Domains	checkDomainTransferability	API 调用
Amazon Route 53 Domains	CheckDomainTransferability	API 调用
Amazon Route 53 Domains	isEmailReachable	API 调用
Amazon Route 53 Domains	searchDomains	API 调用
Amazon Route 53 Domains	sendVerificationMessage	API 调用
Amazon Route 53 Domains	ViewBilling	API 调用
Amazon Route 53 Domains	viewBilling	API 调用
亚马逊 CloudWatch RUM	BatchGetRumMetricDefinitions	API 调用
Amazon Simple Storage Service	echo	API 调用
Amazon Simple Storage Service	GenerateInventory	服务事件
Amazon SageMaker	BatchDescribeModelPackage	API 调用

服务	事件名称	事件类型
Amazon SageMaker	DeleteModelCard	API 调用
Amazon SageMaker	QueryLineage	API 调用
Amazon SageMaker	RenderUiTemplate	API 调用
Amazon SageMaker	Search	API 调用
亚马逊 EventBridge 架构	ExportSchema	API 调用
亚马逊 EventBridge 架构	SearchSchemas	API 调用
Amazon SimpleDB	DomainMetadata	API 调用
AWS Secrets Manager	ValidateResourcePolicy	API 调用
AWS Service Catalog	ScanProvisionedProducts	API 调用
AWS Service Catalog	SearchProducts	API 调用
AWS Service Catalog	SearchProductsAsAdmin	API 调用
AWS Service Catalog	SearchProvisionedProducts	API 调用
Amazon SES	BatchGetMetricData	API 调用
Amazon SES	TestRenderEmailTemplate	API 调用
Amazon SES	TestRenderTemplate	API 调用
Amazon Simple Notification Service	CheckIfPhoneNumber IsOptedOut	API 调用
AWS SQL Workbench	BatchGetNotebookCell	API 调用
AWS SQL Workbench	ExportNotebook	API 调用
Amazon EC2 Systems Manager	ExecuteApi	API 调用

服务	事件名称	事件类型
AWS Systems Manager Incident Manager	DeleteContactChannel	API 调用
AWS IAM Identity Center	IsMemberInGroup	API 调用
AWS IAM Identity Center	SearchGroups	API 调用
AWS IAM Identity Center	SearchUsers	API 调用
AWS STS	AssumeRole	API 调用
AWS STS	AssumeRoleWithSAML	API 调用
AWS STS	AssumeRoleWithWebIdentity	API 调用
AWS STS	DecodeAuthorizationMessage	API 调用
AWS 税务设置	BatchGetTaxExemptions	API 调用
AWS WAFV2	CheckCapacity	API 调用
AWS WAFV2	GenerateMobileSdkReleaseUrl	API 调用
AWS Well-Architected Tool	ExportLens	API 调用
AWS Well-Architected Tool	TagResource	API 调用
AWS Well-Architected Tool	UntagResource	API 调用
AWS Well-Architected Tool	UpdateGlobalSettings	API 调用
Amazon Connect Wisdom	QueryAssistant	API 调用
Amazon Connect Wisdom	SearchContent	API 调用
Amazon Connect Wisdom	SearchSessions	API 调用
Amazon WorkDocs	AbortDocumentVersionUpload	API 调用

服务	事件名称	事件类型
Amazon WorkDocs	AddUsersToGroup	API 调用
Amazon WorkDocs	BatchGetUsers	API 调用
Amazon WorkDocs	CheckAlias	API 调用
Amazon WorkDocs	CompleteDocumentVersionUpload	API 调用
Amazon WorkDocs	CreateAnnotation	API 调用
Amazon WorkDocs	CreateComment	API 调用
Amazon WorkDocs	CreateFeedbackRequest	API 调用
Amazon WorkDocs	CreateFolder	API 调用
Amazon WorkDocs	CreateGroup	API 调用
Amazon WorkDocs	CreateShare	API 调用
Amazon WorkDocs	CreateUser	API 调用
Amazon WorkDocs	DeleteAnnotation	API 调用
Amazon WorkDocs	DeleteComment	API 调用
Amazon WorkDocs	DeleteDocument	API 调用
Amazon WorkDocs	DeleteFeedbackRequest	API 调用
Amazon WorkDocs	DeleteFolder	API 调用
Amazon WorkDocs	DeleteFolderContents	API 调用
Amazon WorkDocs	DeleteGroup	API 调用
Amazon WorkDocs	DeleteOrganizationShare	API 调用
Amazon WorkDocs	DeleteUser	API 调用

服务	事件名称	事件类型
Amazon WorkDocs	DownloadDocumentVersion	API 调用
Amazon WorkDocs	DownloadDocumentVersionUnderlays	API 调用
Amazon WorkDocs	InitiateDocumentVersionUpload	API 调用
Amazon WorkDocs	LogoutUser	API 调用
Amazon WorkDocs	PaginatedOrganizationActivity	API 调用
Amazon WorkDocs	PublishAnnotations	API 调用
Amazon WorkDocs	PublishComments	API 调用
Amazon WorkDocs	RestoreDocument	API 调用
Amazon WorkDocs	RestoreFolder	API 调用
Amazon WorkDocs	SearchGroups	API 调用
Amazon WorkDocs	SearchOrganizationUsers	API 调用
Amazon WorkDocs	TransferUserResources	API 调用
Amazon WorkDocs	UpdateAnnotation	API 调用
Amazon WorkDocs	UpdateComment	API 调用
Amazon WorkDocs	UpdateDocument	API 调用
Amazon WorkDocs	UpdateDocumentVersion	API 调用
Amazon WorkDocs	UpdateFolder	API 调用
Amazon WorkDocs	UpdateGroup	API 调用
Amazon WorkDocs	UpdateOrganization	API 调用

服务	事件名称	事件类型
Amazon WorkDocs	UpdateUser	API 调用
Amazon WorkMail	AssumeImpersonationRole	API 调用
Amazon WorkMail	QueryDnsRecords	API 调用
Amazon WorkMail	SearchMembers	API 调用
Amazon WorkMail	TestAvailabilityConfiguration	API 调用
Amazon WorkMail	TestInboundMailFlowRules	API 调用
Amazon WorkMail	TestOutboundMailFlowRules	API 调用

EventBridge 事件详情参考

EventBridge 本身会发出以下事件。与任何其他 AWS 服务一样，这些事件会自动发送到默认事件总线。

有关所有事件中包含的元数据字段的定义，请参阅the section called “事件结构参考”。

主题

预定活动

以下是该`Scheduled Event`活动的详细信息字段。

之所以包括`source`和`detail-type`字段，是因为它们包含 EventBridge 事件的特定值。有关所有事件中包含的其他元数据字段的定义，请参阅the section called “事件结构参考”。

```
{
  . . .,
  "detail-type": "Scheduled Event",
  "source": "aws.events",
```

```
  . . .,
  "detail": {}
}
```

`detail-type`

标识事件的类型。

对于此事件，此值为`Scheduled Event`。

必需：是

`source`

标识生成事件的服务。对于 EventBridge 事件，此值为`aws.events`。

必需：是

`detail`

包含关于事件信息的 JSON 对象。生成事件的服务决定该字段的内容。

必需：是

此对象中没有`Scheduled Event`事件的必填字段。

Example 预定事件事件示例

```
{
  "version": "0",
  "id": "89d1a02d-5ec7-412e-82f5-13505f849b41",
  "detail-type": "Scheduled Event",
  "source": "aws.events",
  "account": "123456789012",
  "time": "2016-12-30T18:44:49Z",
  "region": "us-east-1",
  "resources": ["arn:aws:events:us-east-1:123456789012:rule/SampleRule"],
  "detail": {}
}
```

架构已创建

以下是该`Schema Created`活动的详细信息字段。

创建架构后，EventBridge 会同时发送Schema Created和Schema Version Created事件。

之所以包括source和detail-type字段，是因为它们包含 EventBridge 事件的特定值。有关所有事件中包含的其他元数据字段的定义，请参阅the section called “事件结构参考”。

```
{
  . . .,
  "detail-type": "Schema Created",
  "source": "aws.schemas",
  . . .,
  "detail": {
    "SchemaName" : "String",
    "SchemaType" : "String",
    "RegistryName" : "String",
    "CreationDate" : "DateTime",
    "Version" : "Number"
  }
}
```

`detail-type`

标识事件的类型。

对于此事件，此值为Schema Created。

必需：是

`source`

标识生成事件的服务。对于 EventBridge 事件，此值为aws.schemas。

必需：是

`detail`

包含关于事件信息的 JSON 对象。生成事件的服务决定该字段的内容。

必需：是

对于此事件，这些数据包括：

SchemaName

架构的名称。

必需：是

SchemaType

架构的类型。

有效值：OpenApi3 | JSONSchemaDraft4

必需：是

RegistryName

包含该架构的注册表的名称。

必需：是

CreationDate

架构的创建日期。

必需：是

Version

架构的版本。

对于Schema Created事件，此值将始终为1。

必需：是

Example 示例 “架构已创建” 事件

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Schema Created",
  "source": "aws.schemas",
  "account": "123456789012",
  "time": "2019-05-31T21:49:54Z",
  "region": "us-east-1",
  "resources": ["arn:aws:schemas:us-east-1::schema/myRegistry/mySchema"],
  "detail": {
    "SchemaName": "mySchema",
    "SchemaType": "OpenApi3",
```

```
    "RegistryName": "myRegistry",
    "CreationDate": "2019-11-29T20:08:55Z",
    "Version": "1"
  }
}
```

架构版本已创建

以下是该Schema Version Created活动的详细信息字段。

创建架构后，EventBridge 会同时发送Schema Created和Schema Version Created事件。

之所以包括source和detail-type字段，是因为它们包含 EventBridge 事件的特定值。有关所有事件中包含的其他元数据字段的定义，请参阅the section called “事件结构参考”。

```
{
  . . .,
  "detail-type": "Schema Version Created",
  "source": "aws.schemas",
  . . .,
  "detail": {
    "SchemaName" : "String",
    "SchemaType" : "String",
    "RegistryName" : "String",
    "CreationDate" : "DateTime",
    "Version" : "Number"
  }
}
```

`detail-type`

标识事件的类型。

对于此事件，此值为Schema Version Created。

必需：是

`source`

标识生成事件的服务。对于 EventBridge 事件，此值为aws.schemas。

必需：是

`detail`

包含关于事件信息的 JSON 对象。生成事件的服务决定该字段的内容。

必需：是

对于此事件，这些数据包括：

`SchemaName`

架构的名称。

必需：是

`SchemaType`

架构的类型。

有效值：`OpenApi3` | `JSONSchemaDraft4`

必需：是

`RegistryName`

包含该架构的注册表的名称。

必需：是

`CreationDate`

架构版本的创建日期。

必需：是

`Version`

架构的版本。

必需：是

Example 示例 “架构版本已创建” 事件

```
{
  "version": "0",
```

```
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Schema Version Created",
  "source": "aws.schemas",
  "account": "123456789012",
  "time": "2019-05-31T21:49:54Z",
  "region": "us-east-1",
  "resources": ["arn:aws:schemas:us-east-1::schema/myRegistry/mySchema"],
  "detail": {
    "SchemaName": "mySchema",
    "SchemaType": "OpenApi3",
    "RegistryName": "myRegistry",
    "CreationDate": "2019-11-29T20:08:55Z",
    "Version": "5"
  }
}
```

通过 Amazon 接收来自 SaaS 合作伙伴的事件 EventBridge

为了能够从 SaaS 合作伙伴应用程序和服务接收事件，您需要合作伙伴提供的合作伙伴事件源。然后，您可以创建合作伙伴事件总线，并将其与合作伙伴事件源匹配。

以下视频介绍了 SaaS 与以下方面的集成 EventBridge：软件即服务 (SaaS) 合作伙伴

主题

- 支持的 SaaS 合作伙伴集成
- 将 Amazon 配置 EventBridge 为接收来自 SaaS 集成的事件
- 创建与 SaaS 合作伙伴事件匹配的规则
- 使用 AWS Lambda 函数 URL 接收事件
- 从 Salesforce 接收事件

支持的 SaaS 合作伙伴集成

EventBridge 支持以下 SaaS 合作伙伴集成：

- Adobe
- Auth0
- Blitline

- BUIDLHub
- Buildkite
- CleverTap
- Datadog
- Epsagon
- Freshworks
- Genesys
- GS2
- Karte
- Kloudless
- Mackerel
- MongoDB
- New Relic
- OneLogin
- Opsgenie
- PagerDuty
- Payshield
- SaaSus Platform
- SailPoint
- Saviynt
- Segment
- Shopify
- SignalFx
- Site24x7
- Stax
- Stripe
- SugarCRM
- SugarCRM
- Symantec
- Thundra

- TriggerMesh
- Whispir
- Zendesk
- Amazon Seller Partner API

合作伙伴事件源在以下区域中可用。

代码	名称
us-east-1	美国东部（弗吉尼亚州北部）
us-east-2	美国东部（俄亥俄州）
us-west-1	美国西部（北加利福尼亚）
us-west-2	美国西部（俄勒冈）
ca-central-1	加拿大（中部）
eu-central-1	欧洲地区（法兰克福）
eu-central-2	欧洲（苏黎世）
eu-west-1	欧洲地区（爱尔兰）
eu-west-2	欧洲地区（伦敦）
eu-west-3	欧洲地区（巴黎）
eu-north-1	Europe (Stockholm)
eu-south-1	欧洲地区（米兰）
eu-south-2	欧洲（西班牙）
af-south-1	非洲（开普敦）
ap-south-1	亚太地区（孟买）
ap-south-2	亚太地区（海得拉巴）

代码	名称
ap-east-1	亚太地区（香港）
ap-northeast-1	亚太地区（东京)
ap-northeast-2	亚太地区 (首尔)
ap-northeast-3	亚太地区（大阪）
ap-southeast-1	亚太地区（新加坡）
ap-southeast-2	亚太地区（悉尼）
ap-southeast-3	亚太地区（雅加达）
ap-southeast-4	亚太地区（墨尔本）
cn-north-1	中国（北京）
cn-northwest-1	中国（宁夏）
me-central-1	中东（阿联酋）
me-south-1	中东（巴林）
sa-east-1	南美洲（圣保罗）
il-central-1	以色列（特拉维夫）

将 Amazon 配置 EventBridge 为接收来自 SaaS 集成的事件

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择合作伙伴事件源。
3. 查找所需的合作伙伴，然后为该合作伙伴选择设置。
4. 要将您的账户 ID 复制到剪贴板，选择复制。
5. 在导航窗格中，选择合作伙伴事件源。
6. 转到合作伙伴的网站，并按照说明，使用您的账户 ID 创建合作伙伴事件源。您创建的事件源仅供您的账户使用。

7. 返回 EventBridge 控制台，在导航窗格中选择合作伙伴事件源。
8. 选择合作伙伴事件源旁边的按钮，然后选择与事件总线关联。

   该事件源的状态从 Pending 更改为 Active，并更新事件总线的名称，以匹配伙伴事件源名称。您现在可以开始创建，以匹配来自该合作伙伴事件源的事件。有关更多信息，请参阅 创建与 SaaS 合作伙伴事件匹配的规则。

   > **Note**
   >
   > 合作伙伴向任何未与事件总线关联的合作伙伴事件源发布的事件都将立即被删除。这些事件不会在静止状态下持续下去。 EventBridge

创建与 SaaS 合作伙伴事件匹配的规则

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择规则。
3. 选择创建规则。
4. 为规则输入名称和描述。

   规则不能与同一区域中的另一个规则和同一事件总线上的名称相同。
5. 对于事件总线，请选择要与此规则关联的事件总线。如果您希望此规则对来自您自己的账户的匹配事件触发，请选择 AWS 默认事件总线。当您账户中的某个 AWS 服务发出一个事件时，它始终会发送到您账户的默认事件总线。
6. 对于规则类型，选择具有事件模式的规则。
7. 选择下一步。
8. 对于事件源，选择其他。
9. （可选）对于示例事件，请选择事件的类型。
10. 在事件模式中，输入 JSON 事件模式。
11. 选择 Next（下一步）。
12. 对于目标类型，选择AWS 服务。
13. 在 “选择目标” 中，选择在 EventBridge 检测到与事件模式匹配的事件时要向其发送信息的 AWS 服务。
14. 显示的字段因您选择的服务而异。根据需要输入此目标类型的特定信息。

15. 对于许多目标类型， EventBridge 需要向目标发送事件的权限。在这些情况下， EventBridge 可以创建规则运行所需的 IAM 角色。请执行以下操作之一：

    - 要自动创建 IAM 角色，请选择为此特定资源创建新角色。
    - 要使用您之前创建的 IAM 角色，请选择使用现有角色，然后从下拉列表中选择现有角色。

16. （可选）对于 Additional settings（其他设置），执行以下操作：

    a. 对于 Maximum age of event（事件的最大时长），输入一分钟（00:01）与 24 小时（24:00）之间的值。

    b. 对于重试尝试，输入 0 到 185 之间的数字。

    c. 对于死信队列，选择是否使用标准的 Amazon SQS 队列作为死信队列。 EventBridge 如果匹配此规则的事件未成功传送到目标，则将其发送到死信队列。请执行以下操作之一：

        - 选择无不使用死信队列。
        - 在当前 AWS 帐户中选择选择一个Amazon SQS队列用作死信队列，然后从下拉列表中选择要使用的队列。
        - 选择选择其他 AWS 账户中的 Amazon SQS 队列作为死信队列，然后输入要使用的队列的 ARN。您必须将基于资源的策略附加到队列，以授予向该队列发送消息的 EventBridge 权限。有关更多信息，请参阅 为死信队列授予权限。

17. （可选）选择 Add another target（添加其他目标），以为此规则添加其他目标。

18. 选择 Next（下一步）。

19. （可选）为规则输入一个或多个标签。有关更多信息，请参阅 亚马逊 EventBridge 标签。

20. 选择 下一步。

21. 查看规则详细信息并选择创建规则。

使用 AWS Lambda 函数 URL 接收事件

> **Note**
>
> 为了让我们的合作伙伴能够访问入站 Webhook，我们正在 AWS 您的账户中创建一个 Open Lambda，通过验证第三方合作伙伴发送的身份验证签名，在 Lambda 应用程序级别上进行保护。请与您的安全团队一起检查此配置。有关更多信息，请参阅 Lambda 函数 URL 的安全性和身份验证模型。

您的 Amazon EventBridge 事件总线可以使用 AWS CloudFormation 模板创建的AWS Lambda 函数 URL 来接收来自支持的 SaaS 提供商的事件。使用函数 URL，事件数据将发送到 Lambda 函数。然后，该函数将这些数据转换为事件，该事件可以由事件总线接收 EventBridge 并发送到事件总线进行处理。事件进入事件总线后，您可以使用规则来筛选事件，应用任何已配置的输入转换，然后将其路由到正确的目标。

> **Note**
>
> 创建 Lambda 函数 URL 会增加您的月度成本。有关更多信息，请参阅AWS Lambda 定价。

要设置与的连接 EventBridge，首先要选择要与之建立连接的 SaaS 提供商。然后，提供您与该提供商一起创建的签名密钥，然后选择要向其发送 EventBridge 事件的事件总线。最后，使用 AWS CloudFormation 模板并创建完成连接所需的资源。

以下 SaaS 提供商目前可用于 EventBridge 使用 Lambda 函数网址：

- GitHub
- Stripe
- Twilio

主题

- 设置 Twilio 连接
- 更新 Webhook 密钥或身份验证令牌
- 更新 Lambda 函数
- 可用事件类型
- 配额、错误代码和传送重试

设置 GitHub 连接

步骤 1：创建 AWS CloudFormation 堆栈

首先，使用 Amazon EventBridge 控制台创建 CloudFormation 堆栈：

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 从导航窗格中选择快速入门。
3. 在使用 Lambda fURL 的入站 Webhook下，选择开始使用。
4. 在 GitHub 下，选择设置。
5. 在步骤 1：选择事件总线下，从下拉列表中选择一个事件总线。此事件总线从您提供给 GitHub 的 Lambda 函数 URL 接收数据。您也可以选择新建事件总线来创建事件总线。
6. 在 “步骤 2：使用进行设置” 下 CloudFormation，选择 “新建 GitHub webhook”。
7. 选择我确认我创建的入站 Webhook 可以公开访问。然后选择确认。
8. 输入堆栈的名称。
9. 在参数下，验证是否列出了正确的事件总线，然后为 GitHubWebhookSecret 指定安全令牌。有关创建安全令牌的更多信息，请参阅 GitHub 文档中的设置您的密钥令牌。
10. 在功能和转换下，选择以下各项：
    - 我承认这 AWS CloudFormation 可能会创建 IAM 资源。
    - 我承认这 AWS CloudFormation 可能会创建带有自定义名称的 IAM 资源。
    - 我承认这 AWS CloudFormation 可能需要以下能力：**CAPABILITY_AUTO_EXPAND**
11. 选择创建堆栈。

步骤 2：创建 GitHub Webhook

接下来，在 GitHub 中创建 Webhook。要完成此步骤，您需要安全令牌和在步骤 2 中创建的 Lambda 函数 URL。有关更多信息，请参阅 GitHub 文档中的创建 Webhook。

设置 Stripe 连接

步骤 1：创建 Stripe 端点

要在 EventBridge 和 Stripe 之间建立连接，请先创建一个 Stripe 端点并记下终端节点的密钥。在步骤 2 中设置堆栈时，您将使用此端点密钥。有关更多信息，请参阅 Stripe 文档中的交互式 Webhook 端点生成器。

> **Note**
>
> 您需要一个虚拟 URL 来设置 Stripe 端点。例如，`www.example.com`。

步骤 2：创建 AWS CloudFormation 堆栈

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中选择快速入门。
3. 在使用 Lambda fURL 的入站 Webhook下，选择开始使用。
4. 在 Stripe 下，选择设置。
5. 在步骤 1：选择事件总线下，从下拉列表中选择一个事件总线。此事件总线从您提供给 Stripe 的 Lambda 函数 URL 接收数据。您也可以选择新建事件总线来创建事件总线。
6. 在 “步骤 2：使用进行设置” 下 CloudFormation，选择 “新建 Stripe webhook”。
7. 选择我确认我创建的入站 Webhook 可以公开访问。然后选择确认。
8. 输入堆栈的名称。
9. 在参数下，验证是否列出了正确的事件总线，然后输入您在步骤 1 中创建的 StripeWebhookSecret。
10. 在功能和转换下，选择以下各项：
    - 我承认这 AWS CloudFormation 可能会创建 IAM 资源。
    - 我承认这 AWS CloudFormation 可能会创建带有自定义名称的 IAM 资源。
    - 我承认这 AWS CloudFormation 可能需要以下能力：**`CAPABILITY_AUTO_EXPAND`**
11. 选择创建堆栈。

步骤 3：更新 Stripe 端点

现在，您已经创建了 Lambda 函数 URL，请更新 Stripe 端点以向 Lambda 函数 URL 发送事件。

设置 Twilio 连接

步骤 1：查找您的 Twilio 身份验证令牌

要在Twilio和之间建立连接 EventBridge，请先Twilio使用Twilio账户的身份验证令牌或密钥将连接设置为。有关更多信息，请参阅 Twilio 文档中的身份验证令牌及其更改方法。

步骤 2：创建 AWS CloudFormation 堆栈

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中选择快速入门。
3. 在使用 Lambda fURL 的入站 Webhook下，选择开始使用。
4. 在 Twilio 下，选择设置。
5. 在步骤 1：选择事件总线下，从下拉列表中选择一个事件总线。此事件总线从您提供给 Twilio 的 Lambda 函数 URL 接收数据。您也可以选择新建事件总线来创建事件总线。
6. 在 “步骤 2：使用进行设置” 下 CloudFormation，选择 “新建 Twilio webhook”。
7. 选择我确认我创建的入站 Webhook 可以公开访问。然后选择确认。
8. 输入堆栈的名称。
9. 在参数下，验证是否列出了正确的事件总线，然后输入您在步骤 1 中创建的 TwilioWebhookSecret。
10. 在功能和转换下，选择以下各项：
    - 我承认这 AWS CloudFormation 可能会创建 IAM 资源。
    - 我承认这 AWS CloudFormation 可能会创建带有自定义名称的 IAM 资源。
    - 我承认这 AWS CloudFormation 可能需要以下功能：CAPABILITY_AUTO_EXPAND
11. 选择创建堆栈。

步骤 3：创建 Twilio Webhook

设置 Lambda 函数 URL 后，您需要将其提供给 Twilio，以便发送事件数据。有关更多信息，请参阅 Twilio 文档中的使用 Twilio 配置您的公共 URL。

更新 Webhook 密钥或身份验证令牌

更新 GitHub 密钥

> **Note**
>
> GitHub 不支持同时拥有两个密钥。当 AWS CloudFormation 堆栈中的GitHub密钥和密钥不同步时，您可能会遇到资源停机情况。 GitHub由于签名不正确，在密钥不同步时发送的消息将失败。等到GitHub和 CloudFormation 密钥同步，然后重试。

1. 新建 GitHub 密钥。有关更多信息，请参阅 GitHub 文档中加密机密。
2. 打开 AWS CloudFormation 控制台，网址为 https://console.aws.amazon.com/cloudformation。
3. 从导航窗格中，选择堆栈。
4. 为 Webhook 选择堆栈，其中包含要更新的密钥。
5. 选择更新。
6. 确保选中使用当前模板，然后选择下一步。
7. 在下方 GitHubWebhookSecret，清除 “使用现有值”，输入您在步骤 1 中创建的新GitHub密钥，然后选择 “下一步”。
8. 选择下一步。
9. 选择更新堆栈。

密钥传播可能最多需要一个小时。为了缩短停机时间，您可以刷新 Lambda 执行上下文。

更新 Stripe 密钥

1. 在 Stripe 控制面板的 Webhooks 部分，选择滚动密钥，并将到期时间至少延迟两 (2) 小时。有关更多信息，请参阅 Stripe 文档中部署端点密钥。
2. 打开 AWS CloudFormation 控制台，网址为 https://console.aws.amazon.com/cloudformation。
3. 从导航窗格中，选择堆栈。
4. 为 Webhook 选择堆栈，其中包含要更新的密钥。
5. 选择更新。
6. 确保选中使用当前模板，然后选择下一步。

7. 在下方 StripeWebhookSecret，清除 “使用现有值”，输入您在步骤 1 中创建的新Stripe密钥，然后选择 “下一步”。
8. 选择下一步。
9. 选择更新堆栈。

Stripe 将在轮换期间同时发送旧签名和新签名。

更新 Twilio 密钥

> **Note**
>
> Twilio 不支持同时拥有两个密钥。当 AWS CloudFormation 堆栈中的Twilio密钥和密钥不同步时，您可能会遇到资源停机情况。 Twilio由于签名不正确，在密钥不同步时发送的消息将失败。等到Twilio和 CloudFormation 密钥同步，然后重试。

1. 新建 Twilio 密钥。有关更多信息，请参阅 Twilio 文档中的身份验证令牌及其更改方法。
2. 打开 AWS CloudFormation 控制台，网址为 https://console.aws.amazon.com/cloudformation。
3. 从导航窗格中，选择堆栈。
4. 为 Webhook 选择堆栈，其中包含要更新的密钥。
5. 选择更新。
6. 确保选中使用当前模板，然后选择下一步。
7. 在下方 TwilioWebhookSecret，清除 “使用现有值”，输入您在步骤 1 中创建的新Twilio密钥，然后选择 “下一步”。
8. 选择下一步。
9. 选择更新堆栈。

密钥传播可能最多需要一个小时。为了缩短停机时间，您可以刷新 Lambda 执行上下文。

更新 Lambda 函数

CloudFormation 堆栈创建的 Lambda 函数创建基本的 webhook。如果您想针对特定用例（例如自定义日志）自定义 Lambda 函数，请使用 CloudFormation 控制台访问该函数，然后使用 Lambda 控制台更新 Lambda 函数代码。

访问 Lambda 函数

1. 打开 AWS CloudFormation 控制台，网址为 https://console.aws.amazon.com/cloudformation。
2. 从导航窗格中，选择堆栈。
3. 为 Webhook 选择堆栈，其中包含要更新的 Lambda 函数。
4. 选择资源选项卡。
5. 要在 Lambda 控制台中打开 Lambda 函数，请在物理 ID 下选择 Lambda 函数的 ID。

现在您已获得了 Lambda 函数，请使用 Lambda 控制台更新函数代码。

更新 Lambda 函数代码

1. 在操作下，选择导出函数。
2. 选择下载部署包并将文件保存到您的计算机中。
3. 解压缩部署包 .zip 文件，更新 `app.py` 文件，然后压缩更新后的部署包，确保包含原始 .zip 文件中的所有文件。
4. 在 Lambda 控制台中，选择代码选项卡。
5. 在 Code source（代码源）下，选择 Upload from（上载自）。
6. 选择 .zip file (.zip 文件)，然后选择 Upload file (上载文件)。
   - 在文件选择器中，选择您更新的文件，然后依次选择打开和保存。
7. 在操作下，选择发布新版本。

可用事件类型

事件总线目前支持以下 CloudFormation 事件类型：

- GitHub— 支持所有事件类型。
- Stripe - 支持所有事件类型。
- Twilio - 支持事件后 Webhook。

配额、错误代码和传送重试

配额

Webhook 的传入请求数量受底层 AWS 服务的限制。下表包括相关的配额。

服务	限额
AWS Lambda	默认：10 个并发执行 有关配额的更多信息（包括如何请求增加配额），请参阅 Lambda 配额。
AWS Secrets Manager	默认：每秒 5,000 个请求 有关配额的更多信息（包括如何请求增加配额），请参阅 AWS Secrets Manager 配额。 **Note** 使用 AWS Secrets Manager Python 缓存客户端可以最大限度地减少每秒的请求数。
Amazon EventBridge	操作的最大条目大小为 256KB。 PutEvents EventBridge 强制执行基于区域的费率配额。有关更多信息，请参阅 ???。

错误代码

发生错误时，每项 AWS 服务都会返回特定的错误代码。下表包括相关的错误代码。

服务	错误代码	描述
AWS Lambda	429 “” TooManyRequestsException	超出并发执行配额。
AWS Secrets Manager	500 “Internal Server Error”	超出每秒请求数配额。
Amazon EventBridge	500 “Internal Server Error”	超出该区域的费率配额。

活动重新传送

发生错误时，您可以重试传送受影响的事件。每个 SaaS 提供商都有不同的重试步骤。

GitHub

可使用 GitHub Webhook API 检查任何 Webhook 调用的传送状态，并在需要时重新传送事件。有关更多信息，请参阅 GitHub 文档：

- 组织 - 为组织 Webhook 重新传送
- 存储库 - 为存储库 Webhook 重新传送
- 应用 - 为应用 Webhook 重新传送

Stripe

Stripe 尝试在长达三天的时间内传送 Webhook，并呈指数级退避。有关更多信息，请参阅 Stripe 文档：

- 传送尝试和重试
- 处理错误

Twilio

Twilio 用户可以使用连接覆盖来自定义事件重试选项。有关更多信息，请参阅 Twilio 文档中的 Webhook（HTTP 回调）：连接覆盖。

从 Salesforce 接收事件

您可以通过以下方式使用 Amazon EventBridge 接收来自Salesforce以下的事件：

- 使用Salesforce's事件总线中继功能直接在 EventBridge 合作伙伴事件总线上接收事件。
- 通过在 A mazon AppFlow 中配置Salesforce用作数据源的流程。 AppFlow 然后，Amazon 使用合作伙伴事件总线向发送Salesforce事件。 EventBridge

您可以使用 API 目标将事件信息发送到 Salesforce。事件发送到 Salesforce 后，即可通过流或 Apex 触发器进行处理。有关设置 Salesforce API 目标的更多信息，请参阅 ???。

主题

- 使用事件总线中继从 Salesforce 接收事件
- Salesforce通过使用 Amazon 接收事件 AppFlow

使用事件总线中继从 Salesforce 接收事件

步骤 1：设置Salesforce事件总线中继和 EventBridge 合作伙伴事件源

在上创建事件中继配置时Salesforce，Salesforce会创建一个处于待处理状态的合作伙伴事件源。EventBridge

配置 Salesforce 事件总线中继

1. 设置 REST API 工具
2. （可选）定义平台事件
3. 为自定义平台活动创建频道
4. 创建频道成员，以关联自定义平台事件
5. 创建命名凭证
6. 创建事件中继配置

第 2 步：在 EventBridge 控制台中激活Salesforce合作伙伴事件源并启动事件中继

1. 在 EventBridge 控制台中打开合作伙伴事件源页面。
2. 选择您在步骤 1 中创建的 Salesforce 合作伙伴事件源。

3. 选择与事件总线关联。
4. 验证合作伙伴事件总线的名称。
5. 选择关联。
6. 启动事件中继

现在，您已经设置并启动了事件总线中继并配置了合作伙伴事件源，您可以创建一个对事件做出反应的 EventBridge 规则，以筛选数据并将其发送到目标。

Salesforce通过使用 Amazon 接收事件 AppFlow

Amazon AppFlow 将活动封装在活动信封Salesforce中。 EventBridge 以下示例显示了 EventBridge 合作伙伴Salesforce事件总线接收的事件。

```
{
    "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": "0056g000003XW7AAAW",
            "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"
    }
}
```

步骤 1： AppFlow 将 Amazon 配置Salesforce为合作伙伴事件源

要向发送事件 EventBridge，您首先需要 AppFlow 将 Amazon 配置Salesforce为合作伙伴事件源。

1. 在 Amazon AppFlow 控制台中，选择创建流程。
2. 在流详细信息部分，在流名称中输入流的名称。
3. （可选）输入流的描述，然后选择下一步。
4. 在源详细信息下，从源名称下拉列表中选择 Salesforce，然后选择连接以创建新连接。
5. 在连接到 Salesforce 对话框中，为 Salesforce 环境选择生产或沙盒。
6. 在连接名称字段中，输入连接的唯一名称，然后选择继续。
7. 在 Salesforce 对话框中，执行以下操作：
   a. 输入您的 Salesforce 登录凭证，登录 Salesforce。
   b. 为 Amazon AppFlow 要处理的数据类型选择Salesforce事件。
8. 在选择Salesforce事件下拉列表中，选择要发送到的事件类型 EventBridge。
9. 对于目的地，请选择 Amazon EventBridge。
10. 选择创建新的合作伙伴事件源。
11. （可选）为合作伙伴事件源指定唯一的后缀。
12. 选择生成合作伙伴事件源。
13. 选择一个 Amazon S3 桶来存储大于 256KB 的事件负载文件。
14. 在流触发器部分，确保选中按事件运行流。此设置可确保在发生新的 Salesforce 事件时执行流。
15. 选择下一步。
16. 要进行字段映射，请选择直接映射所有字段。也可以从源字段名称列表中选择您感兴趣的字段。

    有关字段映射的更多信息，请参阅映射数据字段。
17. 选择下一步。
18. （可选）在 Amazon 中为数据字段配置筛选条件 AppFlow。
19. 选择下一步。
20. 检查设置，然后选择创建流。

配置流程后，Amazon AppFlow 会创建一个新的合作伙伴事件源，然后您需要将其与账户中的合作伙伴事件总线相关联。

步骤 2：配置 EventBridge 为接收Salesforce事件

在按照本节的说明进行操作之前，请确保已配置从以目标 EventBridge 为目标Salesforce的事件触发的 Amazon AppFlow 流程。

配置 EventBridge 为接收Salesforce事件

1. 在 EventBridge 控制台中打开合作伙伴事件源页面。
2. 选择您在步骤 1 中创建的 Salesforce 合作伙伴事件源。
3. 选择与事件总线关联。
4. 验证合作伙伴事件总线的名称。
5. 选择关联。
6. 在 Amazon AppFlow 控制台中，打开您创建的流程，然后选择激活流程。
7. 在 EventBridge 控制台中打开 “规则” 页面。
8. 选择创建规则。
9. 为规则输入唯一名称。
10. 在定义模式部分，选择事件模式。
11. 在事件匹配模式下，选择服务提供的预定义模式。
12. 在服务提供商部分，选择所有事件。
13. 在选择事件总线中，选择自定义或合作伙伴事件总线。
14. 选择您与 Amazon AppFlow 合作伙伴事件源关联的事件总线。
15. 在 “选择目标” 中，选择规则运行时要执行的 AWS 服务。一个规则最多可以有五个目标。
16. 选择创建。

目标服务会接收为您的账户配置的所有 Salesforce 事件。要筛选事件或将某些事件发送到不同的目标，您可以使用事件模式中基于内容的筛选。

> **Note**
>
> 对于大于 256KB 的事件，Amazon AppFlow 不会将完整事件发送至。 EventBridge相反，Amazon AppFlow 会将事件放入您账户的 S3 存储桶中，然后向发送一个 EventBridge 带有指向 Amazon S3 存储桶指针的事件。您可以使用此指针从桶中获取完整事件。

调试 Amazon EventBridge 事件传送

事件交付问题可能很难识别，EventBridge 提供了几种调试和从事件交付失败中恢复的方法。

主题

事件重试策略和使用死信队列

有时，事件无法成功传送到规则中指定的目标。例如，当目标资源不可用、 EventBridge 缺乏对目标资源的权限或由于网络状况而导致时，就会发生这种情况。如果由于可重试的错误而无法成功将事件传送到目标，则 EventBridge 会重试发送该事件。您可以在目标的重试策略设置中设置尝试的时间长度和重试次数。默认情况下， EventBridge 会重试发送事件 24 小时，最多 185 次，并出现指数级退缩和抖动或随机延迟。如果在用尽所有重试尝试后仍未传送事件，则该事件将被丢弃，并且 EventBridge不会继续处理该事件。为避免在事件未能传送到目标后丢失事件，您可以配置死信队列 (DLQ)，并将所有失败事件发送到该队列，以便日后处理。

EventBridge DLQ 是标准的 Amazon SQS 队列 EventBridge ，用于存储无法成功传送到目标的事件。创建规则并添加目标时，可以选择是否使用 DLQ。配置 DLQ 时，可以保留任何未成功传送的事件。然后，您可以解决导致事件传送失败的问题，并在之后处理事件。

事件错误的处理方式不同。有些事件会在不进行任何重试的情况下被丢弃或发送到 DLQ。例如，由于缺少对目标的权限或目标资源已不存在而导致的错误，在采取行动解决潜在问题之前，所有重试尝试都将失败。与其重试，不如 EventBridge 将这些事件直接发送到 DLQ（如果有的话）。

当事件交付失败时，会向 Amazon CloudWatch 指标 EventBridge 发布一个表明目标`invocation`失败的事件。如果您使用 DLQ，则会将其他指标发送到 CloudWatch包括`InvocationsSentToDLQ`和。`InvocationsFailedToBeSentToDLQ`

您的 DLQ 中的每条消息都将包含以下自定义属性：

- `RULE_ARN`
- `TARGET_ARN`
- `ERROR_CODE`

  以下是 DLQ 可能返回的错误代码示例：

  - `CONNECTION_FAILURE`
  - `CROSS_ACCOUNT_INGESTION_FAILED`
  - `CROSS_REGION_INGESTION_FAILED`
  - `ERROR_FROM_TARGET`
  - `EVENTS_IN_BATCH_REQUEST_REJECTED`
  - `EVENTS_IN_BATCH_REQUEST_REJECTED`
  - `FAILED_TO_ASSUME_ROLE`
  - `INTERNAL_ERROR`

  - INVALID_JSON
  - INVALID_PARAMETER
  - NO_PERMISSIONS
  - NO_RESOURCE
  - RESOURCE_ALREADY_EXISTS
  - RESOURCE_LIMIT_EXCEEDED
  - RESOURCE_MODIFICATION_COLLISION
  - SDK_CLIENT_ERROR
  - THIRD_ACCOUNT_HOP_DETECTED
  - THIRD_REGION_HOP_DETECTED
  - THROTTLING
  - TIMEOUT
  - TRANSIENT_ASSUME_ROLE
  - UNKNOWN
- ERROR_MESSAGE
- EXHAUSTED_RETRY_CONDITION

  可能返回以下条件：

  - MaximumRetryAttempts
  - MaximumEventAgeInSeconds
- RETRY_ATTEMPTS

以下视频介绍了 DLQ 的设置：使用死信队列 (DLQ)

主题

- 使用死信队列的注意事项
- 为死信队列授予权限
- 如何从死信队列中重新发送事件

使用死信队列的注意事项

为配置 DLQ 时，请考虑以下几点。 EventBridge

- 仅支持标准队列。你不能在中使用 FIFO 队列来获取 DLQ。 EventBridge
- EventBridge 消息中包含事件元数据和消息属性，包括：错误代码、错误消息、用尽重试条件、规则 ARN、重试尝试和目标 ARN。您可以使用这些值来识别事件和失败原因。
- 同一账户中 DLQ 的权限：
  - 如果您使用控制台向规则添加目标，并在同一个账户中选择一个 Amazon SQS 队列，则会将授予您 EventBridge 访问该队列的基于资源的策略附加到该队列中。
  - 如果您使用 EventBridge API 的`PutTargets`操作为规则添加或更新目标，并且在同一账户中选择了 Amazon SQS 队列，则必须手动向所选队列授予权限。要了解更多信息，请参阅为死信队列授予权限。
- 使用不同 AWS 账户的 Amazon SQS 队列的权限。
  - 如果您通过控制台创建规则，则不会显示其他账户的队列供您选择。您必须提供其他账户中队列的 ARN，然后手动附加基于资源的策略，为该队列授予权限。要了解更多信息，请参阅为死信队列授予权限。
  - 如果您使用 API 创建规则，则必须手动将基于资源的策略附加到用作死信队列的另一账户中的 SQS 队列。要了解更多信息，请参阅为死信队列授予权限。
- 您使用的 Amazon SQS 队列必须位于创建规则的区域。

为死信队列授予权限

当您为规则的目标配置 DLQ 时， EventBridge 会将调用失败的事件发送到选定的 Amazon SQS 队列。要成功将事件传送到队列， EventBridge 必须具有执行此操作的权限。当您使用 EventBridge 控制台为规则配置目标并选择 DLQ 时，权限会自动添加。如果您使用 API 创建规则，或者使用其他 AWS 账户中的队列，则必须手动创建授予所需权限的基于资源的策略，然后将其附加到队列。

以下基于资源的策略演示了如何授予向 Amazon SQS 队列发送事件消息所需的权限。 EventBridge 策略示例向 EventBridge 服务授予使用该`SendMessage`操作向名为 “MyEventDLQ” 的队列发送消息的权限。队列必须位于 us-west-2 区域，账户为 123456789012。 AWS 该`Condition`声明仅允许来自名为 “MyTestRule” 的规则的请求，该规则是在us-west-2区域创建的，账户为123456789012。 AWS

```
{
  "Sid": "Dead-letter queue permissions",
  "Effect": "Allow",
```

```
  "Principal": {
     "Service": "events.amazonaws.com"
  },
  "Action": "sqs:SendMessage",
  "Resource": "arn:aws:sqs:us-west-2:123456789012:MyEventDLQ",
  "Condition": {
    "ArnEquals": {
      "aws:SourceArn": "arn:aws:events:us-west-2:123456789012:rule/MyTestRule"
    }
  }
}
```

要将策略附加到队列，请使用 Amazon SQS 控制台，打开队列，然后选择访问策略并编辑该策略。您也可以使用 AWS CLI，了解更多信息，请参阅Amazon SQS 权限。

如何从死信队列中重新发送事件

可以通过两种方式将消息移出 DLQ：

- 避免编写 Amazon SQS 使用者逻辑 - 将 DLQ 设置为 Lambda 函数的事件源以耗尽 DLQ。
- 编写 Amazon SQS 使用者逻辑 — 使用 Amazon SQS API AWS 、SDK AWS CLI 或编写自定义使用者逻辑，用于轮询、处理和删除 DLQ 中的消息。

亚马逊 EventBridge 事件模式

事件模式与它们匹配的事件具有相同的结构。规则使用事件模式来选择事件，并将事件发送到目标。事件模式匹配或不匹配事件。

> **Important**
>
> 在中 EventBridge，可以创建可能导致 higher-than-expected 收费和限制的规则。例如，您可能无意中创建了一条规则，导致无限循环，规则会以递归方式触发，不会结束。假设您创建了一条规则，来检测 S3 桶中的 ACL 更改，然后触发软件以将 ACL 更改为所需状态。如果编写该规则时不小心，则 ACL 的后续更改将再次触发该规则，从而产生无限循环。
> 有关如何编写精确的规则和事件模式，以最大限度地减少此类意外结果的指导，请参阅???和???。

以下视频介绍了事件模式的基础知识：如何筛选事件

主题

- 创建事件模式
- 事件和事件模式示例
- 在 Amazon EventBridge 事件模式中匹配空值和空字符串
- Amazon EventBridge 事件模式中的数组
- 在 Amazon EventBridge 事件模式中筛选内容
- 使用 EventBridge 沙盒测试事件模式
- 定义 Amazon EventBridge 事件模式时的最佳实践

以下事件显示了一个来自 Amazon EC2 的简单 AWS 事件。

```
{
  "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"
  }
}
```

以下事件模式可处理所有 Amazon EC2 `instance-termination` 事件。

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

创建事件模式

要创建事件模式，您要指定希望事件模式匹配的事件字段。仅指定用于匹配的字段。前面的事件模式示例仅提供三个字段的值：顶级字段`"source"`和`"detail-type"`，以及`"detail"`对象`"state"`字段内的字段。 EventBridge 应用规则时会忽略事件中的所有其他字段。

要使事件模式匹配事件，事件必须包含模式中列出的所有字段名。字段名必须还显示在具有相同嵌套结构的事件中。

在编写事件模式来匹配事件时，您可以使用 `TestEventPattern` API 或 `test-event-pattern` CLI 命令测试模式是否匹配正确的事件。有关更多信息，请参阅TestEventPattern。

匹配事件值

在事件模式中，要匹配的值在 JSON 数组中，用方括号（“[”、“]”）括起，因此您可以提供多个值。例如，要匹配来自 Amazon EC2 或的事件 AWS Fargate，您可以使用以下模式，该模式匹配`"source"`字段值为`"aws.ec2"`或的事件`"aws.fargate"`。

```
{
```

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

创建事件模式时的注意事项

构造事件模式时需要考虑以下事项：

- EventBridge 忽略事件中未包含在事件模式中的字段。结果是，事件模式中没有出现的字段用 `"*": "*"` 通配符表示。
- 事件模式匹配的值遵循 JSON 规则。可以包括用引号 (") 括起来的字符串、数字和关键字 `true`、`false` 和 `null`。
- 对于字符串， EventBridge 使用精确 character-by-character 匹配而不进行大小写折叠或任何其他字符串规范化。
- 对于数字， EventBridge 使用字符串表示形式。例如，300、300.0 和 3.0e2 不相等。
- 如果为同一 JSON 字段指定了多个模式，则 EventBridge 只使用最后一个模式。
- 请注意，在 EventBridge 编译事件模式以供使用时，它使用点 (.) 作为连接字符。

  这意味着 EventBridge 将以下事件模式视为相同：

  ```
  ## has no dots in keys
  { "detail" : { "state": { "status": [ "running" ] } } }

  ## has dots in keys
  { "detail" : { "state.status": [ "running" ] } }
  ```

  这两个事件模式都将匹配以下两个事件：

  ```
  ## has no dots in keys
  { "detail" : { "state": { "status": "running" } } }

  ## has dots in keys
  { "detail" : { "state.status": "running"  } }
  ```

  > **Note**
  >
  > 这描述了当前的 EventBridge 行为，不应依赖它来保持不变。

- 包含重复字段的事件模式无效。如果模式包含重复字段，则 EventBridge 仅考虑最终字段值。

例如，以下事件模式将匹配同一个事件：

```
## has duplicate keys
{
  "source": ["aws.s3"],
  "source": ["aws.sns"],
  "detail-type": ["AWS API Call via CloudTrail"],
  "detail":  {
      "eventSource": ["s3.amazonaws.com"],
      "eventSource": ["sns.amazonaws.com"]
  }
}

## has unique keys
{
  "source": ["aws.sns"],
  "detail-type": ["AWS API Call via CloudTrail"],
  "detail": { "eventSource": ["sns.amazonaws.com"] }
}
```

并将 EventBridge 以下两个事件视为相同：

```
## has duplicate keys
{
  "source": ["aws.s3"],
  "source": ["aws.sns"],
  "detail-type": ["AWS API Call via CloudTrail"],
  "detail":  [
    {
      "eventSource": ["s3.amazonaws.com"],
      "eventSource": ["sns.amazonaws.com"]
    }
  ]
}

## has unique keys
{
  "source": ["aws.sns"],
  "detail-type": ["AWS API Call via CloudTrail"],
  "detail": [
    { "eventSource": ["sns.amazonaws.com"] }
  ]
```

```
}
```

> **Note**
>
> 这描述了当前的 EventBridge 行为，不应依赖它来保持不变。

事件模式中使用的比较操作

以下是中所有可用的比较运算符的摘要 EventBridge。

比较运算符仅适用于叶节点，`$or` 和 `anything-but` 除外。

Comparison（比较）	示例	Rule syntax（规则语法）
And	位置为“纽约”，日期为“星期一”	`"Location": [ "New York" ], "Day": ["Monday"]`
除此之外什么都行	状态是除了 “初始化” 之外的任何值。	`"state": [ { "anything-but": "initializing" } ]`
除此之外的任何东西（开头为）	地区不在美国。	`"Region": [ { "anything-but": {"prefix": "us-" } } ]`
除此之外的任何东西（结尾为）	FileName 不以.png 扩展名结尾。	`"FileName": [ { "anything-but": { "suffix": ".png" } } ]`
除此之外的任何东西（忽略大小写）	状态是除了 “初始化” 或任何其他大小写变体（例如 “初始化”）之外的任何值。	`"state": : [{ "anything-but": { "equals-ignore-case": "initializing" }}]}`
开头	地区在美国。	`"Region": [ {"prefix": "us-" } ]`

Comparison（比较）	示例	Rule syntax（规则语法）
开头为（忽略大小写）	无论大小写如何，服务名称都以字母“eventb”开头。	`{"service" : [{ "prefix": { "equals-ignore-case": "eventb" }}]}`
空	LastName 是空的。	`"LastName": [""]`
等于	名字为“Alice”	`"Name": [ "Alice" ]`
等于（忽略大小写）	名字为“Alice”	`"Name": [ { "equals-ignore-case": "alice" } ]`
结束	FileName 以.png 扩展名结尾	`"FileName": [ { "suffix": ".png" } ]`
结尾为（忽略大小写）	服务名称以字母 “tbridge” 或任何其他大小写变体（例如 “TBRIDGE”）结尾。	`{"service" : [{ "suffix": { "equals-ignore-case": "tBridge" }}]}`
存在	ProductName 存在	`"ProductName": [ { "exists": true } ]`
不存在	ProductName 不存在	`"ProductName": [ { "exists": false } ]`
非	天气是除“下雨”以外的任何天气	`"Weather": [ { "anything-but": [ "Raining" ] } ]`
Null	用户 ID 为空	`"UserID": [ null ]`
数值（等于）	价格为 100	`"Price": [ { "numeric": [ "=", 100 ] } ]`
数值（范围）	价格大于 10，且小于等于 20	`"Price": [ { "numeric": [ ">", 10, "<=", 20 ] } ]`
Or	PaymentType 是 “贷方” 或 “借记卡”	`"PaymentType": [ "Credit", "Debit"]`

Comparison（比较）	示例	Rule syntax（规则语法）
或（多个字段）	位置为“纽约”，或日期为“星期一”。	`"$or": [ { "Location": [ "New York" ] }, { "Day": [ "Monday" ] } ]`
通配符	位于“dir”文件夹中的任何扩展名为 .png 的文件	`"FileName": [ { "wildcard": "dir/*.png" } ]`

事件和事件模式示例

您可以使用所有 JSON 数据类型和值来匹配事件。以下示例展示事件和与之匹配的事件模式。

字段匹配

您可以匹配字段的值。请考虑以下 Amazon EC2 Auto Scaling 事件。

```
{
  "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
  }
}
```

对于以上事件，您可以使用 `"responseElements"` 字段进行匹配。

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

```
  }
}
```

值匹配

请考虑以下 Amazon Macie 事件，该事件已被截断。

```
{
  "version": "0",
  "id": "0948ba87-d3b8-c6d4-f2da-732a1example",
  "detail-type": "Macie Finding",
  "source": "aws.macie",
  "account": "123456789012",
  "time": "2021-04-29T23:12:15Z",
  "region":"us-east-1",
  "resources": [

  ],
  "detail": {
    "schemaVersion": "1.0",
    "id": "64b917aa-3843-014c-91d8-937ffexample",
    "accountId": "123456789012",
    "partition": "aws",
    "region": "us-east-1",
    "type": "Policy:IAMUser/S3BucketEncryptionDisabled",
    "title": "Encryption is disabled for the S3 bucket",
    "description": "Encryption is disabled for the Amazon S3 bucket. The data in the
 bucket isn’t encrypted
        using server-side encryption.",
    "severity": {
        "score": 1,
        "description": "Low"
    },
    "createdAt": "2021-04-29T15:46:02Z",
    "updatedAt": "2021-04-29T23:12:15Z",
    "count": 2,
.
.
.
```

以下事件模式匹配严重性分数为 1 且计数为 2 的任何事件。

```
{
```

```
  "source": ["aws.macie"],
  "detail-type": ["Macie Finding"],
  "detail": {
    "severity": {
      "score": [1]
    },
    "count":[2]
  }
}
```

在 Amazon EventBridge 事件模式中匹配空值和空字符串

> **Important**
>
> 在中 EventBridge，可以创建可能导致 higher-than-expected 收费和限制的规则。例如，您可能无意中创建了一条规则，导致无限循环，规则会以递归方式触发，不会结束。假设您创建了一条规则，来检测 S3 桶中的 ACL 更改，然后触发软件以将 ACL 更改为所需状态。如果编写该规则时不小心，则 ACL 的后续更改将再次触发该规则，从而产生无限循环。
> 有关如何编写精确的规则和事件模式，以最大限度地减少此类意外结果的指导，请参阅???和???。

您可以创建一种事件模式，与事件中具有 Null 值或空字符串的字段匹配。考虑以下示例 事件。

查看最佳实践，以避免超出预期的费用和节流

```
{
  "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
   }
}
```

要匹配 `eventVersion` 值为空字符串的事件，请使用以下事件模式，它可匹配上一事件。

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

要匹配 `responseElements` 值为 Null 的事件，请使用以下事件模式，它可匹配上一事件。

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

> **Note**
>
> 在模式匹配中，Null 值和空字符串是不可互换的。匹配空字符串的事件模式不匹配 null 值。

Amazon EventBridge 事件模式中的数组

事件模式中每个字段的值都是包含一个或多个值的数组。如果数组中的任何值与事件中的值相匹配，则事件模式与该事件匹配。如果事件中的值为数组，则在事件模式数组与事件数组的交集不为空时，事件模式匹配。

> **Important**
>
> 在中 EventBridge，可以创建可能导致 higher-than-expected 收费和限制的规则。例如，您可能无意中创建了一条规则，导致无限循环，规则会以递归方式触发，不会结束。假设您创建了一条规则，来检测 S3 桶中的 ACL 更改，然后触发软件以将 ACL 更改为所需状态。如果编写该规则时不小心，则 ACL 的后续更改将再次触发该规则，从而产生无限循环。
> 有关如何编写精确的规则和事件模式，以最大限度地减少此类意外结果的指导，请参阅???和???。

例如，考虑包含以下字段的事件模式。

```
"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",
]
```

上一事件模式与包括以下字段的事件相匹配，因为事件模式数组中的第一项与事件数组中的第二项匹配。

```
"resources": [
   "arn:aws:autoscaling:us-east-1:123456789012:autoScalingGroup:eb56d16b-bbf0-401d-
b893-d5978ed4a025:autoScalingGroupName/ASGTerminate",
   "arn:aws:ec2:us-east-1:123456789012:instance/i-b188560f"
]
```

在 Amazon EventBridge 事件模式中筛选内容

Amazon EventBridge 支持使用事件模式进行声明式内容筛选。通过内容筛选，您可以编写仅在非常具体的条件下匹配事件的复杂事件模式。例如，您可以创建事件模式，在以下情况下与事件匹配：

- 事件的某个字段在特定的数值范围内。
- 事件来自特定 IP 地址。
- 事件 JSON 中不存在特定字段。

> **Important**
>
> 在中 EventBridge，可以创建可能导致 higher-than-expected 收费和限制的规则。例如，您可能无意中创建了一条规则，导致无限循环，规则会以递归方式触发，不会结束。假设您创建了一条规则，来检测 S3 桶中的 ACL 更改，然后触发软件以将 ACL 更改为所需状态。如果编写该规则时不小心，则 ACL 的后续更改将再次触发该规则，从而产生无限循环。
> 有关如何编写精确的规则和事件模式，以最大限度地减少此类意外结果的指导，请参阅???和???。

筛选类型

- 前缀匹配
- 后缀匹配
- Anything-but 匹配
- 数值匹配
- IP 地址匹配
- Exists 匹配
- E quals-ignore-case 匹配
- 使用通配符进行匹配
- 具有多个匹配的复杂示例
- 具有 $or 匹配的复杂示例

前缀匹配

您可以根据事件源中值的前缀匹配事件。您可以对字符串值使用前缀匹配。

例如，以下事件模式将匹配 "time" 字段以 "2017-10-02" 开头的任何事件，例如 "time": "2017-10-02T18:43:48Z"。

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

忽略大小写时进行前缀匹配

无论值开头的字符大小写如何，您也可以将前缀值与结合使用，来equals-ignore-case匹配前缀值prefix.

例如，以下事件模式将匹配service字段以字符串开头的任何事件EventB，以及EVENTBeventb、或这些字符的任何其他大小写。

```
{
  "detail": {"service" : [{ "prefix": { "equals-ignore-case": "EventB" }}]}
}
```

后缀匹配

您可以根据事件源中值的后缀匹配事件。您可以对字符串值使用后缀匹配。

例如，以下事件模式将匹配 "FileName" 字段以 .png 文件扩展名结尾的任何事件。

```
{
  "FileName": [ { "suffix": ".png" } ]
}
```

忽略大小写时进行后缀匹配

无论值结尾的字符大小写如何，您也可以匹配后缀值，同时使用equals-ignore-case与 suffix.

例如，以下事件模式将匹配FileName字段以字符串结尾的任何事件.png，但也匹配这些字符.PNG的任何其他大小写。

```
{
  "detail": {"FileName" : [{ "suffix": { "equals-ignore-case": ".png" }}]}
}
```

Anything-but 匹配

Anything-but 匹配可匹配除规则中提供的内容之外的任何内容。

您可以将 Anything-but 匹配与字符串和数值一起使用，包括仅包含字符串或仅包含数值的列表。

以下事件模式显示了 Anything-but 匹配与字符串和数字。

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

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

以下事件模式显示了 Anything-but 匹配与一组字符串。

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

以下事件模式显示了 Anything-but 匹配与一组数字。

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

除了在忽略大小写的情况下匹配之外什么都没有

也可以与结合使用 `equals-ignore-case`anything-but，以匹配字符串值，而不考虑字符大小写。

以下事件模式匹配包含字符串 “初始化”、“初始化”、“初始化” 或这些字符的任何其他大写字母的state字段。

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

您也可以equals-ignore-case与结合使用anything-but来匹配值列表：

```
{
  "detail": {"state" : [{ "anything-but": { "equals-ignore-case": ["initializing",
 "stopped"] }}]}
}
```

除了前缀匹配之外什么都没有

以下事件模式显示了与 "state" 字段中没有 "init" 前缀的任何事件相匹配的 Anything-but 匹配。

> **Note**
>
> Anything-but 匹配只能处理单个前缀，不能处理一组前缀。

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

除了后缀匹配之外什么都没有

无论字符大小写如何，都可以与结合使用suffixanything-but来匹配字符串值。

> **Note**
>
> 除了匹配之@@ 外的任何东西都只能使用单个后缀，而不是列表。

以下事件模式匹配以结尾的FileName字段的任何值.txt。

```
{
  "detail": {
    "FileName": [ { "anything-but": { "suffix": ".txt" } } ]
  }
}
```

数值匹配

数值匹配适用于 JSON 数字值。仅限于 -5.0e9 和 +5.0e9（含）之间的值，精度为 15 位（小数点右侧为六位）。

以下代码显示的事件模式的数值匹配，仅匹配所有字段均为真的事件。

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

IP 地址匹配

可以将 IP 地址匹配用于 IPv4 和 IPv6 地址。以下事件模式显示的 IP 地址匹配，匹配以 10.0.0 开头并以 0 到 255 之间的数字结尾的 IP 地址。

```
{
  "detail": {
    "sourceIPAddress": [ { "cidr": "10.0.0.0/24" } ]
  }
}
```

Exists 匹配

Exists 匹配 用于确定事件的 JSON 中存在或不存在某个字段。

Exists 匹配仅适用于叶节点。它对于中间节点不起作用。

以下事件模式与任何具有 `detail.state` 字段的事件相匹配。

```
{
```

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

上一事件模式与以下事件匹配。

```
{
  "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-abcd1111",
    "state": "pending"
  }
}
```

上一事件模式与以下事件不匹配，因为它没有 `detail.state` 字段。

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

E quals-ignore-case 匹配

无论大小写如何，E quals-ignore-case 匹配都适用于字符串值。

以下事件模式匹配 `detail-type` 字段与指定字符串相匹配（无论大小写如何）的任何事件。

```
{
```

```
  "detail-type": [ { "equals-ignore-case": "ec2 instance state-change notification" } ]
}
```

上一事件模式与以下事件匹配。

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

使用通配符进行匹配

您可以使用通配符 (*) 匹配事件模式中的字符串值。

> **Note**
>
> 目前，只有事件总线规则支持通配符。

在事件模式中使用通配符的注意事项：

- 可以在给定的字符串值中指定任意数量的通配符；但是，不支持连续通配符。
- EventBridge 支持使用反斜杠字符 (\) 来指定通配符过滤器中的文字 * 和\ 字符：
  - 字符串 * 代表文字 * 字符
  - 字符串 \\ 代表文字 \ 字符

  不支持使用反斜杠对其他字符进行转义。

通配符和事件模式的复杂性

规则使用通配符的复杂程度是有限的。如果规则过于复杂，则`InvalidEventPatternException`在尝试创建规则时 EventBridge 返回。如果您的规则生成此类错误，请考虑使用以下指南来降低事件模式的复杂性：

- 减少使用的通配符数量

  仅在确实需要匹配多个可能值的位置使用通配符。例如，考虑以下事件模式，您想匹配同一区域中的事件总线：

  ```
  {
  "EventBusArn": [ { "wildcard": "*:*:*:*:*:event-bus/*" } ]
  }
  ```

  在上述情况下，ARN 的许多部分将直接取决于您的事件总线所在的区域。因此，如果您使用 us-east-1 区域，则以下示例可能是不太复杂，但仍可与所需值匹配的模式：

  ```
  {
  "EventBusArn": [ { "wildcard": "arn:aws:events:us-east-1:*:event-bus/*" } ]
  }
  ```

- 减少通配符后出现的重复字符序列

  使用通配符后多次出现相同的字符序列，会增加处理事件模式的复杂性。请修改您的事件模式，最大限度地减少重复序列。例如，考虑以下示例，该示例与任何用户的文件名为 doc.txt 的文件相匹配：

  ```
  {
  "FileName": [ { "wildcard": "/Users/*/dir/dir/dir/dir/dir/doc.txt" } ]
  }
  ```

  如果您知道 doc.txt 文件只会出现在指定路径中，则可以通过以下方式减少重复的字符序列：

  ```
  {
  "FileName": [ { "wildcard": "/Users/*/doc.txt" } ]
  }
  ```

具有多个匹配的复杂示例

您可以将多个匹配规则组合为更复杂的事件模式。例如，以下事件模式组合了 anything-but 和 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**
>
> 在构建事件模式时，如果您多次包含一个密钥，则最后一次引用将用于评估事件。例如，对于以下模式：
>
> ```
> {
>   "detail": {
>     "location": [ { "prefix": "us-" } ],
>     "location": [ { "anything-but": "us-east" } ]
>   }
> }
> ```
>
> 只有在评估 location 时才会考虑 { "anything-but": "us-east" }。

具有 $or 匹配的复杂示例

您还可以创建复杂的事件模式，检查多个字段中是否有任何 字段值匹配。$or 用于创建事件模式，匹配多个字段中有任何值匹配的情况。

请注意，在 $or 构造中，您可以在各个字段的模式匹配中包含其他筛选器类型，例如数值匹配和数组。

满足以下任意条件时，匹配以下事件模式：

- c-count 字段大于 0 或小于等于 5。
- d-count 字段小于 10。
- x-limit 字段等于 3.018e2。

```
{
```

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

> **Note**
>
> 如果使用 $or 会导致超过 1000 个规则组合，接受事件模式（例如 PutRule、CreateArchive、UpdateArchive 和 TestEventPattern）的 API 会引发 InvalidEventPatternException。
> 要确定事件模式中规则组合的数量，请将事件模式中每个 $or 数组的参数总数相乘。例如，上面的模式包含一个 $or 数组，带有三个参数，因此规则组合的总数也是三个。如果您再添加一个包含两个参数的 $or 数组，则规则组合总数将为六个。

使用 EventBridge 沙盒测试事件模式

规则使用事件模式来选择事件，并将事件发送到目标。事件模式与它们匹配的事件具有相同的结构。事件模式匹配或不匹配事件。

定义事件模式通常是创建新规则或编辑现有规则这一大型过程中的一环。但是 EventBridge，使用中的沙盒，您可以快速定义事件模式并使用示例事件来确认模式与所需事件相匹配，而无需创建或编辑规则。测试完事件模式后， EventBridge 可以选择直接从沙箱中使用该事件模式创建新规则。

有关事件模式的更多信息，请参阅 ???。

> **Important**
>
> 在中 EventBridge，可以创建可能导致 higher-than-expected 收费和限制的规则。例如，您可能无意中创建了一条规则，导致无限循环，规则会以递归方式触发，不会结束。假设您创建了一条规则，来检测 S3 桶中的 ACL 更改，然后触发软件以将 ACL 更改为所需状态。如果编写该规则时不小心，则 ACL 的后续更改将再次触发该规则，从而产生无限循环。
> 有关如何编写精确的规则和事件模式，以最大限度地减少此类意外结果的指导，请参阅???和???。

使用 EventBridge 沙盒测试事件模式

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中选择开发人员资源，然后选择沙盒，再在沙盒页面上选择事件模式选项卡。
3. 对于事件来源，选择AWS 事件或 EventBridge合作伙伴事件。
4. 在示例事件部分，选择要测试事件模式的示例事件类型。

   提供以下示例事件类型：

   - AWS 事件-从支持 AWS 服务的事件中进行选择。
   - EventBridge 合作伙伴活动-从支持的 EventBridge第三方服务（例如 Salesforce）发出的事件中进行选择。
   - 输入我自己的 - 以 JSON 文本输入您自己的事件。

     您也可以使用 AWS 或合作伙伴事件作为创建自己的自定义事件的起点。

     1. 选择AWS 活动或EventBridge 合作伙伴活动。
     2. 使用示例事件下拉列表，选择要用作自定义事件起点的事件。

        EventBridge 显示示例事件。
     3. 选择 复制。
     4. 选择输入我自己的作为事件类型。
     5. 在 JSON 编辑窗格中删除示例事件结构，然后将 AWS 或合作伙伴事件粘贴到原处。
     6. 编辑事件 JSON，创建您自己的示例事件。
5. 选择创建方法。您可以根据 EventBridge 架构或模板创建事件模式，也可以创建自定义事件模式。

   Existing schema

   要使用现有 EventBridge 架构创建事件模式，请执行以下操作：

   1. 在创建方法部分的方法中，选择使用架构。
   2. 在事件模式部分的架构类型中，选择从架构注册表中选择架构。
   3. 对于架构注册表，选择下拉框并输入架构注册表的名称，例如 `aws.events`。您还可以从出现的下拉列表中选择一个选项。
   4. 对于架构，选择下拉框并输入要使用的架构的名称。例如，`aws.s3@ObjectDeleted`。您还可以从出现的下拉列表中选择一个选项。

5. 在模型部分，选择任意属性旁边的编辑按钮，可打开其属性。根据需要设置关系和值字段，然后选择设置保存属性。

> **Note**
>
> 有关属性定义的信息，请选择属性名称旁边的信息图标。有关如何在事件中设置属性的参考，请打开属性对话框的注释部分。
> 要删除某一属性的特性，请选择该属性的编辑按钮，然后选择清除。

6. 选择以 JSON 格式生成事件模式，作为 JSON 文本生成并验证您的事件模式。
7. 要根据您的事件模式测试示例事件，请选择测试模式。

   EventBridge 显示一个消息框，说明您的示例事件是否与事件模式匹配。

   您还可以选择以下任何选项：

   - 复制 - 将事件模式复制到设备的剪贴板。
   - 修饰 - 添加换行符、制表符和空格，使 JSON 文本更易于阅读。

Custom schema

要编写自定义架构并将其转换为事件模式，请执行以下操作：

1. 在创建方法部分的方法中，选择使用架构。
2. 在事件模式部分的架构类型中，选择输入架构。
3. 在文本框中输入您的架构。您必须将架构格式化为有效的 JSON 文本。
4. 在模型部分，选择任意属性旁边的编辑按钮，可打开其属性。根据需要设置关系和值字段，然后选择设置保存属性。

> **Note**
>
> 有关属性定义的信息，请选择属性名称旁边的信息图标。有关如何在事件中设置属性的参考，请打开属性对话框的注释部分。
> 要删除某一属性的特性，请选择该属性的编辑按钮，然后选择清除。

5. 选择以 JSON 格式生成事件模式，作为 JSON 文本生成并验证您的事件模式。
6. 要根据您的事件模式测试示例事件，请选择测试模式。

EventBridge 显示一个消息框，说明您的示例事件是否与事件模式匹配。

您还可以选择以下任何选项：

- 复制 - 将事件模式复制到设备的剪贴板。
- 修饰 - 添加换行符、制表符和空格，使 JSON 文本更易于阅读。

Event pattern

要以 JSON 格式编写自定义事件模式，请执行以下操作：

1. 在创建方法部分的方法中，选择自定义模式（JSON 编辑器）。
2. 在事件模式中，以 JSON 格式的文本输入您的自定义事件模式。
3. 要根据您的事件模式测试示例事件，请选择测试模式。

   EventBridge 显示一个消息框，说明您的示例事件是否与事件模式匹配。

   您还可以选择以下任何选项：

   - 复制 - 将事件模式复制到设备的剪贴板。
   - 修饰 - 添加换行符、制表符和空格，使 JSON 文本更易于阅读。
   - 事件模式表单 - 在模式生成器中打开事件模式。如果无法在模式生成器中按原样渲染图案，则会在模式生成器打开模式生成器之前 EventBridge 发出警告。
6. （可选）要使用此事件模式创建规则，并将该规则分配给特定的事件总线，请选择使用模式创建规则。

   EventBridge 将带您进入创建规则的步骤 1，您可以使用它来创建规则并将其分配给您选择的事件总线。

   请注意，步骤 2 - 生成事件模式包含您已经指定的事件模式信息，您可以接受或更新这些信息。

   有关如何创建规则的更多信息，请参阅???。

定义 Amazon EventBridge 事件模式时的最佳实践

以下是在事件总线规则中定义事件模式时需要考虑的一些最佳实践。

避免编写无限循环

在中 EventBridge，可以创建导致无限循环的规则，即重复触发规则。例如，某规则可能检测到 S3 存储桶上的 ACL 已更改，然后触发软件以将 ACL 更改为所需状态。如果编写该规则时不小心，则 ACL 的后续更改将再次触发该规则，从而产生无限循环。

为了防止出现这些问题，请为规则编写尽可能精确的事件模式，这样它们就只会匹配您实际希望发送到目标的事件。在上面的示例中，您将创建一个事件模式来匹配事件，使触发的操作就不会重新触发相同的规则。例如，在规则中创建一个事件模式，仅在发现 ACL 处于不良状态时才匹配事件，而不是在任何更改之后都匹配事件。有关更多信息，请参阅 ??? 和 ???。

无限循环可能快速导致费用超出预期。它还可能导致节流和事件传送延迟。您可以监控调用速率的上限，在数量出现意外激增时收到警告。

使用预算，在费用超出指定限额时提醒您。有关更多信息，请参阅通过预算管理成本。

使事件模式尽可能精确

您的事件模式越精确，就越有可能只匹配您真正想要匹配的事件，并且在向事件源添加新事件，或更新现有事件以包含新属性时，避免意外匹配。

事件模式可以包括筛选器，匹配以下内容：

- 有关该事件的事件元数据，例如 `source`、`detail-type`、`account` 或`region`。
- 事件数据，即 `detail` 对象内的字段。
- 事件内容，或 `detail` 对象内字段的实际值。

大多数模式都很简单，例如仅指定 `source` 和 `detail-type` 筛选器。但是， EventBridge 模式包括可以灵活地根据事件的任何键或值进行筛选。此外，您可以应用内容筛选器（例如 `prefix` 和 `suffix` 筛选器）来提高模式的精度。有关更多信息，请参阅 ???。

将事件源和详细信息类型指定为筛选器

使用 `source` 和 `detail-type` 元数据字段使事件模式更加精确，可以减少生成无限循环和匹配不想要的事件的概率。

如果需要匹配两个或更多字段中的特定值，请使用 `$or` 比较运算符，而不是在单个值数组中列出所有可能的值。

对于通过传送的事件 AWS CloudTrail，我们建议您使用该eventName字段作为筛选条件。

以下事件模式示例匹配CreateQueue或SetQueueAttributes来自亚马逊简单队列服务CreateKey或来自该 AWS Key Management Service 服务DisableKeyRotation的事件。

```
{
  "detail-type": ["AWS API Call via CloudTrail"],
  "$or": [{
      "source": [
        "aws.sqs"
        ],
      "detail": {
        "eventName": [
          "CreateQueue",
          "SetQueueAttributes"
        ]
      }
    },
    {
      "source": [
        "aws.kms"
        ],
      "detail": {
        "eventName": [
          "CreateKey",
          "DisableKeyRotation"
        ]
      }
    }
  ]
}
```

指定账户和区域作为筛选条件

在事件模式中包含 account 和 region 字段，有助于限制跨账户或跨区域的事件匹配。

指定内容筛选器

基于内容的筛选有助于提高事件模式的精度，同时仍将事件模式的长度保持最短。例如，与列出所有可能的数值相比，基于数值范围进行匹配可能更加合适。

有关更多信息，请参阅 ???。

确定事件模式的范围，将事件源更新纳入考虑范围

创建事件模式时，应考虑到事件架构和事件域可能会随着时间的推移而演变和扩展。再次重申，尽可能精确地设置事件模式，可以帮助您在事件源发生变化或扩展时限制意外匹配。

例如，假设您要与来自新的微服务的事件进行匹配，该服务发布与支付相关的事件。最初，该服务使用域 `acme.payments`，并发布单个事件 `Payment accepted`：

```
{
  "detail-type": "Payment accepted",
  "source": "acme.payments",
  "detail": {
    "type": "credit",
    "amount": "100",
    "date": "2023-06-10",
    "currency": "USD"
    }
  }
}
```

此时，您可以创建与已接受付款的事件相匹配的简单事件模式：

```
{ “source” : “acme.payments” }
```

但是，假设该服务稍后引入了一个针对拒绝付款的新事件：

```
{
  "detail-type": "Payment rejected",
  "source": "acme.payments",
  "detail": {
  }
}
```

在这种情况下，您创建的简单事件模式现在与 `Payment accepted` 和 `Payment rejected` 事件均可匹配。 EventBridge 将两种类型的事件路由到指定的目标进行处理，这可能会导致处理失败和额外的处理成本。

要将事件模式的范围仅限定于 `Payment accepted` 事件，您需要至少同时指定 `source` 和 `detail-type`：

```
{
```

```
  "detail-type": "Payment accepted",
  "source": "acme.payments"
  }
}
```

您还可以在事件模式中指定账户和区域，进一步限制跨账户或跨区域事件何时匹配此规则。

```
{
  "account": "012345678910",
  "source": "acme.payments",
  "region": "AWS-Region",
  "detail-type": "Payment accepted"
}
```

验证事件模式

为确保规则与所需事件相匹配，我们强烈建议您验证您的事件模式。您可以使用 EventBridge 控制台或 API 验证您的事件模式：

- 在 EventBridge 控制台中，您可以在创建规则的过程中创建和测试事件模式，也可以使用沙盒单独创建和测试事件模式。
- 您可以使用TestEventPattern操作以编程方式测试您的事件模式。

亚马逊 EventBridge 规则

您可以指定 EventBridge 如何处理传送到每个事件总线的事件。为此，您需要创建规则。规则指定要将哪些事件发送到哪些目标进行处理。一条规则可以向多个目标发送事件，然后这些目标会并行运行。

可创建两种类型的规则：

- 与事件数据匹配的规则

  您可以根据事件数据标准（称为事件模式）创建与传入事件匹配的规则。事件模式 定义了事件结构和规则匹配的字段。如果事件符合事件模式中定义的标准，则将其 EventBridge 发送到您指定的目标。

  有关更多信息，请参阅???。
- 按计划运行的规则

  您也可以创建按指定间隔向指定目标发送事件的规则。例如，要定期运行 Lambda 函数，可以创建按计划运行的规则。

> **Note**
>
> EventBridge 提供 Amazon S EventBridge cheduler，这是一款无服务器计划程序，允许您通过一个中央托管服务创建、运行和管理任务。 EventBridge Scheduler 具有高度可定制性，与 EventBridge计划规则相比，它具有更高的可扩展性，具有更广泛的目标 API 操作和 AWS 服务。
> 我们建议您使用 EventBridge 调度器按计划调用目标。有关更多信息，请参阅???。

以下视频介绍了规则的基础知识：什么是规则

亚马逊 EventBridge 托管规则

除了您创建的规则外， AWS 服务还可以在您的 AWS 账户中创建和管理这些服务的某些功能所需的 EventBridge 规则。这些策略称为托管式规则。

当服务创建托管规则时，它还可以创建一个IAM 策略，向该服务授予创建规则的权限。以这种方式创建的 IAM 策略，作用范围仅局限于资源级权限，仅允许创建必需的规则。

您可以使用强制删除选项删除托管规则，但只有在确定其他服务不再需要该规则时，才应将其删除。否则，删除托管式规则会导致依赖它的功能停止工作。

创建对事件作出反应的 Amazon EventBridge 规则

要对 Amazon EventBridge 收到的事件执行操作，您可以创建规则。当事件匹配规则中定义的事件模式时，EventBridge 会将事件发送到指定的目标，并触发规则中定义的操作。

以下视频探讨了如何创建不同类型的规则，以及如何对其进行测试：了解规则。

可以使用以下过程创建 Amazon EventBridge 规则，对事件做出响应。

创建对事件做出反应的规则

以下步骤将引导您完成创建规则的过程，当事件发送到指定事件总线时，EventBridge 使用该规则来匹配事件。

步骤

- 定义规则
- 构建事件模式
- 选择目标
- 配置标签并检查规则

定义规则

首先，为您的规则输入用于标识它的名称和描述。您还必须定义事件总线，您的规则将在其中查找与事件模式匹配的事件。

定义规则细节

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Rules (规则)。
3. 选择 Create rule (创建规则)。
4. 输入规则的名称和可选描述。

   规则不能与同一 AWS 区域中和同一事件总线上的另一条规则的名称相同。
5. 对于事件总线，请选择要与此规则关联的事件总线。如果您希望此规则对来自您自己的账户的匹配事件触发，请选择 AWS 默认事件总线。当您账户中的某个 AWS 服务发出一个事件时，它始终会发送到您账户的默认事件总线。

6. 对于 Rule type（规则类型），选择 Rule with an event pattern（具有事件模式的规则）。
7. 选择 Next（下一步）。

构建事件模式

接下来构建事件模式 方法是指定事件源，选择事件模式的基础，然后定义要匹配的属性和值。您也可以生成 JSON 格式的事件模式，并针对示例事件对其进行测试。

构建事件模式

1. 对于 Event source（事件源），选择 AWS 事件或 EventBridge 合作伙伴事件。
2. （可选）在示例事件部分，选择要测试事件模式的示例事件类型。

   提供以下示例事件类型：

   - AWS 事件 - 从支持的 AWS 服务发出的事件中进行选择。
   - EventBridge 合作伙伴事件 - 从支持 EventBridge 的第三方服务（例如 Salesforce）发出的事件中进行选择。
   - 输入我自己的 - 以 JSON 文本输入您自己的事件。

   您也可以使用 AWS 或合作伙伴事件作为创建自定义事件的起点。

   1. 选择 AWS 事件或 EventBridge 合作伙伴事件。
   2. 使用示例事件下拉列表，选择要用作自定义事件起点的事件。

      EventBridge 显示此示例事件。
   3. 选择复制。
   4. 选择输入我自己的作为事件类型。
   5. 在 JSON 编辑窗格中删除示例事件结构，然后将 AWS 或合作伙伴事件粘贴到此处。
   6. 编辑事件 JSON，创建您自己的示例事件。
3. 选择创建方法。您可以根据 EventBridge 架构或模板创建事件模式，也可以创建自定义事件模式。

   Existing schema

   要使用现有 EventBridge 架构创建事件模式，请执行以下操作：

   1. 在创建方法部分的方法中，选择使用架构。
   2. 在事件模式部分的架构类型中，选择从架构注册表中选择架构。

3. 对于架构注册表，选择下拉框并输入架构注册表的名称，例如 `aws.events`。您还可以从出现的下拉列表中选择一个选项。
4. 对于架构，选择下拉框并输入要使用的架构的名称。例如，`aws.s3@ObjectDeleted`。您还可以从出现的下拉列表中选择一个选项。
5. 在模型部分，选择任意属性旁边的编辑按钮，可打开其属性。根据需要设置关系和值字段，然后选择设置保存属性。

> **Note**
>
> 有关属性定义的信息，请选择属性名称旁边的信息图标。有关如何在事件中设置属性的参考，请打开属性对话框的注释部分。
> 要删除某一属性的特性，请选择该属性的编辑按钮，然后选择清除。

6. 选择以 JSON 格式生成事件模式，作为 JSON 文本生成并验证您的事件模式。
7. （可选）要根据您的事件模式测试示例事件，请选择测试模式。

   EventBridge 会显示一个消息框，说明您的示例事件是否与事件模式匹配。

   您还可以选择以下任何选项：

   - 复制 - 将事件模式复制到设备的剪贴板。
   - 修饰 - 添加换行符、制表符和空格，使 JSON 文本更易于阅读。

Custom schema

要编写自定义架构并将其转换为事件模式，请执行以下操作：

1. 在创建方法部分的方法中，选择使用架构。
2. 在事件模式部分的架构类型中，选择输入架构。
3. 在文本框中输入您的架构。您必须将架构格式化为有效的 JSON 文本。
4. 在模型部分，选择任意属性旁边的编辑按钮，可打开其属性。根据需要设置关系和值字段，然后选择设置保存属性。

> **Note**
>
> 有关属性定义的信息，请选择属性名称旁边的信息图标。有关如何在事件中设置属性的参考，请打开属性对话框的注释部分。

要删除某一属性的特性，请选择该属性的编辑按钮，然后选择清除。

5. 选择以 JSON 格式生成事件模式，作为 JSON 文本生成并验证您的事件模式。
6. （可选）要根据您的事件模式测试示例事件，请选择测试模式。

   EventBridge 会显示一个消息框，说明您的示例事件是否与事件模式匹配。

   您还可以选择以下任何选项：

   - 复制 - 将事件模式复制到设备的剪贴板。
   - 修饰 - 添加换行符、制表符和空格，使 JSON 文本更易于阅读。

Event pattern

要以 JSON 格式编写自定义事件模式，请执行以下操作：

1. 在创建方法部分的方法中，选择自定义模式（JSON 编辑器）。
2. 在事件模式中，以 JSON 格式的文本输入您的自定义事件模式。
3. （可选）要根据您的事件模式测试示例事件，请选择测试模式。

   EventBridge 会显示一个消息框，说明您的示例事件是否与事件模式匹配。

   您还可以选择以下任何选项：

   - 复制 - 将事件模式复制到设备的剪贴板。
   - 修饰 - 添加换行符、制表符和空格，使 JSON 文本更易于阅读。
   - 事件模式表单 - 在模式生成器中打开事件模式。如果此模式无法在模式生成器中按原样渲染，EventBridge 会在打开模式生成器之前向您发出警告。

4. 选择 Next（下一步）。

选择目标

选择一个或多个目标，接收与指定模式匹配的事件。目标可以包括 EventBridge 事件总线、EventBridge API 目标（包括 Salesforce 等 SaaS 合作伙伴或另一 AWS 服务）。

选择目标

1. 对于目标类型，请选择以下目标类型之一：

Event bus

要选择 EventBridge 事件总线，请选择EventBridge 事件总线，然后执行以下操作：

- 使用与此规则位于同一 AWS 区域的事件总线：
  1. 选择同一账户和区域中的事件总线。
  2. 对于目标的事件总线，选择下拉框并输入事件总线的名称。您也可以从下拉列表中选择事件总线。

     有关更多信息，请参阅???。
- 使用与此规则不同的 AWS 区域 或账户中的事件总线：
  1. 选择不同账户或区域中的事件总线。
  2. 对于事件总线作为目标，请输入要使用的事件总线的 ARN。

     有关更多信息，请参阅：
     - ???
     - ???

API destination

要使用 EventBridge API 目标，请选择 EventBridge API 目标，然后执行以下任一操作：

- 要使用现有 API 目标，请选择使用现有 API 目标。然后从下拉列表中选择 API 目标。
- 要创建新的 API 目标，请选择创建新的 API 目标。然后为目标提供以下详细信息：
  - 名称 - 为目标键入一个名称。

    名称在您的 AWS 账户 内必须是唯一的。名称最多可以包含 64 个字符。有效字符为 A-Z、a-z、0-9 和 . _ -（连字符）。
  - （可选）描述 - 输入目标的描述。

    描述最多可包含 512 个字符。
  - API 目标端点 - 目标的 URL 端点。

    端点 URL 必须以 **`https`** 开头。可以将 **`*`** 作为路径参数通配符包括在其中。您可以根据目标的 `HttpParameters` 属性设置路径参数。
  - HTTP 方法 - 选择调用端点时使用的 HTTP 方法。

- （可选）每秒调用速率限制 - 输入该目标每秒可接受的调用次数上限。

  该值必须大于零。默认情况下，该值设为 300。

- 连接 - 选择使用新连接或现有连接：
  - 要使用现有连接，请选择使用现有连接，然后从下拉列表中选择连接。
  - 要为此目标创建新连接，请选择创建新连接，然后定义连接的名称、目标类型和授权类型。您还可以为此连接添加可选描述。

有关更多信息，请参阅???。

AWS 服务

要使用 AWS 服务，请选择 AWS 服务，然后执行以下操作：

1. 在选择目标中，选择一个 AWS 服务用作目标。为所选服务提供所需的信息。

> **Note**
>
> 显示的字段因所选服务而异。有关可用目标的更多信息，请参阅 EventBridge 控制台中可用的目标。

2. 对于许多目标类型，EventBridge 需要权限以便将事件发送到目标。在这些情况下，EventBridge 可以创建运行事件所需的 IAM 角色：

   对于执行角色，请执行以下任一操作：

   - 为此规则创建新的执行角色：
     a. 选择为此特定资源创建新角色。
     b. 输入此执行角色的名称，或使用 EventBridge 生成的名称。
   - 为此规则使用现有执行角色：
     a. 选择使用现有角色。
     b. 输入要使用的执行角色的名称，或从下拉列表中选择。

3. （可选）对于其他设置，请指定适用于您的目标类型的任何可选设置：

Event bus

（可选）对于死信队列，选择是否使用标准 Amazon SQS 队列作为死信队列。如果与此规则匹配的事件未成功传递到目标，EventBridge 会将这些事件发送到死信队列。请执行下列操作之一：

- 选择无不使用死信队列。
- 在当前 AWS 帐户中选择选择一个Amazon SQS队列用作死信队列，然后从下拉列表中选择要使用的队列。
- 选择在其他 Amazon SQS 队列中选择其他队列 AWS 帐户作为死信队列，然后输入要使用的队列的 ARN。您必须将基于资源的策略附加到队列，以授予 EventBridge 向其发送消息的权限。

  有关更多信息，请参阅为死信队列授予权限。

API destination

1. （可选）在配置目标输入中，选择针对匹配的事件，要如何自定义发送到目标的文本。选择以下选项之一：
   - 匹配的事件 - EventBridge 会将整个原始源事件发送到目标。这是默认模式。
   - 部分匹配事件 - EventBridge 仅将原始源事件的指定部分发送到目标。

     在指定部分匹配事件下，指定一个 JSON 路径，其中定义了您希望 EventBridge 发送到目标的事件部分。
   - 常量 (JSON 文本) - EventBridge 仅向目标发送指定的 JSON 文本。不会发送原始源事件的任何部分。

     在在 JSON 中指定常量下，指定您希望 EventBridge 发送到目标的 JSON 文本（而不是事件）。
   - 输入转换器 - 配置输入转换器，自定义您希望 EventBridge 向目标发送的文本。有关更多信息，请参阅???。

     a. 选择配置输入转换器。

     b. 按照???中的步骤配置输入转换器。
2. （可选）在重试策略下，指定发生错误后 EventBridge 应如何重试向目标发送事件。

- 事件的最长保留时间 - 输入 EventBridge 保留未处理事件的时间上限（以小时、分钟和秒为单位）。默认为 24 小时。
- 重试次数 - 输入发生错误后，EventBridge 应重试向目标发送事件的次数上限。默认为 185 次。

3. （可选）对于死信队列，选择是否使用标准 Amazon SQS 队列作为死信队列。如果与此规则匹配的事件未成功传递到目标，EventBridge 会将这些事件发送到死信队列。请执行下列操作之一：
   - 选择无不使用死信队列。
   - 在当前 AWS 帐户中选择选择一个Amazon SQS队列用作死信队列，然后从下拉列表中选择要使用的队列。
   - 选择在其他 Amazon SQS 队列中选择其他队列 AWS 帐户作为死信队列，然后输入要使用的队列的 ARN。您必须将基于资源的策略附加到队列，以授予 EventBridge 向其发送消息的权限。

     有关更多信息，请参阅为死信队列授予权限。

AWS service

请注意，EventBridge 可能不会显示特定 AWS 服务的以下所有字段。

1. （可选）在配置目标输入中，选择针对匹配的事件，要如何自定义发送到目标的文本。选择以下选项之一：
   - 匹配的事件 - EventBridge 会将整个原始源事件发送到目标。这是默认模式。
   - 部分匹配事件 - EventBridge 仅将原始源事件的指定部分发送到目标。

     在指定部分匹配事件下，指定一个 JSON 路径，其中定义了您希望 EventBridge 发送到目标的事件部分。
   - 常量 (JSON 文本) - EventBridge 仅向目标发送指定的 JSON 文本。不会发送原始源事件的任何部分。

     在在 JSON 中指定常量下，指定您希望 EventBridge 发送到目标的 JSON 文本（而不是事件）。
   - 输入转换器 - 配置输入转换器，自定义您希望 EventBridge 向目标发送的文本。有关更多信息，请参阅???。

     a. 选择配置输入转换器。

b. 按照???中的步骤配置输入转换器。

2. （可选）在重试策略下，指定发生错误后 EventBridge 应如何重试向目标发送事件。
   - 事件的最长保留时间 - 输入 EventBridge 保留未处理事件的时间上限（以小时、分钟和秒为单位）。默认为 24 小时。
   - 重试次数 - 输入发生错误后，EventBridge 应重试向目标发送事件的次数上限。默认为 185 次。
3. （可选）对于死信队列，选择是否使用标准 Amazon SQS 队列作为死信队列。如果与此规则匹配的事件未成功传递到目标，EventBridge 会将这些事件发送到死信队列。请执行下列操作之一：
   - 选择无不使用死信队列。
   - 在当前 AWS 帐户中选择选择一个Amazon SQS队列用作死信队列，然后从下拉列表中选择要使用的队列。
   - 选择在其他 Amazon SQS 队列中选择其他队列 AWS 帐户作为死信队列，然后输入要使用的队列的 ARN。您必须将基于资源的策略附加到队列，以授予 EventBridge 向其发送消息的权限。

   有关更多信息，请参阅为死信队列授予权限。
4. （可选）选择 Add another target（添加其他目标），以为此规则添加其他目标。
5. 选择 Next（下一步）。

请注意，EventBridge 可能不会显示特定 AWS 服务的以下所有字段。

配置标签并检查规则

最后，为规则输入所需的任何标签，然后检查并创建规则。

配置标签，检查并创建规则

1. （可选）为规则输入一个或多个标签。有关更多信息，请参阅亚马逊 EventBridge 标签。
2. 选择 Next（下一步）。
3. 检查新规则的详细信息。要对任何部分进行更改，请选择该部分旁边的编辑按钮。

   对规则详情感到满意后，选择创建规则。

将 Amazon EventBridge 调度器与 Amazon EventBridge 结合使用

Amazon EventBridge 调度器是一个无服务器调度器，使您能够从一个中央托管服务创建、运行和管理任务。借助 EventBridge 调度器，您可以使用 cron 和 rate 表达式为定期模式创建计划，也可以配置一次性调用。您可以设置灵活的交付时间窗口、定义重试限制，并为失败的 API 调用设置最大保留时间。

EventBridge 调度器具有高度可定制性，与 EventBridge 计划规则相比，可扩展性更高，目标 API 操作和 AWS 服务范围更广。建议您使用 EventBridge 调度器按计划调用目标。

主题

设置执行角色

创建新计划时，EventBridge 调度器必须有权代表您调用其目标 API 操作。您可以使用执行角色授予 EventBridge 调度器这些权限。您附加到计划执行角色的权限策略定义了所需权限。这些权限取决于您希望 EventBridge 调度器调用的目标 API。

使用 EventBridge 调度器控制台创建计划时，EventBridge 调度器会根据选择的目标自动设置执行角色，如以下步骤所示。如果您想使用 EventBridge 调度器 SDK（AWS CLI 或 AWS CloudFormation ）之一创建计划，您必须拥有现有的执行角色，以授予 EventBridge 调度器调用目标所需的权限。有关为计划手动设置执行角色的更多信息，请参阅 EventBridge Scheduler User Guide 中的 Setting up an execution role。

创建计划

使用控制台创建计划

1. 打开 Amazon EventBridge 调度器控制台，网址为：https://console.aws.amazon.com/scheduler/home。
2. 在计划页面，选择创建计划。
3. 在指定计划详细信息页面，在计划名称和描述部分中，执行以下操作：

   a. 对于计划名称，输入计划的名称。例如，**`MyTestSchedule`**。

b. （可选）对于描述，输入对计划的描述。例如，**`My first schedule`**。

c. 对于计划组，从下拉列表中选择一个计划组。如果您没有计划组，选择默认。要创建计划组，选择创建自己的计划。

您可以使用计划组将标签添加到计划组。

4. • 选择计划选项。

出现	请执行此操作...
一次性计划 一次性计划仅在您指定的日期和时间调用一次目标。	对于日期和时间，请执行以下操作： • 输入 YYYY/MM/DD 格式的有效日期。 • 输入 24 小时 hh:mm 格式的时间戳。 • 对于时区，选择时区。
定期计划 定期计划按照您使用 cron 表达式或 rate 表达式指定的速率调用目标。	a. 对于计划类型，执行以下操作之一： • 要使用 cron 表达式定义计划，请选择基于 cron 的计划并输入 cron 表达式。 • 要使用 rate 表达式定义计划，请选择基于 rate 的计划并输入 rate 表达式。 有关 cron 和 rate 表达式的更多信息，请参阅 Amazon EventBridge Scheduler User Guide 中的 Schedule types on EventBridge Scheduler。

出现	请执行此操作...	
	b. 对于灵活的时间窗口，选择关闭以关闭该选项，或者选择一个预定义的时间窗口。例如，如果您选择 15 分钟并且将定期计划设置为每小时调用一次其目标，则该计划将在每小时开始后的 15 分钟内运行。	

5. （可选）如果您在上一步中选择定期计划，在时间范围部分，请执行以下操作：

   a. 对于时区，请选择时区。

   b. 对于开始日期和时间，请输入 YYYY/MM/DD 格式的有效日期，然后指定 24 小时 hh:mm 格式的时间戳。

   c. 对于结束日期和时间，请输入 YYYY/MM/DD 格式的有效日期，然后指定 24 小时 hh:mm 格式的时间戳。

6. 选择 Next（下一步）。

7. 在选择目标页面，选择 EventBridge 调度器调用的 AWS API 操作：

   a. 对于目标 API，请选择模板化目标。

   b. 选择 Amazon EventBridge PutEvents。

   c. 在 PutEvents 下，指定以下选项：

   - 对于 EventBridge 事件总线，从下拉菜单中选择事件总线。例如，**`default`**。

     您也可以选择创建新的事件总线，在 EventBridge 控制台中创建新的事件总线。

   - 在详细信息类型中，输入要匹配的事件的详细信息类型。例如，**`Object Created`**。
   - 在源中，输入作为事件源的服务的名称。

     对于 AWS 服务事件，请将服务前缀指定为源。请勿包含 `aws.` 前缀。例如，对于 Amazon S3 事件，请输入 **`s3`**。

     要确定服务的前缀，请参阅《服务授权参考》中的条件键表。有关源事件值和详细信息类型事件值的更多信息，请参阅???。

- （可选）：在详细信息中输入事件模式，进一步筛选 EventBridge 调度器发送到 EventBridge 的事件。

  有关更多信息，请参阅???。

8. 选择 Next（下一步）。
9. 在 Settings (设置) 页面上，执行以下操作：

   a. 要打开计划，在计划状态下，切换启用计划。

   b. 要为计划配置重试策略，在重试策略和死信队列（DLQ）下，请执行以下操作：

   - 切换重试。
   - 对于事件的最长期限，输入 EventBridge 调度器必须保留未处理事件的最长小时和分钟数。
   - 最长时间为 24 小时。
   - 对于最大重试次数，输入在目标返回错误的情况下，EventBridge 调度器重试计划的最大次数。

     最大值为 185 次重试。

   配置重试策略后，如果计划未能调用其目标，EventBridge 调度器将重新运行该计划。如果已配置，则必须为计划设置最长保留时间和最大重试次数。

   c. 选择 EventBridge 调度器存储未送达事件的位置。

死信队列（DLQ）选项	请执行此操作...	
请勿存储	选择 None。	
将事件存储在创建计划所在的同一 AWS 账户 中	a. 选择在我的 AWS 账户 中选择一个 Amazon SQS 队列作为 DLQ。 b. 选择 Amazon SQS 队列的 Amazon 资源名称（ARN）。	
将事件存储在与创建计划所在不同的 AWS 账户 中	a. 选择在另一个 AWS 账户 中指定一个 Amazon SQS 队列作为 DLQ。	

死信队列（DLQ）选项	请执行此操作...	
	b. 输入 Amazon SQS 队列的 Amazon 资源名称（ARN）。	

d. 要使用客户托管密钥加密目标输入，在加密下，选择自定义加密设置（高级）。

   如果选择此选项，请输入现有的 KMS 密钥 ARN 或选择创建一个 AWS KMS key 以导航到 AWS KMS 控制台。有关 EventBridge 调度器如何加密静态数据的更多信息，请参阅 Amazon EventBridge Scheduler User Guide 中的 Encryption at rest。

e. 要让 EventBridge 调度器为您创建新的执行角色，请选择为此计划创建新角色。然后，在角色名称中输入名称。如果您选择此选项，EventBridge 调度器会将模板化目标所需的必要权限附加到该角色。

10. 选择 Next（下一步）。
11. 在查看并创建计划页面上，查看计划的详细信息。在每个部分中，选择编辑返回到该步骤并编辑其详细信息。
12. 选择创建计划。

    您可以在计划页面上查看新的和现有的计划列表。在状态列下，验证新计划是否已启用。

相关的 资源

有关 EventBridge 调度器的详细信息，请参阅以下内容：

- EventBridge Scheduler User Guide
- EventBridge Scheduler API Reference
- EventBridge Scheduler Pricing

创建按计划运行的 Amazon EventBridge 规则

规则可以响应事件运行，也可以按特定的时间间隔运行。例如，要定期运行 AWS Lambda 函数，可以创建按计划运行的规则。

> **Note**
>
> EventBridge 提供的 Amazon EventBridge 调度器是无服务器调度器，使您能够通过中央托管服务创建、运行和管理任务。EventBridge 调度器具有高度可定制性，与 EventBridge 计划规则相比，可扩展性更高，目标 API 操作和 AWS 服务范围更广。
> 建议您使用 EventBridge 调度器按计划调用目标。有关更多信息，请参阅???。

在 EventBridge 中，您可以创建两种类型的计划规则：

- 按正常频率运行的规则

  EventBridge 会定期运行这些规则；例如，每 20 分钟运行一次。

  要为计划规则指定频率，请定义 rate 表达式。
- 在特定时间运行的规则

  EventBridge 在特定的时间和日期运行这些规则；例如，上午 8:00。PST，每个月的第一个星期一。

  要指定计划规则运行的时间和日期，请定义 cron 表达式。

rate 表达式更易于定义，而 cron 表达式可提供详细的计划控制。例如，使用 cron 表达式，您可以定义在每周或每月的某一天的指定时间运行的规则。相反，rate 表达式以常规速率运行规则，例如每小时一次或每天一次。

所有计划的事件都使用 UTC+0 时区，计划的最小精度为 1 分钟。

> **Note**
>
> EventBridge 不在计划表达式中提供第二级精度。使用 cron 表达式的最高解析精度是一分钟。由于 EventBridge 和目标服务的分布式特性，触发计划规则的时间与目标服务运行目标资源的时间之间，有几秒钟的延迟。

以下视频概述了如何计划任务：使用 EventBridge 创建计划任务

主题

- 创建按计划运行的规则
- Cron 表达式引用
- Rate 表达式引用

创建按计划运行的规则

以下步骤将引导您创建按定期计划运行的 EventBridge 规则。

> **Note**
>
> 您只能使用默认事件总线创建计划规则。

步骤

- 定义规则
- 定义计划
- 选择目标
- 配置标签并检查规则

定义规则

首先，为您的规则输入用于标识它的名称和描述。

定义规则细节

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Rules (规则)。
3. 选择 Create rule (创建规则)。
4. 输入规则的名称和可选描述。

   规则不能与同一 AWS 区域中和同一事件总线上的另一条规则的名称相同。
5. 对于事件总线，选择默认事件总线。您只能使用默认事件总线创建计划规则。
6. 要使规则在创建后立即生效，请确保已启用在选定的事件总线上启用该规则选项。
7. 对于 Rule type（规则类型），选择 Schedule（计划）。

   此时，您可以选择继续创建按计划运行的规则，也可以使用 Amazon EventBridge 调度器。

8. 选择您希望如何继续：

   - 使用 EventBridge 调度器创建您的计划

     > **Note**
     >
     > EventBridge 调度器是一个无服务器调度器，使您能够从一个中央托管服务创建、运行和管理任务。它提供独立于事件总线和规则的一次性和重复性计划功能。EventBridge 调度器具有高度可定制性，与 EventBridge 计划规则相比，可扩展性更高，目标 API 操作和 AWS 服务范围更广。
     > 建议您使用 EventBridge 调度器按计划调用目标。有关更多信息，请参阅《Amazon EventBridge 调度器用户指南》中的什么是 Amazon EventBridge 调度器。

     1. 选择在 EventBridge Scheduler 中继续

        EventBridge 将在 EventBridge 调度器控制台中打开创建计划页面。

     2. 在 EventBridge 调度器控制台中创建计划。

   - 继续使用 EventBridge 为默认事件总线创建计划规则

     1. 选择继续创建规则。

定义计划

接下来，定义计划模式。

定义计划模式

1. 在计划模式中，选择是希望计划在特定时间运行，还是按固定频率运行：

   Specific time

   1. 选择在特定时间运行的精细计划，例如上午 8:00 PST，每个月的第一个星期一。
   2. 对于 Cron 表达式，请指定字段来定义 cron 表达式，EventBridge 应使用该表达式来确定何时执行此计划规则。

      一旦您指定了所有字段，EventBridge 就会显示接下来将执行此计划规则的十个日期。您可以选择以 UTC 或本地时区来显示这些日期。

      有关构造 cron 表达式的更多信息，请参阅 ???。

Regular rate

1. 选择以固定频率运行的计划，例如每 10 分钟运行一次。
2. 在 rate 表达式中，指定值和单位字段，定义 EventBridge 应执行此计划规则的频率。

   有关构造 rate 表达式的更多信息，请参阅 ???。

2. 选择 Next（下一步）。

选择目标

选择一个或多个目标，接收与指定模式匹配的事件。目标可以包括 EventBridge 事件总线、EventBridge API 目标（包括 Salesforce 等 SaaS 合作伙伴或另一 AWS 服务）。

选择目标

1. 对于目标类型，请选择以下目标类型之一：

   Event bus

   要选择 EventBridge 事件总线，请选择EventBridge 事件总线，然后执行以下操作：

   - 使用与此规则位于同一 AWS 区域的事件总线：
     1. 选择同一账户和区域中的事件总线。
     2. 对于目标的事件总线，选择下拉框并输入事件总线的名称。您也可以从下拉列表中选择事件总线。

        有关更多信息，请参阅???。
   - 使用与此规则不同的 AWS 区域 或账户中的事件总线：
     1. 选择不同账户或区域中的事件总线。
     2. 对于事件总线作为目标，请输入要使用的事件总线的 ARN。

        有关更多信息，请参阅：

        - ???
        - ???

API destination

要使用 EventBridge API 目标，请选择 EventBridge API 目标，然后执行以下任一操作：

- 要使用现有 API 目标，请选择使用现有 API 目标。然后从下拉列表中选择 API 目标。
- 要创建新的 API 目标，请选择创建新的 API 目标。然后为目标提供以下详细信息：
  - 名称 - 为目标键入一个名称。

    名称在您的 AWS 账户 内必须是唯一的。名称最多可以包含 64 个字符。有效字符为 A-Z、a-z、0-9 和 . _ -（连字符）。
  - （可选）描述 - 输入目标的描述。

    描述最多可包含 512 个字符。
  - API 目标端点 - 目标的 URL 端点。

    端点 URL 必须以 **`https`** 开头。可以将 **`*`** 作为路径参数通配符包括在其中。您可以根据目标的 `HttpParameters` 属性设置路径参数。
  - HTTP 方法 - 选择调用端点时使用的 HTTP 方法。
  - （可选）每秒调用速率限制 - 输入该目标每秒可接受的调用次数上限。

    该值必须大于零。默认情况下，该值设为 300。
  - 连接 - 选择使用新连接或现有连接：
    - 要使用现有连接，请选择使用现有连接，然后从下拉列表中选择连接。
    - 要为此目标创建新连接，请选择创建新连接，然后定义连接的名称、目标类型和授权类型。您还可以为此连接添加可选描述。

有关更多信息，请参阅???。

AWS 服务

要使用 AWS 服务，请选择 AWS 服务，然后执行以下操作：

1. 在选择目标中，选择一个 AWS 服务用作目标。为所选服务提供所需的信息。

> **Note**
>
> 显示的字段因所选服务而异。有关可用目标的更多信息，请参阅 EventBridge 控制台中可用的目标。

2. 对于许多目标类型，EventBridge 需要权限以便将事件发送到目标。在这些情况下，EventBridge 可以创建运行事件所需的 IAM 角色：

   对于执行角色，请执行以下任一操作：

   - 为此规则创建新的执行角色：
     a. 选择为此特定资源创建新角色。
     b. 输入此执行角色的名称，或使用 EventBridge 生成的名称。
   - 为此规则使用现有执行角色：
     a. 选择使用现有角色。
     b. 输入要使用的执行角色的名称，或从下拉列表中选择。

3. （可选）对于其他设置，请指定适用于您的目标类型的任何可选设置：

   Event bus

   （可选）对于死信队列，选择是否使用标准 Amazon SQS 队列作为死信队列。如果与此规则匹配的事件未成功传递到目标，EventBridge 会将这些事件发送到死信队列。请执行下列操作之一：

   - 选择无不使用死信队列。
   - 在当前 AWS 帐户中选择选择一个Amazon SQS队列用作死信队列，然后从下拉列表中选择要使用的队列。
   - 选择在其他 Amazon SQS 队列中选择其他队列 AWS 帐户作为死信队列，然后输入要使用的队列的 ARN。您必须将基于资源的策略附加到队列，以授予 EventBridge 向其发送消息的权限。

     有关更多信息，请参阅为死信队列授予权限。

API destination

1. （可选）在配置目标输入中，选择针对匹配的事件，要如何自定义发送到目标的文本。选择以下选项之一：
   - 匹配的事件 - EventBridge 会将整个原始源事件发送到目标。这是默认模式。
   - 部分匹配事件 - EventBridge 仅将原始源事件的指定部分发送到目标。

     在指定部分匹配事件下，指定一个 JSON 路径，其中定义了您希望 EventBridge 发送到目标的事件部分。
   - 常量 (JSON 文本) - EventBridge 仅向目标发送指定的 JSON 文本。不会发送原始源事件的任何部分。

     在在 JSON 中指定常量下，指定您希望 EventBridge 发送到目标的 JSON 文本（而不是事件）。
   - 输入转换器 - 配置输入转换器，自定义您希望 EventBridge 向目标发送的文本。有关更多信息，请参阅???。

     a. 选择配置输入转换器。

     b. 按照???中的步骤配置输入转换器。
2. （可选）在重试策略下，指定发生错误后 EventBridge 应如何重试向目标发送事件。
   - 事件的最长保留时间 - 输入 EventBridge 保留未处理事件的时间上限（以小时、分钟和秒为单位）。默认为 24 小时。
   - 重试次数 - 输入发生错误后，EventBridge 应重试向目标发送事件的次数上限。默认为 185 次。
3. （可选）对于死信队列，选择是否使用标准 Amazon SQS 队列作为死信队列。如果与此规则匹配的事件未成功传递到目标，EventBridge 会将这些事件发送到死信队列。请执行下列操作之一：
   - 选择无不使用死信队列。
   - 在当前 AWS 帐户中选择选择一个Amazon SQS队列用作死信队列，然后从下拉列表中选择要使用的队列。
   - 选择在其他 Amazon SQS 队列中选择其他队列 AWS 帐户作为死信队列，然后输入要使用的队列的 ARN。您必须将基于资源的策略附加到队列，以授予 EventBridge 向其发送消息的权限。

     有关更多信息，请参阅为死信队列授予权限。

AWS service

请注意，EventBridge 可能不会显示特定 AWS 服务的以下所有字段。

1. （可选）在配置目标输入中，选择针对匹配的事件，要如何自定义发送到目标的文本。选择以下选项之一：
   - 匹配的事件 - EventBridge 会将整个原始源事件发送到目标。这是默认模式。
   - 部分匹配事件 - EventBridge 仅将原始源事件的指定部分发送到目标。

     在指定部分匹配事件下，指定一个 JSON 路径，其中定义了您希望 EventBridge 发送到目标的事件部分。
   - 常量 (JSON 文本) - EventBridge 仅向目标发送指定的 JSON 文本。不会发送原始源事件的任何部分。

     在在 JSON 中指定常量下，指定您希望 EventBridge 发送到目标的 JSON 文本（而不是事件）。
   - 输入转换器 - 配置输入转换器，自定义您希望 EventBridge 向目标发送的文本。有关更多信息，请参阅???。

     a. 选择配置输入转换器。

     b. 按照???中的步骤配置输入转换器。
2. （可选）在重试策略下，指定发生错误后 EventBridge 应如何重试向目标发送事件。
   - 事件的最长保留时间 - 输入 EventBridge 保留未处理事件的时间上限（以小时、分钟和秒为单位）。默认为 24 小时。
   - 重试次数 - 输入发生错误后，EventBridge 应重试向目标发送事件的次数上限。默认为 185 次。
3. （可选）对于死信队列，选择是否使用标准 Amazon SQS 队列作为死信队列。如果与此规则匹配的事件未成功传递到目标，EventBridge 会将这些事件发送到死信队列。请执行下列操作之一：
   - 选择无不使用死信队列。
   - 在当前 AWS 帐户中选择选择一个Amazon SQS队列用作死信队列，然后从下拉列表中选择要使用的队列。
   - 选择在其他 Amazon SQS 队列中选择其他队列 AWS 帐户作为死信队列，然后输入要使用的队列的 ARN。您必须将基于资源的策略附加到队列，以授予 EventBridge 向其发送消息的权限。

有关更多信息，请参阅为死信队列授予权限。

4. （可选）选择 Add another target（添加其他目标），以为此规则添加其他目标。
5. 选择 Next（下一步）。

配置标签并检查规则

最后，为规则输入所需的任何标签，然后检查并创建规则。

配置标签，检查并创建规则

1. （可选）为规则输入一个或多个标签。有关更多信息，请参阅亚马逊 EventBridge 标签。
2. 选择 Next（下一步）。
3. 检查新规则的详细信息。要对任何部分进行更改，请选择该部分旁边的编辑按钮。

   对规则详情感到满意后，选择创建规则。

Cron 表达式引用

Cron 表达式有六个必填字段，之间以空格分隔。

语法

```
cron(fields)
```

Field	值	通配符
分钟	0-59	, - * /
小时	0-23	, - * /
日期	1-31	, - * ? / L W
月份	1-12 或 JAN-DEC	, - * /
星期几	1-7 或 SUN-SAT	, - * ? L #
年	1970-2199	, - * /

通配符

- , (逗号) 通配符包含其他值。在“月份”字段中，JAN、FEB 和 MAR 包含 January、February 和 March。
- - (破折号) 通配符用于指定范围。在“日”字段中，1-15 包含指定月份的 1 - 15 日。
- * (星号) 通配符包含该字段中的所有值。在“Hours (小时) ”字段中，* 包括每个小时。您不能在“日期”和“星期几”字段中同时使用 *。如果您在一个中使用它，则必须在另一个中使用 ？。
- / (斜杠) 通配符用于指定增量。在“分钟”字段中，您可以输入 1/10 以指定从一个小时的第一分钟开始的每个第十分钟 (例如，第 11 分钟、第 21 分钟和第 31 分钟，依此类推)。
- ? (问号) 通配符用于指定任何。在“日期”字段中，您可以输入 7，如果一周的任何日期都是可接受的，则可以在“日期”字段中输入 ？
- “日期”或“星期几”字段中的 L 通配符用于指定月或周的最后一天。
- “日期”字段中的 **W** 通配符用于指定工作日。在“日期”字段中，**3W** 用于指定最靠近当月的第三周的日。
- “星期几”字段中的 # 通配符用于指定一个月内所指定星期几的特定实例。例如，3#2 指该月的第二个星期二：3 指的是星期二，因为它是每周的第三天，2 是指该月内该类型的第二天。

> **Note**
>
> 如果使用 “#” 字符，则只能在星期字段中定义一个表达式。例如，`"3#1,6#3"` 是无效的，因为它被解释为两个表达式。

限制

- 您无法在同一 Cron 表达式中为日期和星期几字段同时指定值。如果您在其中一个字段中指定值或 * (星号) ，则必须在另一个字段中使用 ? (问号) 。
- 不支持产生的速率快于 1 分钟的 Cron 表达式。

示例

在创建带计划的规则时，可以使用以下示例 cron 字符串。

分钟	小时	日期	月份	星期几	年	含义
0	10	*	*	?	*	每天上午 10:00

分钟	小时	日期	月份	星期几	年	含义
						(UTC+0) 运行
15	12	*	*	?	*	每天中午 12:15 (UTC+0) 运行
0	18	?	*	MON-FRI	*	每星期一到星期五的下午 6:00 (UTC+0) 运行
0	8	1	*	?	*	每月第 1 天上午 8:00 (UTC+0) 运行
0/15	*	*	*	?	*	每 15 分钟运行一次
0/10	*	?	*	MON-FRI	*	从星期一到星期五，每 10 分钟运行一次
0/5	8-17	?	*	MON-FRI	*	每星期一到星期五的上午 8:00 到下午 5:55 (UTC+0) 之间，每隔 5 分钟运行一次

分钟	小时	日期	月份	星期几	年	含义
0/30	20-2	?	*	MON-FRI	*	星期一至星期五，从起始日晚上 10:00 至次日凌晨 2:00 (UTC) 每隔 30 分钟运行一次 星期一凌晨 12:00 至凌晨 2:00 (UTC) 运行。

以下示例创建一条规则，在每天中午 12:00 (UTC+0) 运行。

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

以下示例创建一条规则，在每天下午 2:05pm 和 2:35pm (UTC+0) 运行。

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

以下示例创建一条规则，从 2019 到 2022 年在每个月最后一个周五的上午 10:15 (UTC+0) 运行。

```
aws events put-rule --schedule-expression "cron(15 10 ? * 6L 2019-2022)" --name MyRule3
```

Rate 表达式引用

Rate 表达式在创建计划事件规则时启动，然后按照其定义的计划运行。

Rate 表达式有两个必填字段，之间以空格分隔。

语法

```
rate(value unit)
```

值

正数。

单位

时间单位。需要不同的单位，例如，对于值 1 为 `minute`；对于大于 1 的值 1 为 `minutes`。

有效值：minute | minutes | hour | hours | day | days

限制

如果值等于 1，则单位必须为单数。如果值大于 1，则单位必须为复数。例如，rate(1 hours) 和 rate(5 hour) 无效，而 rate(1 hour) 和 rate(5 hours) 有效。

示例

以下示例说明如何将 Rate 表达式与 AWS CLI `put-rule` 命令结合使用。第一个示例每分钟触发一次规则，第二个示例每 5 分钟触发一次规则，第三个示例每小时触发一次规则，第四个示例每天触发一次规则。

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

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

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

```
aws events put-rule --schedule-expression "rate(1 day)" --name MyRule5
```

禁用或删除 Amazon EventBridge 规则

要阻止规则处理事件或按计划运行，可以删除或禁用该规则。以下步骤将引导您删除或禁用 EventBridge 规则。

删除或禁用规则

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Rules (规则)。

   在 Event bus (事件总线) 下，选择与规则关联的事件总线。
3. 请执行下列操作之一：

   a. 要删除规则，请选择规则旁边的按钮，然后依次选择 Actions、Delete 和 Delete。

      如果该规则是托管规则，请输入规则的名称以确认它是托管规则，删除它可能会在创建此规则的服务中停止功能。要继续，请输入规则名称并选择 Force delete (强制删除)。

   b. 要临时禁用规则，请选择规则旁边的按钮，然后依次选择 Disable (禁用) 和 Disable (禁用)。

      您不能禁用托管规则。

定义 Amazon EventBridge 规则的最佳实践

以下是为事件总线创建规则时需要考虑的一些最佳实践。

为每条规则设置单一目标

虽然您最多可以为给定规则指定五个目标，但如果为每条规则指定单一目标，管理规则将会更加轻松。如果多个目标需要接收同一组事件，我们建议复制该规则，将相同的事件传送到不同目标。这种封装简化了规则的维护：如果事件目标的需求随着时间的推移而出现差异，则可以独立于其他规则更新每条规则，及其事件模式。

设置规则权限

您可以进行设置，使使用事件的应用程序组件或服务能够控制各自规则的管理。客户采用的一种常见架构方法是，使用单独的 AWS 帐户来隔离这些应用程序组件或服务。要使事件从一个账户流向另一个账户，必须在一条事件总线上创建一个规则，将事件路由到另一个账户中的事件总线。您可以进行设置，使使用事件的团队或服务能够控制各自规则的管理。方法是通过资源策略为他们的账户指定适当的权限。这适用于所有账户和区域。

有关更多信息，请参阅???。

有关资源策略的示例，请参阅 GitHub 上的 Amazon EventBridge 多账户设计模式。

监控规则性能

监控您的规则，确保其执行符合您的预期：

- 监控 `TriggeredRules` 指标中是否存在缺失的数据点或异常，可以帮助您检测发布者做出的重大更改的差异。有关更多信息，请参阅???。
- 异常警报或预期计数上限也有助于检测某规则何时与新事件匹配。在启用新的使用案例和功能时，如果包括 AWS 服务和 SaaS 合作伙伴在内的事件发布者引入新事件，就会发生这种情况。如果这些新事件在意料之外，并且导致处理量高于下游目标的处理速度，它们可能会导致事件积压。

  对意外事件的此类处理也可能导致不必要的账单费用。

  当账户超过其每秒目标调用总次数服务限额时，还会触发规则节流。EventBridge 仍会尝试传送与节流规则匹配的事件，并在最长 24 小时内重试，或者按照目标的自定义重试策略中的设置进行重试。您可以使用 `ThrottledRules` 指标检测节流规则并发出警报
- 对于低延迟使用场景，您还可以使用 `IngestionToInvocationStartLatency` 监控延迟，它可以指示事件总线的运行状况。任何超过 30 秒的高延迟时间段都可能表示服务中断或规则节流。

使用 Amazon EventBridge 和 AWS Serverless Application Model 模板

您可以在 EventBridge 控制台中手动构建和测试规则，这有助于在开发过程中完善事件模式。但是，一旦您准备好部署应用程序，使用 AWS SAM 等框架一致地启动所有无服务器资源则更加方便。

我们将使用此示例应用程序来研究如何使用 AWS SAM 模板构建 EventBridge 资源。此示例中的 template.yaml 文件是一个 AWS SAM 模板，其中定义了四个 AWS Lambda 函数，并展示了将 Lambda 函数与 EventBridge 集成的两种不同方式。

有关此示例应用程序的演练，请参阅???。

可通过两种方式将 EventBridge 和 AWS SAM 模板配合使用。对于由一条规则调用一个 Lambda 函数的简单集成，建议使用组合模板方法。如果您的路由逻辑很复杂，或者要连接到 AWS SAM 模板之外的资源，那么分开的模板方法是更好的选择。

方法：

- 组合模板
- 分开的模板

组合模板

第一种方法使用 `Events` 属性来配置 EventBridge 规则。以下示例代码定义了一个事件，它调用您的 Lambda 函数。

> **Note**
>
> 此示例自动在默认事件总线上创建规则，每个 AWS 账户中都有默认事件总线。要将规则与自定义事件总线关联，可以将 `EventBusName` 添加到模板中。

```
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"
```

此 YAML 代码等同于 EventBridge 控制台中的一个事件模式。在 YAML 中，您只需要定义事件模式，AWS SAM 会自动创建具有所需权限的 IAM 角色。

分开的模板

在 AWS SAM 中定义 EventBridge 配置的第二种方法，模板对资源进行了更明确的分隔。

1. 首先，您要定义 Lambda 函数：

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

2. 接下来，使用 `AWS::Events::Rule` 资源定义规则。这些属性定义了事件模式，也可以指定目标。您可以明确定义多个目标。

   ```
   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. 最后，定义一个 AWS::Lambda::Permission 资源，向 EventBridge 授予调用目标的权限。

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

根据 Amazon EventBridge 规则生成 AWS CloudFormation 模板

AWS CloudFormation 可将基础设施视为代码，使您能够以集中且可重复的方式跨账户和区域配置和管理 AWS 资源。CloudFormation 实现这一功能的方式是允许您创建模板，它定义了您要预置和管理的资源。

EventBridge 允许您根据账户中的现有规则生成模板，以此来帮助您快速开始开发 CloudFormation 模板。您可以选择在模板中包含一条或多条规则。然后，您可以使用这些模板作为基础创建资源堆栈，实现 CloudFormation 管理。

有关 CloudFormation 的更多信息，请参阅《AWS CloudFormation 用户指南》。

> **Note**
>
> EventBridge 在生成的模板中不包含托管规则。

您也可以根据现有事件总线生成模板（包括事件总线中包含的规则）。

根据一条或多条规则生成 AWS CloudFormation 模板

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择规则。
3. 在选择事件总线下选择事件总线，其中包含模板中要包含的规则。
4. 在规则下，选择要包含在生成的 AWS CloudFormation 模板中的规则。

   对于单个规则，您也可以选择规则名称以显示其详细信息页面。
5. 选择 CloudFormation 模板，然后选择您希望 EventBridge 生成模板的格式：JSON 或 YAML。

   EventBridge 将显示以所选格式生成的模板。
6. EventBridge 支持您选择下载模板文件或将模板复制到剪贴板。

   - 选择下载，下载模板文件。
   - 要将此模板复制到剪贴板，请选择复制。
7. 要退出模板，请选择取消。

根据使用场景的需要自定义 AWS CloudFormation 模板后，即可使用它在 AWS CloudFormation 中创建堆栈。

使用 Amazon EventBridge 生成的 CloudFormation 模板时的注意事项

使用从 EventBridge 生成的 CloudFormation 模板时，请考虑以下因素：

- EventBridge 在生成的模板中不包含任何密码。

  您可以编辑此模板，以包含模板参数，使用户能够在使用模板创建或更新 CloudFormation 堆栈时指定密码或其他敏感信息。

  此外，用户可以使用 Secrets Manager 在所需区域创建密钥，然后编辑生成的模板以使用动态参数。
- 生成的模板中的目标，与原始事件总线中指定的目标完全相同。如果在使用模板在其他区域创建堆栈之前，未对模板进行适当的编辑，可能会导致跨区域问题。

  此外，生成的模板不会自动创建下游目标。

亚马逊的 EventBridge 目标

目标是一种资源或端点，当事件与为规则定义的事件模式匹配时，它会向其 EventBridge 发送事件。规则会处理事件数据，并将相关信息发送给目标， EventBridge 需要访问目标资源的权限。您最多可以为每条规则定义五个目标。

当您将目标添加到规则，之后该规则很快运行，可能不会立即调用新的或更新的目标。请稍等片刻，以便更改生效。

以下视频介绍了有关目标的基础知识：什么是目标

EventBridge 控制台中可用的目标

您可以在 EventBridge 控制台中为事件配置以下目标：

- API 目标
- API Gateway
- AWS AppSync;
- 批处理作业队列
- CloudWatch 日志组
- CodeBuild 项目
- CodePipeline
- Amazon EBS `CreateSnapshot` API 调用
- EC2 Image Builder
- EC2 `RebootInstances` API 调用
- EC2 `StopInstances` API 调用
- EC2 `TerminateInstances` API 调用
- ECS 任务
- 不同账户或区域中的事件总线
- 同一账户和区域中的事件总线
- Firehose 传输流
- Glue 工作流
- Incident Manager 响应计划

- Inspector 评估模板
- Kinesis 流
- Lambda 函数（异步）
- Amazon Redshift 集群数据 API 查询
- Amazon Redshift Serverless 工作组数据 API 查询
- SageMaker 管道
- Amazon SNS 主题

  EventBridge 不支持 Amazon SNS FIFO（先入先出）主题。
- Amazon SQS 队列
- Step Functions 状态机（异步）
- Systems Manager Automation
- Systems Manager OpsItem
- Systems Manager 运行命令

目标参数

有些目标不会将事件负载中的信息发送给目标，而是将事件视为调用特定 API 的触发器。 EventBridge 使用 Targe t 参数来确定该目标会发生什么。这些功能包括：

- API 目标（发送到 API 目标的数据必须与 API 的结构相匹配。必须使用 `InputTransformer` 对象来确保数据的结构正确。如果要包含原始事件负载，请在 `InputTransformer` 中引用它。）
- API Gateway（发送到 API Gateway 的数据必须与 API 的结构相匹配。必须使用 `InputTransformer` 对象来确保数据的结构正确。如果要包含原始事件负载，请在 `InputTransformer` 中引用它。）
- Amazon EC2 Image Builder
- `RedshiftDataParameters`（Amazon Redshift 数据 API 集群）
- `SageMakerPipelineParameters`（Amazon SageMaker 运行时模型构建管道）

> **Note**
>
> EventBridge 不支持所有 JSON Path 语法并在运行时对其进行评估。支持的语法包括：
>
> - 点表示法（例如 `$.detail`）

- 短划线
- 下划线
- 字母数字字符
- 数组索引
- 通配符 (*)

动态路径参数

一些目标参数支持可选的动态 JSON 路径语法。此语法允许您指定 JSON 路径而不是静态值（例如 `$.detail.state`）。整个值必须是 JSON 路径，不能仅仅是其中的一部分。例如，`RedshiftParameters.Sql` 可以是 `$.detail.state` 但不能是 `"SELECT * FROM $.detail.state"`。这些路径在运行时会动态替换为来自指定路径的事件负载本身的数据。动态路径参数不能引用新值或输入转换生成的转换后的值。动态参数 JSON 路径支持的语法与转换输入时的语法相同。有关更多信息，请参阅 ???。

动态语法可用于以下参数的所有字符串、非枚举字段：

- `EcsParameters`
- `HttpParameters`（`HeaderParameters` 键除外）
- `RedshiftDataParameters`
- `SageMakerPipelineParameters`

权限

要对您拥有的资源进行 API 调用，EventBridge 需要适当的权限。对于 AWS Lambda 和 Amazon SNS 资源，EventBridge 使用基于资源的策略。对于 EC2 实例、Kinesis 数据流和 Step Functions 状态机，EventBridge 使用您在`RoleARN`参数中指定的 IAM 角色。`PutTargets`您可以使用已配置的 IAM 授权调用 API Gateway 端点，但如果您尚未配置授权，该角色是可选的。有关更多信息，请参阅亚马逊 EventBridge 和 AWS Identity and Access Management。

如果另一账户在同一区域中，并且已授予您权限，您可以向该账户发送事件。有关更多信息，请参阅在 AWS 账户之间发送和接收 Amazon EventBridge 事件。

如果您的目标已加密，您必须在 KMS 密钥政策中包含以下部分。

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

EventBridge 目标细节

AWS Batch 作业队列

某些参数 AWS Batch `submitJob`可以通过进行配置BatchParameters。

其他参数可以在事件负载中指定。如果事件负载（通过或通过 InputTransformers）包含以下密钥，则它们将被映射到`submitJob`请求参数：

- `ContainerOverrides: containerOverrides`

  > **Note**
  >
  > 仅包括命令、环境、内存和 vCPU

- `DependsOn: dependsOn`

  > **Note**
  >
  > 仅包括 jobId

- `Parameters: parameters`

CloudWatch 日志组

如果您不使用InputTransformer带有 CloudWatch Logs 目标的，则事件负载将用作日志消息，将事件的源用作时间戳。如果您确实使用了 InputTransformer，则模板必须是：

```
{"timestamp":<timestamp>,"message":<message>}
```

EventBridge 对发送到日志流的条目进行批处理；因此， EventBridge 可能会将单个或多个事件传送到日志流，具体取决于流量。

CodeBuild 项目

如果您使用InputTransformers将输入事件调整为 Target 以匹配 CodeBuild StartBuildRequest结构，则参数将以 1 比 1 映射并传递到。`codeBuild.StartBuild`

Amazon ECS 任务

如果您使用InputTransformers将输入事件塑造成目标以匹配 Amazon ECS RunTask TaskOverride结构，则参数将以 1 比 1 的比分映射并传递到。`ecs.RunTask`

Incident Manager 响应计划

如果匹配的事件来自 CloudWatch 警报，则警报状态更改详细信息将填充到事件管理器 StartIncidentRequest 呼叫的触发器详细信息中。

配置目标

了解如何为 EventBridge 目标配置设置。

目标：

- API 目标
- 亚马逊的 EventBridge 目标 Amazon API Gateway
- AWS AppSync 亚马逊的目标 EventBridge
- HTTP 终端节点目标的连接
- 在 AWS 账户之间发送和接收 Amazon EventBridge 事件
- 在 AWS 区域之间发送和接收 Amazon EventBridge 事件
- 在同一账户和地区的 EventBridge 活动总线之间发送和接收 Amazon 事件

API 目标

Amazon EventBridge API 目标是您可以作为规则目标调用的 HTTP 终端节点，类似于调用 AWS 服务或资源作为目标的方式。使用 API 目标，您可以使用 API 调用在 AWS 服务、集成软件即服务 (SaaS) 应用程序和外部应用程序之间路由事件。 AWS 当您将 API 目标指定为规则的目标时，会为与规则中指定的事件模式匹配的任何事件 EventBridge 调用 HTTP 终端节点，然后随请求一起传送事件信息。使用 EventBridge，您可以对请求使用除 CONNECT 和 TRACE 之外的任何 HTTP 方法。最常用的 HTTP 方法是 PUT 和 POST。您也可以使用输入转换器，将事件自定义为特定 HTTP 端点参数的参数。有关更多信息，请参阅 亚马逊 EventBridge 输入转换。

API 目标不支持私有目标，例如接口 VPC 端点。有关更多信息，请参阅 ???。

> **Important**
>
> EventBridge 向 API 目标端点发出的请求的最大客户端执行超时时间必须为 5 秒。如果目标端点的响应时间超过 5 秒， EventBridge 则请求超时。 EventBridge 重试会将请求超时到重试策略上配置的最大值。默认情况下，上限为 24 小时和 185 次。达到重试次数上限后，会将事件发送到死信队列（如有）。否则，该事件将被丢弃。

以下视频演示了 API 目标的用法：使用 API 目标

本主题内容：

- 创建 API 目标
- 创建规则，向 API 目标发送事件
- API 目标的服务相关角色
- 向 API 目标发送的请求中的标头
- API 目标错误代码
- 调用率如何影响事件传送
- 向 API 目标发送 CloudEvents 事件
- API 目标合作伙伴

创建 API 目标

每个 API 目标都需要一个连接。连接 指定向 API 目标端点授权的授权类型和要使用的凭证。您可以选择现有连接，也可以在创建 API 目标的同时创建连接。有关更多信息，请参阅 ???。

使用 EventBridge 控制台创建 API 目的地

1. AWS 使用具有管理和 EventBridge 打开EventBridge控制台权限的账户登录。
2. 在左侧导航窗格中，选择 API 目标。
3. 向下滚动到 API 目标表，然后选择创建 API 目标。
4. 在创建 API 目标页面上，输入 API 目标的名称。您最多可以使用 64 个大写或小写字母、数字、点 (.)、短划线 (-) 或下划线 (_) 字符。

   在当前区域中，该名称对于您的账户必须是唯一的。
5. 为 API 目标输入描述。
6. 为 API 目标输入 API 目标端点。API 目标端点是事件的 HTTP 调用端点目标。用于此 API 目标的连接中包含的授权信息，用于对此端点进行授权。URL 必须使用 HTTPS。
7. 输入连接到 API 目标端点使用的 HTTP 方法。
8. （可选）在每秒调用速率限制字段中，输入每秒发送到 API 目标端点的调用次数上限。

   您设置的速率限制可能会影响事件的 EventBridge 交付方式。有关更多信息，请参阅 调用率如何影响事件传送。
9. 对于连接，执行以下操作之一：
   - 选择使用现有连接，然后选择要用于此 API 目标的连接。
   - 选择创建新连接，然后输入要创建的连接的详细信息。有关更多信息，请参阅连接。
10. 选择创建。

创建规则，向 API 目标发送事件

创建 API 目标后，可以将其选为某规则的目标。要使用 API 目标作为目标，您必须提供具有正确权限的 IAM 角色。有关更多信息，请参阅 ???。

选择 API 目标作为目标，是创建规则的一部分。

使用控制台创建规则，向 API 目标发送事件

1. 遵循???过程中的步骤。

2. 在该???步骤中，当系统提示您选择 API 目标作为目标类型时：

   a. 选择 EventBridge API 目的地。

   b. 请执行以下操作之一：

      - 选择使用现有 API 目标并选择现有 API 目标
      - 选择创建新 API 目标并指定必要的设置来定义新 API 目标。

        有关指定所需设置的更多信息，请参阅???。

   c. （可选）：要为事件指定标题参数，请在标题参数下选择添加标头参数。

      接下来，指定标头参数的键和值。

   d. （可选）：要为事件指定查询字符串参数，请在查询字符串参数下选择添加查询字符串参数。

      接下来，为查询字符串参数指定键和值。

3. 按照步骤完成规则的创建。

API 目标的服务相关角色

当您为 API 目标创建连接时，名AWS ServiceRoleForAmazonEventBridgeApiDestinations为的服务相关角色会添加到您的账户。 EventBridge 使用服务相关角色在 Secrets Manager 中创建和存储密钥。要向服务相关角色授予必要的权限，请将AmazonEventBridgeApiDestinationsServiceRolePolicy策略EventBridge附加到该角色。该策略授予的权限，仅限于角色与连接密钥交互所必需的权限，不包括其他权限。该角色只能与您账户中的连接进行交互，以管理密钥。

以下策略是 `AmazonEventBridgeApiDestinationsServiceRolePolicy`。

```
{
    "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/*"
        }
    ]
}
```

有关服务相关角色的更多信息，请参阅 IAM 文档中的使用服务相关角色。

以下 AWS 区域支持AmazonEventBridgeApiDestinationsServiceRolePolicy服务相关角色：

- 美国东部（弗吉尼亚州北部）
- 美国东部（俄亥俄州）
- 美国西部（北加利福尼亚）
- 美国西部（俄勒冈州）
- 非洲（开普敦）
- 亚太地区（香港）
- Asia Pacific (Mumbai)
- 亚太地区（大阪）
- 亚太地区（首尔）
- 亚太地区（新加坡）
- 亚太地区（悉尼）
- 亚太地区（东京）
- 加拿大（中部）
- 欧洲地区（法兰克福）
- 欧洲地区（爱尔兰）
- 欧洲地区（伦敦）
- 欧洲地区（米兰）
- 欧洲地区（巴黎）
- Europe (Stockholm)
- 欧洲地区（米兰）
- 南美洲（圣保罗）
- 中国（宁夏）
- 中国（北京）

向 API 目标发送的请求中的标头

以下部分详细介绍如何 EventBridge 处理发往 API 目标的请求中的 HTTP 标头。

向 API 目标发送的请求中包含的标头

除了为用于 API 目标的连接定义的授权标头外，每个请求中 EventBridge 还包括以下标头。

标头键	标头值
用户代理	亚马逊//EventBridgeApiDestinations
Content-Type	如果未指定自定义 “内容类型” 值，则 EventBridge 包括以下默认值作为 “内容类型”： application/json; charset=utf-8
Range	bytes=0-1048575
Accept-Encoding	gzip,deflate
Connection	close
内容长度	表示发送到接收方的实体正文的大小（以字节为单位）的实体标头。
Host	一个请求标头，指定发送请求的服务器的主机和端口号。

向 API 目标发送的请求中无法覆盖的标头

EventBridge 不允许您覆盖以下标题：

- 用户代理
- Range

从 API 目标请求中 EventBridge 移除标头

EventBridge 删除所有 API 目标请求的以下标头：

- A-IM
- Accept-Charset
- Accept-Datetime
- Accept-Encoding
- Cache-Control
- Connection
- Content-Encoding
- 内容长度
- Content-MD5
- Date
- Expect
- Forwarded
- From
- Host
- HTTP2-Settings
- If-Match
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Max-Forwards
- Origin
- Pragma
- Proxy-Authorization
- Range
- Referer
- TE
- Trailer
- Transfer-Encoding
- 用户代理

- Upgrade
- Via
- Warning

API 目标错误代码

当 EventBridge 尝试将事件传送到 API 目标时出现错误时， EventBridge会执行以下操作：

- 将重试与错误代码 409、429 和 5xx 关联的事件。
- 不会重试与错误代码 1xx、2xx、3xx 和 4xx（不包括 429）关联的事件。

EventBridge API 目的地会读取标准的 HTTP 响应标头，`Retry-After`以了解在发出后续请求之前需要等待多长时间。 EventBridge 在定义的重试策略和`Retry-After`标头之间选择更保守的值。如果`Retry-After`值为负，则 EventBridge 停止重试该事件的传送。

调用率如何影响事件传送

如果将每秒调用速率设置为远低于生成的调用数量的值，则可能无法在 24 小时的重试时间内传送此事件。例如，如果您将调用速率设置为每秒 10 次调用，但每秒会生成数千个事件，则很快就会有超过 24 小时的待传送事件积压。为确保不会丢失任何事件，请设置一个死信队列，将调用失败的事件发送到该队列，以便日后可以处理这些事件。有关更多信息，请参阅 事件重试策略和使用死信队列。

向 API 目标发送 CloudEvents 事件

CloudEvents 是针对事件格式的供应商中立规范，旨在提供跨服务、平台和系统的互操作性。您可以使用 EventBridge 在 AWS 服务事件发送到目标（例如 API 目的地） CloudEvents 之前将其转换为。

> **Note**
>
> 以下过程说明了如何将源事件转换为结构化模式。 CloudEvents在 CloudEvents 规范中，结构化模式消息是将整个事件（属性和数据）编码到事件的有效载荷中的消息。

有关 CloudEvents 规范的更多信息，请参阅 cloudeven ts.io。

使用控制台将 AWS 事件转换为 CloudEvents 格式

要将事件转换为传送到目标之前的 CloudEvents 格式，首先要创建事件总线规则。作为定义规则的一部分，在将变换事件发送到您指定的目标之前，您可以使用输入 EventBridge 转换器来处理变换事件。

1. 遵循???过程中的步骤。
2. 在该???步骤中，当系统提示您选择 API 目标作为目标类型时：

   a. 选择 EventBridge API 目的地。

   b. 请执行以下操作之一：

      - 选择使用现有 API 目标并选择现有 API 目标
      - 选择创建新 API 目标并指定必要的设置来定义新 API 目标。

      有关指定所需设置的更多信息，请参阅???。

   c. 为事件指定必需的 Content-Type 标头参数： CloudEvents

      - 在 “标题参数” 下，选择 “添加标题参数”。
      - 对于密钥，请指定`Content-Type`。

      对于值，请指定`application/cloudevents+json; charset=UTF-8`。
3. 为目标指定执行角色。
4. 定义输入转换器，将源事件数据转换为以下 CloudEvents 格式：

   a. 在 “其他设置” 下，对于 “配置目标输入”，选择 “输入变压器”。

   然后选择 “配置输入变压器”。

   b. 在目标输入变压器下，指定输入路径。

   在下面的输入路径中，region 属性是该 CloudEvents 格式的自定义扩展属性。因此，不要求遵守 CloudEvents 规范。

   CloudEvents 允许您使用和创建核心规范中未定义的扩展属性。有关更多信息，包括已知扩展属性的列表，请参阅CloudEvents 规范文档中的CloudEvents 扩展属性 GitHub。

   ```
   {
     "detail": "$.detail",
     "detail-type": "$.detail-type",
     "id": "$.id",
     "region": "$.region",
     "source": "$.source",
     "time": "$.time"
   }
   ```

   c. 在模板中，输入模板以将源事件数据转换为 CloudEvents 格式。

在下面的模板中，region并不是严格要求的，因为输入路径中的region属性是 CloudEvents 规范的扩展属性。

```
{
  "specversion":"1.0",
  "id":<id>,
  "source":<source>,
  "type":<detail-type>,
  "time":<time>,
  "region":<region>,
  "data":<detail>
}
```

5. 按照步骤完成规则的创建。

API 目标合作伙伴

使用以下 AWS 合作伙伴提供的信息为其服务或应用程序配置 API 目标和连接。

Confluent

API 目标调用端点 URL：

通常采用以下格式：

```
https://random-id.region.aws.confluent.cloud:443/kafka/v3/
clusters/cluster-id/topics/topic-name/records
```

有关更多信息，请参阅 Confluent 文档中的查找 REST 端点地址和集群 ID。

支持的授权类型：

基本

需要的其他授权参数：

不适用

Confluent 文档：

制作唱片

适用于 Apache Kafka 的融合 REST 代理

常用的 API 操作：

POST

其他信息：

要将事件数据转换为端点可以处理的消息，请创建目标输入转换器。

- 要在不指定 Kafka 分区密钥的情况下生成记录，请使用以下模板作为输入转换器。不需要输入路径。

```
{
  "value":{
      "type":"JSON",
      "data":aws.events.event.json
  },
}
```

- 要使用事件数据字段作为 Kafka 分区密钥生成记录，请按照下面的输入路径和模板示例进行操作。此示例定义了该orderId字段的输入路径，然后将该字段指定为分区键。

  首先，定义事件数据字段的输入路径：

```
{
  "orderId":"$.detail.orderId"
}
```

  然后，使用输入转换器模板将数据字段指定为分区键：

```
{
  "value":{
    "type":"JSON",
    "data":aws.events.event.json
  },
  "key":{
    "data":"<orderId>",
    "type":"STRING"
  }
}
```

Coralogix

API 目标调用端点 URL

有关端点的完整列表，请参阅 Coralogix API 参考。

支持的授权类型

API 密钥

需要的其他授权参数

标头 "x-amz-event-bridge-access-key"，值为 Coralogix API 密钥

Coralogix 文档

Amazon EventBridge 身份验证

常用的 API 操作

美国：https://ingress.coralogix.us/aws/event-bridge

新加坡：https://ingress.coralogixsg.com/aws/event-bridge

爱尔兰：https://ingress.coralogix.com/aws/event-bridge

斯德哥尔摩：https://ingress.eu2.coralogix.com/aws/event-bridge

印度：https://ingress.coralogix.in/aws/event-bridge

其他信息

这些事件存储为日志条目，`applicationName=[AWS Account]`、`subsystemName=[event.source]`。

Datadog

API 目标调用端点 URL

有关端点的完整列表，请参阅 Datadog API 参考。

支持的授权类型

API 密钥

需要的其他授权参数

无

Datadog 文档

身份验证

常用的 API 操作

POST https://api.datadoghq.com/api/v1/events

POST https://http-intake.logs.datadoghq.com/v1/input

其他信息

端点 URL 因您的 Datadog 组织所在的位置而异。如需了解您的组织的正确 URL，请参阅文档。

Freshworks

API 目标调用端点 URL

有关端点列表，请参阅 https://developers.freshworks.com/documentation/

支持的授权类型

基本、API 密钥

需要的其他授权参数

不适用

Freshworks 文档

身份验证

常用的 API 操作

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

其他信息

无

MongoDB

API 目标调用端点 URL

https://data.mongodb-api.com/app/*App ID*/endpoint/

支持的授权类型

API 密钥

电子邮件/密码

自定义 JWT 身份验证

需要的其他授权参数

无

MongoDB 文档

Atlas 数据 API

端点

自定义 HTTPS 端点

身份验证

常用的 API 操作

无

其他信息

无

New Relic

API 目标调用端点 URL

有关更多信息，请参阅我们的欧洲和美国地区数据中心。

事件

美国 - https://insights-collector.newrelic.com/v1/accounts/*YOUR_NEW_RELIC_ACCOUNT_ID*/events

欧洲 - https://insights-collector.eu01.nr-data.net/v1/accounts/*YOUR_NEW_RELIC_ACCOUNT_ID*/events

指标

美国 - https://metric-api.newrelic.com/metric/v1

欧洲 - https://metric-api.eu.newrelic.com/metric/v1

日志

美国 - https://log-api.newrelic.com/log/v1

欧洲 - https://log-api.eu.newrelic.com/log/v1

跟踪

美国 - https://trace-api.newrelic.com/trace/v1

欧洲 - https://trace-api.eu.newrelic.com/trace/v1

支持的授权类型

API 密钥

New Relic 文档

指标 API

事件 API

日志 API

跟踪 API

常用的 API 操作

指标 API

事件 API

日志 API

跟踪 API

其他信息

- 指标 API 限制
- 事件 API 限制
- 日志 API 限制
- 跟踪 API 限制

Operata

API 目标调用端点 URL：

https://api.operata.io/v2/aws/events/contact-record

支持的授权类型：

基本

需要的其他授权参数：

无

Operata 文档：

- 如何创建、查看、更改和撤销 API 令牌？
- 使用 Amazon EventBridge 调度器管道进行操作 AWS 集成

常用的 API 操作：

```
POST https://api.operata.io/v2/aws/events/contact-record
```

其他信息：

`username` 是 Operata 群组 ID，密码是您的 API 令牌。

Salesforce

API 目标调用端点 URL

Sobject— https *:// myDomainName.my.salesforce.com/services/data/ ### / sobjects//* SobjectEndpoint*

自定义平台事件— *https://myDomainName.my.salesforce.com/services/data/### /sobjects/ /* customPlatformEndpoint*

有关端点的完整列表，请参阅 Salesforce API 参考

支持的授权类型

OAuth 客户端凭证

当返回 401 或 407 响应时，会刷新 OAUTH 令牌。

需要的其他授权参数

Salesforce 连接的应用程序客户端 ID 和客户端密钥。

以下授权端点之一：

- 制作— https://*MyDomainName*.my.salesforce.com。 /services/oauth2/令牌
- 没有增强域名的沙箱 — https://***MyDomainName***— ***SandboxName***.my. salesforce.com/services /oauth2/token
- 带有增强域名的沙箱 — https://— .sandbox.my.salesforce.com/services/oa*MyDomainName*uth2/tok *SandboxName*en

以下键/值对：

密钥	值
grant_type	client_credentials

Salesforce 文档

REST API 开发人员指南

常用的 API 操作

使用对象元数据

使用记录

其他信息

有关说明如何使用 EventBridge 控制台创建连接Salesforce、API 目标以及将信息路由到的规则的教程Salesforce，请参阅???。

Slack

API 目标调用端点 URL

有关端点和其他资源的列表，请参阅使用 Slack Web API

支持的授权类型

OAuth 2.0

当返回 401 或 407 响应时，会刷新 OAUTH 令牌。

当您创建 Slack 应用程序并将其安装到工作区时，将代表您创建一个 OAuth 持有者令牌，用于通过您的 API 目标连接对调用进行身份验证。

需要的其他授权参数

不适用

Slack 文档

基本应用设置

使用 OAuth 进行安装

检索消息

发送消息

使用传入 Webhook 发送消息

常用的 API 操作

https://slack.com/api/chat.postMessage

其他信息

配置 EventBridge 规则时，有两种配置需要突出显示：

- 包括一个标头参数，将内容类型定义为“application/json;charset=utf-8”。
- 使用输入转换器将输入事件映射到 Slack API 的预期输出，即确保发送到 Slack API 的负载具有“channel”和“text”键/值对。

Shopify

API 目标调用端点 URL

有关端点列表以及其他资源和方法，请参阅端点和请求

支持的授权类型

OAuth、API 密钥

> **Note**
>
> 当返回 401 或 407 响应时，会刷新 OAUTH 令牌。

需要的其他授权参数

不适用

Shopify 文档

身份验证和授权概述

常用的 API 操作

POST - /admin/api/2022-01/products.json

GET - admin/api/2022-01/products/{product_id}.json

PUT - admin/api/2022-01/products/{product_id}.json

DELETE - admin/api/2022-01/products/{product_id}.json

其他信息

创建应用

亚马逊 EventBridge webhook 配送

以 Shopify 管理员身份使用自定义应用的访问令牌

产品

Shopify 管理员 API

Splunk

API 目标调用端点 URL

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

支持的授权类型

基本、API 密钥

需要的其他授权参数

无

Splunk 文档

对于这两种授权类型，都需要 HEC 令牌 ID。有关更多信息，请参阅在 Splunk Web 中设置和使用 HTTP 事件收集器。

常用的 API 操作

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

其他信息

API 密钥 — 为配置终端节点时 EventBridge，API 密钥名称为 “授权”，值为 Splunk HEC 令牌 ID。

基本（用户名/密码）— 为配置终端节点时 EventBridge，用户名为 “Splunk”，密码为 Splunk HEC 令牌 ID。

Sumo Logic

API 目标调用端点 URL

每名用户的 HTTP 日志和指标源端点 URL 会有所不同。有关更多信息，请参阅 HTTP 日志和指标源。

支持的授权类型

Sumo Logic 不需要对他们的 HTTP 源进行身份验证，因为 URL 中内置了唯一的密钥。因此，您应确保将此 URL 视为机密。

配置 EventBridge API 目标时，需要授权类型。要满足此要求，请选择 API 密钥并为其指定密钥名称“dummy-key”和密钥值“dummy-value”。

需要的其他授权参数

不适用

Sumo Logic 文档

Sumo Logic已经建立了托管源来收集来自许多 AWS 服务的日志和指标，你可以使用他们网站上的信息来处理这些来源。有关更多信息，请参阅 Amazon Web Services。

如果您要从应用程序生成自定义事件，并希望将其Sumo Logic作为日志或指标发送到，请使用 EventBridge API 目标以及 Sumo Logic HTTP 日志和指标源端点。

- 要注册并创建免费 Sumo Logic 实例，请参阅立即开始免费试用。
- 有关使用 Sumo Logic 的更多信息，请参阅 HTTP 日志和指标源。

常用的 API 操作

POST https://endpoint4.collection.us2.sumologic.com/receiver/v1/http/*UNIQUE_ID_PER_COLLECTOR*

其他信息

无

TriggerMesh

API 目标调用端点 URL

使用 HTTP 事件源主题中的信息来确定端点 URL。端点 URL 包括事件源名称和用户命名空间，格式如下：

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

要在请求中包含端点的基本授权参数。

支持的授权类型

基本

需要的其他授权参数

无

TriggerMesh 文档

HTTP 事件源

常用的 API 操作

不适用

其他信息

无

Zendesk

API 目标调用端点 URL

https://developer.zendesk.com/rest_api/docs/support/tickets

支持的授权类型

基本、API 密钥

需要的其他授权参数

无

Zendesk 文档

安全和认证

常用的 API 操作

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

其他信息

API 请求 EventBridge 会计入你的 Zendesk API 限制。有关您的计划的 Zendesk 限制信息，请参阅使用限制。

为了更好地保护您的账户和数据，我们建议使用 API 密钥，而不是基本的登录凭证身份验证。

亚马逊的 EventBridge 目标 Amazon API Gateway

您可以使用 Amazon API Gateway 创建、发布、维护和监控 API。亚马逊 EventBridge支持向 API Gateway 终端节点发送事件。当您将 API Gateway 端点指定为目标时，发送到该目标的每个事件都将映射一个发送到该端点的请求。

> **Important**
>
> EventBridge 支持使用 API Gateway 边缘优化和区域端点作为目标。目前不支持私有端点。要了解端点的更多信息，请参阅 https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-api-endpoint-types.html。

可以将 API Gateway 目标用于以下使用场景：

- 根据第三方事件调用 API Gateway 中托管的客户指定 AWS 的 API。
- 按计划定期调用端点。

EventBridge JSON 事件信息将作为 HTTP 请求的正文发送到您的终端节点。您可以在目标的 `HttpParameters` 字段中指定其他请求属性，如下所示：

- `PathParameterValues` 列出了与您的端点 ARN 中的任何路径变量顺序对应的值，例如 `"arn:aws:execute-api:us-east-1:112233445566:myapi/dev/POST/pets/*/*"`。
- `QueryStringParameters`表示 EventBridge 附加到调用端点的查询字符串参数。
- `HeaderParameters` 定义要添加到请求中的 HTTP 标头。

> **Note**
>
> 出于安全考虑，不允许使用以下 HTTP 标头密钥：
>
> - 任何以 `X-Amz` 或 `X-Amzn` 为前缀的标头密钥
> - `Authorization`
> - `Connection`
> - `Content-Encoding`
> - `Content-Length`
> - `Host`
> - `Max-Forwards`
> - `TE`
> - `Transfer-Encoding`
> - `Trailer`
> - `Upgrade`
> - `Via`
> - `WWW-Authenticate`
> - `X-Forwarded-For`

动态参数

在调用 API Gateway 目标时，您可以向发送到目标的事件动态添加数据。有关更多信息，请参阅 the section called “目标参数”。

调用重试

与所有目标一样， EventBridge 会重试一些失败的调用。对于 API Gateway， EventBridge 重试以 5xx 或 429 HTTP 状态码发送的响应长达 24 小时，并出现指数级退缩和抖动。之后，在 Amazon 上 EventBridge 发布一个`FailedInvocations`指标 CloudWatch。 EventBridge 不会重试其他 4xx HTTP 错误。

超时

EventBridge 规则 API Gateway 请求的最大客户端执行超时时间必须为 5 秒。如果 API Gateway 的响应 EventBridge 时间超过 5 秒，则会超时请求然后重试。

EventBridge Pipes API Gateway 请求的最大超时时间为 29 秒，这是 API Gateway 的最大超时值。

AWS AppSync 亚马逊的目标 EventBridge

AWS AppSync 使开发人员能够使用安全、无服务器和高性能的 GraphQL 和 Pub/Sub API 将其应用程序和服务连接到数据和事件。借助 AWS AppSync，您可以通过 GraphQL 突变将实时数据更新发布到您的应用程序。 EventBridge 支持为匹配的事件调用有效的 GraphQL 突变操作。当你将 AWS AppSync API 突变指定为目标时，会通过突变操作 AWS AppSync 处理事件，然后突变操作可以触发与该突变相关的订阅。

> **Note**
>
> EventBridge 支持 AWS AppSync 公共 GraphQL 接口。 EventBridge 目前不支持 AWS AppSync 私有 API。

您可以将 AWS AppSync GraphQL API 目标用于以下用例：

- 推送、转换事件数据以及将其存储到您配置的数据源中。
- 向连接的应用程序客户端发送实时通知。

> **Note**
>
> AWS AppSync 目标仅支持使用AWS_IAM授权类型调用 AWS AppSync GraphQL API。

有关 AWS AppSync GraphQL API 的更多信息，请参阅《开发者指南》中的 G raphQL 和 AWS AppSync 架构。AWS AppSync

使用控制台为 EventBridge 规则指定 AWS AppSync 目标

1. 创建或编辑规则。
2. 在目标下，通过依次选择 AWS 服务和 AWS AppSync 来指定目标。
3. 指定要解析和执行的突变操作以及选择集。
   - 选择 AWS AppSync API，然后选择要调用的 GraphQL API 突变。
   - 在配置参数和选择集下，选择使用键值映射或输入转换器创建选择集。

     Key-value mapping

     要使用键值映射来创建选择集，请执行以下操作：
     - 为 API 参数指定变量。每个变量可以是静态值，也可以是事件负载的动态 JSON 路径表达式。
     - 在选择集下，选择要包含在响应中的变量。

     Input transformer

     要使用输入转换器创建选择集，请执行以下操作：
     - 指定用于定义要使用的变量的输入路径。
     - 指定输入模板来定义要传递给目标的信息并为其设置格式。

     有关更多信息，请参阅 ???。
4. 对于执行角色，选择是创建新的角色还是使用现有角色。
5. 完成规则的创建或编辑。

示例：Amazon 的 AWS AppSync 目标 EventBridge

在以下示例中，我们将介绍如何为 EventBridge 规则指定 AWS AppSync 目标，包括定义输入转换以格式化要交付的事件。

假设你有一个 AWS AppSync GraphQL APIEc2EventAPI，由以下架构定义：

```
type Event {
    id: ID!
    statusCode: String
    instanceId: String
}

type Mutation {
    pushEvent(id: ID!, statusCode: String!, instanceId: String): Event
}

type Query {
    listEvents: [Event]
}

type Subscription {
    subscribeToEvent(id: ID, statusCode: String, instanceId: String): Event
        @aws_subscribe(mutations: ["pushEvent"])
}
```

使用此 API 的应用程序客户端可以订阅由 pushEvent 突变触发的 subscribeToEvent 订阅。

你可以创建一个带有通过pushEvent突变向 AppSync API 发送事件的目标的 EventBridge 规则。当调用突变时，任何订阅的客户端都将收到该事件。

要将此 API 指定为 EventBridge 规则的目标，您需要执行以下操作：

1. 将规则目标的 Amazon 资源名称（ARN）设置为 Ec2EventAPI API 的 GraphQL 端点 ARN。
2. 将突变 GraphQL 操作指定为目标参数：

```
mutation CreatePushEvent($id: ID!, $statusCode: String, $instanceId: String) {
  pushEvent(id: $input, statusCode: $statusCode, instanceId: $instanceId) {
    id
    statusCode
    instanceId
  }
}
```

   您的突变选择集必须包含您希望在 GraphQL 订阅中订阅的所有字段。
3. 配置输入转换器以指定在操作中如何使用来自匹配事件的数据。

假设您选择了“EC2 Instance Launch Successful”示例事件：

```
{
  "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": ["arn:aws:autoscaling:us-
east-1:123456789012:autoScalingGroup:eb56d16b-bbf0-401d-b893-
d5978ed4a025:autoScalingGroupName/sampleLuanchSucASG", "arn:aws:ec2:us-
east-1:123456789012:instance/i-b188560f"],
  "detail": {
    "StatusCode": "InProgress",
    "AutoScalingGroupName": "sampleLuanchSucASG",
    "ActivityId": "9cabb81f-42de-417d-8aa7-ce16bf026590",
    "Details": {
      "Availability Zone": "us-east-1b",
      "Subnet ID": "subnet-95bfcebe"
    },
    "RequestId": "9cabb81f-42de-417d-8aa7-ce16bf026590",
    "EndTime": "2015-11-11T21:31:47.208Z",
    "EC2InstanceId": "i-b188560f",
    "StartTime": "2015-11-11T21:31:13.671Z",
    "Cause": "At 2015-11-11T21:31:10Z a user request created an AutoScalingGroup
 changing the desired capacity from 0 to 1.  At 2015-11-11T21:31:11Z an instance was
 started in response to a difference between desired and actual capacity, increasing
 the capacity from 0 to 1."
  }
}
```

您可以使用目标输入转换器的输入路径，定义以下变量供模板使用：

```
{
  "id": "$.id",
  "statusCode": "$.detail.StatusCode",
  "EC2InstanceId": "$.detail.EC2InstanceId"
}
```

编写输入转换器模板以定义 EventBridge 传递给 AWS AppSync 突变操作的变量。模板的计算结果必须为 JSON。根据我们的输入路径，您可以编写以下模板：

```
{
  "id": <id>,
  "statusCode": <statusCode>,
  "instanceId": <EC2InstanceId>
}
```

HTTP 终端节点目标的连接

连接定义用于连接到给定 HTTP 端点的授权方法和凭据。 EventBridge 当您配置授权设置并创建连接时，它会在中创建一个密钥 AWS Secrets Manager 来安全地存储授权信息。您还可以根据自己的 HTTP 终端节点目标添加要包含在连接中的其他参数。

通过以下方式使用连接：

- API 目标

  创建 API 目标时，要指定用于该目标的连接。您可以从账户中选择现有连接，也可以在创建 API 目标时创建连接。

连接的授权方法

EventBridge 连接支持以下授权方法：

- 基本
- API 密钥

  对于基本和 API 密钥授权，将为您 EventBridge 填充所需的授权标头。

- OAuth

  要获得 OAuth 授权， EventBridge 还要将您的客户端 ID 和密钥交换为访问令牌，然后对其进行安全管理。

  当返回 401 或 407 响应时，会刷新 OAUTH 令牌。

创建连接时，还可以包括授权端点所需的标头、正文和查询参数。如果对多个 HTTP 端点的授权相同，则可以将同一个连接用于多个 HTTP 端点。

当您创建连接并添加授权参数时， EventBridge 会在中创建密钥 AWS Secrets Manager。存储和访问 Secrets Manager 密钥的成本包含在使用 API 目标的费用中。要详细了解在 API 目标中使用密钥的最佳实践，请参阅CloudFormation 用户指南AWS::Events::ApiDestination中的。

> **Note**
>
> 要成功创建或更新连接，必须使用具有 Secrets Manager 使用权限的账户。所需的权限包含在 AmazonEventBridgeFullAccess 策略 中。在您的账户中为该连接创建的服务相关角色将获得相同的权限。

为 HTTP 终端节点目标创建连接

使用 EventBridge 控制台创建用于 HTTP 端点的连接

1. AWS 使用具有管理和 EventBridge 打开EventBridge 控制台权限的账户登录。
2. 在左侧导航窗格中，选择 API 目标。
3. 向下滚动到 API 目标表，然后选择连接选项卡。
4. 选择创建连接。
5. 在创建连接页面上，为连接输入连接名称。
6. 为连接输入描述。
7. 在授权类型中，选择授权类型，用于为使用此连接的 API 目标指定的 HTTP 端点授权连接。请执行以下操作之一：
   - 选择基本(用户名/密码)，然后输入用于授权 HTTP 端点的用户名和密码。
   - 选择 OAuth 客户端证书，然后输入用于对端点进行授权的授权端点、HTTP 方法、客户端 ID 和客户密钥。

     在 OAuth Http 参数下，添加要包含的所有其他参数，以便使用授权端点进行授权。从下拉列表中选择一个参数，然后输入键和值。要包括其他参数，请选择添加参数。

     在调用 Http 参数下，添加要包含在授权请求中的任何其他参数。要添加参数，从下拉列表中选择一个参数，然后输入键和值。要包括其他参数，请选择添加参数。
   - 选择 API 密钥，然后输入用于 API 密钥授权的 API 密钥名称和关联值。

在调用 Http 参数下，添加要包含在授权请求中的任何其他参数。要添加参数，从下拉列表中选择一个参数，然后输入键和值。要包括其他参数，请选择添加参数。

8. 选择创建。

使用 EventBridge 控制台编辑连接

您可以编辑现有连接。

使用 EventBridge 控制台编辑连接

1. AWS 使用具有管理和 EventBridge 打开EventBridge 控制台权限的账户登录。
2. 在左侧导航窗格中，选择 API 目标。
3. 向下滚动到 API 目标表，然后选择连接选项卡。
4. 在连接表中，选择要编辑的连接。
5. 在连接详细信息页面上，选择编辑。
6. 更新连接的值，然后选择更新。

使用控制台取消对连接的授权 EventBridge

取消对连接的授权后，会删除所有授权参数。删除授权参数会从连接中删除密钥，因此您无需创建新连接即可重复使用该密钥。

> **Note**
>
> 您必须更新使用取消授权连接的任何 HTTP 终端节点，才能使用不同的连接成功向 HTTP 端点发送请求。

取消对连接的授权

1. AWS 使用具有管理和 EventBridge 打开EventBridge 控制台权限的账户登录。
2. 在左侧导航窗格中，选择 API 目标。
3. 向下滚动到 API 目标表，然后选择连接选项卡。
4. 在连接表中，选择连接。
5. 在连接详细信息页面上，选择取消授权。

6. 在取消对连接的授权? 对话框中，输入连接的名称，然后选择取消授权。

连接状态将更改为正在取消授权，直到该过程完成。然后，状态更改为已取消授权。现在，您可以编辑连接，添加新的授权参数。

在 AWS 账户之间发送和接收 Amazon EventBridge 事件

您可以配置 EventBridge 为在 AWS 账户中的事件总线之间发送和接收事件。当您配置 EventBridge 为在账户之间发送或接收事件时，可以指定哪些 AWS 账户可以向您的账户中的事件总线发送事件或从事件总线接收事件。您还可以允许或拒绝事件，它们来自与事件总线关联的特定规则，或来自特定来源的事件。有关更多信息，请参阅使用 Amazon EventBridge 资源策略简化跨账户访问

> **Note**
>
> 如果使用 AWS Organizations，则可以指定组织并向该组织中的所有账户授予访问权限。此外，在向其他账户发送事件时，发送事件总线必须附加 IAM 角色。有关更多信息，请参阅《AWS Organizations用户指南》中的什么是AWS Organizations 。

> **Note**
>
> 如果您使用 Incident Manager 响应计划作为目标，则在默认情况下，与您的账户共享的所有响应计划都可用。

只要目标区域是支持的跨区域目标区域，您就可以在所有区域的同一区域的账户中的事件总线之间以及不同区域的账户之间发送和接收事件。 AWS

配置 EventBridge 为向不同账户中的事件总线发送事件或从事件总线接收事件的步骤包括以下内容：

- 在接收者账户上，编辑事件总线上的权限，以允许指定 AWS 账户、组织或所有 AWS 账户向接收者账户发送事件。
- 在发送方 账户中，设置一个或多个将接收方账户的事件总线作为目标的规则。

  如果发送者账户继承了从 AWS 组织发送事件的权限，则发送者账户还必须具有 IAM 角色，其策略使其能够向接收者账户发送事件。如果您使用创建 AWS Management Console 以接收者账户中的事件总线为目标的规则，则该角色将自动创建。如果使用 AWS CLI，则必须手动创建角色。
- 在接收方 账户中，设置一个或多个匹配来自发送方账户的事件的规则。

从一个账户发送到另一个账户的事件将作为自定义事件向发送账户收取费用。不向接收账户收费。有关更多信息，请参阅 Amazon EventBridge 定价。

如果接收方账户设置了一条将从发送方账户接收的事件发送到第三个账户的规则，则这些事件不会发送到第三个账户。

以下视频讲述了账户之间的路由事件：将事件路由到其他 AWS 账户中的公交车

授予权限以允许来自其他 AWS 账户的事件

要接收来自其他账户或组织的事件，您必须先编辑将要接收事件的事件总线的权限。默认事件总线接受来自 AWS 服务、其他授权 AWS 账户和`PutEvents`呼叫的事件。需使用附加到事件总线的基于资源的策略来授予或拒绝事件总线的权限。在该策略中，您可以使用账户 ID 向其他 AWS 账户授予权限，也可以向使用 AWS 组织 ID 的组织授予权限。要了解有关事件总线权限的更多信息，包括示例策略，请参阅 Amazon EventBridge 事件总线的权限。

> **Note**
>
> EventBridge 现在要求所有新的跨账户事件总线目标添加 IAM 角色。这仅适用于 2023 年 3 月 2 日之后创建的事件总线目标。在该日期之前创建的没有 IAM 角色的应用程序不受影响。但是，我们建议添加 IAM 角色，授予用户访问另一账户中资源的权限，因为这样可以确保使用服务控制策略 (SCP) 应用组织边界，从而确定谁可以从您的组织中的账户发送和接收事件。

> **Important**
>
> 如果您选择接收来自所有 AWS 账户的事件，请谨慎创建规则，使其仅匹配要从其他账户接收的事件。要创建更安全的规则，请确保每个规则的事件模式都包含一个 `Account` 字段，其中包含要从其接收事件的一个或多个账户的账户 ID。其事件模式包含“账户”字段的规则与从在 `Account` 字段中未列出的账户发送的事件不匹配。有关更多信息，请参阅 亚马逊 EventBridge 活动。

AWS 账户间事件规则

如果您的账户设置为接收来自其他 AWS 账户的事件总线的事件，则可以编写与这些事件相匹配的规则。将规则的事件模式设置为与您从其他账户的事件总线接收的事件相匹配。

除非您在规则的事件模式中指定 account，否则您的账户中与您从其他账户事件总结收到的事件进行匹配的任何规则（包括新规则和现有规则）都基于这些事件触发。如果您要从另一账户的事件总线接收事件，并且希望仅对从您自己的账户生成的事件模式触发规则，则必须添加 account 并将您自己的账户 ID 指定为规则的事件模式。

如果您将账户设置为在所有 AWS 账户中接受来自活动总线的事件，我们强烈建议您在 AWS 账户中的每 EventBridge 条规则中添加account相应规则。这样可以防止您的账户中的规则触发来自未知 AWS 账户的事件。在规则中指定 account 字段时，可以在该字段中指定多个 AWS 账户的账户 ID。

要在您已授予权限的 AWS 账户中任何事件总线的匹配事件上触发规则，请不要在规则account字段中指定*。这样做不会匹配任何事件，因为 * 从不显示在事件的 account 字段中。相反，只需忽略规则的 account 字段即可。

创建在 AWS 账户之间发送事件的规则

将其他账户中的事件总线指定为目标，是创建规则的一部分。

使用控制台创建向其他 AWS 账户发送事件的规则

1. 遵循???过程中的步骤。
2. 在???步骤中，当系统提示您选择目标类型时：
   a. 选择EventBridge 活动总线。
   b. 选择不同账户或区域中的事件总线。
   c. 对于事件总线作为目标，请输入要使用的事件总线的 ARN。
3. 根据过程中的步骤，完成规则创建。

在 AWS 区域之间发送和接收 Amazon EventBridge 事件

您可以配置 EventBridge 为在 AWS 区域之间发送和接收事件。您还可以允许或拒绝事件，它们来自特定区域、与事件总线关联的特定规则或特定来源。有关更多信息，请参阅 Ama zon 引入跨区域事件路由 EventBridge

以下区域是支持的目标区域：

- 非洲（开普敦）区域
- 亚太地区（香港）区域
- Asia Pacific（Tokyo）Region

- 亚太地区（首尔）区域
- 亚太地区（大阪）区域
- 亚太地区（孟买）区域
- 亚太地区（新加坡）区域
- 亚太地区（悉尼）区域
- 加拿大（中部）区域
- 欧洲地区（法兰克福）区域
- 欧洲地区（斯德哥尔摩）区域
- 欧洲地区（米兰）
- 欧洲地区（爱尔兰）区域
- 欧洲地区（伦敦）区域
- 欧洲地区（巴黎）区域
- 中东（阿联酋）区域
- 中东（巴林）区域
- 南美洲（圣保罗）区域
- US East (N. Virginia) Region
- 美国东部（俄亥俄州）区域
- 美国西部（北加利福尼亚）区域
- 美国西部（俄勒冈州）区域
- 亚太地区（雅加达）区域
- 亚太地区（墨尔本）区域
- 以色列（特拉维夫）区域

以下视频介绍了使用 https://console.aws.amazon.com/events/、和 AWS Serverless Application Model：跨区域事件路由 AWS CloudFormation，在区域之间路由事件

创建将事件发送到其他 AWS 区域的规则

将另一个 AWS 区域的事件总线指定为目标是创建规则的一部分。

使用控制台创建向其他 AWS 账户发送事件的规则

1. 遵循???过程中的步骤。
2. 在???步骤中，当系统提示您选择目标类型时：
   a. 选择EventBridge 活动总线。
   b. 选择不同账户或区域中的事件总线。
   c. 对于事件总线作为目标，请输入要使用的事件总线的 ARN。
3. 根据过程中的步骤，完成规则创建。

在同一账户和地区的 EventBridge 活动总线之间发送和接收 Amazon 事件

您可以配置 EventBridge 为在同一 AWS 账户和区域的事件总线之间发送和接收事件。

当您配置 EventBridge 为在事件总线之间发送或接收事件时，您可以在发送者事件总线上使用 IAM 角色向发送者事件总线授予向接收器事件总线发送事件的权限。您可以在接收方 事件总线上使用基于资源的策略，授予接收方 事件总线权限，接收来自发送方 事件总线的事件。您还可以允许或拒绝事件，它们来自特定事件总线、与事件总线关联的特定规则或特定来源。有关事件总线权限的更多信息，包括示例策略，请参阅 Amazon EventBridge 事件总线的权限。

配置 EventBridge 为向您的账户中的事件总线发送事件或在事件总线之间接收事件的步骤包括以下内容：

- 要使用现有 IAM 角色，您需要向发送方事件总线授予对接收方事件总线的权限，或向接收方事件总线授予对发送方事件总线的权限。
- 在发送方 事件总线中，设置一个或多个规则，将接收方事件总线作为目标并创建 IAM 角色。如需应附加到该角色的策略示例，请参阅???。
- 在接收方 事件总线上，编辑权限，允许从另一个事件总线传递事件。
- 在接收方 事件中，设置一个或多个规则，匹配来自发送方事件总线的事件。

> **Note**
>
> EventBridge 无法将从发送者事件总线接收的事件路由到第三条事件总线。

从一条事件总线发送到另一条事件总线的事件，按自定义事件收费。有关更多信息，请参阅 Amazon EventBridge 定价。

创建规则，将事件发送到同一 AWS 账户和区域中的不同事件总线

要将事件发送到另一事件总线，您需要创建一个规则，以该事件总线作为目标。将同一 AWS 账户和区域中的事件总线指定为目标是创建规则的一部分。

使用控制台创建规则，将事件发送到同一 AWS 账户和区域中的不同事件总线

1. 遵循???过程中的步骤。
2. 在???步骤中，当系统提示您选择目标类型时：
   a. 选择EventBridge 活动总线。
   b. 在同一个 AWS 账户和地区中选择事件总线。
   c. 对于事件总线作为目标，从下拉列表中选择一个事件总线。
3. 根据过程中的步骤，完成规则创建。

亚马逊 EventBridge 输入转换

在将信息 EventBridge传递给规则目标之前，您可以自定义事件中的文本。可以使用控制台或 API 中的输入转换器定义变量，使用 JSON 路径来引用原始事件源中的值。转换后的事件将发送到目标，而不是原始事件。但是，动态路径参数必须引用原始事件，而不是转换后的事件。您最多可以定义 100 个变量，同时从输入中为每个变量分配一个值。然后，您可以在输入模板 中以 <*variable-name*> 形式使用这些变量。

使用输入转换器的教程请参阅???。

> **Note**
>
> EventBridge 不支持所有 JSON Path 语法并在运行时对其进行评估。支持的语法包括：
>
> - 点表示法（例如 `$.detail`）
> - 短划线
> - 下划线
> - 字母数字字符
> - 数组索引
> - 通配符 (*)

本主题内容：

预定义变量

您可以在不定义 JSON 路径的情况下使用预定义的变量。这些变量是保留的，您不能使用这些名称创建变量。

- `aws.events.rule-arn`— 规则的亚马逊资源名称 (ARN)。 EventBridge
- `aws.events.rule-name`— EventBridge 规则的名称。
- `aws.events.event.ingestion-time`— 接收事件的时间 EventBridge。这是 ISO 8601 时间戳。此变量由生成 EventBridge ，无法覆盖。
- `aws.events.event` - 作为 JSON 的原始事件负载（不带 `detail` 字段）。只能用作 JSON 字段的值，因为其内容不会被转义。
- `aws.events.event.json` - 作为 JSON 的完整原始事件负载。（带 `detail` 字段）。只能用作 JSON 字段的值，因为其内容不会被转义。

输入转换示例

以下是 Amazon EC2 事件示例。

```
{
  "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"
  }
}
```

在控制台中定义规则时，选择配置输入下的输入转换器选项。此选项显示两个文本框：一个用于 Input Path (输入路径) ，一个用于 Input Template (输入模板)。

输入路径 用于定义变量。使用 JSON 路径来引用事件中的项目，并将这些值存储在变量中。例如，您可以通过在第一个文本框中输入以下内容来创建一个输入路径，以引用示例事件中的值。您也可以使用方括号和索引从数组中获取项目。

> **Note**
>
> EventBridge 在运行时替换输入转换器以确保有效的 JSON 输出。因此，请在引用 JSON 路径参数的变量前后加引号，但不要在引用 JSON 对象或数组的变量前后加引号。

```
{
  "timestamp" : "$.time",
  "instance" : "$.detail.instance-id",
  "state" : "$.detail.state",
  "resource" : "$.resources[0]"
}
```

这些代码会定义四个变量，<timestamp>、<instance>、<state> 和 <resource>。您可以在创建输入模板 时引用这些变量。

输入模板 是您要传递给目标的信息的模板。您可以创建将字符串或 JSON 传递到目标的模板。使用上一个事件和输入路径，以下输入模板 示例将事件转换为示例输出，然后再将其路由到目标。

描述	模板	输出
简单字符串	`"instance <instance> is in <state>"`	`"instance i-0123456789 is in RUNNING"`
带转义引号的字符串	`"instance \"<instance>\" is in <state>"`	`"instance \"i-0123456789\" is in RUNNING"` 请注意，这是 EventBridge 控制台中的行为。 AWS CLI 对斜杠字符进行转义，结果为 `"instance "i-0123456789" is in RUNNING"` 。
简单的 JSON	`{ "instance" : <instance>, "state": <state>`	`{ "instance" : "i-0123456789", "state": "RUNNING"`

描述	模板	输出
	} 	}

带有字符串和变量的 JSON	{
	 "instance" : <instance
	>,
	 "state": "<state>",
	 "instanceStatus":
	 "instance \"<instance>
	\" is in <state>"
	}
		{
		 "instance" : "i-012345
		6789",
		 "state": "RUNNING",
		 "instanceStatus":
		 "instance \"i-01234
		56789\" is in RUNNING"
		}

混合变量和静态信息的 JSON	{
	 "instance" :
	<instance>,
	 "state": [9, <state>,
	true],
	 "Transformed" : "Yes"
	}
		{
		 "instance" :
		"i-0123456789",
		 "state": [
		 9,
		 "RUNNING",
		 true
		],
		 "Transformed" : "Yes"
		}

在 JSON 中包含保留变量	{
	 "instance" :
	<instance>,
	 "state": <state>,
	 "ruleArn" : <aws.even
	ts.rule-arn>,
	 "ruleName" :
	<aws.events.rule-n
	ame>,
	 "originalEvent" :
	<aws.events.event.
	json>
	}
		{
		 "instance" :
		"i-0123456789",
		 "state": "RUNNING",
		 "ruleArn" : "arn:aws:
		events:us-east-2:1
		23456789012:rule/e
		xample",
		 "ruleName" :
		"example",
		 "originalEvent" : {
		 ... // commented for
		brevity
		 }
		}

描述	模板	输出
在字符串中包含保留变量	`"<aws.events.rule-name> triggered"`	`"example triggered"`
亚马逊 CloudWatch 日志组	`{ "timestamp" : <timestamp>, "message": "instance \"<instance>\" is in <state>" }`	`{ "timestamp" : 2015-11-11T21:29:54Z, "message": "instance "i-0123456789" is in RUNNING }`

使用 EventBridge API 转换输入

有关使用 EventBridge API 转换输入的信息，请参阅使用输入转换器从事件中提取数据并将该数据输入到目标。

使用转换输入 AWS CloudFormation

有关使用 AWS CloudFormation 转换输入的信息，请参见AWS::Events::Rule InputTransformer。

转换输入的常见问题

以下是转换输入时的一些常见问题 EventBridge：

- 对于字符串，需要引号。
- 为模板创建 JSON 路径时不进行验证。
- 如果您指定一个变量，匹配在事件中不存在的 JSON 路径，则不会创建该变量，并且不会在输出中显示该变量。
- 像 `aws.events.event.json` 这样的 JSON 属性只能用作 JSON 字段的值，不能在其他字符串中内联。
- EventBridge 填充目标的输入模板时，不会转义输入路径提取的值。
- 如果 JSON 路径引用了 JSON 对象或数组，但变量是在字符串中引用的，则 EventBridge 删除所有内部引号以确保字符串有效。例如，对于<detail>指向的变量`$.detail`，“Detail is<detail>” 将导致从对象中 EventBridge 删除引号。

因此，如果要根据单个 JSON 路径变量输出 JSON 对象，则必须将其作为键。在本示例中，`{"detail": <detail>}`。

- 表示字符串的变量不需要引号。允许使用它们，但在转换过程中 EventBridge 会自动为字符串变量值添加引号，以确保转换输出是有效的 JSON。 EventBridge 不会为表示 JSON 对象或数组的变量添加引号。不要为表示 JSON 对象或数组的变量添加引号。

  例如，以下输入模板包含的变量表示字符串和 JSON 对象：

  ```
  {
    "ruleArn" : <aws.events.rule-arn>,
    "ruleName" : <aws.events.rule-name>,
    "originalEvent" : <aws.events.event.json>
  }
  ```

  生成带有正确引号的有效 JSON：

  ```
  {
    "ruleArn" : "arn:aws:events:us-east-2:123456789012:rule/example",
    "ruleName" : "example",
    "originalEvent" : {
      ... // commented for brevity
    }
  }
  ```

- 对于（非 JSON）文本输出为多行字符串，请用双引号将输入模板中的每行单独行括起来。

  例如，如果您要将 Finding Amazon Inspector 事件与以下事件模式进行匹配：

  ```
  {
    "detail": {
      "severity": ["HIGH"],
      "status": ["ACTIVE"]
    },
    "detail-type": ["Inspector2 Finding"],
    "source": ["inspector2"]
  }
  ```

  并使用以下输入路径：

  ```
  {
  ```

```
  "account": "$.detail.awsAccountId",
  "ami": "$.detail.resources[0].details.awsEc2Instance.imageId",
  "arn": "$.detail.findingArn",
  "description": "$.detail.description",
  "instance": "$.detail.resources[0].id",
  "platform": "$.detail.resources[0].details.awsEc2Instance.platform",
  "region": "$.detail.resources[0].region",
  "severity": "$.detail.severity",
  "time": "$.time",
  "title": "$.detail.title",
  "type": "$.detail.type"
}
```

你可以使用下面的输入模板来生成多行字符串输出：

```
"<severity> severity finding <title>"
"Description: <description>"
"ARN: \"<arn>\""
"Type: <type>"
"AWS Account: <account>"
"Region: <region>"
"EC2 Instance: <instance>"
"Platform: <platform>"
"AMI: <ami>"
```

在创建规则的过程中配置输入转换器

作为创建规则的一部分，您可以指定一个输入转换器，用于 EventBridge 在将匹配事件发送到指定目标之前处理这些事件。您可以为 AWS 服务或 API 目标的目标配置输入转换器。

创建目标输入转换器，作为规则的一部分

1. 按???中的步骤操作，创建一条规则。
2. 在步骤 3 - 选择目标中，展开其他设置。
3. 对于配置目标输入，在下拉列表中选择输入转换器。

   单击配置输入转换器。

   EventBridge 显示配置输入变压器对话框。

4. 在示例事件部分，选择要测试事件模式的示例事件类型。您可以选择 AWS 活动、合作伙伴活动或输入自己的自定义活动。

   AWS events

   从支持的 AWS 服务发出的事件中进行选择。

   1. 选择 AWS 事件。
   2. 在示例事件下，选择所需 AWS 的事件。活动按 AWS 服务组织。

      当您选择一个事件时， EventBridge 会填充示例事件。

      例如，如果您选择 S3 对象已创建，则 EventBridge 会显示示例 S3 对象已创建事件。
   3. （可选）您也可以选择复制，将示例事件复制到设备的剪贴板。

   Partner events

   从支持的 EventBridge第三方服务（例如 Salesforce）发出的事件中进行选择。

   1. 选择EventBridge 合作伙伴活动。
   2. 在示例事件下，选择所需合作伙伴事件。事件是按合作伙伴组织的。

      当您选择一个事件时， EventBridge 会填充示例事件。
   3. （可选）您也可以选择复制，将示例事件复制到设备的剪贴板。

   Enter your own

   以 JSON 文本输入您自己的事件。

   1. 选择输入您自己的。
   2. EventBridge 使用必需事件属性的模板填充示例事件。
   3. 根据需要编辑示例事件并添加内容。示例事件必须是有效的 JSON。
   4. （可选）您还可以选择以下任何选项：
      - 复制 - 将示例事件复制到设备的剪贴板。
      - 修饰 - 添加换行符、制表符和空格，使 JSON 文本更易于阅读。
5. （可选）展开示例输入路径、模板和输出部分，查看以下示例：

- 如何使用 JSON 路径来定义表示事件数据的变量
- 如何在输入转换器模板中使用这些变量
- EventBridge 发送到目标的结果输出

有关输入转换的更多详细示例，请参阅???。

6. 在目标输入转换器部分，定义要在输入模板中使用的任何变量。

   变量使用 JSON 路径引用原始事件源中的值。然后，您可以在输入模板中引用这些变量，以便在 EventBridge 传递到目标的转换事件中包含来自原始源事件的数据。您最多可以定义 100 个变量。输入转换器必须是有效的 JSON。

   例如，假设您选择了 AWS 事件 S3 Object Created 作为此输入转换器的示例事件。然后，您可以定义以下变量，在模板中使用：

   ```
   {
     "requester": "$.detail.requester",
     "key": "$.detail.object.key",
     "bucket": "$.detail.bucket.name"
   }
   ```

   （可选）您也可以选择复制，将输入转换器复制到设备的剪贴板。

7. 在 “模板” 部分中，撰写要用于确定 EventBridge 传递给目标的内容的模板。

   您可以使用 JSON 字符串、静态信息、您定义的变量以及保留变量。有关输入转换的更多详细示例，请参阅???。

   例如，假设您在前一个示例中定义了变量。然后，您可以撰写以下模板，引用这些变量，以及保留变量和静态信息。

   ```
   {
       "message": "<requester> has created the object \"<key>\" in the bucket
    \"<bucket>\"",
       "RuleName": <aws.events.rule-name>,
       "ruleArn" : <aws.events.rule-arn>,
       "Transformed": "Yes"
   }
   ```

   （可选）您也可以选择复制，将模板复制到设备的剪贴板。

8. 要测试您的模板，请选择生成输出。

   EventBridge 根据输入模板处理示例事件，并显示在 Output 下生成的转换后的输出。这些信息 EventBridge 将代替原始源事件传递给目标。

   上述示例输入模板生成的输出将如下所示：

   ```
   {
       "message": "123456789012 has created the object "example-key" in the bucket
    "example-bucket"",
       "RuleName": rule-name,
       "ruleArn" : arn:aws:events:us-east-1:123456789012:rule/rule-name,
       "Transformed": "Yes"
   }
   ```

   （可选）您也可以选择复制，将生成的输出复制到设备的剪贴板。

9. 选择确认
10. 按???中的其余步骤操作，创建一条规则。

使用 EventBridge 沙盒测试目标输入变压器

在将信息 EventBridge传递给规则目标之前，您可以使用输入转换器自定义事件中的文本。

在创建新规则或编辑现有规则时，配置输入转换器通常是指定目标这一大型过程中的一环。但是 EventBridge，使用中的沙盒，您可以快速配置输入变压器并使用示例事件来确认您获得了所需的输出，而无需创建或编辑规则。

有关输入转换的更多信息，请参阅 ???。

测试目标输入转换器

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在开发人员资源下，选择沙盒，然后在沙盒页面上选择目标输入转换器选项卡。
3. 在示例事件部分，选择要测试事件模式的示例事件类型。您可以选择 AWS 活动、合作伙伴活动或输入自己的自定义活动。

   AWS events

   从支持的 AWS 服务发出的事件中进行选择。

1. 选择 AWS 事件。
2. 在示例事件下，选择所需 AWS 的事件。活动按 AWS 服务组织。

   当您选择一个事件时， EventBridge 会填充示例事件。

   例如，如果您选择 S3 对象已创建，则 EventBridge 会显示示例 S3 对象已创建事件。
3. （可选）您也可以选择复制，将示例事件复制到设备的剪贴板。

Partner events

从支持的 EventBridge第三方服务（例如 Salesforce）发出的事件中进行选择。

1. 选择EventBridge 合作伙伴活动。
2. 在示例事件下，选择所需合作伙伴事件。事件是按合作伙伴组织的。

   当您选择一个事件时， EventBridge 会填充示例事件。
3. （可选）您也可以选择复制，将示例事件复制到设备的剪贴板。

Enter your own

以 JSON 文本输入您自己的事件。

1. 选择输入您自己的。
2. EventBridge 使用必需事件属性的模板填充示例事件。
3. 根据需要编辑示例事件并添加内容。示例事件必须是有效的 JSON。
4. （可选）您还可以选择以下任何选项：
   - 复制 - 将示例事件复制到设备的剪贴板。
   - 修饰 - 添加换行符、制表符和空格，使 JSON 文本更易于阅读。

4. （可选）展开示例输入路径、模板和输出部分，查看以下示例：

- 如何使用 JSON 路径来定义表示事件数据的变量
- 如何在输入转换器模板中使用这些变量
- EventBridge 发送到目标的结果输出

有关输入转换的更多详细示例，请参阅???。

5. 在目标输入转换器部分，定义要在输入模板中使用的任何变量。

   变量使用 JSON 路径引用原始事件源中的值。然后，您可以在输入模板中引用这些变量，以便在 EventBridge 传递到目标的转换事件中包含来自原始源事件的数据。您最多可以定义 100 个变量。输入转换器必须是有效的 JSON。

   例如，假设您选择了 AWS 事件 S3 Object Created 作为此输入转换器的示例事件。然后，您可以定义以下变量，在模板中使用：

   ```
   {
     "requester": "$.detail.requester",
     "key": "$.detail.object.key",
     "bucket": "$.detail.bucket.name"
   }
   ```

   （可选）您也可以选择复制，将输入转换器复制到设备的剪贴板。

6. 在 “模板” 部分中，撰写要用于确定 EventBridge 传递给目标的内容的模板。

   您可以使用 JSON 字符串、静态信息、您定义的变量以及保留变量。有关输入转换的更多详细示例，请参阅???。

   例如，假设您在前一个示例中定义了变量。然后，您可以撰写以下模板，引用这些变量，以及保留变量和静态信息。

   ```
   {
       "message": "<requester> has created the object \"<key>\" in the bucket
    \"<bucket>\"",
       "RuleName": <aws.events.rule-name>,
       "ruleArn" : <aws.events.rule-arn>,
       "Transformed": "Yes"
   }
   ```

   （可选）您也可以选择复制，将模板复制到设备的剪贴板。

7. 要测试您的模板，请选择生成输出。

   EventBridge 根据输入模板处理示例事件，并显示在 Output 下生成的转换后的输出。这些信息 EventBridge 将代替原始源事件传递给目标。

   上述示例输入模板生成的输出将如下所示：

```
{
    "message": "123456789012 has created the object "example-key" in the bucket
 "example-bucket"",
    "RuleName": rule-name,
    "ruleArn" : arn:aws:events:us-east-1:123456789012:rule/rule-name,
    "Transformed": "Yes"
}
```

（可选）您也可以选择复制，将生成的输出复制到设备的剪贴板。

Amazon EventBridge 存档和重放

在 EventBridge 中，您可以创建事件存档，这样日后就可以轻松重放这些事件。例如，您可能希望重放事件来从错误中恢复，或验证应用程序中的新功能。

> **Note**
>
> 在将事件发布到事件总线和事件到达存档之间可能会有延迟。我们建议您将重放存档事件延迟 10 分钟，以确保重放所有事件。

以下视频演示了存档和重放的用法：创建存档和重放

主题

- 存档 Amazon EventBridge 事件
- 重放存档的 Amazon EventBridge 事件

存档 Amazon EventBridge 事件

在 EventBridge 中创建存档时，您可以指定事件模式，确定将哪些事件发送到存档。EventBridge 会将与事件模式匹配的事件发送到存档。您还可以设置保留期，在事件被丢弃之前将其存储到存档中。

默认情况下，EventBridge 使用 AWS 拥有的 CMK，使用 256 位高级加密标准 (AES-256) 加密存档中的事件数据，这样有助于保护您的数据，防止未经授权的访问。

> **Note**
>
> 过期事件通常每 24 小时从 `DescribeArchive` 操作的 `EventCount` 和 `SizeBytes` 值中扣除一次。因此，最近过期的事件可能不会反映在这些值中。

创建用于所有事件的存档

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在左侧导航窗格中，选择存档。
3. 选择创建存档。
4. 在存档详细信息下，输入档案的名称。在所选区域中，该名称对于您的账户必须是唯一的。

   在创建存档后无法更改名称。
5. （可选）输入存档的描述。
6. 对于源，选择发出事件并发送到存档的事件总线。
7. 对于保留期，执行以下操作之一：
   - 选择无限期，可将事件保留在存档中，永远不会删除。
   - 输入保留事件的天数。在指定的天数之后，EventBridge 将从存档中删除事件。
8. 选择 Next（下一步）。
9. 在事件模式下，选择未在筛选事件。
10. 选择创建存档。

创建具有某种事件模式的存档

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在左侧导航窗格中，选择存档。

3. 选择创建存档。
4. 在存档详细信息下，输入档案的名称。在所选区域中，该名称对于您的账户必须是唯一的。

   在创建存档后无法更改名称。
5. （可选）输入存档的描述。
6. 对于源，选择发出事件并发送到存档的事件总线。
7. 对于保留期，执行以下操作之一：
   - 选择无限期*，可将事件保留在存档中，永远不会删除。
   - 输入保留事件的天数。在指定的天数之后，EventBridge 将从存档中删除事件。
8. 选择 Next（下一步）。
9. 在事件模式下，选择正在通过事件模式匹配来筛选事件。
10. 请执行下列操作之一：
    - 选择模式生成器，然后选择服务提供商。如果选择 AWS，还要选择要在模式中使用的 AWS 服务名称和事件类型。
    - 选择 JSON 编辑器，手动创建模式。您也可以从规则中复制模式，然后粘贴到 JSON 编辑器中。
11. 选择创建存档。

要确认事件已成功发送到存档，您可以使用 EventBridge API 的 `DescribeArchive` 操作，了解 `EventCount` 是否反映存档中的事件数量。如果为 0，则存档中没有事件。

重放存档的 Amazon EventBridge 事件

创建存档后，您可以重放存档中的事件。例如，如果您使用其他功能更新应用程序，则可以重放历史事件，以确保重新处理该事件，以保持应用程序的一致性。您还可使用存档，为新功能重放事件。重放事件时，您可以指定要从哪个存档中重放事件、要重放事件的开始和结束时间、事件总线或重放事件的一条或多条规则。

事件的重放顺序不一定要与添加到存档中的顺序相同。重放根据事件中的时间处理要重放的事件，并以一分钟的间隔进行重放。如果您指定了涵盖 20 分钟时间范围的事件开始时间和事件结束时间，会首先从该 20 分钟范围的第一分钟开始重放事件。然后重放第二分钟的事件。您可以使用 EventBridge API 的 `DescribeReplay` 操作来确定重放的进度。`EventLastReplayedTime` 可返回上次重放事件的时间戳。

事件的重放基于 AWS 账户的每秒 `PutEvents` 事务限制，但与该限制是分开的。您可以请求提高 PutEvents 的限制。有关更多信息，请参阅 Amazon EventBridge 配额。

> **Note**
>
> 每个 AWS 区域的每个账户最多可以有 10 个处于活动状态的并发重放。

开始活动重放

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在左侧导航窗格中，选择重放。
3. 选择启动新的重放。
4. 输入重放的名称和描述（可选）。
5. 对于来源，选择要重放事件的存档。
6. 对于目的地，您只能将事件重放到发出事件的同一事件总线。
7. 在指定规则部分，请指定下列值之一：
   - 选择所有规则，将事件重放到所有规则。
   - 选择指定规则，然后选择要将事件重放到的一条或多条规则。
8. 在重放时间范围下，指定开始时间和结束时间的日期、时间和时区。仅重放开始时间和结束时间之间发生的事件。
9. 选择启动重放。

重放存档中的事件时，重放的状态为已完成。

如果您启动重放后想要中断它，只要重放状态为正在启动或正在运行，就可以取消。

取消重放

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在左侧导航窗格中，选择重放。
3. 选择要取消的重放。
4. 选择取消。

Amazon EventBridge Pipes

Amazon EventBridge Pipes 可将源与目标连接起来。管道用于在支持的源和目标之间进行点对点集成，并支持高级转换和富集。开发事件驱动架构时，它可减少对专业知识和集成代码的需求，实现公司应用程序的一致性。要设置管道，请选择源、添加可选筛选、定义可选富集，然后为事件数据选择目标。

> **Note**
>
> 您也可以使用事件总线路由事件。事件总线非常适合在事件驱动的服务之间进行多对多事件路由。有关更多信息，请参阅???。

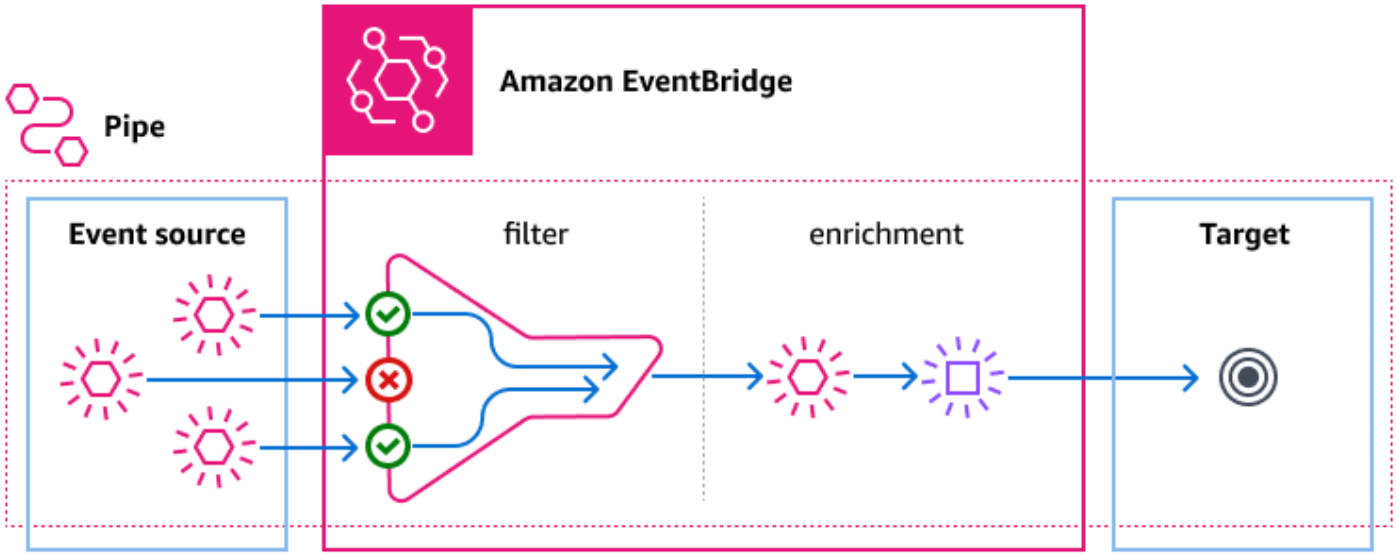
EventBridge Pipes 的工作原理

概括地说，EventBridge Pipes 的工作原理如下：

1. 您在账户中创建一个管道。这包括：
   - 指定一个支持的事件源，您希望管道从中接收事件。
   - （可选）配置筛选器，使管道仅处理从源接收到的事件的一部分。
   - （可选）配置一个富集步骤，在将事件数据发送到目标之前对其进行增强。
   - 指定一个支持的目标，您希望管道向其发送事件。
2. 事件源开始向管道发送事件，管道处理事件，然后发送到目标。
   - 如果您配置了筛选器，管道会评估事件，仅在事件与该筛选器匹配时才将其发送到目标。

     您只需为符合筛选条件的事件付费。
   - 如果您配置了富集，管道会在将事件发送到目标之前对事件执行此富集。

     如果对事件进行批处理，富集会保持批次中事件的顺序。



例如，管道可以用来创建电子商务系统。假设您有一个 API，其中包含客户信息（例如送货地址）。

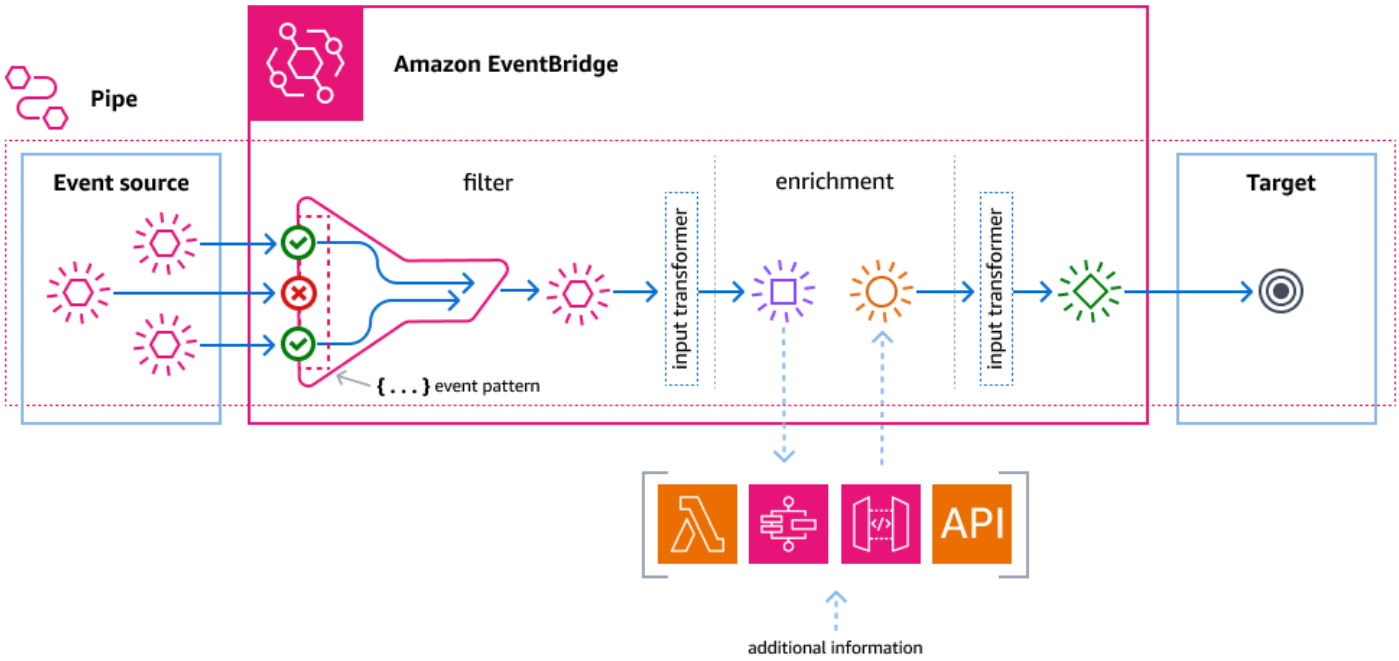
1. 您可以使用以下项创建管道：
   - Amazon SQS 收到订单消息队列，作为事件源。
   - EventBridge API 目标，作为富集
   - AWS Step Functions 状态机，作为目标
2. 然后，当 Amazon SQS 收到订单消息出现在队列中时，该消息将发送到您的管道。
3. 管道将该数据发送到 EventBridge API 目标进行富集，后者会返回该订单的客户信息。
4. 最后，管道将富集的数据发送到处理订单的 AWS Step Functions 状态机。

EventBridge Pipes 概念

以下是 EventBridge Pipes 中基本组件的详细介绍。

竖线

管道将事件从单一源路由到单一目标。管道还能够筛选特定事件，以及在将事件数据发送到目标之前对其进行富集。



来源

EventBridge Pipes 接收来自各种源的事件数据，对这些数据应用可选的筛选和富集，然后将其发送到目标。如果源对发送到管道的事件强制规定执行顺序，该顺序将在发送到目标的整个过程中保持不变。

更多有关来源的信息，请参阅 ???。

筛选器

管道可以筛选给定源的事件，仅处理其中的一部分事件。要在管道中配置筛选，您需要定义一个事件模式，管道使用该模式来确定要将哪些事件发送到目标。

您只需为符合筛选条件的事件付费。

有关更多信息，请参阅???。

富集

通过 EventBridge Pipes 的富集步骤，您可以在将数据从源发送到目标之前，对其进行增强。例如，您可能会收到票证已创建 事件，但其中不包含完整票证数据。使用富集，您可以使用 Lambda 函数调用 `get-ticket` API，以获取完整的票证详情。然后，管道可以将该信息发送到目标。

有关富集事件数据的更多信息，请参阅 ???。

目标

筛选和富集事件数据后，您可以指定管道将其发送到特定目标，例如 Amazon Kinesis 流或 Amazon CloudWatch 日志组。有关可用目标的列表，请参阅 ???。

数据在增强之后、通过管道将其发送到目标之前，您可以对其进行转换。有关更多信息，请参阅???。

多个管道可以将事件发送到同一目标，每个管道可有不同的源。

您也可以同时使用管道和事件总线，将事件发送到多个目标。一个常见的使用场景是创建一个管道，它以事件总线为目标；该管道将事件发送到此事件总线，然后事件总线会将这些事件发送到多个目标。例如，您可以创建一个管道，将 DynamoDB 流作为源，将事件总线作为目标。管道接收来自 DynamoDB 流的事件，并将它们发送到事件总线，然后事件总线根据您在事件总线中指定的规则，将它们发送到多个目标。

Amazon EventBridge Pipes 的权限

设置管道时，您可以使用现有的执行角色，也可以由 EventBridge 为您创建具有所需权限的执行角色。EventBridge Pipes 所需的权限因源类型而异，如下所示。如果您要设置自己的执行角色，必须自行添加这些权限。

> **Note**
>
> 如果您不确定访问源所需的确切权限范围，请使用 EventBridge Pipes 控制台创建新角色，然后检查策略中列出的操作。

主题

- DynamoDB 执行角色权限
- Kinesis 执行角色权限
- Amazon MQ 执行角色权限
- Amazon MSK 执行角色权限
- 自托管 Apache Kafka 执行角色权限
- Amazon SQS 执行角色权限
- 富集和目标权限

DynamoDB 执行角色权限

对于 DynamoDB Streams，EventBridge Pipes 需要以下权限才能管理与您的 DynamoDB 数据流相关的资源。

- dynamodb:DescribeStream
- dynamodb:GetRecords
- dynamodb:GetShardIterator
- dynamodb:ListStreams

要将失败批次的记录发送到管道死信队列，您的管道执行角色需要以下权限：

- sqs:SendMessage

Kinesis 执行角色权限

对于 Kinesis，EventBridge Pipes 需要以下权限才能管理与您的 Kinesis 数据流相关的资源。

- kinesis:DescribeStream
- kinesis:DescribeStreamSummary
- kinesis:GetRecords
- kinesis:GetShardIterator
- kinesis:ListShards
- kinesis:ListStreams
- kinesis:SubscribeToShard

要将失败批次的记录发送到管道死信队列，您的管道执行角色需要以下权限：

- sqs:SendMessage

Amazon MQ 执行角色权限

对于 Amazon MQ，EventBridge Pipes 需要以下权限才能管理与您的 Amazon MQ 消息代理相关的资源。

- mq:DescribeBroker
- secretsmanager:GetSecretValue
- ec2:CreateNetworkInterface
- ec2:DeleteNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:DescribeSecurityGroups
- ec2:DescribeSubnets
- ec2:DescribeVpcs
- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents

Amazon MSK 执行角色权限

对于 Amazon MSK，EventBridge 需要以下权限才能管理与您的 Amazon MSK 主题相关的资源。

> **Note**
>
> 如果您使用基于 IAM 角色的身份验证，则除了下面列出的权限外，您的执行角色还需要 ??? 中列出的权限。

- kafka:DescribeClusterV2
- kafka:GetBootstrapBrokers
- ec2:CreateNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:DescribeVpcs
- ec2:DeleteNetworkInterface
- ec2:DescribeSubnets
- ec2:DescribeSecurityGroups
- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents

自托管 Apache Kafka 执行角色权限

对于自托管 Apache Kafka，EventBridge 需要以下权限才能管理与您的自托管 Apache Kafka 流相关的资源。

所需的权限

要在 Amazon CloudWatch Logs 中创建日志，并将日志存储到日志组，管道必须在它的执行角色中具有以下权限：

- `logs:CreateLogGroup`
- `logs:CreateLogStream`
- `logs:PutLogEvents`

可选权限

您的管道还可能需要权限来：

- 描述您的 Secrets Manager 密钥。
- 访问 AWS Key Management Service（AWS KMS）客户管理的密钥。
- 访问 Amazon VPC。

Secrets Manager 和 AWS KMS 权限

根据您为 Apache Kafka 代理配置的访问控制类型，您的管道可能需要访问您的 Secrets Manager 密钥或解密 AWS KMS 客户管理的密钥的权限。要连接到这些资源，函数的执行角色必须具有以下权限：

- `secretsmanager:GetSecretValue`
- `kms:Decrypt`

VPC 权限

如果只有 VPC 内的用户才能访问您的自托管 Apache Kafka 集群，您的管道必须具有访问 Amazon VPC 资源的权限。这些资源包括您的 VPC、子网、安全组和网络接口。要连接到这些资源，管道的执行角色必须具有以下权限：

- `ec2:CreateNetworkInterface`

- ec2:DescribeNetworkInterfaces
- ec2:DescribeVpcs
- ec2:DeleteNetworkInterface
- ec2:DescribeSubnets
- ec2:DescribeSecurityGroups

Amazon SQS 执行角色权限

对于 Amazon SQS，EventBridge 需要以下权限才能管理与您的 Amazon SQS 队列相关的资源。

- sqs:ReceiveMessage
- sqs:DeleteMessage
- sqs:GetQueueAttributes

富集和目标权限

为了对您拥有的资源执行 API 调用，EventBridge Pipes 需要相应权限。EventBridge Pipes 将您在管道中指定的 IAM 角色用于富集，将 IAM 主体 `pipes.amazonaws.com` 用于目标调用。

创建亚马逊 EventBridge 管道

EventBridge Pipes 使您能够在源和目标之间创建 point-to-point 集成，包括高级事件转换和扩展。要创建 EventBridge 管道，请执行以下步骤：

1. ???
2. ???
3. ???
4. ???
5. ???

有关如何使用 CLI 创建管道的信息，请参阅 AWS CL AWS I 命令参考中的 create- pipe。

指定源

首先，请指定您希望管道接收事件的源。

使用控制台指定管道源

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择管道。
3. 选择创建管道。
4. 输入管道的名称。
5. （可选）添加管道的描述。
6. 在构建管道选项卡上，对于源，选择要为此管道指定的源类型，然后配置此源。

   配置属性因您所选的源类型而异：

   Confluent

   要将 Confluent Cloud 直播配置为来源，请使用控制台

   1. 对于源，选择 Confluent Cloud。
   2. 对于引导服务器，输入代理的 `host:port` 配对地址。
   3. 对于主题名称，输入管道将要读取的主题名称。
   4. （可选）对于 VPC，选择所需的 VPC。然后，对于 VPC 子网，选择所需的子网。对于 VPC 安全组，选择安全组。
   5. 对于“身份验证-可选”，打开“使用身份验证”并执行以下操作：
      a. 对于身份验证方法，选择身份验证类型。
      b. 在密钥中选择密钥。

      有关更多信息，请参阅 Confluent 文档中的向 Confluent 云资源进行身份验证。
   6. （可选）对于其他设置 - 可选，执行以下操作：
      a. 对于起始位置，请选择以下选项之一：
         - 最新 - 开始读取分片中包含最新记录的流。
         - 修剪水平线 - 开始读取分片中包含最后一条未修剪过的记录的流。这是分片中最早的记录。
      b. 对于批次大小 - 可选，输入每个批次的最大记录数。默认值是 100。
      c. 对于批次时段 - 可选，输入在继续操作之前收集记录的最大秒数。

DynamoDB

1. 为源选择 DynamoDB。
2. 对于 DynamoDB 流，选择要用作源的流。
3. 对于起始位置，请选择以下选项之一：
   - 最新 - 开始读取分片中包含最新记录的流。
   - 修剪水平线 - 开始读取分片中包含最后一条未修剪过的记录的流。这是分片中最早的记录。
4. （可选）对于其他设置 - 可选，执行以下操作：
   a. 对于批次大小 - 可选，输入每个批次的最大记录数。默认值是 100。
   b. 对于批次时段 - 可选，输入在继续操作之前收集记录的最大秒数。
   c. 对于每个分片的并发批次 - 可选，输入同一分片中可以同时读取的批次数。
   d. 对于部分批次项目失败时，请选择以下选项：
      - AUTOMATIC_BISECT - 将每个批次一分为二，并分别重试，直到所有记录都处理完毕或批次中还剩下一条失败消息。

      > **Note**
      >
      > 如果不选择 AUTOMATIC_BISECT，则可以返回特定的失败记录，只会重试这些记录。

Kinesis

使用控制台配置 Kinesis 源

1. 对于源，选择 Kinesis。
2. 对于 Kinesis 流，选择要用作源的流。
3. 对于起始位置，请选择以下选项之一：
   - 最新 - 开始读取分片中包含最新记录的流。
   - 修剪水平线 - 开始读取分片中包含最后一条未修剪过的记录的流。这是分片中最早的记录。

- 在时间戳处 - 从指定时间开始读取流。在时间戳下，使用 YYYY/MM/DD 和 hh:mm:ss 格式输入日期和时间。

4. （可选）对于其他设置 - 可选，执行以下操作：

   a. 对于批次大小 - 可选，输入每个批次的最大记录数。默认值是 100。

   b. （可选）对于批次时段 - 可选，输入在继续操作之前收集记录的最大秒数。

   c. 对于每个分片的并发批次 - 可选，输入同一分片中可以同时读取的批次数。

   d. 对于部分批次项目失败时，请选择以下选项：

   - AUTOMATIC_BISECT - 将每个批次一分为二，并分别重试，直到所有记录都处理完毕或批次中还剩下一条失败消息。

> **Note**
>
> 如果不选择 AUTOMATIC_BISECT，则可以返回特定的失败记录，只会重试这些记录。

Amazon MQ

使用控制台配置 Amazon MQ 源

1. 对于源，选择 Amazon MQ。
2. 对于 Amazon MQ 代理，选择要用作源的流。
3. 对于队列名称，输入管道将要读取的队列名称。
4. 对于身份验证方法，选择 BASIC_AUTH。
5. 在密钥中选择密钥。
6. （可选）对于其他设置 - 可选，执行以下操作：

   a. 对于批次大小 - 可选，输入每个批次的最大消息数。默认值是 100。

   b. 对于批次时段 - 可选，输入在继续操作之前收集记录的最大秒数。

Amazon MSK

使用控制台配置 Amazon MSK 源

1. 对于源，选择 Amazon MSK。
2. 对于 Amazon MSK 集群，选择要使用的集群。

3. 对于主题名称，输入管道将要读取的主题名称。
4. （可选）对于使用者组 ID，输入希望管道加入的使用者组的 ID。
5. （可选）对于身份验证 - 可选，启用使用身份验证并执行以下操作：
   a. 对于身份验证方法，选择所需的类型。
   b. 在密钥中选择密钥。
6. （可选）对于其他设置 - 可选，执行以下操作：
   a. 对于批次大小 - 可选，输入每个批次的最大记录数。默认值是 100。
   b. 对于批次时段 - 可选，输入在继续操作之前收集记录的最大秒数。
   c. 对于起始位置，请选择以下选项之一：
      - 最新 - 开始读取分片中包含最新记录的主题。
      - 修剪水平线 - 开始读取分片中包含最后一条未修剪过的记录的和题。这是分片中最早的记录。

> **Note**
>
> 修剪水平线与 Apache Kafka 中的最早相同。

Self managed Apache Kafka

使用控制台配置自托管 Apache Kafka 源

1. 对于源，选择自我托管式 Apache Kafka。
2. 对于引导服务器，输入代理的 `host:port` 配对地址。
3. 对于主题名称，输入管道将要读取的主题名称。
4. （可选）对于 VPC，选择所需的 VPC。然后，对于 VPC 子网，选择所需的子网。对于 VPC 安全组，选择安全组。
5. （可选）对于身份验证 - 可选，启用使用身份验证并执行以下操作：
   a. 对于身份验证方法，选择身份验证类型。
   b. 在密钥中选择密钥。
6. （可选）对于其他设置 - 可选，执行以下操作：
   a. 对于起始位置，请选择以下选项之一：
      - 最新 - 开始读取分片中包含最新记录的流。

- 修剪水平线 - 开始读取分片中包含最后一条未修剪过的记录的流。这是分片中最早的记录。

b. 对于批次大小 - 可选，输入每个批次的最大记录数。默认值是 100。

c. 对于批次时段 - 可选，输入在继续操作之前收集记录的最大秒数。

Amazon SQS

使用控制台配置 Amazon SQS 源

1. 对于源，选择 SQS。
2. 对于 SQS 队列，请选择要使用的队列。
3. （可选）对于其他设置 - 可选，执行以下操作：

a. 对于批次大小 - 可选，输入每个批次的最大记录数。默认值是 100。

b. 对于批次时段 - 可选，输入在继续操作之前收集记录的最大秒数。

配置事件筛选（可选）

您可以向管道添加筛选功能，这样就可以只将部分事件从源发送到目标。

使用控制台配置筛选

1. 选择筛选。
2. 在示例事件 - 可选下，您将看到一个示例事件，可以使用它来构建您自己的事件模式，您也可以选择输入您自己的，输入您自己的事件。
3. 在事件模式下，输入要用于筛选事件的事件模式。有关构造过滤器的更多信息，请参阅???。

以下是一个事件模式示例，仅发送城市字段中的值为西雅图的事件。

```
{
  "data": {
    "City": ["Seattle"]
  }
}
```

现在已对事件进行筛选，您可以为管道添加可选的富集和目标。

定义事件富集（可选）

您可以将用于丰富的事件数据发送到 Lambda 函数、 AWS Step Functions 状态机、Amazon API Gateway 或 API 目标。

选择富集

1. 选择富集。
2. 在详细信息下，为服务选择要用于富集的服务和相关设置。

您也可以先转换数据，再发送以进行增强。

（可选）定义输入转换器

1. 选择富集输入转换器 - 可选。
2. 对于示例事件/事件有效负载，请选择示例事件类型。
3. 对于转换器，请输入转换器语法，例如 `"Event happened at <$.detail.field>."`，其中 `<$.detail.field>` 是对示例事件中某个字段的引用。您也可以双击示例事件中的某个字段，将其添加到转换器中。
4. 对于输出，请确认输出符合您的要求。

现在，数据已经过筛选和增强，您必须定义要将事件数据发送到的目标。

配置目标

配置目标

1. 选择目标。
2. 在详细信息下，为目标服务选择目标。显示的字段因您选择的目标而异。根据需要输入此目标类型的特定信息。

您也可以先转换数据，再发送到目标。

（可选）定义输入转换器

1. 选择目标输入变压器 - 可选。
2. 对于示例事件/事件有效负载，请选择示例事件类型。

3. 对于转换器，请输入转换器语法，例如 "Event happened at <$.detail.field>."，其中 <$.detail.field> 是对示例事件中某个字段的引用。您也可以双击示例事件中的某个字段，将其添加到转换器中。
4. 对于输出，请确认输出符合您的要求。

现在，管道已配置完毕，请确保其设置配置正确。

配置管道设置

默认情况下管道处于活动状态，但您可以将其停用。您还可以指定管道的权限、设置管道日志记录和添加标签。

配置管道设置

1. 选择管道设置选项卡。
2. 默认情况下，新创建的管道创建好就处于活动状态。如果要创建非活动管道，请在激活下，为激活管道关闭活动。
3. 在权限下，对于执行角色，执行以下操作之一：

   a. 要为此管道 EventBridge 创建新的执行角色，请选择为此特定资源创建新角色。在角色名称下，您可以选择编辑角色名称。

   b. 要使用现有的执行角色，请选择使用现有角色。在角色名称下，选择角色。
4. （可选）如果您已将 Kinesis 或 DynamoDB 流指定为管道源，则可以配置重试策略和死信队列 (DLQ)。

   对于重试策略和死信队列 - 可选，请执行以下操作：

   在重试策略下，执行以下操作：

   a. 如果要启用重试策略，请打开重试。默认情况下，新创建的管道未启用重试策略。

   b. 对于 Maximum age of event（事件的最大时长），输入一分钟（00:01）与 24 小时（24:00）之间的值。

   c. 对于重试尝试，输入 0 到 185 之间的数字。

   d. 如果要使用死信队列 (DLQ)，请打开死信队列，选择您的方法，然后选择要使用的队列或主题。默认情况下，新创建的管道不使用 DLQ。
5. （可选）在日志 - 可选下，您可以设置 EventBridge Pipes 如何向支持的服务发送日志信息，包括如何配置这些日志。

有关管道记录日志的更多信息，请参阅 ???。

CloudWatch 默认情况下，日志被选为日志目标，ERROR日志级别也是如此。因此，默认情况下，Pip EventBridge es 会创建一个新的 CloudWatch 日志组，向该组发送包含详细ERROR级别的日志记录。

要让 Pip EventBridge es 将日志记录发送到任何支持的日志目的地，请执行以下操作：

a. 在日志-可选下，选择要将日志记录传送到的目的地。

b. 对于日志级别，选择 EventBridge 要包含在日志记录中的信息级别。默认情况下选中 ERROR 日志级别。

   有关更多信息，请参阅 ???。

c. 如果 EventBridge 要在日志记录中包含事件负载信息以及服务请求和响应信息，请选择 “包括执行数据”。

   有关更多信息，请参阅 ???。

d. 配置您选择的每个日志目标：

   对于 CloudWatch Logs 日志，请在CloudWatch 日志下执行以下操作：

   - 对于CloudWatch 日志组，请选择是否 EventBridge 创建新的日志组，或者您可以选择现有日志组或指定现有日志组的 ARN。
   - 对于新的日志组，请根据需要编辑日志组名称。

   CloudWatch 默认情况下，日志处于选中状态。

   对于 Firehose 流日志，在Firehose 流日志下，选择该 Firehose 流。

   对于 Amazon S3 日志，在 S3 日志下执行以下操作：

   - 输入要用作日志目标的桶的名称。
   - 输入存储桶所有者的 AWS 账户 ID。
   - 输入 EventBridge 创建 S3 对象时要使用的任何前缀文本。

     有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的使用前缀组织对象。
   - 选择 EventBridge 要如何格式化 S3 日志记录：

- `json`：JSON
- `plain`：纯文本
- `w3c`：W3C 扩展日志记录文件格式

6. （可选）在标签 - 可选下，选择添加新标签，然后为规则输入一个或多个标签。有关更多信息，请参阅 ???。
7. 选择创建管道。

验证配置参数

创建管道后， EventBridge 验证以下配置参数：

- IAM 角色 — 由于创建管道后无法更改管道的来源，因此需要 EventBridge 验证提供的 IAM 角色是否可以访问该源。

> **Note**
>
> EventBridge 不会对浓缩或目标执行相同的验证，因为它们可以在创建管道后进行更新。

- 批处理- EventBridge 验证源的批量大小是否超过目标的最大批次大小。如果是，则 EventBridge 需要较小的批次大小。此外，如果目标不支持批处理，则无法 EventBridge 为源配置批处理。
- 扩充 — EventBridge 验证 API Gateway 和 API 目标丰富版的批量大小是否为 1，因为仅支持批量大小为 1。

启动或停止管道

默认情况下，管道为 `Running` 状态，处理创建好的事件。

如果您使用 Amazon SQS、Kinesis 或 DynamoDB 源创建管道，创建管道通常需要一两分钟。

如果您使用 Amazon MSK、自托管 Apache Kafka 或 Amazon MQ 源创建管道，创建管道最多需要十分钟。

使用控制台创建不处理事件的管道

- 关闭激活管道设置。

创建不以编程方式处理事件的管道

- 在您的 API 调用中，将 `DesiredState` 设置为 `Stopped`。

使用控制台启动或停止现有管道

- 在管道设置选项卡的激活下，为激活管道打开或关闭活动。

以编程方式启动或停止现有管道

- 在 API 调用中，将 `DesiredState` 参数设置为 `RUNNING` 或 `STOPPED`。

在管道处于 `STOPPED` 状态和不再处理事件之间，可能会有延迟：

- 对于 Amazon SQS 和流源，这种延迟通常少于两分钟。
- 对于 Amazon MQ 和 Apache Kafka 源，这种延迟可能长达十五分钟。

Amazon Pi EventBridge pes 来源

EventBridge Pipes 接收来自各种来源的事件数据，对这些数据应用可选的过滤器和扩充功能，然后将其发送到目的地。

如果源对发送到 Pip EventBridge es 的事件强制执行顺序，则该顺序将在到达目标的整个过程中保持不变。

可以将以下 AWS 服务指定为 Pip EventBridge es 的来源：

- Amazon DynamoDB 流
- Amazon Kinesis 流
- Amazon MQ 代理
- Amazon MSK 流
- Amazon SQS 队列
- 阿帕奇 Kafka 直播

  当你将 Apache Kafka 流指定为管道源时，你可以指定自己管理的 Apache Kafka 流，也可以指定由第三方提供商管理的 Apache Kafka 流，例如：

  - Confluent Cloud

- CloudKarafka
- Redpanda

Amazon DynamoDB 流作为源

您可以使用 EventBridge Pipes 接收 DynamoDB 流中的记录。然后，您可以选择筛选或增强这些记录，然后再将它们发送到目标进行处理。在设置管道时，您可以选择 Amazon DynamoDB Streams 的特定设置。将数据发送到目标时，EventBridge Pipes 会保持数据流中记录的顺序。

> **Important**
>
> 禁用作为管道源的 DynamoDB 流会导致该管道变得不可用，即使您随后重新启用流该管道也不再可用。这是因为：
>
> - 您无法停止、启动或更新源已禁用的管道。
> - 创建管道后，您无法使用新源更新此管道。当您重新启用 DynamoDB 流时，该流会被分配一个新的 Amazon 资源名称（ARN），并且不再与您的管道关联。
>
> 如果您确实重新启用了 DynamoDB 流，则需要使用该流的新 ARN 创建新的管道。

示例事件

以下示例事件显示了管道接收到的信息。您可以使用此事件来创建和筛选您的事件模式，或定义输入转换。并非所有字段都可以筛选。有关可筛选字段的更多信息，请参阅 ???。

```
[
  {
    "eventID": "1",
    "eventVersion": "1.0",
    "dynamodb": {
      "Keys": {
        "Id": {
          "N": "101"
        }
      },
      "NewImage": {
        "Message": {
          "S": "New item!"
```

```
        },
        "Id": {
          "N": "101"
        }
      },
      "StreamViewType": "NEW_AND_OLD_IMAGES",
      "SequenceNumber": "111",
      "SizeBytes": 26
    },
    "awsRegion": "us-west-2",
    "eventName": "INSERT",
    "eventSourceARN": "arn:aws:dynamodb:us-east-1:111122223333:table/EventSourceTable",
    "eventSource": "aws:dynamodb"
  },
  {
    "eventID": "2",
    "eventVersion": "1.0",
    "dynamodb": {
      "OldImage": {
        "Message": {
          "S": "New item!"
        },
        "Id": {
          "N": "101"
        }
      },
      "SequenceNumber": "222",
      "Keys": {
        "Id": {
          "N": "101"
        }
      },
      "SizeBytes": 59,
      "NewImage": {
        "Message": {
          "S": "This item has changed"
        },
        "Id": {
          "N": "101"
        }
      },
      "StreamViewType": "NEW_AND_OLD_IMAGES"
    },
    "awsRegion": "us-west-2",
```

```
    "eventName": "MODIFY",
    "eventSourceARN": "arn:aws:dynamodb:us-east-1:111122223333:table/EventSourceTable",
    "eventSource": "aws:dynamodb"
  }
]
```

轮询和批处理流

EventBridge 以每秒四次的基本频率轮询 DynamoDB 流中的分片来获取记录。如果有可用记录，EventBridge 会处理事件并等待结果。如果处理成功，EventBridge 将恢复轮询，直到收到更多记录。

默认情况下，EventBridge 会在记录可用时立即调用您的管道。如果 EventBridge 从源中读取的批次只有一条记录，则只会处理一个事件。为避免处理数量较少的记录，您可以配置批处理时段，让管道缓冲最多五分钟的记录。处理事件前，EventBridge 会持续从源中读取记录，直到收集完整批次、批处理时段到期或批次达到 6MB 的有效负载时为止。

您还可以通过将来自各个分区的多个批次并行处理来增加并发性。EventBridge 可以同时处理每个分片中的多达 10 个批次。如果您增加每个分片的并发批次数，EventBridge 仍然可以确保分区密钥级别的顺序处理。

配置 `ParallelizationFactor` 设置，同时处理 Kinesis 或 DynamoDB 数据流一个分片中的多个管道执行。您可以指定 EventBridge 通过从 1（默认值）到 10 的并行化因子从分片中轮询的并发批次数。例如，假设您将 `ParallelizationFactor` 设置为 2，则最多可以有 200 个并发 EventBridge 管道执行来处理 100 个 Kinesis 数据分片。这有助于在数据量不稳定并且 `IteratorAge` 较高时纵向扩展处理吞吐量。请注意，如果使用 Kinesis 聚合，并行化因子将不起作用。

轮询和流的起始位置

请注意，管道创建和更新期间的流源轮询最终将是一致的。

- 在管道创建期间，可能需要几分钟才能开始轮询来自流的事件。
- 在管道更新源轮询配置期间，可能需要几分钟才能停止和重新开始轮询来自流的事件。

这意味着，如果您指定 `LATEST` 作为流的起始位置，在创建或更新管道期间，管道可能会错过发送的事件。为确保不会错过任何事件，请将流的起始位置指定为 `TRIM_HORIZON`。

报告批处理项目失败

在 EventBridge 使用和处理来自源的流式数据时，默认情况下，仅在批处理完全成功时，才会在批次的最高序列号处设置检查点。为避免重新处理失败批次中已成功处理的消息，您可以配置富集或目标，返回对象来指示哪些消息处理成功、哪些失败。这称为部分批处理响应。

有关更多信息，请参阅???。

成功和失败的条件

如果返回以下任意一项，EventBridge 会将此批次视为完全成功：

- 空的 `batchItemFailure` 列表
- Null `batchItemFailure` 列表
- 空的 `EventResponse`
- Null `EventResponse`

如果返回以下任意一项，EventBridge 会将此批次视为完全失败：

- 空字符串 `itemIdentifier`
- Null `itemIdentifier`
- 包含错误密钥名的 `itemIdentifier`

EventBridge 会根据您的重试策略在失败时重试。

Amazon Kinesis 流作为源

您可以使用 EventBridge Pipes 接收 Kinesis 数据流中的记录。然后，您可以选择筛选或增强这些记录，然后再将它们发送到可用的目标之一进行处理。在设置管道时，可以选择特定于 Kinesis 的设置。将数据发送到目标时，EventBridge Pipes 会保持数据流中记录的顺序。

Kinesis 数据流是一组分区。每个分片包含一系列数据记录。使用者 是一种处理 Kinesis 数据流中的数据的应用程序。您可以将 EventBridge Pipe 映射到共享吞吐量使用者（标准迭代器）或具有增强扇出功能的专用吞吐量使用者。

对于标准迭代器，EventBridge 使用 HTTP 协议轮询 Kinesis 流中的每个分片以查找记录。管道与分片的其他使用者共享读取吞吐量。

为了最大限度地减少延迟并最大限度地提高读取吞吐量，您可以创建具有增强扇出功能的数据流使用者。流使用者将获得与每个分片的专用连接，这不会影响从流中读取信息的其他应用程序。如果您有许多应用程序读取相同的数据，或者您正在重新处理具有大记录的流，则专用吞吐量可以提供帮助。Kinesis 通过 HTTP/2 向 EventBridge 推送记录。有关 Kinesis 数据流的信息，请参阅[读取 Amazon Kinesis Data Streams 中的数据](#)。

示例事件

以下示例事件显示了管道接收到的信息。您可以使用此事件来创建和筛选您的事件模式，或定义输入转换。并非所有字段都可以筛选。有关可筛选字段的更多信息，请参阅 [???](#)。


```
[
  {
    "kinesisSchemaVersion": "1.0",
    "partitionKey": "1",
    "sequenceNumber": "49590338271490256608559692538361571095921575989136588898",
    "data": "SGVsbG8sIHRoaXMgaXMgYSB0ZXN0Lg==",
    "approximateArrivalTimestamp": 1545084650.987
    "eventSource": "aws:kinesis",
    "eventVersion": "1.0",
    "eventID":
 "shardId-000000000006:49590338271490256608559692538361571095921575989136588898",
    "eventName": "aws:kinesis:record",
    "invokeIdentityArn": "arn:aws:iam::123456789012:role/lambda-role",
    "awsRegion": "us-east-2",
    "eventSourceARN": "arn:aws:kinesis:us-east-2:123456789012:stream/lambda-stream"
  },
  {
    "kinesisSchemaVersion": "1.0",
    "partitionKey": "1",
    "sequenceNumber": "49590338271490256608559692540925702759324208523137515618",
    "data": "VGhpcyBpcyBvbmx5IGEgdGVzdC4=",
    "approximateArrivalTimestamp": 1545084711.166
    "eventSource": "aws:kinesis",
    "eventVersion": "1.0",
    "eventID":
 "shardId-000000000006:49590338271490256608559692540925702759324208523137515618",
    "eventName": "aws:kinesis:record",
    "invokeIdentityArn": "arn:aws:iam::123456789012:role/lambda-role",
    "awsRegion": "us-east-2",
    "eventSourceARN": "arn:aws:kinesis:us-east-2:123456789012:stream/lambda-stream"
  }
```

```
]
```

轮询和批处理流

EventBridge 以每秒四次的基本频率轮询 Kinesis 流中的分片来获取记录。如果有可用记录，EventBridge 会处理事件并等待结果。如果处理成功，EventBridge 将恢复轮询，直到收到更多记录。

默认情况下，EventBridge 会在记录可用时立即调用您的管道。如果 EventBridge 从源中读取的批次只有一条记录，则只会处理一个事件。为避免处理数量较少的记录，您可以配置批处理时段，让管道缓冲最多五分钟的记录。处理事件前，EventBridge 会持续从源中读取记录，直到收集完整批次、批处理时段到期或批次达到 6MB 的有效负载时为止。

您还可以通过将来自各个分区的多个批次并行处理来增加并发性。EventBridge 可以同时处理每个分片中的多达 10 个批次。如果您增加每个分片的并发批次数，EventBridge 仍然可以确保分区密钥级别的顺序处理。

配置 `ParallelizationFactor` 设置，同时处理 Kinesis 或 DynamoDB 数据流一个分片中的多个管道执行。您可以指定 EventBridge 通过从 1（默认值）到 10 的并行化因子从分片中轮询的并发批次数。例如，假设您将 `ParallelizationFactor` 设置为 2，则最多可以有 200 个并发 EventBridge 管道执行来处理 100 个 Kinesis 数据分片。这有助于在数据量不稳定并且 `IteratorAge` 较高时纵向扩展处理吞吐量。请注意，如果使用 Kinesis 聚合，并行化因子将不起作用。

轮询和流的起始位置

请注意，管道创建和更新期间的流源轮询最终将是一致的。

- 在管道创建期间，可能需要几分钟才能开始轮询来自流的事件。
- 在管道更新源轮询配置期间，可能需要几分钟才能停止和重新开始轮询来自流的事件。

这意味着，如果您指定 `LATEST` 作为流的起始位置，在创建或更新管道期间，管道可能会错过发送的事件。为确保不会错过任何事件，请将流的起始位置指定为 `TRIM_HORIZON` 或 `AT_TIMESTAMP`。

报告批处理项目失败

在 EventBridge 使用和处理来自源的流式数据时，默认情况下，仅在批处理完全成功时，才会在批次的最高序列号处设置检查点。为避免重新处理失败批次中已成功处理的消息，您可以配置富集或目标，返回对象来指示哪些消息处理成功、哪些失败。这称为部分批处理响应。

有关更多信息，请参阅 ???。

成功和失败的条件

如果返回以下任意一项，EventBridge 会将此批次视为完全成功：

- 空的 `batchItemFailure` 列表
- Null `batchItemFailure` 列表
- 空的 `EventResponse`
- Null `EventResponse`

如果返回以下任意一项，EventBridge 会将此批次视为完全失败：

- 空字符串 `itemIdentifier`
- Null `itemIdentifier`
- 包含错误密钥名的 `itemIdentifier`

EventBridge 会根据您的重试策略在失败时重试。

将 Amazon MQ 消息代理作为源

您可使用 EventBridge Pipes 接收来自 Amazon MQ 消息代理的记录。然后，您可以选择筛选或增强这些记录，然后再将它们发送到可用的目标之一进行处理。在设置管道时，可以选择特定于 Amazon MQ 的设置。将数据发送到目标时，EventBridge Pipes 会保持消息代理中记录的顺序。

Amazon MQ 是一项托管消息代理服务，用于 Apache ActiveMQ 和 RabbitMQ。消息代理允许软件应用程序和组件使用各种编程语言、操作系统和正式消息收发协议进行通信，使用主题或队列作为事件目标。

Amazon MQ 还可以通过安装 ActiveMQ 代理或 RabbitMQ 代理，代表您管理 Amazon Elastic Compute Cloud (Amazon EC2) 实例。安装代理后，它会为您的实例提供不同的网络拓扑和其他基础设施需求。

Amazon MQ 源有以下配置限制：

- 跨账户 - EventBridge 不支持跨账户处理。您不能使用 EventBridge 处理来自另一 AWS 账户中的 Amazon MQ 消息代理的记录。

- 身份验证 - 对于 ActiveMQ，仅支持 ActiveMQ SimpleAuthenticationPlugin。对于 RabbitMQ，仅支持 PLAIN 身份验证机制。要管理凭证，请使用 AWS Secrets Manager。有关 ActiveMQ 身份验证的更多信息，请参阅《Amazon MQ 开发人员指南》中的使用 LDAP 集成 ActiveMQ 代理。
- 连接配额 - 代理具有每个有线级协议允许的最大连接数。此配额基于代理实例类型。有关更多信息，请参阅《Amazon MQ 开发人员指南》中的 *Amazon MQ 中的配额* 的代理部分。
- 连接 - 您可以在公有或私有虚拟私有云 (VPC) 中创建代理。对于私有 VPC，您的管道需要具备对 VPC 的访问权限才能接收消息。
- 事件目标 - 仅支持队列目标。但是，您可以使用虚拟主题，在与管道交互时与，虚拟主题在内部与主题行为一致，在外部与队列行为一致。有关更多信息，请参阅 Apache ActiveMQ 网站上的虚拟目标和 RabbitMQ 网站上的虚拟主机。
- 网络拓扑 - 对于 ActiveMQ，管道仅支持一个单实例或备用代理。对于 RabbitMQ，每个管道只支持一个单实例代理或集群部署。单实例代理需要一个失效转移端点。有关这些代理部署模式的更多信息，请参阅《Amazon MQ 开发人员指南》中的 Active MQ 代理架构和 RabbitMQ 代理架构。
- 协议 - 支持的协议取决于您使用的 Amazon MQ 集成。
  - 对于 ActiveMQ 集成，EventBridge 使用 OpenWire/Java Message Service (JMS) 协议来使用消息。任何其他协议都不支持消息使用。EventBridge 仅支持 JMS 协议中的 TextMessage 和 BytesMessage 操作。有关 OpenWire 协议的更多信息，请参阅 Apache ActiveMQ 网站上的 OpenWire。
  - 对于 RabbitMQ 集成，EventBridge 使用 AMQP 0-9-1 协议来使用消息。消息的使用不支持任何其他协议。有关 RabbitMQ 的 AMQP 0-9-1 协议实施的详细信息，请参阅 RabbitMQ 网站上的 AMQP 0-9-1 完整参考指南。

EventBridge 自动支持 Amazon MQ 支持的最新版本 ActiveMQ 和 RabbitMQ。有关受支持的最新版本，请参阅《Amazon MQ 开发人员指南》中的 Amazon MQ 发布说明。

> **Note**
>
> 默认情况下，Amazon MQ 代理有一个每周维护时段。代理在该时段内无法使用。对于没有备用的代理，EventBridge 在时间窗结束前不会处理消息。

示例事件

以下示例事件显示了管道接收到的信息。您可以使用此事件来创建和筛选您的事件模式，或定义输入转换。并非所有字段都可以筛选。有关可筛选字段的更多信息，请参阅 ???。

ActiveMQ

```
[
  {
    "eventSource": "aws:amq",
    "eventSourceArn": "arn:aws:mq:us-
west-2:112556298976:broker:test:b-9bcfa592-423a-4942-879d-eb284b418fc8",
    "messageID": "ID:b-9bcfa592-423a-4942-879d-eb284b418fc8-1.mq.us-
west-2.amazonaws.com-37557-1234520418293-4:1:1:1:1",
    "messageType": "jms/text-message",
    "data": "QUJDOkFBQUE=",
    "connectionId": "myJMSCoID",
    "redelivered": false,
    "destination": {
      "physicalname": "testQueue"
    },
    "timestamp": 1598827811958,
    "brokerInTime": 1598827811958,
    "brokerOutTime": 1598827811959
  },
  {
    "eventSource": "aws:amq",
    "eventSourceArn": "arn:aws:mq:us-
west-2:112556298976:broker:test:b-9bcfa592-423a-4942-879d-eb284b418fc8",
    "messageID": "ID:b-9bcfa592-423a-4942-879d-eb284b418fc8-1.mq.us-
west-2.amazonaws.com-37557-1234520418293-4:1:1:1:1",
    "messageType": "jms/bytes-message",
    "data": "3DTOOW7crj51prgVLQaGQ82S48k=",
    "connectionId": "myJMSCoID1",
    "persistent": false,
    "destination": {
      "physicalname": "testQueue"
    },
    "timestamp": 1598827811958,
    "brokerInTime": 1598827811958,
    "brokerOutTime": 1598827811959
  }
]
```

RabbitMQ

```
[
  {
```

```
    "eventSource": "aws:rmq",
    "eventSourceArn": "arn:aws:mq:us-
west-2:111122223333:broker:pizzaBroker:b-9bcfa592-423a-4942-879d-eb284b418fc8",
    "eventSourceKey": "pizzaQueue::/",
    "basicProperties": {
      "contentType": "text/plain",
      "contentEncoding": null,
      "headers": {
        "header1": {
          "bytes": [
            118,
            97,
            108,
            117,
            101,
            49
          ]
        },
        "header2": {
          "bytes": [
            118,
            97,
            108,
            117,
            101,
            50
          ]
        },
        "numberInHeader": 10
      },
      "deliveryMode": 1,
      "priority": 34,
      "correlationId": null,
      "replyTo": null,
      "expiration": "60000",
      "messageId": null,
      "timestamp": "Jan 1, 1970, 12:33:41 AM",
      "type": null,
      "userId": "AIDACKCEVSQ6C2EXAMPLE",
      "appId": null,
      "clusterId": null,
      "bodySize": 80
    },
    "redelivered": false,
```

```
    "data": "eyJ0aW1lb3V0IjowLCJkYXRhIjoiQ1pybWYwR3c4T3Y0YnFMUXhENEUifQ=="
  }
]
```

使用者组

为了与 Amazon MQ 进行交互，EventBridge 会创建一个可以从 Amazon MQ 代理中读取的使用者组。使用与管道 UUID 相同的 ID 创建使用者组。

对于 Amazon MQ 源，EventBridge 会将记录合并为批处理，然后通过单个有效负载将其发送到您的函数。要控制行为，您可以配置批处理时段和批处理大小。EventBridge 会提取消息，直到出现下列情况之一时：

- 处理后的记录达到 6 MB 的有效负载大小上限。
- 批处理时间窗过期。
- 记录数达到完整批次大小。

EventBridge 会将您的批次转换为单个有效负载，然后调用您的函数。消息既不会永久保存，也不会反序列化。相反，使用者组会将其作为字节 BLOB 进行检索。然后以 base64 格式编码为 JSON 有效负载。如果管道为批次中的任何消息返回错误，EventBridge 将重试整批消息，直到处理成功或消息过期为止。

网络配置

默认情况下，在 `PubliclyAccessible` 标志设置为 false 时创建 Amazon MQ 代理。只有在 `PubliclyAccessible` 设置为 true 时，代理才会接收公有 IP 地址。要获得对管道的完全访问权限，您的代理必须使用公有端点，或提供对 VPC 的访问权限。

如果您的 Amazon MQ 代理不可公开访问，EventBridge 必须能够访问与该代理关联的 Amazon Virtual Private Cloud (Amazon VPC) 资源。要访问您的 Amazon MQ 代理的 VPC，EventBridge 需要您的源子网的出站互联网访问权限。对于公有子网，它必须是托管 NAT 网关。对于私有子网，它可以是 NAT 网关，也可以是您自己的 NAT。确保此 NAT 具有公共 IP 地址，可以连接到互联网。

使用以下规则配置您的 Amazon VPC 安全组（至少）：

- 入站规则 – 对于没有公共可访问性的代理，允许指定为源的安全组的所有端口上的所有流量。对于具有公共可访问性的代理，允许所有目标的所有端口上的所有流量。
- 出站规则 – 允许所有目标的所有端口上的所有流量传输。

> **Note**
>
> 可通过 Amazon MQ API 发现您的 Amazon VPC 配置。在设置过程中，您不需要对其进行配置。

Amazon Managed Streaming for Apache Kafka 主题作为源

您可以使用 EventBridge Pipes 从 Amazon Managed Streaming for Apache Kafka (Amazon MSK) 主题接收记录。您可以选择筛选或增强这些记录，然后再将它们发送到可用的目标之一进行处理。在设置管道时，可以选择特定于 Amazon MSK 的设置。将数据发送到目标时，EventBridge Pipes 会保持消息代理中记录的顺序。

Amazon MSK 是一项完全托管的服务，可构建并运行应用程序，使用 Apache Kafka 处理流数据。Amazon MSK 简化了运行 Apache Kafka 的集群的设置、扩展和管理。您可以使用 Amazon MSK 配置您的应用程序，以适用于多个可用区并保证 AWS Identity and Access Management (IAM) 的安全性。Amazon MSK 支持多个开源版本的 Kafka。

Amazon MSK 作为源，运行方式与使用 Amazon Simple Queue Service (Amazon SQS) 或 Amazon Kinesis 相似。EventBridge 在内部轮询来自源的新消息，然后同步调用目标。EventBridge 批量读取消息，并将这些消息作为事件有效负载提供给您的函数。最大批处理大小可配置。（默认值为 100 个消息。）

对于基于 Apache Kafka 的源，EventBridge 支持处理控制参数，例如批处理时段和批次大小。

EventBridge 按顺序读取各个分区的消息。EventBridge 处理各个批次后，会提交该批次中消息的偏移量。如果管道的目标为批次中的任何消息返回错误，EventBridge 将重试整批消息，直到处理成功或消息过期为止。

EventBridge 调用目标时会在事件中发送一批消息。事件负载包含一个消息数组。每个数组项目都包含 Amazon MSK 主题和分区标识符的详细信息，以及时间戳和 base64 编码的消息。

示例事件

以下示例事件显示了管道接收到的信息。您可以使用此事件来创建和筛选您的事件模式，或定义输入转换。并非所有字段都可以筛选。有关可筛选字段的更多信息，请参阅 ???。

```
[
  {
```

```
    "eventSource": "aws:kafka",
    "eventSourceArn": "arn:aws:kafka:sa-east-1:123456789012:cluster/
vpc-2priv-2pub/751d2973-a626-431c-9d4e-d7975eb44dd7-2",
    "eventSourceKey": "mytopic-0",
    "topic": "mytopic",
    "partition": "0",
    "offset": 15,
    "timestamp": 1545084650987,
    "timestampType": "CREATE_TIME",
    "key":"abcDEFghiJKLmnoPQRstuVWXyz1234==",
    "value":"SGVsbG8sIHRoaXMgaXMgYSB0ZXN0Lg==",
    "headers": [
      {
        "headerKey": [
          104,
          101,
          97,
          100,
          101,
          114,
          86,
          97,
          108,
          117,
          101
        ]
      }
    ]
  }
]
```

轮询和流的起始位置

请注意，管道创建和更新期间的流源轮询最终将是一致的。

- 在管道创建期间，可能需要几分钟才能开始轮询来自流的事件。
- 在管道更新源轮询配置期间，可能需要几分钟才能停止和重新开始轮询来自流的事件。

这意味着，如果您指定 LATEST 作为流的起始位置，在创建或更新管道期间，管道可能会错过发送的事件。为确保不会错过任何事件，请将流的起始位置指定为 TRIM_HORIZON。

MSK 集群身份验证

EventBridge 需要访问 Amazon MSK 集群、检索记录和执行其他任务的权限。Amazon MSK 支持通过多种选项来控制客户端对 MSK 集群的访问。有关在哪种情况下使用哪种身份验证方法的更多信息，请参阅 ???。

集群访问选项

- 未经身份验证的访问
- SASL/SCRAM 身份验证
- 基于 IAM 角色的身份验证
- 双向 TLS 身份验证
- 配置 mTLS 密钥
- EventBridge 如何选择引导代理

未经身份验证的访问

我们建议仅使用未经身份验证的访问权限进行开发。仅当集群禁用基于 IAM 角色的身份验证时，未经身份验证的访问权限才有效。

SASL/SCRAM 身份验证

Amazon MSK 支持使用传输层安全性协议（TLS）加密进行简单身份验证和安全层/加盐质疑应答身份验证机制（SASL/SCRAM）身份验证。为使 EventBridge 连接到集群，可以将身份验证凭证（登录凭证）存储在 AWS Secrets Manager 密钥中。

有关使用 Secrets Manager 的更多信息，请参阅《Amazon Managed Streaming for Apache Kafka 开发人员指南》中的使用 AWS Secrets Manager 进行用户名和密码身份验证。

Amazon MSK 不支持 SASL/PLAIN 身份验证。

基于 IAM 角色的身份验证

您可以使用 IAM 来验证连接到 MSK 集群的客户端的身份。如果 IAM 身份验证在您的 MSK 集群上处于活动状态，并且您没有为身份验证提供密钥，EventBridge 将自动默认使用 IAM 身份验证。要创建和部署基于 IAM 用户或角色的策略，请使用 IAM 控制台或 API。有关更多信息，请参阅《Amazon Managed Streaming for Apache Kafka 开发人员指南》中的 IAM 访问控制。

要允许 EventBridge 连接到 MSK 集群、读取记录和执行其他必要操作，请将以下权限添加到管道的执行角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:Connect",
                "kafka-cluster:DescribeGroup",
                "kafka-cluster:AlterGroup",
                "kafka-cluster:DescribeTopic",
                "kafka-cluster:ReadData",
                "kafka-cluster:DescribeClusterDynamicConfiguration"
            ],
            "Resource": [
                "arn:aws:kafka:region:account-id:cluster/cluster-name/cluster-uuid",
                "arn:aws:kafka:region:account-id:topic/cluster-name/cluster-uuid/topic-
name",
                "arn:aws:kafka:region:account-id:group/cluster-name/cluster-
uuid/consumer-group-id"
            ]
        }
    ]
}
```

您可以将这些权限范围限定为特定集群、主题和组。有关更多信息，请参阅《Amazon Managed Streaming for Apache Kafka 开发人员指南》中的 Amazon MSK Kafka 操作。

双向 TLS 身份验证

双向 TLS（mTLS）在客户端和服务器之间提供双向身份验证。客户端向服务器发送证书以便服务器验证客户端，而服务器又向客户端发送证书以便客户端验证服务器。

对于 Amazon MSK，EventBridge 充当客户端。您可以配置客户端证书（作为 Secrets Manager 中的密钥），使用 MSK 集群中的代理对 EventBridge 进行身份验证。客户端证书必须由服务器信任存储中的证书颁发机构 (CA) 签名。MSK 集群会向 EventBridge 发送服务器证书，以便使用 EventBridge 对代理进行身份验证。服务器证书必须由 AWS 信任存储中的 CA 签名。

Amazon MSK 不支持自签名服务器证书，因为 Amazon MSK 中的所有代理都使用由 Amazon Trust Services CA 签名的公有证书（EventBridge 默认信任这些证书）。

有关适用于 Amazon MSK 的 mTLS 的更多信息，请参阅《Amazon Managed Streaming for Apache Kafka 开发人员指南》中的双向 TLS 身份验证。

配置 mTLS 密钥

CLIENT_CERTIFICATE_TLS_AUTH 密钥需要证书字段和私有密钥字段。对于加密的私有密钥，密钥需要私有密钥密码。证书和私有密钥必须采用 PEM 格式。

> **Note**
>
> EventBridge 支持 PBES1（而不是 PBES2）私有密钥加密算法。

证书字段必须包含证书列表，首先是客户端证书，然后是任何中间证书，最后是根证书。每个证书都必须按照以下结构在新行中启动：

```
-----BEGIN CERTIFICATE-----
        <certificate contents>
-----END CERTIFICATE-----
```

Secrets Manager 支持最多包含 65536 字节的密钥，这为长证书链提供了充足的空间。

私有密钥必须采用 PKCS #8 格式，并具有以下结构：

```
-----BEGIN PRIVATE KEY-----
         <private key contents>
-----END PRIVATE KEY-----
```

对于加密的私有密钥，请使用以下结构：

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
          <private key contents>
-----END ENCRYPTED PRIVATE KEY-----
```

以下示例显示使用加密私有密钥进行 mTLS 身份验证的密钥内容。对于加密的私有密钥，您可以在密钥中包含私有密钥密码。

```
{
 "privateKeyPassword": "testpassword",
 "certificate": "-----BEGIN CERTIFICATE-----
MIIE5DCCAsygAwIBAgIRAPJdwaFaNRrytHBto0j5BA0wDQYJKoZIhvcNAQELBQAw
...
j0Lh4/+1HfgyE2KlmII36dg4IMzNjAFEBZiCRoPimO40s1cRqtFHXoal0QQbIlxk
```

```
cmUuiAii9R0=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIFgjCCA2qgAwIBAgIQdjNZd6uFf9hbNC5RdfmHrzANBgkqhkiG9w0BAQsFADBb
...
rQoiowbbk5wXCheYSANQIfTZ6weQTgiCHCCbuuMKNVS95FkXm0vqVD/YpXKwA/no
c8PH3PSoAaRwMMgOSA2ALJvbRz8mpg==
-----END CERTIFICATE-----",
 "privateKey": "-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFKzBVBgkqhkiG9w0BBQ0wSDAnBgkqhkiG9w0BBQwwGgQUiAFcK5hT/X7Kjmgp
...
QrSekqF+kWzmB6nAfSzgO9IaoAaytLvNgGTckWeUkWn/V0Ck+LdGUXzAC4RxZnoQ
zp2mwJn2NYB7AZ7+imp0azDZb+8YG2aUCiyqb6PnnA==
-----END ENCRYPTED PRIVATE KEY-----"
}
```

EventBridge 如何选择引导代理

EventBridge 根据集群中可用的身份验证方法，以及您是否提供用于身份验证的密钥来选择引导代理。如果您为 mTLS 或 SASL/SCRAM 提供了密钥，EventBridge 将自动选择该身份验证方法。如果不提供密钥，EventBridge 会选择在集群中处于活动状态的最强身份验证方法。下面是 EventBridge 选择代理的优先级顺序，从最强到最弱的身份验证：

- mTLS（为 mTLS 提供的密钥）
- SASL/SCRAM（为 SASL/SCRAM 提供的密钥）
- SASL IAM（未提供密钥，IAM 身份验证处于活动状态）
- 未经身份验证的 TLS（未提供密钥，IAM 身份验证未处于活动状态）
- 纯文本（未提供密钥，IAM 身份验证和未经身份验证的 TLS 均未处于活动状态）

> **Note**
>
> 如果 EventBridge 无法连接到最安全的代理类型，则不会尝试连接到其他（较弱）代理类型。如果希望 EventBridge 选择较弱的代理类型，请停用集群中所有更强的身份验证方法。

网络配置

EventBridge 必须具有对与 Amazon MSK 集群关联的 Amazon Virtual Private Cloud (Amazon VPC) 资源的访问权限。要访问您的 Amazon MSK 集群的 VPC，EventBridge 需要您的源子网的出站互联网访

问权限。对于公有子网，它必须是托管 NAT 网关。对于私有子网，它可以是 NAT 网关，也可以是您自己的 NAT。确保此 NAT 具有公共 IP 地址，可以连接到互联网。

使用以下规则配置您的 Amazon VPC 安全组（至少）：

- 入站规则 – 允许为源指定之安全组的 Amazon MSK 代理端口（9092 对应纯文本，9094 对应 TLS，9096 对应 SASL，9098 对应 IAM）上的所有流量。
- 出站规则 – 允许所有目标的端口 443 上的所有流量传输。对于为源指定的安全组，允许 Amazon MSK 代理端口（9092 对应纯文本，9094 对应 TLS，9096 对应 SASL，9098 对应 IAM）上的所有流量。

> **Note**
>
> 可通过 Amazon MSK API 发现您的 Amazon VPC 配置。在设置过程中，您不需要对其进行配置。

可自定义的使用者组 ID

将 Apache Kafka 设置为事件源时，您可以指定使用者组 ID。此使用者组 ID 是您希望管道加入的 Apache Kafka 使用者组的现有标识符。您可以使用此功能将任何正在进行的 Apache Kafka 记录处理设置从其他使用者迁移到 EventBridge。

如果指定了使用者组 ID，并且该使用者组中还有其他活跃的轮询器，则 Apache Kafka 会向所有使用者分发消息。换句话说，EventBridge 不会收到 Apache Kafka 主题的所有消息。如果希望 EventBridge 处理主题中的所有消息，请关闭该使用者组中的任何其他轮询器。

此外，如果指定了使用者组 ID，而 Apache Kafka 找到了具有相同 ID 的有效现有使用者组，则 EventBridge 会忽略管道的 `StartingPosition` 参数。相反，EventBridge 开始根据使用者组的已提交偏移量处理记录。如果指定了使用者组 ID，而 Apache Kafka 找不到现有使用者组，则 EventBridge 会使用指定的 `StartingPosition` 配置源。

在所有 Apache Kafka 事件源中，您指定的使用者组 ID 必须是唯一的。在使用指定的使用者组 ID 创建管道后，无法更新此值。

Amazon MSK 源的自动扩缩

当您最初创建 Amazon MSK 源时，EventBridge 会分配一个使用者来处理 Apache Kafka 主题中的所有分区。每个使用者都使用多个并行运行的处理器来处理增加的工作负载。此外，EventBridge 会根据

工作负载自动增加或缩减使用者的数量。要保留每个分区中的消息顺序，使用者的最大数量为主题中每个分区一个使用者。

EventBridge 会按一分钟的间隔时间来评估主题中所有分区的使用者偏移滞后。如果延迟太高，则分区接收消息的速度比 EventBridge 处理消息的速度更快。如有必要，EventBridge 会在主题中添加或删除使用者。增加或移除使用者的扩缩过程会在评估完成后的三分钟内进行。

如果您的目标过载，EventBridge 会减少使用者的数量。此操作通过减少使用者可以检索和发送到管道的消息数，来减少管道的工作负载。

Apache Kafka 作为来源进行直播

Apache Kafka 是开源事件流平台，支持数据管道和流分析等工作负载。你可以使用适用于 A pache Kafka 的亚马逊托管流媒体（亚马逊 MSK），也可以使用自我管理的 Apache Kafka 集群。 AWS 用术语来说，自我管理集群是指未由托管的任何 Apache Kafka 集群。 AWS这包括您自己管理的集群，以及由第三方提供商（例如Confluent CloudCloudKarafka、或）托管的集群Redpanda。

有关集群其他 AWS 托管选项的更多信息，请参阅AWS 大数据博客上的《运行 Apache Kafka 的最佳实践》。 AWS

Apache Kafka 作为来源的运作方式与使用亚马逊简单队列服务 (Amazon SQS) 或亚马逊 Kinesis 类似。 EventBridge在内部轮询来自源的新消息，然后同步调用目标。 EventBridge 批量读取消息，并将其作为事件负载提供给您的函数。最大批处理大小可配置。（默认值为 100 个消息。）

对于基于 Apache Kafka 的源， EventBridge 支持处理控制参数，例如批处理窗口和批处理大小。

EventBridge 当它调用你的管道时，会在事件参数中发送一批消息。事件负载包含一个消息数组。每个数组项目都包含 Apache Kafka 主题和 Apache Kafka 分区标识符的详细信息，以及时间戳和 base64 编码的消息。

示例事件

以下示例事件显示了管道接收到的信息。您可以使用此事件来创建和筛选您的事件模式，或定义输入转换。并非所有字段都可以筛选。有关可筛选字段的更多信息，请参阅 ???。

```
[
  {
    "eventSource": "SelfManagedKafka",
    "bootstrapServers": "b-2.demo-cluster-1.a1bcde.c1.kafka.us-
east-1.amazonaws.com:9092,b-1.demo-cluster-1.a1bcde.c1.kafka.us-
east-1.amazonaws.com:9092",
    "eventSourceKey": "mytopic-0",
```

```
    "topic": "mytopic",
    "partition": 0,
    "offset": 15,
    "timestamp": 1545084650987,
    "timestampType": "CREATE_TIME",
    "key":"abcDEFghiJKLmnoPQRstuVWXyz1234==",
    "value":"SGVsbG8sIHRoaXMgaXMgYSB0ZXN0Lg==",
    "headers": [
      {
        "headerKey": [
          104,
          101,
          97,
          100,
          101,
          114,
          86,
          97,
          108,
          117,
          101
        ]
      }
    ]
  }
]
```

Apache Kafka 集群身份验证

EventBridge Pipes 支持多种通过自我管理的 Apache Kafka 集群进行身份验证的方法。请确保将 Apache Kafka 集群配置为使用支持的下列身份验证方法之一。有关 Apache Kafka 安全的更多信息，请参阅 Apache Kafka 文档的安全部分。

VPC 访问

如果您使用的是自我管理的 Apache Kafka 环境，只有您的 VPC 中的 Apache Kafka 用户才能访问您的 Apache Kafka 代理，则必须在 Apache Kafka 源中配置亚马逊虚拟私有云（亚马逊 VPC）。

SASL/SCRAM 身份验证

EventBridge Pipes 支持带有传输层安全 (TLS) 加密的简单身份验证和安全层/加盐质询响应身份验证机制 (SASL/SCRAM) 身份验证。 EventBridge Pipes 发送加密的凭据以向集群进行身份验证。有关 SASL/SCRAM 身份验证的更多信息，请参阅 RFC 5802。

EventBridge 管道支持采用 TLS 加密的 SASL/PLAIN 身份验证。使用 SASL/PLAIN 身份验证，Pi EventBridge pes 会将凭据以明文（未加密）的形式发送到服务器。

为了 SASL 身份验证，需要将登录凭证作为密钥存储在 AWS Secrets Manager中。

双向 TLS 身份验证

双向 TLS（mTLS）在客户端和服务器之间提供双向身份验证。客户端向服务器发送证书以便服务器验证客户端，而服务器又向客户端发送证书以便客户端验证服务器。

在自我管理的 Apache Kafka 中，Pip EventBridge es 充当客户端。您可以配置客户端证书（作为 Secrets Manager 中的密钥）来向 Apache Kafka 代理对 Pip EventBridge es 进行身份验证。客户端证书必须由服务器信任存储中的证书颁发机构 (CA) 签名。

Apache Kafka 集群向 Pipes 发送服务器证书，用于使用 Pip EventBridge es 对 Apache Kafka 代理进行身份验证。 EventBridge 服务器证书可以是公有 CA 证书。也可以是私有 CA/自签名证书。公有 CA 证书必须由 Pip EventBridge es 信任存储区中的 CA 签名。对于私有 CA/自签名证书，您可以配置服务器根 CA 证书（作为 Secrets Manager 中的密钥）。 EventBridge Pipes 使用根证书来验证 Apache Kafka 代理。

有关 mTLS 的更多信息，请参阅为作为源的 Amazon MSK 引入双向 TLS 身份验证。

配置客户端证书密钥

CLIENT_CERTIFICATE_TLS_AUTH 密钥需要证书字段和私有密钥字段。对于加密的私有密钥，密钥需要私有密钥密码。证书和私有密钥必须采用 PEM 格式。

> **Note**
>
> EventBridge Pipes 支持 PBES1（但不是 PBES2）私钥加密算法。

证书字段必须包含证书列表，首先是客户端证书，然后是任何中间证书，最后是根证书。每个证书都必须按照以下结构在新行中启动：

```
-----BEGIN CERTIFICATE-----
        <certificate contents>
-----END CERTIFICATE-----
```

Secrets Manager 支持最多包含 65536 字节的密钥，这为长证书链提供了充足的空间。

私有密钥必须采用 PKCS #8 格式，并具有以下结构：

```
-----BEGIN PRIVATE KEY-----
         <private key contents>
-----END PRIVATE KEY-----
```

对于加密的私有密钥，请使用以下结构：

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
          <private key contents>
-----END ENCRYPTED PRIVATE KEY-----
```

以下示例显示使用加密私有密钥进行 mTLS 身份验证的密钥内容。对于加密的私有密钥，可以在密钥中包含私有密钥密码。

```
{
 "privateKeyPassword": "testpassword",
 "certificate": "-----BEGIN CERTIFICATE-----
MIIE5DCCAsygAwIBAgIRAPJdwaFaNRrytHBto0j5BA0wDQYJKoZIhvcNAQELBQAw
...
j0Lh4/+1HfgyE2KlmII36dg4IMzNjAFEBZiCRoPimO40s1cRqtFHXoal0QQbIlxk
cmUuiAii9R0=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIFgjCCA2qgAwIBAgIQdjNZd6uFf9hbNC5RdfmHrzANBgkqhkiG9w0BAQsFADBb
...
rQoiowbbk5wXCheYSANQIfTZ6weQTgiCHCCbuuMKNVS95FkXm0vqVD/YpXKwA/no
c8PH3PSoAaRwMMgOSA2ALJvbRz8mpg==
-----END CERTIFICATE-----",
 "privateKey": "-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFKzBVBgkqhkiG9w0BBQ0wSDAnBgkqhkiG9w0BBQwwGgQUiAFcK5hT/X7Kjmgp
...
QrSekqF+kWzmB6nAfSzgO9IaoAaytLvNgGTckWeUkWn/V0Ck+LdGUXzAC4RxZnoQ
zp2mwJn2NYB7AZ7+imp0azDZb+8YG2aUCiyqb6PnnA==
-----END ENCRYPTED PRIVATE KEY-----"
}
```

配置服务器根 CA 证书密钥

如果您的 Apache Kafka 代理使用 TLS 加密（具有由私有 CA 签名的证书），则创建此密钥。您可以将 TLS 加密用于 VPC、SASL/SCRAM、SASL/PLAIN 或 mTLS 身份验证。

服务器根 CA 证书密钥需要一个字段，其中包含 PEM 格式的 Apache Kafka 代理的根 CA 证书。以下示例显示密钥的结构。

```
{
     "certificate": "-----BEGIN CERTIFICATE-----
  MIID7zCCAtegAwIBAgIBADANBgkqhkiG9w0BAQsFADCBmDELMAkGA1UEBhMCVVMx
  EDAOBgNVBAgTB0FyaXpvbmExEzARBgNVBAcTClNjb3R0c2RhbGUxJTAjBgNVBAoT
  HFN0YXJmaWVsZCBUZWNobm9sb2dpZXMsIEluYy4xOzA5BgNVBAMTMlN0YXJmaWVs
  ZCBTZXJ2aWNlcyBSb290IENlcnRpZmljYXRlIEF1dG...
  -----END CERTIFICATE-----"
```

网络配置

如果您使用的是使用私有 VPC 连接的自我管理的 Apache Kafka 环境，则 EventBridge 必须有权访问与您的 Apache Kafka 代理关联的亚马逊虚拟私有云（亚马逊 VPC）资源。要访问您的 Apache Kafka 集群的 VPC， EventBridge 需要您的源子网的出站互联网访问权限。对于公有子网，它必须是托管 NAT 网关。对于私有子网，它可以是 NAT 网关，也可以是您自己的 NAT。确保此 NAT 具有公共 IP 地址，可以连接到互联网。

使用以下规则配置您的 Amazon VPC 安全组（至少）：

- 入站规则 - 允许为源指定之安全组的 Apache Kafka 代理端口（9092 对应纯文本，9094 对应 TLS，9096 对应 SASL，9098 对应 IAM）上的所有流量。
- 出站规则 – 允许所有目标的端口 443 上的所有流量传输。允许为源指定之安全组的 Apache Kafka 代理端口（9092 对应纯文本，9094 对应 TLS，9096 对应 SASL，9098 对应 IAM）上的所有流量。

使用 Apache Kafka 来源进行消费者自动扩展

最初创建 Apache Kafka 源时， EventBridge 会分配一个使用者来处理 Kafka 主题中的所有分区。每个使用者都使用多个并行运行的处理器来处理增加的工作负载。此外，还可以根据工作负载 EventBridge 自动增加或缩小使用者的数量。要保留每个分区中的消息顺序，使用者的最大数量为主题中每个分区一个使用者。

每隔一分钟， EventBridge 评估主题中所有分区的消费偏移延迟。如果延迟太高，则表示分区接收消息的速度 EventBridge 快于处理消息的速度。如有必要，在主题中 EventBridge 添加或删除消费者。增加或移除使用者的扩缩过程会在评估完成后的三分钟内进行。

如果您的目标超负荷，则 EventBridge 减少消费者的数量。此操作通过减少使用者可以检索和发送到函数的消息数来减少函数的工作负载。

将 Amazon Simple Queue Service 作为源

您可以使用 Pip EventBridge es 接收来自亚马逊 SQS 队列的记录。然后，您可以选择筛选或增强这些记录，然后再将它们发送到可用的目标进行处理。

您可以使用管道来处理亚马逊简单队列服务 (Amazon SQS) 队列中的消息。 EventBridge 管道支持标准队列和先进先出 (FIFO) 队列。在 Amazon SQS 中，您可以通过将来自一个应用程序组件的任务发送到一个队列中并异步处理它们来进行分载。

EventBridge 轮询队列并使用包含队列消息的事件同步调用您的管道。 EventBridge 批量读取消息，每批次调用管道一次。当您的管道成功处理批处理时， EventBridge 会将其消息从队列中删除。

默认情况下，可以同时 EventBridge 轮询队列中最多 10 条消息，然后将该批次发送到您的管道。为避免在记录数量较少的情况下调用管道，您可以配置批处理时间窗，让事件源缓冲最多 5 分钟的记录。在调用管道之前， EventBridge 继续轮询来自 Amazon SQS 标准队列的消息，直到出现以下情况之一：

- 批处理时间窗过期。
- 已达到调用负载大小配额。
- 已达到配置的批次大小上限。

> **Note**
>
> 如果您使用的是批处理窗口，并且您的 Amazon SQS 队列流量较低，则 EventBridge 可能需要等待 20 秒钟才能调用管道。即使您将批处理时间窗设置为少于 20 秒，情况依然如此。对于 FIFO 队列，记录包含与重复数据消除和顺序相关的其他属性。

EventBridge 读取批次时，消息会保留在队列中，但在队列的可见性超时时间内会被隐藏。如果您的管道成功处理了批处理，则会从队列中 EventBridge 删除消息。默认情况下，如果您的管道在处理某个批次时遇到错误，则该批次中的所有消息都会在队列中重新可见。因此，管道代码必须能够多次处理同一条消息，而不会产生意外的副作用。您可以在管道响应中包括批处理项目失败次数，来修改此再处理行为。以下示例显示了包含两条消息的批次事件。

示例事件

以下示例事件显示了管道接收到的信息。您可以使用此事件来创建和筛选您的事件模式，或定义输入转换。并非所有字段都可以筛选。有关可筛选字段的更多信息，请参阅 ???。

标准队列


```
[
  {
    "messageId": "059f36b4-87a3-44ab-83d2-661975830a7d",
    "receiptHandle": "AQEBwJnKyrHigUMZj6rYigCgxlaS3SLy0a...",
    "body": "Test message.",
    "attributes": {
      "ApproximateReceiveCount": "1",
      "SentTimestamp": "1545082649183",
      "SenderId": "AIDAIENQZJOLO23YVJ4VO",
      "ApproximateFirstReceiveTimestamp": "1545082649185"
    },
    "messageAttributes": {},
    "md5OfBody": "e4e68fb7bd0e697a0ae8f1bb342846b3",
    "eventSource": "aws:sqs",
    "eventSourceARN": "arn:aws:sqs:us-east-2:123456789012:my-queue",
    "awsRegion": "us-east-2"
  },
  {
    "messageId": "2e1424d4-f796-459a-8184-9c92662be6da",
    "receiptHandle": "AQEBzWwaftRI0KuVm4tP+/7q1rGgNqicHq...",
    "body": "Test message.",
    "attributes": {
      "ApproximateReceiveCount": "1",
      "SentTimestamp": "1545082650636",
      "SenderId": "AIDAIENQZJOLO23YVJ4VO",
      "ApproximateFirstReceiveTimestamp": "1545082650649"
    },
    "messageAttributes": {},
    "md5OfBody": "e4e68fb7bd0e697a0ae8f1bb342846b3",
    "eventSource": "aws:sqs",
    "eventSourceARN": "arn:aws:sqs:us-east-2:123456789012:my-queue",
    "awsRegion": "us-east-2"
  }
]
```


FIFO 队列


```
[
  {
    "messageId": "11d6ee51-4cc7-4302-9e22-7cd8afdaadf5",
    "receiptHandle": "AQEBBX8nesZEXmkhsmZeyIE8iQAMig7qw...",
```

```
    "body": "Test message.",
    "attributes": {
      "ApproximateReceiveCount": "1",
      "SentTimestamp": "1573251510774",
      "SequenceNumber": "18849496460467696128",
      "MessageGroupId": "1",
      "SenderId": "AIDAIO23YVJENQZJOL4VO",
      "MessageDeduplicationId": "1",
      "ApproximateFirstReceiveTimestamp": "1573251510774"
    },
    "messageAttributes": {},
    "md5OfBody": "e4e68fb7bd0e697a0ae8f1bb342846b3",
    "eventSource": "aws:sqs",
    "eventSourceARN": "arn:aws:sqs:us-east-2:123456789012:fifo.fifo",
    "awsRegion": "us-east-2"
  }
]
```

扩展和处理

对于标准队列， EventBridge 使用长轮询来轮询队列，直到队列变为活动状态。当有消息可用时，最多 EventBridge 读取五个批次并将其发送到您的管道。如果消息仍然可用，则将正在读取批处理的进程的数量每分钟最多 EventBridge 增加 300 个实例。管道可以同时处理的最大批次数量为 1,000。

对于 FIFO 队列，按接收消息的顺序向管道 EventBridge 发送消息。向 FIFO 队列发送消息时，需要指定消息组 ID。Amazon SQS 便于按顺序将同一个群组中的消息传送到 EventBridge。 EventBridge 将收到的消息分组，一次只能为一个群组发送一批。如果您的管道返回错误，则在 EventBridge 收到来自同一组的其他消息之前，管道会尝试对受影响的消息进行所有重试。

配置队列以与 Pip EventBridge es 配合使用

创建一个 Amazon SQS 队列，用作管道的源。然后配置队列，让管道有时间处理每批事件，并留出 EventBridge 时间在扩展时重试以响应限制错误。

为使您的管道有时间处理每批记录，请将源队列的可见性超时至少设置为管道富集和目标组件合并运行时的六倍。如果您的管道在处理前一批次时受到限制，则额外的时间允许重试。 EventBridge

如果您的管道多次都未能处理某条消息，Amazon SQS 可以将其发送到某个死信队列。当您的管道返回错误时，请将其 EventBridge 保留在队列中。在发生可见性超时之后， EventBridge 重新接收消息。要在多次接收之后将消息发送到第二个队列，请在源队列上配置死信队列。

> **Note**
>
> 确保在源队列上配置死信队列，而不是在管道上配置。您在管道上配置的死信队列用于管道的异步调用队列，而不是用于源队列。

如果您的管道返回错误，或者由于达到并发上限而无法调用，则处理可能会成功，但需要额外尝试。要在将消息发送到死信队列之前给予更多处理机会，请将源队列重新驱动策略的 `maxReceiveCount` 至少设置为 5。

报告批处理项目失败

在 EventBridge 使用和处理来自源的流数据时，默认情况下，它会检查批次的最高序号，但前提是批处理完全成功时。为避免重新处理失败批次中已成功处理的消息，您可以配置富集或目标，返回对象来指示哪些消息处理成功、哪些失败。这称为部分批处理响应。

有关更多信息，请参阅???。

成功和失败的条件

如果您返回以下任意一项，则将批次 EventBridge 视为完全成功：

- 空的 `batchItemFailure` 列表
- Null `batchItemFailure` 列表
- 空的 `EventResponse`
- Null `EventResponse`

如果您返回以下任何内容，则会将批次 EventBridge 视为完全失败：

- 空字符串 `itemIdentifier`
- Null `itemIdentifier`
- 包含错误密钥名的 `itemIdentifier`

EventBridge 根据您的重试策略重试失败。

亚马逊 EventBridge 管道过滤

使用 Pip EventBridge es，您可以过滤给定源的事件并仅处理其中的一部分。这种筛选的工作方式与使用事件模式对 EventBridge 事件总线或 Lambda 事件源映射进行筛选的方式相同。有关事件模式的更多信息，请参阅 ???。

筛选标准 `FilterCriteria` 对象是由筛选条件 `Filters` 列表组成的结构。每个筛选条件都是用于定义筛选模式 (`Pattern`) 的结构。`Pattern` 是 JSON 筛选条件规则的字符串表示。`FilterCriteria` 对象与以下示例类似：

```
{
  "Filters": [
    {"Pattern": "{ \"Metadata1\": [ rule1 ], \"data\": { \"Data1\": [ rule2 ] }}"
    }
  ]
}
```

为了更清楚起见，以下是在纯 JSON 中展开的筛选条件 `Pattern` 的值。

```
{
  "Metadata1": [ pattern1 ],
  "data": {"Data1": [ pattern2 ]}
}
```

`FilterCriteria` 对象的主要部分是元数据属性和数据属性。

- 元数据属性是指事件对象的字段。在示例中，`FilterCriteria.Metadata1` 表示元数据属性。
- 数据属性是指事件主体的字段。在示例中，`FilterCriteria.Data1` 表示数据属性。

例如，假设您的 Kinesis 流包含这样的事件：

```
{
  "kinesisSchemaVersion": "1.0",
  "partitionKey": "1",
  "sequenceNumber": "49590338271490256608559692538361571095921575989136588898",
  "data": {"City": "Seattle",
    "State": "WA",
    "Temperature": "46",
    "Month": "December"
  },
```

```
  "approximateArrivalTimestamp": 1545084650.987
}
```

当事件流经您的管道时，采用 base64 编码 data 字段将如下所示：

```
{
  "kinesisSchemaVersion": "1.0",
  "partitionKey": "1",
  "sequenceNumber": "49590338271490256608559692538361571095921575989136588898",
  "data": "SGVsbG8sIHRoaXMgaXMgYSB0ZXN0Lg==",
  "approximateArrivalTimestamp": 1545084650.987
  "eventSource": "aws:kinesis",
  "eventVersion": "1.0",
  "eventID":
 "shardId-000000000006:49590338271490256608559692538361571095921575989136588898",
  "eventName": "aws:kinesis:record",
  "invokeIdentityArn": "arn:aws:iam::123456789012:role/lambda-role",
  "awsRegion": "us-east-2",
  "eventSourceARN": "arn:aws:kinesis:us-east-2:123456789012:stream/lambda-stream"
},
```

Kinesis 事件的元数据属性 是 data 对象之外的任何字段，例如 partitionKey 或 sequenceNumber。

Kinesis 事件的数据属性 是 data 对象中的任何字段，例如 City 或 Temperature。

当您进行筛选以匹配此事件时，可以针对解码后的字段使用筛选条件。例如，要筛选 partitionKey 和City，需要使用以下筛选条件：

```
{ "partitionKey": [ "1" ], "data": { "City": [ "Seattle" ] }
```

创建事件过滤器时，Pi EventBridge pes 可以访问事件内容。这些内容要么是 JSON 转义的，例如 Amazon SQS body 字段，要么是 base64 编码的，例如 Kinesis data 字段。如果您的数据是有效的 JSON，则您的输入模板或目标参数的 JSON 路径可以直接引用内容。例如，如果 Kinesis 事件源是有效的 JSON，您可以使用 <$.data.someKey> 引用变量。

创建事件模式时，您可以根据源 API 发送的字段进行筛选，而不是根据轮询操作添加的字段进行筛选。以下字段不能用于事件模式：

- awsRegion
- eventSource

- eventSourceARN
- eventVersion
- eventID
- eventName
- invokeIdentityArn
- eventSourceKey

消息和数据字段

每个 EventBridge Pipe 源都包含一个包含核心消息或数据的字段。我们将它们称为消息 字段或数据 字段。这些字段之所以特别，是因为它们可能是 JSON 转义或 base64 编码的，但当它们是有效 JSON 时，可以使用 JSON 模式对其进行筛选，就像正文没有被转义一样。这些字段的内容也可以无缝地用于输入转换器。

正确筛选 Amazon SQS 消息

如果 Amazon SQS 消息不符合您的筛选标准，则 EventBridge 会自动将该消息从队列中删除。您无需在 Amazon SQS 中手动删除这些消息。

对于 Amazon SQS，消息 body 可以是任何字符串。但如果您的 FilterCriteria 期望 body 为有效的 JSON 格式，则可能会导致问题。反之亦然，如果传入的消息 body 为有效的 JSON 格式，但您的筛选标准期望 body 为纯字符串，会导致出现意外行为。

要避免此问题，请确保 FilterCriteria 中 body 的格式与您从队列中收到的消息中的 body 的期望格式一致。在筛选您的邮件之前， EventBridge 会自动评估传入消息的格式body和您的过滤器模式。body如果不匹配，则 EventBridge 丢弃该消息。下表汇总了此评估：

传入消息 **body** 格式	筛选条件模式 **body** 格式	导致的操作
纯字符串	纯字符串	EventBridge 根据您的筛选条件进行筛选。
纯字符串	数据属性中没有筛选条件模式	EventBridge 根据您的筛选条件进行筛选（仅限于其他元数据属性）。
纯字符串	有效 JSON	EventBridge 丢弃消息。

传入消息 **body** 格式	筛选条件模式 **body** 格式	导致的操作
有效 JSON	纯字符串	EventBridge 丢弃消息。
有效 JSON	数据属性中没有筛选条件模式	EventBridge 根据您的筛选条件进行筛选（仅限于其他元数据属性）。
有效 JSON	有效 JSON	EventBridge 根据您的筛选条件进行筛选。

如果您未body将其包含在中`FilterCriteria`，则 EventBridge 跳过此检查。

正确筛选 Kinesis 和 DynamoDB 消息

在筛选标准处理了 Kinesis 或 DynamoDB 记录后，流迭代器会跳过此记录。如果记录不符合筛选标准，您无需从事件源手动删除记录。保留期结束后，Kinesis 和 DynamoDB 会自动删除这些旧记录。如果您希望提前删除记录，请参阅更改数据保留期。

要正确筛选流事件源中的事件，数据字段和数据字段的筛选条件都必须为有效的 JSON 格式。（对于 Kinesis，数据字段为 `data`。对于 DynamoDB，数据字段为 `dynamodb`。）如果任一字段都不是有效的 JSON 格式，则 EventBridge 会丢弃该消息或引发异常。下表汇总了具体行为：

传入数据格式（**`data`** 或 **`dynamodb`**）	数据属性中的筛选条件模式格式	导致的操作
有效 JSON	有效 JSON	EventBridge 根据您的筛选条件进行筛选。
有效 JSON	数据属性中没有筛选条件模式	EventBridge 根据您的筛选条件进行筛选（仅限于其他元数据属性）。
有效 JSON	非 JSON	EventBridge 在管道或更新时抛出异常。数据属性的筛选条件模式必须为有效的 JSON 格式。

传入数据格式（**data** 或 **dynamodb**）	数据属性中的筛选条件模式格式	导致的操作
非 JSON	有效 JSON	EventBridge 丢掉了记录。
非 JSON	数据属性中没有筛选条件模式	EventBridge 根据您的筛选条件进行筛选（仅限于其他元数据属性）。
非 JSON	非 JSON	EventBridge 在创建或更新管道时会引发异常。数据属性的筛选条件模式必须为有效的 JSON 格式。

正确筛选 Amazon Managed Streaming for Apache Kafka、自托管 Apache Kafka 和 Amazon MQ 消息。

对于 Amazon MQ 来源，消息字段为 `data`。对于 Apache Kafka 源（Amazon MSK 和自托管 Apache Kafka），有两个消息字段：`key` 和 `value`。

EventBridge 丢弃与过滤器中包含的所有字段不匹配的邮件。对于 Apache Kafka，成功调用该函数后，为匹配和不匹配的消息 EventBridge 提交偏移量。对于 Amazon MQ，在成功调用该函数后 EventBridge 确认匹配的消息，并在筛选不匹配的消息时确认这些消息。

Apache Kafka 和 Amazon MQ 消息必须是 UTF-8 编码的字符串，可以是纯字符串或 JSON 格式。那是因为在应用筛选条件之前，将 Apache Kafka 和 Amazon MQ 字节数组 EventBridge 解码为 UTF-8。如果您的消息使用其他编码（例如 UTF-16 或 ASCII），或者消息格式与格式不匹配，则仅 EventBridge 处理元数据过滤器。`FilterCriteria`下表汇总了具体行为：

传入消息格式（**data** 或 **key** 和 **value**）	消息属性的筛选条件模式格式	导致的操作
纯字符串	纯字符串	EventBridge 根据您的筛选条件进行筛选。

传入消息格式（**data** 或 **key** 和 **value**）	消息属性的筛选条件模式格式	导致的操作
纯字符串	数据属性中没有筛选条件模式	EventBridge 根据您的筛选条件进行筛选（仅限于其他元数据属性）。
纯字符串	有效 JSON	EventBridge 根据您的筛选条件进行筛选（仅限于其他元数据属性）。
有效 JSON	纯字符串	EventBridge 根据您的筛选条件进行筛选（仅限于其他元数据属性）。
有效 JSON	数据属性中没有筛选条件模式	EventBridge 根据您的筛选条件进行筛选（仅限于其他元数据属性）。
有效 JSON	有效 JSON	EventBridge 根据您的筛选条件进行筛选。
非 UTF-8 编码字符串	JSON、纯字符串或无模式	EventBridge 根据您的筛选条件进行筛选（仅限于其他元数据属性）。

Lambda ESM 和 Pipes 之间的区别 EventBridge

筛选事件时，Lambda ESM 和 Pip EventBridge es 的操作方式通常相同。主要区别在于 ESM 有效负载中不存在 `eventSourceKey` 字段。

Amazon EventBridge Pipes 事件富集

通过 EventBridge Pipes 的富集步骤，您可以在将数据从源发送到目标之前，对其进行增强。例如，您可能会收到票证已创建 事件，但其中不包含完整票证数据。使用富集，您可以使用 Lambda 函数调用 `get-ticket` API，以获取完整的票证详情。然后，管道可以将该信息发送到目标。

在 EventBridge 中设置管道时，您可以配置以下富集：

- API 目标
- Amazon API Gateway
- Lambda 函数
- Step Functions 状态机

> **Note**
>
> EventBridge Pipes 仅支持快速工作流程作为富集。

EventBridge 同步调用富集是因为在调用目标之前，它必须等待富集的响应。

富集响应的大小上限为 6MB。

您还可以在发送数据进行增强之前，对从源接收到的数据进行转换。有关更多信息，请参阅???。

使用富集筛选事件

EventBridge Pipes 会将富集响应直接传递给配置的目标。其中包括支持批处理的目标的数组响应。有关批处理行为的更多信息，请参阅 ???。您还可以使用富集作为筛选器，使传递的事件数少于从源接收到的事件数。如果您不想调用目标，请返回一个空响应，例如 ""、{} 或 []。

> **Note**
>
> 如果要使用空负载调用目标，请返回一个包含空 JSON [{}] 的数组。

调用富集

EventBridge 同步调用富集（调用类型设置为 REQUEST_RESPONSE）是因为在调用目标之前，它必须等待富集的响应。

> **Note**
>
> 对于 Step Functions 状态机，EventBridge 仅支持快速工作流程作为富集，因为它们可以同步调用。

Amazon EventBridge Pipes 目标

您可以将管道中的数据发送到特定目标。在 EventBridge 中设置管道时，您可以配置以下目标：

- API 目标
- API Gateway
- 批处理作业队列
- CloudWatch 日志组
- ECS 任务
- 同一账户和区域中的事件总线
- Firehose 传输流
- Inspector 评估模板
- Kinesis 流
- Lambda 函数（同步或异步）
- Redshift 集群数据 API 查询
- SageMaker 管道
- Amazon SNS 主题（不支持 SNS FIFO 主题）
- Amazon SQS 队列
- Step Functions 状态机
  - 快速工作流程（同步或异步）
  - 标准工作流程（异步）

目标参数

有些目标服务不会将事件负载发送给目标，而是将事件视为调用特定 API 的触发器。EventBridge 使用 `PipeTargetParameters` 指定向 API 发送哪些信息。这些功能包括：

- API 目标（发送到 API 目标的数据必须与 API 的结构相匹配。必须使用 `InputTemplate` 对象来确保数据的结构正确。如果要包含原始事件负载，请在 `InputTemplate` 中引用它。）
- API Gateway（发送到 API Gateway 的数据必须与 API 的结构相匹配。必须使用 `InputTemplate` 对象来确保数据的结构正确。如果要包含原始事件负载，请在 `InputTemplate` 中引用它。）
- `PipeTargetRedshiftDataParameters`（Amazon Redshift 数据 API 集群）

- PipeTargetSageMakerPipelineParameters（Amazon SageMaker 运行时系统模型构建管道）
- PipeTargetBatchJobParameters (AWS Batch)

> **Note**
>
> EventBridge 不支持所有 JSON 路径语法，并在运行时对其进行评估。支持的语法包括：
>
> - 点表示法（例如 $.detail）
> - 短划线
> - 下划线
> - 字母数字字符
> - 数组索引
> - 通配符 (*)

动态路径参数

EventBridge Pipes 目标参数支持可选的动态 JSON 路径语法。您可以使用此语法指定 JSON 路径而不是静态值（例如 $.detail.state）。整个值必须是 JSON 路径，不能仅仅是其中的一部分。例如，RedshiftParameters.Sql 可以是 $.detail.state 但不能是 "SELECT * FROM $.detail.state"。这些路径在运行时会动态替换为来自指定路径的事件负载本身的数据。动态路径参数不能引用新值或输入转换生成的转换后的值。动态参数 JSON 路径支持的语法与转换输入时的语法相同。有关更多信息，请参阅???。

动态语法可用于所有 EventBridge Pipes 富集和目标参数的所有字符串、非枚举字段，但以下参数除外：

- PipeTargetCloudWatchLogsParameters.LogStreamName
- PipeTargetEventBridgeEventBusParameters.EndpointId
- PipeEnrichmentHttpParameters.HeaderParameters
- PipeTargetHttpParameters.HeaderParameters

例如，要将管道 Kinesis 目标的 PartitionKey 设置为源事件中的自定义密钥，请将 KinesisTargetParameter.PartitionKey 设置为：

- "$.data.*someKey*"，适用于 Kinesis 源
- "$.body.*someKey*"，适用于 Amazon SQS 源

然后，如果事件负载是有效的 JSON 字符串，例如 {"*someKey*":"*someValue*"}，EventBridge 会从 JSON 路径中提取该值并将其用作目标参数。在此示例中，EventBridge 会将 Kinesis PartitionKey 设置为 "*someValue*"。

权限

为了对您拥有的资源执行 API 调用，EventBridge Pipes 需要相应权限。EventBridge Pipes 将您在管道中指定的 IAM 角色用于富集，将 IAM 主体 pipes.amazonaws.com 用于目标调用。

调用目标

EventBridge 具有以下方法用于调用目标：

- 同步（调用类型设置为 REQUEST_RESPONSE）- EventBridge 在继续操作之前等待目标的响应。
- 异步（调用类型设置为 FIRE_AND_FORGET）- EventBridge 不会等待收到响应后再继续操作。

默认情况下，对于排序源的管道，EventBridge 会同步调用目标，因为在继续处理下一事件之前需要目标的响应。

如果某个源不强制执行排序（例如标准 Amazon SQS 队列），EventBridge 可以同步或异步调用支持的目标。

使用 Lambda 函数和 Step Functions 状态机，您可以配置调用类型。

> **Note**
>
> 对于 Step Functions 状态机，必须异步调用标准工作流程。

EventBridge Pipes 目标的具体信息

AWS Batch 作业队列

所有 AWS Batch submitJob 参数都使用 BatchParameters 显式配置，与所有 Pipe 参数一样，这些参数可以使用传入事件负载的 JSON 路径进行动态配置。

CloudWatch 日志组

无论您是否使用输入转换器，事件负载都将用作日志消息。您可以通过 PipeTarget 中的 CloudWatchLogsParameters 设置 Timestamp（或目标的显式 LogStreamName）。与所有管道参数一样，这些参数可以使用传入事件负载的 JSON 路径进行动态配置。

Amazon ECS 任务

所有 Amazon ECS runTask 参数都是通过 EcsParameters 显式配置的。与所有管道参数一样，这些参数可以使用传入事件负载的 JSON 路径进行动态配置。

Lambda 函数和 Step Functions 工作流程

Lambda 和 Step Functions 没有批处理 API。要处理来自管道源的事件批次，批次将转换为 JSON 数组，并作为输入传递给 Lambda 或 Step Functions 目标。有关更多信息，请参阅???。

Amazon Pip EventBridge es 批处理和并发

批处理行为

EventBridge Pipes 支持从源和支持它的目标进行批处理。此外，AWS Lambda 和 AWS Step Functions还支持对批处理进行富集。由于不同的服务支持不同程度的批处理，因此不能为管道配置比目标支持的大小更大的批处理。例如，Amazon Kinesis 流源支持的批次大小上限为 10,000 条记录，但 Amazon Simple Queue Service 支持将每批最多 10 条消息作为目标。因此，从 Kinesis 流到 Amazon SQS 队列的管道，在源上配置的批次大小上限可以为 10。

如果您使用不支持批处理的富集或目标配置管道，则无法在源上激活批处理。

在源上激活批处理时，JSON 记录数组将通过管道传递，然后映射到支持的富集或目标中的批处理 API。输入转换器分别应用于数组中的每个 JSON 记录，而不是整个数组。有关这些数组的示例，请参阅 ???并选择特定源。即使批次大小为 1，Pipes 也会将批处理 API 用于支持的富集或目标。如果富集或目标没有批处理 API，但收到完整的 JSON 负载，例如 Lambda 和 Step Functions，则整个 JSON 数组将在一个请求中发送。即使批次大小为 1，请求也将作为 JSON 数组发送。

如果在源位置将某管道配置为进行批处理，并且目标也支持批处理，则可以从富集中返回 JSON 项目的数组。该数组可以包含比原始源更短或更长的数组。但是，如果该数组大于目标支持的批次大小，管道将不会调用目标。

支持的可批处理目标

目标	最大批次大小
CloudWatch 日志	10000
EventBridge 活动巴士	10
Firehose 直播	500
Kinesis 流	500
Lambda 函数	客户定义
Step Functions 状态机	客户定义
Amazon SNS 主题	10
Amazon SQS 队列	10

以下富集和目标将接收完整的批处理事件负载进行处理，并受限于事件总负载大小、而不是批次大小：

- Step Functions 状态机（262144 个字符）
- Lambda 函数 (6MB)

部分批处理故障

对于 Amazon SQS 和流源，例如 Kinesis 和 DynamoDB，Pipes 支持对目标故障进行部分 EventBridge 批量故障处理。如果目标支持批处理并且只有部分批处理成功，则 EventBridge 会自动重试对剩余的有效负载进行批处理。对于内容最 up-to-date 丰富的内容，此重试贯穿整个管道，包括重新调用任何已配置的丰富内容。

不支持对富集进行部分批处理故障处理。

对于 Lambda 和 Step Functions 目标，您还可以从目标返回具有已定义结构的负载，来指定部分故障。这表示需要重试的事件。

部分故障负载结构的示例

```
{
```

```
  "batchItemFailures": [
    {
      "itemIdentifier": "id2"
    },
    {
      "itemIdentifier": "id4"
    }
]
```

在此示例中，`itemIdentifier` 与您的目标所处理事件的原始源 ID 相匹配。对于 Amazon SQS，此 ID 为 `messageId`。对于 Kinesis 和 DynamoDB，此 ID 为 `eventID`。为了 EventBridge 使 Pipes 能够充分处理来自目标的部分批处理故障，这些字段需要包含在扩充返回的任何数组有效载荷中。

吞吐量和并发行为

管道接收并传递到富集或目标的每个事件或事件批次，都被视为管道执行。处于 `STARTED` 状态的管道会持续轮询来自源的事件，并根据可用的积压和配置的批处理设置向上和向下扩展。

有关并发管道执行的配额，以及每个账户和区域的管道数量，请参阅 ???。

默认情况下，单个管道可扩展到以下最大并发执行数，具体取决于来源：

- DynamoDB - 并发执行最多为管道中配置的 `ParallelizationFactor` 乘以流中的分片数。
- Apache Kafka - 并发执行最多为主题的分区数量，最高可达 1000 个。
- Kinesis - 并发执行最多为管道中配置的 `ParallelizationFactor` 乘以流中的分片数。
- Amazon MQ - 5
- Amazon SQS - 1250

如果您需要更高的最大轮询吞吐量或并发限制，请联系支持人员。

> **Note**
>
> 执行限制被视为尽力执行的安全限制。尽管轮询不会被限制在这些值以下，但管道或账户的突增值可能会高于这些建议值。

管道执行时间限制为最长 5 分钟，包括富集和目标处理。此限制目前无法提高。

对于严格排序的源（例如 Amazon SQS FIFO 队列、Kinesis 和 DynamoDB Streams 或 Apache Kafka 主题），管道在并发处理能力方面受到源配置的进一步限制，例如 FIFO 队列的消息组 ID 数量，或

Kinesis 队列的分片数量。由于排序在这些约束条件下得到严格保证，因此具有排序源的管道不能超过这些并发限制。

Amazon EventBridge Pipes 输入转换

Amazon EventBridge Pipes 在向富集和目标传递数据时，支持可选的输入转换器。您可以使用输入转换器来重塑 JSON 事件输入负载，以满足富集或目标服务的需求。对于 Amazon API Gateway 和 API 目标，您可以通过这种方式将输入事件转变为 API 的 RESTful 模型。输入转换器作为 `InputTemplate` 参数建模。它们可以是自由文本、事件负载的 JSON 路径或包含事件负载内联 JSON 路径的 JSON 对象。对于富集，事件负载来自源。对于目标，如果在管道中配置了富集，则事件负载是从富集中返回的内容。除了事件负载中的服务特定数据外，您还可以使用 `InputTemplate` 中的保留变量来引用管道的数据。

要访问数组中的项目，请使用方括号表示法。

> **Note**
>
> EventBridge 不支持所有 JSON 路径语法，并在运行时对其进行评估。支持的语法包括：
>
> - 点表示法（例如 `$.detail`）
> - 短划线
> - 下划线
> - 字母数字字符
> - 数组索引
> - 通配符 (*)

以下是引用 Amazon SQS 事件负载的示例 `InputTemplate` 参数：

静态字符串

```
InputTemplate: "Hello, sender"
```

JSON 路径

```
InputTemplate: <$.attributes.SenderId>
```

动态字符串

```
InputTemplate: "Hello, <$.attributes.SenderId>"
```

静态 JSON

```
InputTemplate: >
{
  "key1": "value1",
  "key2": "value2",
  "key3": "value3",
}
```

动态 JSON

```
InputTemplate: >
{
  "key1": "value1"
  "key2": <$.body.key>,
  "d": <aws.pipes.event.ingestion-time>
}
```

使用方括号表示法访问数组中的项目。

```
InputTemplate: >
{
  "key1": "value1"
  "key2": <$.body.Records[3]>,
  "d": <aws.pipes.event.ingestion-time>
}
```

> **Note**
>
> EventBridge 会在运行时替换输入转换器，以确保有效的 JSON 输出。因此，请在引用 JSON 路径参数的变量前后加引号，但不要在引用 JSON 对象或数组的变量前后加引号。

预留变量

输入模板可以使用以下预留变量：

- `<aws.pipes.pipe-arn>` - 管道的 Amazon 资源名称 (ARN)。

- <aws.pipes.pipe-name> - 管道的名称。
- <aws.pipes.source-arn> - 管道事件源的 ARN。
- <aws.pipes.enrichment-arn> - 管道富集的 ARN。
- <aws.pipes.target-arn> - 管道目标的 ARN。
- <aws.pipes.event.ingestion-time> - 输入转换器收到事件的时间。这是 ISO 8601 时间戳。这个时间对于富集输入转换器和目标输入转换器来说是不同的，具体取决于富集完成事件处理的时间。
- <aws.pipes.event> - 输入转换器接收到的事件。

  对于富集输入转换器，这是来自源的事件。它包含来自源的原始负载，以及其他特定于服务的元数据。如需特定于此服务的示例，请参阅 ???中的主题。

  对于目标输入转换器，这是由富集返回的事件（如果已配置富集），不包含其他元数据。因此，富集返回的负载可能是非 JSON。如果未在管道中配置富集，则这是来自源的事件，包含元数据。
- <aws.pipes.event.json> - 与aws.pipes.event 相同，但是只有当原始负载（来自源或由富集返回）为 JSON 时，该变量才有值。如果管道有编码字段，例如 Amazon SQS body 字段或 Kinesis data，则会对这些字段进行解码，并转换为有效的 JSON。由于该变量未被转义，因此只能用作 JSON 字段的值。有关更多信息，请参阅???。

输入转换示例

以下是 Amazon EC2 事件的示例，我们可以将其用作示例事件。

```
{
  "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"
```

```
  }
}
```

让我们使用以下 JSON 作为转换器。

```
{
  "instance" : <$.detail.instance-id>,
  "state": <$.detail.state>,
  "pipeArn" : <aws.pipes.pipe-arn>,
  "pipeName" : <aws.pipes.pipe-name>,
  "originalEvent" : <aws.pipes.event.json>
}
```

以下是生成的输出：

```
{
  "instance" : "i-0123456789",
  "state": "RUNNING",
  "pipeArn" : "arn:aws:pipe:us-east-1:123456789012:pipe/example",
  "pipeName" : "example",
  "originalEvent" : {
    ... // commented for brevity
  }
}
```

隐式正文数据解析

传入负载中的以下字段可能是 JSON 转义的（例如 Amazon SQS body 对象），也可能是 base64 编码的，例如 Kinesis `data` 对象。对于筛选和输入转换，EventBridge 会将这些字段转换为有效的 JSON，以便直接引用子值。例如，适用于 Kinesis 的 `<$.data.someKey>`。

要让目标接收原始负载（不包含任何其他元数据），请针对特定于源的正文数据使用输入转换器。例如，适用于 Amazon SQS 的 `<$.body>` 或适用于 Kinesis 的 `<$.data>`。如果原始负载是有效的 JSON 字符串（例如 `{"key": "value"}`），则针对特定于源的正文数据使用输入转换器，将导致原始源负载中的引号被删除。例如，在传送到目标时，`{"key": "value"}` 会变成 `"{key: value}"`。如果您的目标需要有效的 JSON 负载（例如 EventBridge Lambda 或 Step Functions），将导致传送失败。要让目标接收原始源数据而不生成无效 JSON，请将源正文数据输入转换器包装在 JSON 中。例如，`{"data": <$.data>}`。

隐式正文解析还可用于动态填充大多数管道目标或富集参数的值。有关更多信息，请参阅???。

> **Note**
>
> 如果原始负载是有效的 JSON，此字段将包含未转义、非 base64 编码的 JSON。但是，如果负载不是有效的 JSON，EventBridge 会对下面列出的字段进行 base64 编码，Amazon SQS 除外。

- Active MQ - `data`
- Kinesis - `data`
- Amazon MSK - `key` 和 `value`
- Rabbit MQ - `data`
- 自托管 Apache Kafka - `key` 和 `value`
- Amazon SQS - `body`

转换输入的常见问题

以下是在 EventBridge 管道中转换输入时会遇到的一些常见问题：

- 对于字符串，需要引号。
- 为模板创建 JSON 路径时不进行验证。
- 如果您指定一个变量，匹配在事件中不存在的 JSON 路径，则不会创建该变量，并且不会在输出中显示该变量。
- 像 `aws.pipes.event.json` 这样的 JSON 属性只能用作 JSON 字段的值，不能在其他字符串中内联。
- 在为目标填充输入模板 时，EventBridge 不会转义输入路径 提取的值。
- 如果 JSON 路径引用一个 JSON 对象或数组，但在字符串中引用了该变量，EventBridge 会删除所有内部引号，以确保字符串有效。例如，"Body is <$.body>" 将导致 EventBridge 从对象中删除引号。

  因此，如果要根据单个 JSON 路径变量输出 JSON 对象，则必须将其作为键。在本示例中，`{"body": <$.body>}`。
- 表示字符串的变量不需要引号。这是允许的，但是 EventBridge Pipes 在转换过程中会自动为字符串变量值添加引号，以确保转换输出是有效的 JSON。EventBridge Pipes 不会为表示 JSON 对象或数组的变量添加引号。不要为表示 JSON 对象或数组的变量添加引号。

例如，以下输入模板包含的变量表示字符串和 JSON 对象：

```
{
  "pipeArn" : <aws.pipes.pipe-arn>,
  "pipeName" : <aws.pipes.pipe-name>,
  "originalEvent" : <aws.pipes.event.json>
}
```

生成带有正确引号的有效 JSON：

```
{
  "pipeArn" : "arn:aws:events:us-east-2:123456789012:pipe/example",
  "pipeName" : "example",
  "originalEvent" : {
    ... // commented for brevity
  }
}
```

- 对于 Lambda 或 Step Functions 富集或目标，即使批次大小为 1，批次也会作为 JSON 数组传送到目标。但是，输入转换器仍将应用于 JSON 数组中的单个记录，而不是整个数组。有关更多信息，请参阅???。

记录 Amazon Pi EventBridge pes

EventBridge 管道记录使您可以让 Pipes 向支持的 AWS 服务发送详细说明管道性能的记录。EventBridge 使用日志可深入了解管道的执行性能，并帮助进行故障排除和调试。

您可以选择以下 AWS 服务作为 Pip EventBridge es 将记录传送到的日志目的地：

- CloudWatch 日志

  EventBridge 将日志记录传送到指定的 CloudWatch 日志日志组。

  使用 CloudWatch 日志将您使用的所有系统、应用程序和 AWS 服务的日志集中到一个高度可扩展的服务中。有关更多信息，请参阅 Amazon 日志用户指南中的使用日志组和 CloudWatch 日志流。

- Firehose 直播日志

  EventBridge 将日志记录传送到 Firehose 传输流。

Amazon Data Firehose 是一项完全托管的服务，用于将实时流数据传输到某些 AWS 服务以及受支持的第三方服务提供商拥有的任何自定义 HTTP 终端节点或 HTTP 终端节点等目的地。有关更多信息，请参阅亚马逊数据 Firehose 用户指南中的创建亚马逊数据 Firehose 传输流。

- Amazon S3 日志

  EventBridge 将日志记录作为 Amazon S3 对象传送到指定的存储桶。

  Amazon S3 是一项对象存储服务，提供行业领先的可扩展性、数据可用性、安全性和性能。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的在 Amazon S3 中上传、下载和使用对象。

Amazon Pi EventBridge pes 日志的工作原理

管道执行是管道接收并传递到富集和/或目标的一个事件或一批事件。如果启用，则会在处理事件批处理时为其执行的每个执行步骤 EventBridge 生成一条日志记录。记录中包含的信息适用于事件批次，无论是单个事件还是最多 10,000 个事件。

可以在管道源和目标上配置事件批次的大小。有关更多信息，请参阅 ???。

发送到每个日志目标的记录数据都是相同的。

如果配置了 Amazon CloudWatch Logs 目标，则传送到所有目标的日志记录限制为 256kb。字段会根据需要被截断。

您可以通过以下方式自定义 EventBridge 发送到所选日志目标的记录：

- 您可以指定日志级别，该级别决定了 EventBridge 将记录发送到所选日志目标的执行步骤。有关更多信息，请参阅???。
- 您可以指定 Pip EventBridge es 是否在相关的执行步骤的记录中包含执行数据。这些数据包括：
  - 事件批次的有效负载
  - 发送到 AWS 浓缩服务或目标服务的请求
  - AWS 浓缩服务或目标服务返回的响应

  有关更多信息，请参阅???。

指定 EventBridge 管道日志级别

您可以指定将记录 EventBridge 发送到所选日志目标的执行步骤的类型。

从以下详情级别中进行选择，将其包含在日志记录中。日志级别适用于为管道指定的所有日志目标。每个日志级别都包含之前日志级别的执行步骤。

- OFF — EventBridge 不向任何指定的日志目标发送任何记录。这是默认设置。
- ERROR- EventBridge 将与管道执行期间生成的错误相关的所有记录发送到指定的日志目的地。
- INFO — EventBridge 将与错误相关的所有记录以及管道执行期间执行的选择其他步骤发送到指定的日志目的地。
- TRACE — EventBridge 将管道执行中任何步骤中生成的所有记录发送到指定的日志目的地。

在 EventBridge 控制台中， CloudWatch 默认选择日志作为日志目标，ERROR日志级别也是如此。因此，默认情况下，Pip EventBridge es 会创建一个新的 CloudWatch 日志组，向该组发送包含详细ERROR级别的日志记录。以编程方式配置日志时，不会选择默认值。

下表列出了每个日志级别中包含的执行步骤。

步骤	跟踪	信息	错误	关闭
执行已失败	x	x	x	
执行部分失败	x	x	x	
执行已启动	x	x		
执行已成功	x	x		
执行受限	x	x	x	
执行超时	x	x	x	
富集调用失败	x	x	x	
已跳过富集调用	x	x		
已启动富集调用	x			
富集调用成功	x			
进入富集阶段	x	x		

步骤	跟踪	信息	错误	关闭
富集阶段失败	x	x	x	
富集阶段成功	x	x		
富集转换失败	x	x	x	
已启动富集转换	x			
富集转换成功	x			
目标调用失败	x	x	x	
目标调用部分失败	x	x	x	
已跳过目标调用	x			
已启动目标调用	x			
目标调用成功	x			
进入目标阶段	x	x		
目标阶段失败	x	x	x	
目标阶段部分失败	x	x	x	
已跳过目标阶段	x			
目标阶段成功	x	x		
目标转换失败	x	x	x	
已启动目标转换	x			
目标转换成功	x			

在 Pip EventBridge es 日志中包含执行数据

您可以为指定 EventBridge ，以便在其生成的记录中包含执行数据。执行数据包括表示事件批次有效负载的字段，以及发送到富集和目标的请求和来自它们的响应。

执行数据对于故障排除和调试很有用。payload 字段包含批次中每个事件的实际内容，使您可以将单个事件与特定的管道执行关联起来。

如果您选择包含执行数据，为管道指定的所有日志目标都会包含。

> **Important**
>
> 这些字段可能包含敏感信息。 EventBridge 在记录期间不尝试编辑这些字段的内容。

在包括执行数据时， EventBridge 将以下字段添加到相关记录中：

- **payload**

  表示管道正在处理的事件批次的内容。

  EventBridge 在可能已更新事件批次内容的步骤中生成的记录中包含该payload字段。此操作包括以下步骤：

  - EXECUTION_STARTED
  - ENRICHMENT_TRANSFORMATION_SUCCEEDED
  - ENRICHMENT_STAGE_SUCCEEDED
  - TARGET_TRANSFORMATION_SUCCEEDED
  - TARGET_STAGE_SUCCEEDED

- **awsRequest**

  表示作为 JSON 字符串发送到富集或目标的请求。对于发送到 API 目标的请求，表示发送到该端点的 HTTP 请求。

  EventBridge 将该awsRequest字段包括在浓缩和定向的最后阶段生成的记录中；也就是说，在针对特定的浓缩或目标服务执行或试图执行请求之后 EventBridge 。此操作包括以下步骤：

  - ENRICHMENT_INVOCATION_FAILED
  - ENRICHMENT_INVOCATION_SUCCEEDED
  - TARGET_INVOCATION_FAILED

  - TARGET_INVOCATION_PARTIALLY_FAILED
  - TARGET_INVOCATION_SUCCEEDED
- **awsResponse**

  表示富集或目标以 JSON 格式返回的响应。对于发送到 API 目标的请求，表示从端点返回的 HTTP 响应。

  与一样awsRequest，在浓缩和定向的最后阶段生成的记录中 EventBridge 包括该awsResponse字段；也就是说，在执行或试图执行针对特定浓缩或目标服务的请求并收到回复之后 EventBridge 。此操作包括以下步骤：

  - ENRICHMENT_INVOCATION_FAILED
  - ENRICHMENT_INVOCATION_SUCCEEDED
  - TARGET_INVOCATION_FAILED
  - TARGET_INVOCATION_PARTIALLY_FAILED
  - TARGET_INVOCATION_SUCCEEDED

有关管道执行步骤的讨论，请参阅 ???。

截断 Pipes 日志记录中的 EventBridge执行数据

如果您选择在管道的日志记录中 EventBridge 包含执行数据，则记录可能会超过 256 KB 的大小限制。为防止出现这种情况，请按以下顺序 EventBridge自动截断执行数据字段。 EventBridge 在继续截断下一个字段之前，会完全截断每个字段。 EventBridge 只需从数据字符串末尾删除字符即可截断字段数据；不会尝试根据数据重要性进行截断，截断会使 JSON 格式失效。

- payload
- awsRequest
- awsResponse

EventBridge 如果在事件中截断字段，则该truncatedFields字段将包含被截断的数据字段的列表。

Pip EventBridge es 日志记录中的错误报告

EventBridge 还包括表示失败状态的管道执行步骤中的错误数据（如果有）。这些步骤包括：

- ExecutionThrottled

- ExecutionTimeout
- ExecutionFailed
- ExecutionPartiallyFailed
- EnrichmentTransformationFailed
- EnrichmentInvocationFailed
- EnrichmentStageFailed
- TargetTransformationFailed
- TargetInvocationFailed
- TargetInvocationPartiallyFailed
- TargetStageFailed
- TargetStagePartiallyFailed

EventBridge 管道执行步骤

了解管道执行步骤的流程，可以帮助您使用日志对管道的性能进行故障排除或调试。

管道执行是管道接收并传递到富集或目标的一个事件或一批事件。如果启用，则会在处理事件批处理时为其执行的每个执行步骤 EventBridge 生成一条日志记录。

整体来看，执行包含两个阶段 或一系列步骤：富集和目标。每个阶段都由转换和调用步骤组成。

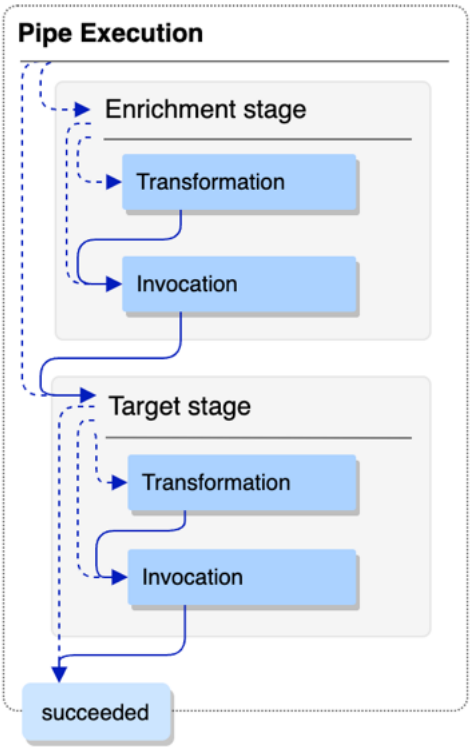
成功执行管道的主要步骤遵循以下流程：

- 管道执行开始。
- 如果您为事件指定了富集，执行将进入富集阶段。如果您未指定富集，执行将进入目标阶段。

  在富集阶段，管道会执行您指定的任何转换，然后调用富集。
- 在目标阶段，管道会执行您指定的任何转换，然后调用目标。

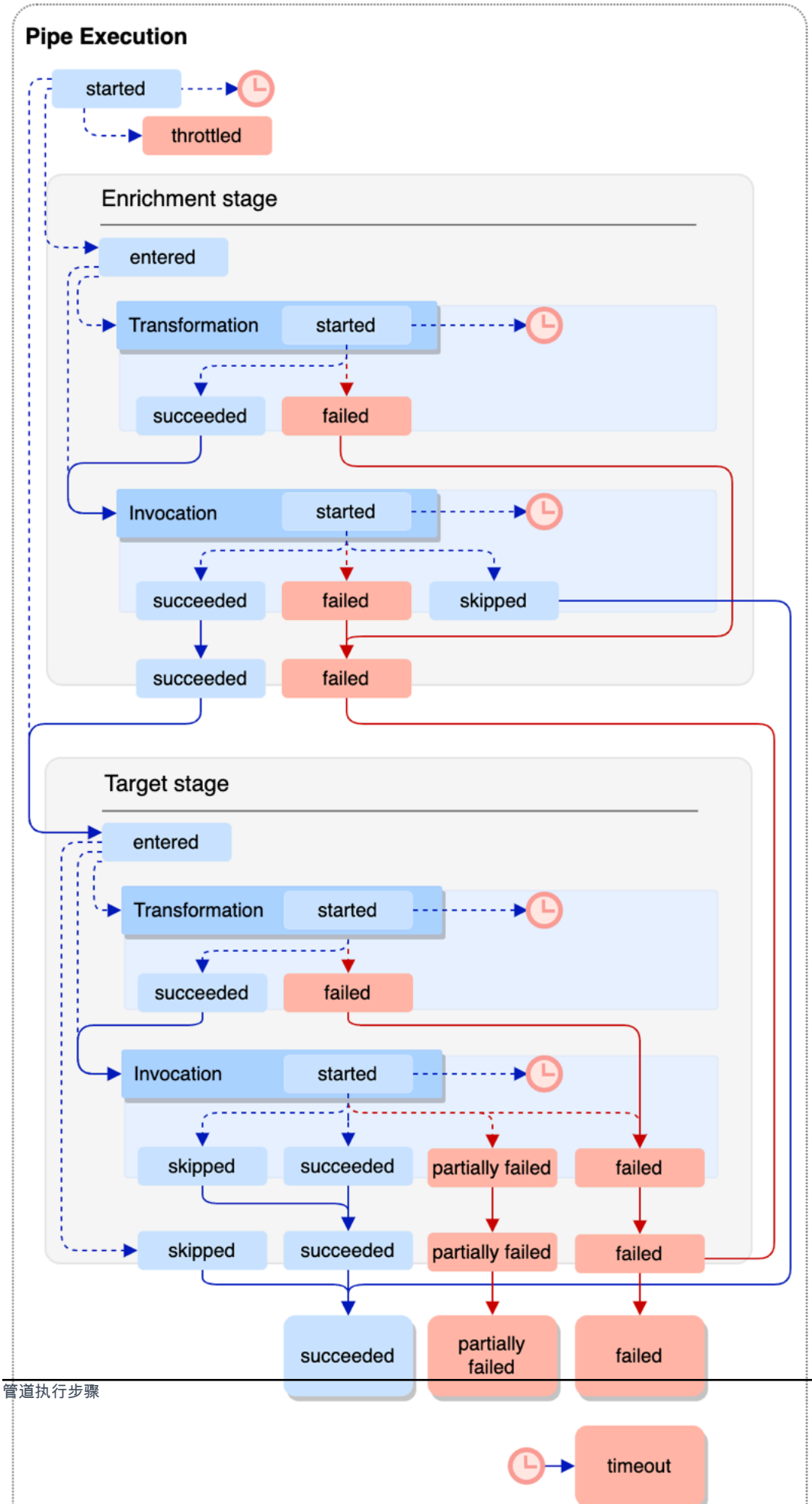
  如果您未指定转换或目标，执行会跳过目标阶段。
- 管道执行成功完成。

下图演示了这个流程。分支路径以虚线表示。



下图展示了管道执行流程的详细视图，包含所有可能的执行步骤。同样，分支路径以虚线表示。

有关管道执行步骤的完整列表，请参阅???。


Pipe Execution
started
throttled
Enrichment stage
entered
Transformation
started
succeeded
failed
Invocation
started
succeeded
failed
skipped
succeeded
failed
Target stage
entered
Transformation
started
succeeded
failed
Invocation
started
skipped
succeeded
partially failed
failed
skipped
succeeded
partially failed
failed
succeeded
partially failed
failed
timeout

请注意，目标调用可能会导致批次部分失败。有关更多信息，请参阅???。

EventBridge 管道日志架构参考

以下参考详细介绍了 Pip EventBridge es 日志记录的架构。

每条日志记录代表一个管道执行步骤，如果已将管道源和目标配置为批处理，最多可包含 10,000 个事件。

有关更多信息，请参阅???。

```
{
    "executionId": "guid",
    "timestamp": "date_time",
    "messageType": "execution_step",
    "resourceArn": "arn:aws:pipes:region:account:pipe/pipe-name",
    "logLevel": "TRACE | INFO | ERROR",
    "payload": "{}",
    "awsRequest": "{}"
    "awsResponse":"{}"
    "truncatedFields": ["awsRequest","awsResponse","payload"],
    "error": {
        "httpStatusCode": code,
        "message": "error_message",
        "details": "",
        "awsService": "service_name",
        "requestId": "service_request_id"
    }
}
```

executionId

管道执行的 ID。

管道执行是管道接收并传递到富集或目标的一个事件或一批事件。有关更多信息，请参阅???。

timestamp

发出日志事件的日期和时间。

单位：毫秒

messageType

生成记录的管道执行步骤。

有关管道执行步骤的更多信息，请参阅 ???。

resourceArn

管道的 Amazon 资源名称 (ARN)。

logLevel

为管道日志指定的详情级别。

有效值：ERROR | INFO | TRACE

有关更多信息，请参阅???。

payload

管道正在处理的事件批次的内容。

EventBridge 仅当您已指定在该管道的日志中包含执行数据时，才包含此字段。有关更多信息，请参阅 ???。

> **Important**
> 这些字段可能包含敏感信息。 EventBridge 在记录期间不尝试编辑这些字段的内容。

有关更多信息，请参阅???。

awsRequest

发送到富集或目标服务的 JSON 格式的请求 对于发送到 API 目标的请求，表示发送到该端点的 HTTP 请求。

EventBridge 仅当您已指定在该管道的日志中包含执行数据时，才包含此字段。有关更多信息，请参阅 ???。

> **Important**
> 这些字段可能包含敏感信息。 EventBridge 在记录期间不尝试编辑这些字段的内容。

有关更多信息，请参阅???。

awsResponse

富集或目标以 JSON 格式返回的响应。对于发送到 API 目标的请求，表示从端点返回的 HTTP 响应，不是 API 目标服务本身返回的响应

EventBridge 仅当您已指定在该管道的日志中包含执行数据时，才包含此字段。有关更多信息，请参阅 ???。

> **Important**
>
> 这些字段可能包含敏感信息。 EventBridge 在记录期间不尝试编辑这些字段的内容。

有关更多信息，请参阅???。

truncatedFields

为了使记录保持在 256 KB 大小限制以下，所有执行数据字段的列表都 EventBridge 已被截断。

如果 EventBridge 不必截断任何执行数据字段，则此字段存在，但是。`null`

有关更多信息，请参阅???。

error

包含在此管道执行步骤中生成的任何错误的相关信息。

如果在此管道执行步骤中未生成错误，该字段仍存在，但为 `null`。

httpStatusCode

调用的服务返回的 HTTP 状态代码。

message

调用的服务返回的错误消息。

详细信息

调用的服务返回的任何详细错误信息。

awsService

被调用服务的名称。

requestId

来自被调用服务的此请求的 ID。

使用 AWS CloudTrail 和亚马逊日志记录和监控 Ama EventBridge zon Pip CloudWatch es

您可以使用指标记录 EventBridge Pipes 调用、使用 CloudTrail 和监控管道的运行状况。 CloudWatch

CloudWatch 指标

EventBridge Pipes CloudWatch 每分钟向 Amazon 发送从管道执行被限制到成功调用目标的所有指标。

指标	描述
`Concurrency`	管道的并发执行数。 有效尺寸： AwsAccountId 单位：无
`Duration`	管道执行所花费的时长。 有效尺寸： PipeName 单位：毫秒
`EventCount`	管道已处理的事件数。 有效尺寸： PipeName 单位：无
`EventSize`	调用管道的事件的有效负载的大小。 有效尺寸： PipeName 单位：字节
`Execution Throttled`	管道执行被限制的次数。 **Note** 如果没有限制执行，此值将为 `0`。

指标	描述
	有效尺寸： AwsAccountId， PipeName 单位：无
`Execution Timeout`	在完成执行之前，管道中有多少次执行超时。 Note 如果没有执行超时，此值将为 `0`。 有效尺寸： PipeName 单位：无
`ExecutionFailed`	管道中有多少次执行失败。 Note 如果没有执行失败，此值将为 `0`。 有效尺寸： PipeName 单位：无
`Execution Partially Failed`	管道中有多少次执行部分失败。 Note 如果没有执行部分失败，此值将为 `0`。 有效尺寸： PipeName 单位：无

指标	描述
EnrichmentStageDuration	富集阶段花了多长时间完成。 有效尺寸： PipeName 单位：毫秒
EnrichmentStageFailed	管道的富集阶段有多少次执行失败。 Note 如果没有执行失败，此值将为 0。 有效尺寸： PipeName 单位：无
Invocations	调用总数。 有效尺寸： AwsAccountId， PipeName 单位：无
TargetStageDuration	目标阶段花了多长时间完成。 有效尺寸： PipeName 单位：毫秒
TargetStageFailed	管道的目标阶段有多少次执行失败。 Note 如果没有执行失败，此值将为 0。 有效尺寸： PipeName 单位：无

指标	描述
TargetStagePartiallyFailed	管道的目标阶段有多少次执行部分失败。 Note 如果没有目标阶段执行部分失败，此值将为 0。 有效尺寸： PipeName 单位：无
TargetStageSkipped	管道目标阶段跳过的执行次数（例如，由于富集返回的有效负载为空）。 有效尺寸： PipeName 单位：计数

CloudWatch 指标的维度

CloudWatch 指标具有维度或可排序的属性，如下所示。

维度	描述
AwsAccountId	按账户 ID 筛选可用指标。
PipeName	按管道名称筛选可用指标。

Amazon EventBridge Pipes 错误处理和故障排除

重试行为和错误处理

如果源服务、富集或目标服务或 EventBridge 出现任何可重试的 AWS 故障，EventBridge Pipes 会自动重试富集和目标调用。但是，如果富集或目标客户实施返回故障，管道轮询吞吐量将逐渐退避。对于几乎持续的 4xx 错误（例如 IAM 的授权问题或资源缺失），可以自动禁用管道，并在 StateReason 中提供一条消息进行解释。

管道调用错误和重试行为

调用管道时，可能会出现两种主要类型的错误：管道内部错误 和客户调用错误。

管道内部错误

管道内部错误是由 EventBridge Pipes 服务管理的调用的某些方面导致的错误。

这些类型的错误可能包括以下问题：

- 尝试调用客户目标服务时 HTTP 连接失败
- 管道服务本身的可用性暂时下降。

通常，EventBridge Pipes 会无限次重试内部错误，仅在源中的记录过期后才会停止。

对于具有流源的管道，EventBridge Pipes 不会将内部错误的重试次数计入流源的重试策略中指定的重试次数上限。对于具有 Amazon SQS 源的管道，EventBridge Pipes 不会将内部错误的重试次数计入 Amazon SQS 源的接收计数上限。

客户调用错误

客户调用错误是由于用户管理的配置或代码而导致的错误。

这些类型的错误可能包括以下问题：

- 管道的权限不足，无法调用目标。
- 同步调用的客户 Lambda、Step Functions、API 目标或 API Gateway 端点中的逻辑错误。

对于客户调用错误，EventBridge Pipes 会执行以下操作：

- 对于具有流源的管道，EventBridge Pipes 会重试，直至达到管道重试策略中配置的重试次数上限，或者直到最长记录期限到期（以先到者为准）。
- 对于具有 Amazon SQS 源的管道，EventBridge Pipes 会重试一个客户错误，直至达到源队列的接收计数上限。
- 对于具有 Apache Kafka 或 Amazon MQ 源的管道，EventBridge 会像重试内部错误一样重试客户错误。

对于具有计算目标的管道，您必须同步调用管道，这样 EventBridge Pipes 才能知道客户计算逻辑引发的任何运行时错误，并重试此类错误。Pipes 无法重试从 Step Functions 标准工作流程的逻辑中引发的错误，因为必须异步调用此目标。

对于 Amazon SQS 和流源（例如 Kinesis 和 DynamoDB），EventBridge Pipes 支持对目标故障进行部分批处理故障处理。有关更多信息，请参阅部分批处理故障。

管道 DLQ 行为

管道从源继承死信队列 (DLQ) 行为：

- 如果源 Amazon SQS 队列配置了 DLQ，则消息将在尝试了指定次数后自动传送到该队列。
- 对于流源，例如 DynamoDB 和 Kinesis 流，您可以为管道配置 DLQ 并路由事件。DynamoDB 和 Kinesis 流源支持使用 Amazon SQS 队列和 Amazon SNS 主题作为 DLQ 目标。

如果您为具有 Kinesis 或 DynamoDB 源的管道指定 `DeadLetterConfig`，请确保管道的 `MaximumRecordAgeInSeconds` 属性小于源事件的 `MaximumRecordAge` 属性。`MaximumRecordAgeInSeconds` 控制管道轮询器何时放弃事件并将其传递给 DLQ，`MaximumRecordAge` 控制消息在被删除之前在源流中的可见时长。因此，请将 `MaximumRecordAgeInSeconds` 设置为小于源 `MaximumRecordAge` 的值，这样在事件发送到 DLQ 到事件被源自动删除之间，您就有足够的时间来确定事件进入 DLQ 的原因。

对于 Amazon MQ 源，可以直接在消息代理上配置 DLQ。

EventBridge Pipes 不支持流源的先进先出 (FIFO) DLQ。

EventBridge Pipes 不支持 Amazon MSK 流和自托管 Apache Kafka 流源的 DLQ。

管道故障状态

创建、删除和更新管道是异步操作，可能会导致失败状态。同样，管道可能会因故障而被自动禁用。在所有情况下，管道 `StateReason` 都可提供信息来帮助排除故障。

以下是 `StateReason` 可能值的示例：

- 未找到流。要恢复处理，请删除管道，然后创建一个新管道。
- 管道没有执行队列操作所需的权限（sqs:ReceiveMessage、sqs:DeleteMessage 和 sqs:GetQueueAttributes）
- 连接错误。您的 VPC 必须能够连接到管道。您可以通过配置 NAT 网关来提供访问权限。有关如何设置 NAT 网关的信息，请查看 AWS 文档。

- MSK 集群没有关联的安全组

管道可能会自动停止并更新 StateReason。可能的原因包括：

- 一个 Step Functions 标准工作流程配置为富集。
- 一个 Step Functions 标准工作流程配置为同步调用的目标。

自定义加密故障

如果将源配置为使用 AWS KMS 自定义加密密钥 (CMK)，而不使用 AWS 托管 AWS KMS 密钥，则必须为管道的执行角色明确授予解密权限。为此，请在自定义 CMK 策略中添加以下额外权限：

```
  {
      "Sid": "Allow Pipes access",
      "Effect": "Allow",
      "Principal": {
          "AWS": "arn:aws:iam::01234567890:role/service-role/
Amazon_EventBridge_Pipe_DDBStreamSourcePipe_12345678"
      },
      "Action": "kms:Decrypt",
      "Resource": "*"
  }
```

请将上述角色替换为您管道的执行角色。

以上要求适用于使用 AWS KMS CMK 的所有管道源，包括：

- Amazon DynamoDB Streams
- Amazon Kinesis Data Streams
- Amazon MQ
- Amazon MSK
- Amazon SQS

教程：创建可筛选源事件的 EventBridge 管道

在本教程中，您将创建一个管道，将 DynamoDB 流源连接到 Amazon SQS 队列目标。其中包括为管道指定事件模式，用于筛选要传送到队列的事件。然后，您将测试管道以确保只传送所需的事件。

先决条件：创建源和目标

在创建管道之前，您需要创建管道要连接的源和目标。在本例中，Amazon DynamoDB 数据流作为管道源，Amazon SQS 队列作为管道目标。

要简化此步骤，您可以使用 AWS CloudFormation 预置源和目标资源。方法是创建一个 CloudFormation 模板，定义以下资源：

- 管道源

  名为 `pipe-tutorial-source` 的 Amazon DynamoDB 表，启用了一个流，提供有关 DynamoDB 表中项目更改情况的的排序信息流。
- 管道目标

  名为 `pipe-tutorial-target` 的 Amazon SQS 队列，用于从您的管道接收 DynamoDB 事件流。

创建 CloudFormation 模板，预置管道资源

1. 复制下方 ??? 部分中的 JSON 模板文本。
2. 将模板另存为 JSON 文件（例如，`~/pipe-tutorial-resources.json`）。

接下来，使用您刚刚创建的模板文件来预置 CloudFormation 堆栈。

> **Note**
>
> 创建 CloudFormation 堆栈后，您需要为其预置的 AWS 资源付费。

使用 AWS CLI 预置教程先决条件

- 运行以下 CLI 命令，其中 `--template-body` 指定模板文件的位置：

```
aws cloudformation create-stack --stack-name pipe-tuturial-resources --template-
body file://~/pipe-tutorial-resources.json
```

使用 CloudFormation 控制台预置教程先决条件

1. 打开 AWS CloudFormation 控制台，地址：https://console.aws.amazon.com/cloudformation。

2. 依次选择堆栈、选择创建堆栈、使用新资源(标准)。

   CloudFormation 会显示创建堆栈向导。
3. 对于先决条件 - 准备模板，保留默认值，模板已准备就绪已选中。
4. 在指定模板下选择上传模板文件，然后选择文件并选择下一步。
5. 配置堆栈及其要预置的资源：

   - 对于堆栈名称，输入 `pipe-tuturial-resources`。
   - 在参数中，请保留 DynamoDB 表和 Amazon SQS 队列的默认名称。
   - 选择 Next（下一步）。
6. 选择下一步，然后选择提交。

   CloudFormation 创建堆栈并预置模板中定义的资源。

有关 CloudFormation 的更多信息,请参阅《AWS CloudFormation 用户指南》中的什么是 AWS CloudFormation？。

步骤 1：创建管道

预置管道源和目标后，您现在可以创建管道来连接这两个服务。

使用 EventBridge 控制台创建管道

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在导航窗格中，选择管道。
3. 选择创建管道。
4. 对于名称，将管道命名为 `pipe-tutorial`。
5. 指定 DynamoDB 数据流源：

   a. 在详细信息下，对于源，选择 DynamoDB 数据流。

      EventBridge 将显示特定于 DynamoDB 的源配置设置。

   b. 对于 DynamoDB 流，选择 `pipe-tutorial-source`。

      将起始位置保留为默认值 `Latest`。

   c. 选择 Next（下一步）。
6. 指定并测试用于筛选事件的事件模式：

通过筛选，您可以控制管道将哪些事件发送到富集或目标。管道仅将与事件模式匹配的事件发送到富集或目标。

有关更多信息，请参阅???。

> **Note**
>
> 您只需为发送到富集或目标的事件付费。

a. 在示例事件 - 可选 下，保持 AWS 事件的选中状态，并确保选中 DynamoDB 流示例事件 1。

   这是您将用来测试我们的事件模式的示例事件。

b. 在事件模式下，输入以下事件模式：

```
{
  "eventName": ["INSERT", "MODIFY"]
}
```

c. 选择测试模式。

   EventBridge 会显示一条消息，说明示例事件与事件模式匹配。这是因为示例事件的 `eventName` 值为 `INSERT`。

d. 选择 Next（下一步）。

7. 选择下一步，跳过指定富集的操作。

   在此示例中，您无需选择富集。富集支持您选择一项服务，增强来自源的数据，然后再将其发送到目标。有关更多信息，请参阅 ???。

8. 将您的 Amazon SQS 队列指定为管道目标：

   a. 在详细信息下，目标服务选择 Amazon SQS 队列。

   b. 对于队列，选择 `pipe-tutorial-target`。

   c. 将目标输入转换器部分留空。

      有关更多信息，请参阅???。

9. 选择创建管道

EventBridge 会创建管道并显示管道详细信息页面。一旦管道的状态更新为 Running，即说明管道已准备就绪。

步骤 2：确认管道筛选器事件

管道已设置完毕，但尚未从表中接收事件。

要测试管道，您需要更新 DynamoDB 表中的条目。每次更新都会生成事件，DynamoDB 流会将这些事件发送到我们的管道。有些事件会匹配您指定的事件模式，有些则不会。然后，您可以检查 Amazon SQS 队列，确保管道仅传送了与我们的事件模式匹配的事件。

更新表项目，生成事件

1. 从 https://console.aws.amazon.com/dynamodb/ 打开 DynamoDB 控制台。
2. 在左侧导航栏上，选择表。选择 pipe-tutorial-source 表。

   DynamoDB 显示 pipe-tutorial-source 的表详细信息页面。
3. 选择浏览表项目，然后选择创建项目。

   DynamoDB 会显示创建项目页面。
4. 在属性下，创建一个新的表项目：
   a. 在专辑中输入 Album A。
   b. 在艺术家中输入 Artist A。
   c. 选择 Create Item (创建项目)。
5. 更新表项目：
   a. 在返回的项目下，选择 Album A。
   b. 选择添加新属性，然后选择字符串。
   c. 输入 Song 的新值，值为 Song A。
   d. 选择 Save changes（保存更改）。
6. 删除表项目：
   a. 在返回的项目下，选中 Album A。
   b. 从操作菜单中选择删除项目。

您已对表项目进行了三次更新；这会为 DynamoDB 数据流生成三个事件：

- 创建项目时会生成 INSERT 事件。
- 为项目添加属性时会生成 MODIFY 事件。
- 删除项目时会生成 REMOVE 事件。

但是，您为管道指定的事件模式应筛选掉所有不是 INSERT 或 MODIFY 的事件。接下来，确认管道已将预期的事件传送到队列。

确认预期的事件已传送到队列。

1. 通过以下网址打开 Amazon SQS 控制台：https://console.aws.amazon.com/sqs/。
2. 选择 pipe-tutorial-target 队列。

   Amazon SQS 会显示队列详情页面。
3. 选择发送和接收消息，然后在接收消息下选择轮询消息。

   队列会轮询管道，然后列出它收到的事件。
4. 选择事件名称，查看已传送的事件 JSON。

队列中应该有两个事件：一个 eventName 为 INSERT，另一个 eventName 为 MODIFY。但是，管道没有传送删除表项目的事件，因为该事件的 eventName 为 REMOVE，与您在管道中指定的事件模式不匹配。

步骤 3：清理资源

首先，删除管道本身。

使用 EventBridge 控制台删除管道

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在导航窗格中，选择管道。
3. 选择 pipe-tutorial 管道并选择删除。

然后删除 CloudFormation 堆栈，以免因继续使用其中预置的资源而被收费。

使用 AWS CLI 删除教程先决条件

- 运行以下 CLI 命令，其中 `--stack-name` 指定堆栈的名称：

```
aws cloudformation delete-stack --stack-name pipe-tuturial-resources
```

使用 AWS CloudFormation 控制台删除教程先决条件

1. 打开 AWS CloudFormation 控制台，地址：https://console.aws.amazon.com/cloudformation。
2. 在堆栈页面上，选择堆栈，然后选择删除。
3. 选择删除确认您的操作。

用于生成先决条件的 AWS CloudFormation 模板

使用下面的 JSON 创建 CloudFormation 模板，用于预置本教程所需的源和目标资源。

```
{
  "AWSTemplateFormatVersion": "2010-09-09",

 "Description" : "Provisions resources to use with the EventBridge Pipes tutorial. You
 will be billed for the AWS resources used if you create a stack from this template.",

  "Parameters" : {
    "SourceTableName" : {
      "Type" : "String",
      "Default" : "pipe-tutorial-source",
      "Description" : "Specify the name of the table to provision as the pipe source,
 or accept the default."
    },
  "TargetQueueName" : {
    "Type" : "String",
    "Default" : "pipe-tutorial-target",
    "Description" : "Specify the name of the queue to provision as the pipe target, or
 accept the default."
  }
},
  "Resources": {
    "PipeTutorialSourceDynamoDBTable": {
      "Type": "AWS::DynamoDB::Table",
      "Properties": {
```

```
        "AttributeDefinitions": [{
            "AttributeName": "Album",
            "AttributeType": "S"
          },
          {
            "AttributeName": "Artist",
            "AttributeType": "S"
          }

        ],
        "KeySchema": [{
            "AttributeName": "Album",
            "KeyType": "HASH"

          },
          {
            "AttributeName": "Artist",
            "KeyType": "RANGE"
          }
        ],
        "ProvisionedThroughput": {
          "ReadCapacityUnits": 10,
          "WriteCapacityUnits": 10
        },
        "StreamSpecification": {
          "StreamViewType": "NEW_AND_OLD_IMAGES"
        },
        "TableName": { "Ref" : "SourceTableName" }
      }
    },
    "PipeTutorialTargetQueue": {
      "Type": "AWS::SQS::Queue",
      "Properties": {
        "QueueName": { "Ref" : "TargetQueueName" }
      }
    }
  }
}
```

从 Pi AWS CloudFormation p EventBridge es 生成模板

AWS CloudFormation 通过将基础设施视为代码，使您能够以集中且可重复的方式跨账户和地区配置和管理您的 AWS 资源。 CloudFormation 通过允许您创建模板来实现此目的，模板定义了您要配置和管理的资源。

EventBridge 允许您使用账户中的现有管道生成模板，以此来帮助您快速开始开发 CloudFormation 模板。您可以选择在模板中包含一条或多条管道。然后，您可以使用这些模板作为创建 CloudFormation 管理资源堆栈的基础。

有关的更多信息 CloudFormation，请参阅《 AWS CloudFormation 用户指南》。

对于事件总线，您可以根据事件总线和事件总线规则生成 CloudFormation 模板。

EventBridge 管道模板中包含的资源

EventBridge 生成 CloudFormation 模板时，它会为每个选定的管道创建一个AWS::Pipes::Pipe资源。此外，在上述条件下还 EventBridge 包括以下资源：

- AWS::Events::ApiDestination

  如果您的管道包含 API 目的地（无论是作为扩充还是目标），请将其作为 AWS::Events::ApiDestination资源 EventBridge包含在 CloudFormation 模板中。

- AWS::Events::EventBus

  如果您的管道包含事件总线作为目标，则将其作为 AWS::Events::EventBus 资源 EventBridge 包含在 CloudFormation 模板中。

- AWS::IAM::Role

  如果在配置管道时 EventBridge 创建了新的执行角色，则可以选择将该角色作为 AWS::IAM::Role 资源 EventBridge 包含在模板中。 EventBridge 不包括您创建的角色。（无论哪种情况，AWS::Pipes::Pipe 资源的`RoleArn`属性都包含角色的 ARN。）

使用从 EventBridge Pip CloudFormation es 生成的模板时的注意事项

使用从中生成的 CloudFormation 模板时，请考虑以下因素 EventBridge：

- EventBridge 在生成模板中不包含任何密码。

您可以编辑模板以包含模板参数，使用户能够在使用模板创建或更新 CloudFormation堆栈时指定密码或其他敏感信息。

此外，用户可以使用 Secrets Manager 在所需区域创建密钥，然后编辑生成的模板以使用动态参数。

- 生成的模板中的目标，与原始管道中指定的目标完全相同。如果在使用模板在其他区域创建堆栈之前，未对模板进行适当的编辑，可能会导致跨区域问题。

  此外，生成的模板不会自动创建下游目标。

从 Pi CloudFormation p EventBridge es 生成模板

要使用 EventBridge 控制台从一个或多个管道生成 CloudFormation 模板，请执行以下操作：

从一个或多个管道生成 CloudFormation 模板

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择管道。
3. 在 “管道” 下，选择要包含在生成的 CloudFormation 模板中的一个或多个管道。

   对于单条管道，您也可以选择管道名称以显示其详细信息页面。
4. 选择 “CloudFormation 模板”，然后选择 EventBridge 要生成模板的格式：JSON 或 YAML。

   EventBridge 显示以选定格式生成的模板。
5. 如果您为任何选定的管道 EventBridge 创建了新的执行角色，并且想要 EventBridge 将这些角色包含在模板中，请选择 “包括控制台代表您创建的 IAM 角色”。
6. EventBridge 允许您选择下载模板文件或将模板复制到剪贴板。

   - 选择下载，下载模板文件。
   - 要将此模板复制到剪贴板，请选择复制。
7. 要退出模板，请选择取消。

通过全局端点和事件复制使应用程序具备区域容错能力

您可以使用 Amazon EventBridge 全局端点提高应用程序的可用性。全局端点使您的应用程序具有区域容错能力，且无需支付额外费用。首先，您需要向端点分配 Amazon Route 53 运行状况检查。启动失效转移后，运行状况检查会报告“运行不正常”状态。在失效转移启动后的几分钟内，所有自定义事件都将路由到辅助区域中的事件总线，并由该事件总线进行处理。一旦运行状况检查报告“正常”状态，事件将由主区域中的事件总线处理。

使用全局端点时，可以启用事件复制。事件复制使用托管规则将所有自定义事件发送到主区域和辅助区域中的事件总线。

> **Note**
>
> 如果您使用的是自定义总线，则需要在每个区域的相同账户中，使用名称相同的自定义总线，这样失效转移才能正常运行。

主题

恢复时间和恢复点目标

恢复时间目标 (RTO) 是在故障后辅助区域开始接收事件所花费的时间。对于 RTO，该时间包括触发 CloudWatch 警报和更新 Route 53 运行状况检查状态的时间段。恢复点目标 (RPO) 可衡量故障期间未处理的数据。对于 RPO，该时间包括未复制到辅助区域，且在服务或区域恢复之前停留在主区域的事件。使用全局端点，如果您遵循我们的警报配置规范性指南，则可以预计 RTO 和 RPO 为 360 秒，最长为 420 秒。

事件复制

辅助区域中的事件以异步方式处理。这意味着不能保证两个区域中的事件会同时处理。触发失效转移后，事件由辅助区域处理，并在主区域可用时由主区域处理。启用事件复制会增加您的月度费用。有关更多信息，请参阅 Amazon EventBridge 定价。

我们建议在设置全局端点时启用事件复制，原因如下：

- 事件复制可帮助您验证全局端点配置是否正确。这有助于确保在失效转移时为您提供保障。
- 要自动从失效转移事件中恢复，需要事件复制。如果您未启用事件复制，则必须手动将 Route 53 运行状况检查重置为“正常”，然后事件才会返回主区域。

复制的事件负载

以下是复制的事件负载示例：

> **Note**
>
> 对于 `region`，列出了复制事件的源区域。

```
{
    "version": "0",
    "id": "a908baa3-65e5-ab77-367e-527c0e71bbc2",
    "detail-type": "Test",
    "source": "test.service.com",
    "account": "0123456789",
    "time": "1900-01-01T00:00:00Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:events:us-east-1:0123456789:endpoint/MyEndpoint"
    ],
    "detail": {
        "a": "b"
    }
}
```

创建全局端点

完成以下步骤，设置全局端点：

1. 确保在主区域和辅助区域中有匹配的事件总线和规则。
2. 创建 Route 53 运行状况检查，监控您的事件总线。要在创建运行状况检查时获得帮助，请在创建全局端点时选择新建运行状况检查。
3. 创建您的全局端点。

设置 Route 53 运行状况检查后，即可创建全局端点。

使用控制台创建全局端点

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在导航窗格中，选择全局端点。
3. 选择 Create Endpoint（创建端点）。
4. 输入端点的名称和说明。
5. 对于主区域中的事件总线，请选择要与端点关联的事件总线。
6. 对于辅助区域，请选择失效转移时要将事件转到的区域。

> **Note**
>
> 辅助区域中的事件总线会自动填充且不可编辑。

7. 对于触发失效转移和恢复的 Route 53 运行状况检查，请选择端点将监控的运行状况检查。如果您还没有运行状况检查，请选择新建运行状况检查，打开 AWS CloudFormation 控制台，并使用 CloudFormation 模板创建运行状况检查。

> **Note**
>
> 缺少数据将导致运行状况检查失败。如果您只需要间歇性地发送事件，请考虑使用更长的 MinimumEvaluationPeriod，或者将丢失的数据视为“缺失”而不是“超出”。

8. （可选）对于事件复制，请执行以下操作：

   a. 选择事件复制已启用。

b. 对于执行角色，选择创建新的 AWS Identity and Access Management 角色或使用现有角色。执行以下操作：

- 选择 Create a new role for this specific resource。您也可以更新角色名称，创建新角色。
- 选择使用现有角色。然后，执行角色选择要使用的所需角色。

9. 选择 Create（创建）。

使用 API 创建全局端点

要使用 EventBridge API 创建全局端点，请参阅《Amazon Eventbridge API 参考》中的 CreateEndpoint。

使用 AWS CloudFormation 创建全局端点

要使用 AWS CloudFormation API 创建全局端点，请参阅《AWS CloudFormation 用户指南》中的 AWS::Events::Endpoints。

使用 AWS 开发工具包处理全局端点

> **Note**
>
> 即将推出对 C++ 的支持。

用 AWS 开发工具包处理全局端点时，请牢记以下几点：

- 您需要针对特定开发工具包安装 AWS 通用运行时 (CRT) 库。如果没有安装 CRT，会收到一条异常消息，指示需要安装的内容。有关更多信息，请参阅下列内容：
  - AWS 通用运行时 (CRT) 库
  - awslabs/aws-crt-java
  - awslabs/aws-crt-nodejs
  - awslabs/aws-crt-python
- 创建全局端点后，您需要将 `endpointId` 和 `EventBusName` 添加到您使用的任何 `PutEvents` 调用中。
- 全局端点支持签名版本 4A。此版本的 SigV4 允许对多个 AWS 区域签署请求。这对于可能导致从多个区域之一访问数据的 API 操作非常有用。使用 AWS 开发工具包时，您可以提供凭证，对全局端

点的请求将使用签名版本 4A，无需进行其他配置。有关 SigV4A 的更多信息，请参阅 AWS 一般参考中的签署 AWS API 请求。

可用区

以下区域支持全局端点。

- 美国东部（弗吉尼亚州北部）
- 美国东部（俄亥俄州）
- 美国西部（北加利福尼亚）
- 美国西部（俄勒冈州）
- 加拿大（中部）
- 欧洲地区（法兰克福）
- 欧洲地区（爱尔兰）
- 欧洲地区（伦敦）
- 欧洲地区（米兰）
- 欧洲地区（巴黎）
- 欧洲地区（斯德哥尔摩）
- 亚太地区（孟买）
- 亚太地区（大阪）
- 亚太地区（首尔）
- 亚太地区（新加坡）
- 亚太地区（悉尼）
- 亚太地区（东京）
- 南美洲（圣保罗）

使用 Amazon EventBridge 全局端点的最佳实践

在设置全局端点时，建议采用以下最佳实践。

主题

- 启用事件复制
- 防止事件节流

- 在 Amazon Route 53 运行状况检查中使用订阅用户指标

启用事件复制

我们强烈建议您在分配给全局端点的辅助区域启用复制，并处理事件。这样可以确保辅助区域中的应用程序配置正确。您还应启用复制，以确保出现的问题得到缓解后自动恢复到主区域。

事件 ID 可能会随着 API 调用而变化，因此跨区域关联事件需要有一个不可变的唯一标识符。在进行与使用者有关的设计时，还应考虑到幂等性。这样一来，如果您要复制事件或从存档中重放事件，在两个区域中处理事件就不会产生任何副作用。

防止事件节流

为防止事件被节流，我们建议您更新 `PutEvents` 和目标限制，使其在各区域保持一致。

在 Amazon Route 53 运行状况检查中使用订阅用户指标

请避免在 Amazon Route 53 运行状况检查中包含订阅用户指标。如果某订阅用户遇到问题，但主区域中所有其他订阅用户都运行正常，包含这些指标可能会导致发布者失效转移到辅助区域。如果您的一个订阅用户未能处理主区域中的事件，则应启用复制，以确保辅助区域的订阅用户能够成功处理事件。

用于设置 Route 53 运行状况检查的 AWS CloudFormation 模板

使用全局端点时，必须进行 Route 53 运行状况检查，以监控各区域的状态。以下模板定义了一条 Amazon CloudWatch 警报，并用它来定义 Route 53 运行状况检查。

主题

- 用于定义 Route 53 运行状况检查的 AWS CloudFormation 模板
- CloudWatch 警报模板属性
- Route 53 运行状况检查模板属性

用于定义 Route 53 运行状况检查的 AWS CloudFormation 模板

使用以下模板定义您的 Route 53 运行状况检查。

```
Description: |-
  Global endpoints health check that will fail when the average Amazon EventBridge
  latency is above 30 seconds for a duration of 5 minutes. Note, missing data will
```

```
  cause the health check to fail, so if you only send events intermittently, consider
  changing the heath check to use a longer evaluation period or instead treat missing
  data as 'missing' instead of 'breaching'.

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label:
          default: "Global endpoint health check alarm configuration"
        Parameters:
          - HealthCheckName
          - HighLatencyAlarmPeriod
          - MinimumEvaluationPeriod
          - MinimumThreshold
          - TreatMissingDataAs
    ParameterLabels:
      HealthCheckName:
        default: Health check name
      HighLatencyAlarmPeriod:
        default: High latency alarm period
      MinimumEvaluationPeriod:
        default: Minimum evaluation period
      MinimumThreshold:
        default: Minimum threshold
      TreatMissingDataAs:
        default: Treat missing data as

Parameters:
  HealthCheckName:
    Description: Name of the health check
    Type: String
    Default: LatencyFailuresHealthCheck
  HighLatencyAlarmPeriod:
    Description: The period, in seconds, over which the statistic is applied. Valid
 values are 10, 30, 60, and any multiple of 60.
    MinValue: 10
    Type: Number
    Default: 60
  MinimumEvaluationPeriod:
    Description: The number of periods over which data is compared to the specified
 threshold. You must have at least one evaluation period.
    MinValue: 1
    Type: Number
    Default: 5
```

```
  MinimumThreshold:
    Description: The value to compare with the specified statistic.
    Type: Number
    Default: 30000
  TreatMissingDataAs:
    Description: Sets how this alarm is to handle missing data points.
    Type: String
    AllowedValues:
      - breaching
      - notBreaching
      - ignore
      - missing
    Default: breaching

Mappings:
  "InsufficientDataMap":
    "missing":
      "HCConfig": "LastKnownStatus"
    "breaching":
      "HCConfig": "Unhealthy"

Resources:
  HighLatencyAlarm:
      Type: AWS::CloudWatch::Alarm
      Properties:
        AlarmDescription: High Latency in Amazon EventBridge
        MetricName: IngestionToInvocationStartLatency
        Namespace: AWS/Events
        Statistic: Average
        Period: !Ref HighLatencyAlarmPeriod
        EvaluationPeriods: !Ref MinimumEvaluationPeriod
        Threshold: !Ref MinimumThreshold
        ComparisonOperator: GreaterThanThreshold
        TreatMissingData: !Ref TreatMissingDataAs

  LatencyHealthCheck:
      Type: AWS::Route53::HealthCheck
      Properties:
        HealthCheckTags:
          - Key: Name
            Value: !Ref HealthCheckName
        HealthCheckConfig:
          Type: CLOUDWATCH_METRIC
          AlarmIdentifier:
```

```
            Name:
              Ref: HighLatencyAlarm
            Region: !Ref AWS::Region
          InsufficientDataHealthStatus: !FindInMap [InsufficientDataMap, !Ref
 TreatMissingDataAs, HCConfig]

Outputs:
  HealthCheckId:
    Description: The identifier that Amazon Route 53 assigned to the health check when
 you created it.
    Value: !GetAtt LatencyHealthCheck.HealthCheckId
```

事件 ID 可能会随着 API 调用而变化，因此跨区域关联事件需要有一个不可变的唯一标识符。在进行与使用者有关的设计时，还应考虑到幂等性。这样一来，如果您要复制事件或从存档中重放事件，在两个区域中处理事件就不会产生任何副作用。

CloudWatch 警报模板属性

> **Note**
>
> 对于所有 **`editable`** 字段，请考虑您的每秒吞吐量。如果您只是间歇性地发送事件，请考虑将运行状况检查更改为使用更长的评估期，或者将缺失的数据视为 `missing`，而不是 `breaching`。

模板的 CloudWatch 警报部分使用了以下属性：

指标	描述
`AlarmDescription`	警报的描述。 默认值：**`High Latency in Amazon EventBridge`**
`MetricName`	与警报关联的指标的名称。这是基于指标的警报所必需的。对于基于数学表达式的警报，您应改为使用 `Metrics`，并且无法指定 `MetricName` 。 默认：IngestionToInvocationStartLatency

指标	描述
Namespace	与警报关联的指标的命名空间。这是基于指标的警报所必需的。对于基于数学表达式的警报，您无法指定 Namespace ，而应改为使用 Metrics。 默认值：AWS/Events
Statistic	与警报关联的指标的统计数据，而不是百分位数。 默认：平均值
Period	以秒为单位的周期，每经过该时长，即应用统计数据。这是基于指标的警报所必需的。有效值为 10、30、60，以及 60 的任何倍数。 默认值：**60**
EvaluationPeriods	其间的数据将与指定阈值进行比较的期间数。如果您设置的警报需要连续超出多个数据点才能触发警报，则此值将指定该数字。如果要设置“N 个中的 M 个”警报，则此值为 N，DatapointsToAlarm 为 M。 默认值：**5**
Threshold	要与指定的统计数据进行比较的值。 默认值：**30,000**
ComparisonOperator	将指定统计数据与阈值进行比较时使用的算术运算。指定的统计值将用作第一个操作数。 默认值：GreaterThanThreshold
TreatMissingData	设置该警报应如何处理缺失数据点。 有效值：breaching 、notBreaching 、ignore 和 missing 默认值：breaching

Route 53 运行状况检查模板属性

> **Note**
>
> 对于所有 **editable** 字段，请考虑您的每秒吞吐量。如果您只是间歇性地发送事件，请考虑将运行状况检查更改为使用更长的评估期，或者将缺失的数据视为 missing，而不是 breaching。

模板的 Route 53 运行状况检查部分使用了以下属性：

指标	描述
HealthCheckName	运行状况检查的名称。 默认值：**LatencyFailuresHealthCheck**
InsufficientDataHealthStatus	当 CloudWatch 没有充足的指标数据来确定警报状态时，您希望 Amazon Route 53 分配给运行状况检查的状态。 有效值： • Healthy：Route 53 将运行状况检查视为正常。 • Unhealthy ：Route 53 将运行状况检查视为不正常。 • LastKnownStatus ：Route 53 使用 CloudWatch 上次有足够数据确定警报状态时运行状况检查的状态。对于没有上一个已知状态的新运行状况检查，运行状况检查的默认状态为运行良好。 默认：运行不正常 **Note** 此字段将根据 TreatMissingData 字段的输入进行更新。如果TreatingMissingData 设置为 Missing，它将更新为 LastKnownStatus 。如果 TreatingMissingData 设置为 Breaching ，则更新为 Unhealthy 。

亚马逊 EventBridge 架构

架构定义了发送到的事件的结构 EventBridge。 EventBridge 为 AWS 服务生成的所有事件提供架构。您还可以创建或上传架构，或者直接从事件总线中的事件推断架构。一旦您有了事件的架构，就可以下载常用编程语言的代码绑定，加快开发速度。您可以使用 EventBridge 控制台、API 或直接在 IDE 中使用工具包来处理架构的代码绑定并管理架构。 AWS 要构建使用事件的无服务器应用，请使用 AWS Serverless Application Model。

> **Note**
>
> 使用输入转换器功能时，会通过架构发现来推断原始事件，而不是发送到目标的转换后事件。

EventBridge 支持 OpenAPI 3 和 jsonSchema Draft4 格式。

对于适用于AWSVS Code 的AWS Toolkit JetBrains 和 Toolkit for VS Code，您可以直接在 IDE 中浏览或搜索架构并下载架构的代码绑定。

以下视频概述了架构和架构注册表：使用架构注册表

主题

- 架构注册表 API 属性值屏蔽
- 正在查找 Amazon EventBridge 架构
- Amazon EventBridge 架构注册表
- 创建亚马逊 EventBridge 架构
- 亚马逊 EventBridge 代码绑定

架构注册表 API 属性值屏蔽

用于创建架构注册表的事件，其某些属性值可能包含敏感的客户信息。为了保护客户信息，这些值将用星号 (*) 屏蔽。由于我们要屏蔽这些值，所以 EventBridge 建议不要构建明确依赖以下属性或其值的应用程序：

- CreateSchema— `requestParameters` 身体的`Content`属性

- GetDiscoveredSchema— `requestParameters` 身体的`Events`属性和`responseElements`身体的`Content`属性
- SearchSchemas— 的`keywords`财产 `requestParameters`
- UpdateSchema— 的`Content`财产 `requestParameters`

正在查找 Amazon EventBridge 架构

EventBridge 包括生成事件的所有 AWS 服务的架构。您可以在 EventBridge 控制台中找到这些架构，也可以使用 API 操作SearchSchemas找到它们。

在控制台中查找 AWS 服务架构 EventBridge

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择 Schemas (架构)。
3. 在架构页面上，选择 AWS 事件架构注册表。
   <result>

   将显示第一页的可用架构。
   </result>
4. 要查找架构，请在搜索 AWS 事件架构中输入搜索词。

   搜索将返回可用架构的名称和内容的匹配项，并显示哪些架构版本包含匹配项。
5. 通过选择事件架构的名称来打开此架构。

Amazon EventBridge 架构注册表

架构注册表是架构的容器。架构注册表收集和组织架构，以便您的架构位于逻辑组中。默认架构注册表为：

- 所有架构-来自 AWS 事件、已发现和自定义架构注册表的所有架构。
- AWS 事件架构注册表-内置架构。
- 已发现的架构注册表 - 架构发现发现的架构。

可以创建自定义注册表，整理您创建或上传的架构。

创建自定义注册表

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择架构，然后选择创建注册表。
3. 在注册表详细信息页面上，输入名称。
4. （可选）输入新注册表的描述。
5. 选择创建。

要在新注册表中创建自定义架构，请选择创建自定义架构。要向注册表中添加架构，请在创建新架构时选择该注册表。

要使用 API 创建注册表，请使用 `CreateRegistry`。有关更多信息，请参阅 Amazon EventBridge 架构注册表 API 参考。

有关通过使用 EventBridge 架构注册表的信息 AWS CloudFormation，请参阅中的EventSchemas 资源类型参考 AWS CloudFormation。

创建亚马逊 EventBridge 架构

可以使用 OpenAPI 规范或 JSONSchema Draft4 规范的 JSON 文件来创建架构。您可以使用模板或根据事件的 JSON 生成架构，在中 EventBridge 创建或上传自己的架构。您也可以从事件总线中的事件推断出架构。要使用架构注册表 API 创建 EventBridge 架构，请使用 `CreateSchema`API 操作。

在 OpenAPI 3 和 JSONSchema Draft4 格式之间进行选择时，请考虑以下区别：

- JSONSchema 格式支持 OpenAPI 不支持的其他关键字，例如 `$schema, additionalItems`。
- 关键字的处理方式略有不同，例如 `type` 和 `format`。
- OpenAPI 不支持 JSON 文档中的 JSONSchema Hyper-Schema 超链接。
- OpenAPI 的工具往往侧重于构建时，而 JSONSchema 的工具往往侧重于运行时操作，例如用于架构验证的客户端工具。

我们建议使用 JSonSchema 格式来实现客户端验证，以便发送的事件 EventBridge 符合架构。您可以使用 JSONSchema 为有效的 JSON 文档定义合同，然后在发送关联事件之前使用 JSON 架构验证器。

有了新架构后，您可以下载代码绑定，以帮助使用该架构为事件创建应用程序。

主题

使用模板创建架构

您可以根据模板创建架构，也可以直接在 EventBridge控制台中编辑模板。要获取模板，请从控制台下载该模板。您可以编辑模板，使架构与您的事件匹配。然后通过控制台上传您的新模板。

下载架构模板

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择 Schema registry (架构注册表)。
3. 在架构模板下的入门部分，选择下载。

或者，您也可以从以下代码示例中复制 JSON 模板。

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

上传架构模板

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择架构，然后选择创建架构。
3. （可选）选择或创建架构注册表。
4. 在架构详细信息下，输入架构的名称。
5. （可选）为架构输入描述。
6. 对于架构类型，请选择 OpenAPI 3.0 或 JSON Schema Draft 4。
7. 在创建 选项卡上，将架构文件拖动到文本框中，或粘贴架构源。
8. 选择创建。

直接在控制台中编辑架构模板

在控制台中编辑架构

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择架构，然后选择创建架构。
3. （可选）选择或创建架构注册表。
4. 在架构详细信息下，输入架构的名称。
5. 对于架构类型，请选择 OpenAPI 3.0 或 JSON Schema Draft 4。
6. （可选）为创建的架构输入描述。
7. 在创建选项卡上，选择加载模板。
8. 在文本框中编辑模板，使架构与您的事件匹配。
9. 选择创建。

根据事件的 JSON 创建架构

如果您有事件的 JSON，则可以自动为该类型的事件创建架构。

基于事件的 JSON 创建架构

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择架构，然后选择创建架构。

3. （可选）选择或创建架构注册表。
4. 在 Schema details (架构详细信息) 下，输入架构的名称。
5. （可选）输入所创建架构的描述。
6. 架构类型选择 OpenAPI 3.0。

   根据事件的 JSON 创建架构时，不能使用 JSONSchema。
7. 选择 Discover from JSON (从 JSON 中发现)
8. 在 JSON 下的文本框中，粘贴或拖动事件的 JSON 源。

   例如，对于执行失败，您可以粘贴此 AWS Step Functions 事件的源代码。

```
{
    "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": [
      "arn:aws:states:us-east-1:012345678912:execution:state-machine-
name:execution-name"
    ],
    "detail": {
        "executionArn": "arn:aws:states:us-east-1:012345678912:execution:state-
machine-name:execution-name",
        "stateMachineArn": "arn:aws:states:us-
east-1:012345678912:stateMachine:state-machine",
        "name": "execution-name",
        "status": "FAILED",
        "startDate": 1551225146847,
        "stopDate": 1551225151881,
        "input": "{}",
        "output": null
    }
}
```

9. 选择 发现架构。
10. EventBridge 为该事件生成一个 OpenAPI 架构。例如，以下架构是为上一 Step Functions 事件生成的。

```
{
  "openapi": "3.0.0",
  "info": {
    "version": "1.0.0",
    "title": "StepFunctionsExecutionStatusChange"
  },
  "paths": {},
  "components": {
    "schemas": {
      "AWSEvent": {
        "type": "object",
        "required": ["detail-type", "resources", "detail", "id", "source", "time",
 "region", "version", "account"],
        "x-amazon-events-detail-type": "Step Functions Execution Status Change",
        "x-amazon-events-source": "aws.states",
        "properties": {
          "detail": {
            "$ref": "#/components/schemas/StepFunctionsExecutionStatusChange"
          },
          "account": {
            "type": "string"
          },
          "detail-type": {
            "type": "string"
          },
          "id": {
            "type": "string"
          },
          "region": {
            "type": "string"
          },
          "resources": {
            "type": "array",
            "items": {
              "type": "string"
            }
          },
          "source": {
            "type": "string"
          },
          "time": {
            "type": "string",
            "format": "date-time"
```

```
          },
          "version": {
            "type": "string"
          }
        }
      },
      "StepFunctionsExecutionStatusChange": {
        "type": "object",
        "required": ["output", "input", "executionArn", "name", "stateMachineArn",
 "startDate", "stopDate", "status"],
        "properties": {
          "executionArn": {
            "type": "string"
          },
          "input": {
            "type": "string"
          },
          "name": {
            "type": "string"
          },
          "output": {},
          "startDate": {
            "type": "integer",
            "format": "int64"
          },
          "stateMachineArn": {
            "type": "string"
          },
          "status": {
            "type": "string"
          },
          "stopDate": {
            "type": "integer",
            "format": "int64"
          }
        }
      }
    }
  }
}
```

11. 生成架构后，选择创建。

根据事件总线中的事件创建架构

EventBridge 可以通过发现事件来推断架构。要推断架构，您可以针对事件总线开启事件发现，并将每个唯一架构添加到架构注册表中，包括跨账户事件的架构。由发现的架构 EventBridge 显示在 “架构” 页面上的 “已发现架构” 注册表中。

如果事件总线上的事件内容发生变化，则 EventBridge 会创建相关 EventBridge 架构的新版本。

> **Note**
>
> 在事件总线上启用事件发现会产生成本。每个月处理的前 500 万个事件是免费的。

> **Note**
>
> EventBridge 默认情况下，根据跨账户事件推断架构，但您可以通过更新属性将其禁用。`cross-account`有关更多信息，请参阅《 EventBridge 架构注册表 API 参考》中的 Discoverers。

在事件总线上启用架构发现

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择 Event Buses (事件总线)。
3. 请执行以下操作之一：
   - 要在默认事件总线上启用发现，请选择开始发现。
   - 要在自定义事件总线上启用发现，请选择自定义事件总线的单选按钮，然后选择开始发现。

亚马逊 EventBridge 代码绑定

您可以为事件架构生成代码绑定，以加快 Golang、Java、Python 和中的开发速度。 TypeScript代码绑定适用于 AWS 服务事件，您创建的架构，以及基于事件总线中的事件生成的架构。您可以使用 EventBridge 控制台、架构注册表 API 或在 IDE 中使用 AWS 工具包为 EventBridge 架构生成代码绑定。

从架构生成代码绑定 EventBridge

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择 Schemas (架构)。
3. 通过浏览架构注册表或搜索架构，找到您希望进行代码绑定的架构。
4. 选择架构名称。
5. 在架构详细信息页面的版本部分，选择下载代码绑定。
6. 在 Download code bindings (下载代码绑定) 页上，选择要下载的代码绑定的语言。
7. 选择 Download (下载)。

   开始下载可能需要几秒钟。下载的文件是所选语言的代码绑定 zip 文件。

Amazon EventBridge 相关服务和工具

Amazon EventBridge 与其他 AWS 服务和工具配合使用，可处理事件或调用资源，作为规则的目标。有关 EventBridge 与其他 AWS 服务和工具集成的更多信息，请参阅以下主题：

主题

- 将 Amazon EventBridge 与接口 VPC 端点配合使用
- Amazon EventBridge 与 AWS X-Ray 集成
- EventBridge 与 AWS 集成应用程序测试套件一起使用
- 在 AWS CloudFormation 堆栈中加入 Amazon EventBridge 资源

将 Amazon EventBridge 与接口 VPC 端点配合使用

如果您使用 Amazon Virtual Private Cloud (Amazon VPC) 托管 AWS 资源，则可以在您的 VPC 和 EventBridge 之间建立私有连接。您的 VPC 中的资源可以使用此连接与 EventBridge 进行通信。

借助 VPC，您可以控制您的网络设置，如 IP 地址范围、子网、路由表和网络网关。要将 VPC 连接到 EventBridge，请为 EventBridge 定义一个接口 VPC 端点。该端点提供了到 EventBridge 的可靠、可扩展的连接，无需互联网网关、网络地址转换 (NAT) 实例或 VPN 连接。有关更多信息，请参阅 Amazon VPC 用户指南中的什么是 Amazon VPC。

接口 VPC 端点由 AWS PrivateLink 提供支持，后者可将弹性网络接口与私有 IP 地址相结合，支持 AWS 服务之间的私有通信。有关更多信息，请参阅 AWS PrivateLink 和 VPC 端点。

当您使用私有接口 VPC 端点时，请自定义您的 VPC 使用该端点发送到 EventBridge 的事件。然后，EventBridge 会根据您配置的规则和目标，将这些事件发送到其他 AWS 服务。将事件发送到其他服务后，您可以通过该服务的公共端点或 VPC 端点接收这些事件。例如，如果您创建了一条规则，向 Amazon SQS 队列发送事件，则可以为 Amazon SQS 配置接口 VPC 端点，在不使用公共端点的情况下，接收来自您的 VPC 中该队列的消息。

可用性

EventBridge 当前在以下区域支持 VPC 端点：

- 美国东部（俄亥俄）
- 美国东部（弗吉尼亚州北部）
- 美国西部（北加利福尼亚）
- 美国西部（俄勒冈州）
- 非洲（开普敦）
- 亚太地区（孟买）
- 亚太地区（海得拉巴）
- 亚太地区（香港）
- 亚太地区（新加坡）
- 亚太地区（悉尼）
- 亚太地区（雅加达）
- 亚太地区（墨尔本）
- 亚太地区（东京）

- 亚太地区 (首尔)
- 亚太地区（大阪）
- 加拿大（中部）
- 加拿大西部（卡尔加里）
- 中国（北京）
- 中国（宁夏）
- 欧洲地区（法兰克福）
- 欧洲（苏黎世）
- 欧洲地区（爱尔兰）
- 欧洲地区（伦敦）
- 欧洲地区（米兰）
- 欧洲（西班牙）
- 欧洲地区（巴黎）
- 欧洲（斯德哥尔摩）
- 中东（阿联酋）
- 中东（巴林）
- 南美洲（圣保罗）
- 以色列（特拉维夫）
- AWS GovCloud（美国西部）
- AWS GovCloud（美国东部）

为 EventBridge 创建 VPC 端点

要将 EventBridge 与您的 VPC 结合使用，请为 EventBridge 创建一个接口 VPC 端点，然后选择 com.amazonaws.***Region***.events 作为服务名称。有关更多信息，请参阅 Amazon VPC 用户指南中的创建接口终端节点。

EventBridge Pipes 具体信息

EventBridge Pipes 不提供对接口 VPC 端点的全面支持。要将 VPC 中的以下源与 EventBridge Pipes 配合使用，请参阅以下内容：

- Amazon MSK 网络配置

- 自托管 Apache Kafka 网络配置
- Amazon MQ 网络配置

Amazon EventBridge 与 AWS X-Ray 集成

您可以使用 AWS X-Ray 来跟踪在 EventBridge 中传递的事件。EventBridge 将原始跟踪标头传递给目标，供目标服务跟踪、分析和调试。

只有事件来自可传递跟踪上下文的 `PutEvents` 请求时，EventBridge 才能为该事件传递跟踪标头。X-Ray 不会跟踪来自第三方合作伙伴、计划事件或 AWS 服务的事件，这些事件源也不会出现在您的 X-Ray 服务图中。

X-Ray 会验证跟踪标头，无效的跟踪标头会被丢弃，但仍会处理该事件。

Important

对于传送到调用目标的事件，跟踪标头不可用。

- 如果您有事件存档，则跟踪标头对于存档事件不可用。如果您重放存档事件，其中不包括跟踪标头。
- 如果您有死信队列 (DLQ)，跟踪标头包含在将事件发送到 DLQ 的 `SendMessage` 请求中。如果您使用 `ReceiveMessage` 从 DLQ 中检索事件（消息），则与该事件关联的跟踪标头将包含在 Amazon SQS 消息属性中，但不包含在事件消息中。

有关 EventBridge 事件节点如何连接源和目标服务的信息，请参阅《AWS X-Ray 开发人员指南》中的在 X-Ray 服务图中查看源和目标。

您可以通过 EventBridge 传递以下跟踪标头信息：

- 默认 HTTP 标头 - 对于所有调用目标，X-Ray SDK 会自动将跟踪标头填充为 `X-Amzn-Trace-Id` HTTP 标头。要了解有关默认 HTTP 标头的更多信息，请参阅《AWS X-Ray 开发人员指南》中的跟踪标头。
- **`TraceHeader`** 系统属性 - `TraceHeader` 是 EventBridge 保留的 PutEventsRequestEntry 属性，用于将 X-Ray 跟踪标头传送到目标。如果您还使用 `PutEventsRequestEntry`，`PutEventsRequestEntry` 会覆盖 HTTP 跟踪标头。

Note

跟踪标头不会计入 `PutEventsRequestEntry` 事件大小。有关更多信息，请参阅计算 Amazon EventBridge PutEvents 活动条目大小。

以下视频演示了如何将 X-Ray 和 EventBridge 配合使用：将 AWS X-Ray 用于跟踪

EventBridge 与 AWS 集成应用程序测试套件一起使用

当您创建由 Lambda EventBridge 或 Step Functions 等无服务器服务组成的应用程序时，您的许多架构组件无法部署到您的桌面，而只能存在于云中 AWS 。与使用本地部署的应用程序相比，这些类型的应用程序受益于基于云的自动测试策略。 AWS 集成应用程序测试套件 (AWS IATK) 可帮助您为应用程序实施其中一些策略。

AWS IATK 是一个软件库，可帮助您为基于云的应用程序编写自动测试。

EventBridge 与 AWS IATK 集成

您可以将 EventBridge 事件和事件总线与 AWS IATK 配合使用来实现自动测试，包括：

实现测试设备

要为事件驱动型架构编写集成测试，请通过将应用程序分解为子系统来建立逻辑边界。一种实用的子系统测试方法是创建测试设备，也就是您专门为测试子系统而创建的资源。

例如，集成测试可以通过向子系统进程传递输入测试事件来启动子系统进程。 AWS IATK 可以为您创建一个用于监听输出事件 EventBridge 的测试工具。（在幕后，该工具由一条将输出事件转发到 Amazon SQS 的 EventBridge 规则组成。）然后，您的集成测试会查询测试设备，以检查输出内容并确定测试是通过还是失败。

生成模拟事件

AWS IATK 使您能够从存储在架构注册表中的架构生成模拟事件。 EventBridge 这允许您生成模拟事件并使用生成的事件调用任何使用方（例如 Lambda 函数或 Step Functions 状态机）。

有关更多信息，请参阅上的AWS 集成应用程序测试套件概述 GitHub。

在 AWS CloudFormation 堆栈中加入 Amazon EventBridge 资源

AWS CloudFormation 可将基础设施视为代码，使您能够以集中且可重复的方式跨账户和区域配置和管理 AWS 资源。CloudFormation 实现这一功能的方式是允许您创建模板，它定义了您要预置和管理的资源。这些资源可能包括 EventBridge 构件，例如事件总线和规则、管道、架构和计划等。使用这些资源将 EventBridge 功能包含在您通过 CloudFormation 预置和管理的技术堆栈中。

AWS CloudFormation 中的 Amazon EventBridge 资源

EventBridge 在以下资源命名空间中提供可用于 CloudFormation 模板的资源：

- AWS::Events

  模板示例包括：

  - 为 PagerDuty 创建 API 目标
  - 为 Slack 创建 API 目标
  - 使用 ApiKey 授权参数创建连接
  - 使用 OAuth 授权参数创建连接
  - 创建带有事件复制的全局端点
  - 使用多个委托人和操作的拒绝策略
  - 使用自定义事件总线向组织授予权限
  - 创建跨区域规则
  - 创建包含目标的死信队列的规则
  - 定期调用 Lambda 函数
  - 调用 Lambda 函数以响应事件
  - 通知主题以响应日志条目
- AWS::EventSchemas
- AWS::Pipes

  模板示例包括：

  - 使用事件筛选条件创建管道
- AWS::Scheduler

为 AWS CloudFormation 模板生成 Amazon EventBridge 资源定义

为了帮助您快速开始开发 CloudFormation 模板，EventBridge 控制台让您可以根据账户中的现有事件总线、规则和管道创建 CloudFormation 模板。

- ???
- ???
- ???

使用 EventBridge 管理 AWS CloudFormation 堆栈事件

除了在 CloudFormation 堆栈中包含 EventBridge 资源外，您还可以使用 EventBridge 管理由 CloudFormation 堆栈本身生成的事件。每当对堆栈执行创建、更新、删除或偏差检测操作时，CloudFormation 都可以向 EventBridge 发送事件。CloudFormation 还会向 EventBridge 发送事件，以更改堆栈集和堆栈集实例的状态。您可以使用 EventBridge 规则将事件路由到定义的目标。

有关更多信息，请参阅《AWS CloudFormation 用户指南》中的使用 EventBridge 管理 CloudFormation 事件。

Amazon EventBridge 教程

EventBridge 与多项 AWS 服务和多个 SaaS 合作伙伴相集成。这些教程旨在帮助您熟悉 EventBridge 的基础知识，以及如何将其融入您的无服务器架构。

教程:

- Amazon EventBridge 入门教程
- Amazon EventBridge 与其他 AWS 服务集成的教程
- 用于与 SaaS 提供商集成的 Amazon EventBridge 教程

Amazon EventBridge 入门教程

以下教程可帮助您了解 EventBridge 的功能及其使用方式。

教程:

- 存档并重放 Amazon EventBridge 事件
- 创建 Amazon EventBridge 示例应用程序
- 教程：使用 EventBridge 架构注册表下载事件的代码绑定
- 教程：使用输入转换器自定义 EventBridge 传递给事件目标的内容

存档并重放 Amazon EventBridge 事件

您可以使用 EventBridge，通过规则将事件路由到特定 AWS Lambda 函数。

在本教程中，您将使用 Lambda 控制台创建一个函数，用作 EventBridge 规则的目标。然后，您将创建一个存档和一条规则，使用 EventBridge 控制台存档测试事件。该存档中有事件后，您将重放它们。

步骤：

- 步骤 1：创建 Lambda 函数
- 步骤 2：创建存档
- 步骤 3：创建规则
- 步骤 4：发送测试事件
- 步骤 5：重放事件
- 步骤 6：清除资源

步骤 1：创建 Lambda 函数

首先，创建 Lambda 函数以记录事件。

创建 Lambda 函数：

1. 打开 AWS Lambda 控制台，地址：https://console.aws.amazon.com/lambda/。
2. 选择创建函数。
3. 选择 Author from scratch (从头开始创作)。
4. 输入 Lambda 函数的名称和说明。例如，将函数命名为 `LogScheduledEvent`。
5. 将其余选项保留为默认值，然后选择创建函数。
6. 在函数页面的代码选项卡上，双击 index.js。
7. 使用以下代码替换现有 JavaScript 代码：

```
'use strict';

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

```
};
```

8. 选择 Deploy（部署）。

步骤 2：创建存档

接下来，创建存档以保存所有测试事件。

创建存档

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在导航窗格中，选择存档。
3. 选择创建存档。
4. 输入存档的名称和说明。例如，将存档命名为 `ArchiveTest`。
5. 将其余选项保留为默认值，然后选择下一步。
6. 选择创建存档。

步骤 3：创建规则

创建规则，用于存档发送到事件总线的事件。

创建 规则

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Rules (规则)。
3. 选择 Create rule (创建规则)。
4. 为规则输入名称和描述。例如，将规则命名为 `ARTestRule`。

   规则不能与同一区域中的另一个规则和同一事件总线上的名称相同。
5. 对于 Event bus（事件总线），请选择要与此规则关联的事件总线。如果您希望此规则匹配来自您账户的事件，请选择默认。当您账户中的某个 AWS 服务发出一个事件时，它始终会发送到您账户的默认事件总线。
6. 对于 Rule type（规则类型），选择 Rule with an event pattern（具有事件模式的规则）。
7. 选择 Next（下一步）。
8. 对于 Event source（事件源），选择 Other（其他）。

9. 对于事件模式，输入以下代码：

```
{
  "detail-type": [
    "customerCreated"
  ]
}
```

10. 选择 Next（下一步）。
11. 对于 Target types（目标类型），选择 AWS service（服务）。
12. 对于选择目标，从下拉列表中选择 Lambda 函数。
13. 在函数中，选择您在步骤 1：创建 Lambda 函数部分创建的 Lambda 函数。在此示例中，选择 `LogScheduledEvent`。
14. 选择 Next（下一步）。
15. 选择 Next（下一步）。
16. 查看规则详细信息并选择 Create rule（创建规则）。

步骤 4：发送测试事件

现在，您已经设置了存档和规则，我们将发送测试事件，以确保存档可正常运行。

> **Note**
>
> 事件可能需要一些时间才能进入存档。

发送测试事件（控制台）

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Event Buses (事件总线)。
3. 在默认事件总线图块中，选择操作、发送事件。
4. 输入事件源。例如，`TestEvent`。
5. 在详细信息类型中，输入 `customerCreated`。
6. 在事件详细信息中，输入 `{}`。
7. 选择 Send (发送)。

步骤 5：重放事件

测试事件进入存档后，您可以重放它们。

重放存档的事件（控制台）

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在导航窗格中，选择重放。
3. 选择启动新的重放。
4. 输入重放的名称和说明。例如，将重放命名为 `ReplayTest`。
5. 对于源，选择您在步骤 2：创建存档部分创建的存档。
6. 在重放时间范围部分，执行以下操作。
   a. 在开始时间中，选择您发送测试事件的日期和发送前的时间。例如，`2021/08/11` 和 `08:00:00`。
   b. 结束时间选择当前日期和时间。例如，`2021/08/11` 和 `09:15:00`。
7. 选择启动重放。

步骤 6：清除资源

除非您想要保留为本教程创建的资源，否则可立即将其删除。请删除您不再使用的 AWS 资源，这样可防止您的 AWS 账户产生不必要的费用。

删除 Lambda 函数

1. 打开 Lamba 控制台的函数页面。
2. 选择您创建的函数。
3. 依次选择 Actions 和 Delete。
4. 选择 Delete (删除)。

删除 EventBridge 存档

1. 在 EventBridge 控制台中打开存档页面。
2. 选择您创建的存档。
3. 选择 Delete (删除)。
4. 输入存档名称，然后选择删除。

删除 EventBridge 规则

1. 在 EventBridge 控制台中打开规则页面。
2. 选择您创建的规则。
3. 选择 Delete。
4. 选择 Delete。

创建 Amazon EventBridge 示例应用程序

您可以使用 EventBridge，通过规则将事件路由到特定 Lambda 函数。

在本教程中，您将使用 AWS CLI、Node.js 和 GitHub 存储库中的代码来创建以下内容：

- 一个 AWS Lambda 函数，为银行 ATM 交易生成事件。
- 三个 Lambda 函数，用作 EventBridge 规则的目标。
- 以及根据事件模式将创建的事件路由到正确的下游函数的规则。

此示例使用 AWS SAM 模板来定义 EventBridge 规则。要了解有关如何在 EventBridge 中使用 AWS SAM 模板的更多信息，请参阅 ???。

在存储库中，atmProducer 子目录包含 `handler.js`，代表生成事件的 ATM 服务。这段代码是用 Node.js 编写的 Lambda 处理程序，它使用这行 JavaScript 代码通过 AWS 开发工具包将事件发布到 EventBridge。

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

此目录还包含 `events.js`，在条目数组中列出了几个测试交易。在 JavaScript 中，单个事件的定义如下：

```
{
  // Event envelope fields
  Source: 'custom.myATMapp',
  EventBusName: 'default',
  DetailType: 'transaction',
  Time: new Date(),

  // Main event body
  Detail: JSON.stringify({
    action: 'withdrawal',
    location: 'MA-BOS-01',
    amount: 300,
    result: 'approved',
    transactionId: '123456',
    cardPresent: true,
    partnerBank: 'Example Bank',
    remainingFunds: 722.34
  })
```

```
}
```

事件的 Detail 部分指定了交易属性。其中包括 ATM 的位置、金额、合作银行和交易结果。

atmConsumer 子目录中的 `handler.js` 文件包含三个函数：

```
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))
}
```

每个函数都会接收交易事件，这些事件通过 `console.log` 语句记录到 Amazon CloudWatch Logs 中。消费者函数独立于生产者运行，它们不知道事件的源。

路由逻辑包含在 EventBridge 规则中，这些规则由应用程序的 AWS SAM 模板部署。这些规则会评估传入的事件流，并将匹配的事件路由到目标 Lambda 函数。

这些规则使用的事件模式是 JSON 对象，与它们匹配的事件具有相同的结构。以下是其中一条规则的事件模式。

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

步骤：

- 先决条件

先决条件

完成本教程需要以下资源：

- 一个 AWS 账户。创建一个 AWS 账户（如果还没有账户）。
- 已安装 AWS CLI。要安装 AWS CLI，请参阅安装、更新和卸载 AWS CLI 版本 2。
- 已安装 Node.js 12.x。要安装 Node.js，请参阅下载。

步骤 1：创建应用程序

要设置示例应用程序，您需要使用 AWS CLI 和 Git 来创建所需的 AWS 资源。

创建应用程序

1. 登录 AWS。
2. 在本地计算机上安装 Git 并安装 AWS Serverless Application Model CLI。
3. 创建一个新目录，然后在终端中导航到该目录。
4. 在命令行输入 `git clone https://github.com/aws-samples/amazon-eventbridge-producer-consumer-example`。
5. 在命令行中运行以下命令：

```
cd ./amazon-eventbridge-producer-consumer-example
sam deploy --guided
```

6. 在终端上执行以下操作：

   a. 对于 **Stack Name**，输入堆栈的名称。例如，将堆栈命名为 Test。
   b. 对于 **AWS Region**，输入区域。例如，us-west-2。
   c. 对于 **Confirm changes before deploy**，输入 Y。
   d. 对于 **Allow SAM CLI IAM role creation**，输入 Y。
   e. 对于 **Save arguments to configuration file**，输入 Y。

f. 对于 **SAM configuration file**，输入 samconfig.toml。

g. 对于 **SAM configuration environment**，输入 default。

步骤 2：运行应用程序

您已设置资源，现在可以利用控制台测试功能。

运行应用程序

1. 在您部署 AWS SAM 应用程序的同一区域中打开 Lambda 控制台。
2. 有四个 Lambda 函数的前缀为 atm-demo。选择 atmProducerFn 函数，然后选择操作、测试。
3. 对于名称，输入 Test。
4. 选择 Test（测试）。

步骤 3：检查日志并验证应用程序运行是否正常

现在，应用程序已运行，您将使用控制台查看 CloudWatch Logs。

查看日志

1. 在您运行 AWS SAM 应用程序的同一区域中打开 CloudWatch 控制台。
2. 选择 Logs（日志），然后选择 Log groups（日志组）。
3. 选择包含 atmConsumerCase1 的日志组。您会看到两个数据流，分别代表 ATM 批准的两笔交易。选择要查看输出的日志流。
4. 返回日志组列表，然后选择包含 atmConsumerCase2 的日志组。您将看到两个数据流，分别代表与纽约 位置筛选器匹配的两笔交易。
5. 返回日志组列表，然后选择包含 atmConsumerCase3 的日志组。打开流，查看被拒绝的交易。

步骤 4：清理资源

除非您想要保留为本教程创建的资源，否则可立即将其删除。请删除您不再使用的 AWS 资源，这样可防止您的 AWS 账户产生不必要的费用。

删除 EventBridge 规则

1. 在 EventBridge 控制台中打开规则页面。

2. 选择您创建的规则。
3. 选择 Delete。
4. 选择 Delete。

删除 Lambda 函数

1. 打开 Lamba 控制台的函数页面。
2. 选择您创建的函数。
3. 依次选择 Actions 和 Delete。
4. 选择 Delete (删除)。

删除 CloudWatch Logs 日志组

1. 打开 CloudWatch 控制台。
2. 依次选择日志、日志组。
3. 选择在本教程中创建的日志组。
4. 依次选择 Actions（操作）和 Delete log group(s)（删除日志组）。
5. 选择 Delete。

教程：使用 EventBridge 架构注册表下载事件的代码绑定

您可以为事件架构生成代码绑定，以加快 Golang、Java、Python 和 TypeScript 的开发。对于现有 AWS 服务、您创建的架构以及基于事件总线中的事件生成的架构，您可以获取代码绑定。您可以使用以下方式之一为架构生成代码绑定：

- EventBridge 控制台
- EventBridge 架构注册表 API
- 带有 AWS 工具包的 IDE

在本教程中，对于 AWS 服务的事件，您要从 EventBridge 架构生成和下载代码绑定。

从 EventBridge 架构生成代码绑定

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Schemas (架构)。
3. 选择 AWS 事件架构注册表选项卡。
4. 浏览架构注册表或搜索架构，找到您希望对其进行代码绑定的 AWS 服务的架构。
5. 选择架构名称。
6. 在架构详细信息页面的版本部分，选择下载代码绑定。
7. 在 Download code bindings (下载代码绑定) 页上，选择要下载的代码绑定的语言。
8. 选择 Download (下载)。

   开始下载可能需要几秒钟。下载文件将是所选语言的代码绑定的 .zip 文件。
9. 解压缩下载的文件，将其添加到您的项目中。

   下载的程序包中包含一个 README 文件，介绍了如何在各种框架中配置程序包的依赖关系。

在您自己的代码中使用这些代码绑定，有助于使用此 EventBridge 事件快速构建应用程序。

教程：使用输入转换器自定义 EventBridge 传递给事件目标的内容

在 EventBridge 中，将事件传递给规则的目标之前，可以使用输入转换器自定义事件的文本。

方法是从事件中定义 JSON 路径，并将其输出分配给不同的变量。然后，您可以在输入模板中使用这些变量。不能对字符 < 和 > 进行转义。有关更多信息，请参阅亚马逊 EventBridge 输入转换。

> **Note**
>
> 如果您指定一个变量以匹配在事件中不存在的 JSON 路径，则不会创建该变量，并且不会在输出中显示该变量。

在本教程中，您使用 `detail-type: "customerCreated"` 创建与事件匹配的规则。输入转换器将变量 `type` 映射到事件中的 $.detail-type JSON 路径。然后，EventBridge 将此变量放入输入模板 "This event was <type>." 结果是以下 Amazon SNS 消息。

```
"This event was of customerCreated type."
```

步骤：

- 步骤 1：创建一个 Amazon SNS 主题
- 步骤 2：创建 Amazon SNS 订阅
- 步骤 3：创建规则
- 步骤 4：发送测试事件
- 步骤 5：确认成功
- 步骤 6：清除资源

步骤 1：创建一个 Amazon SNS 主题

创建一个主题，接收来自 EventBridge 的事件。

要创建主题，请执行以下操作

1. 通过以下网址打开 Amazon SNS 控制台：https://console.aws.amazon.com/sns/v3/home。
2. 在导航窗格中，选择 Topics（主题）。
3. 选择 Create topic（创建主题）。

4. 对于 Type（类型），选择 Standard（标准）。
5. 输入 **eventbridge-IT-test** 作为主题名称。
6. 选择 Create topic（创建主题）。

步骤 2：创建 Amazon SNS 订阅

创建订阅，接收包含转换信息的电子邮件。

创建订阅

1. 通过以下网址打开 Amazon SNS 控制台：https://console.aws.amazon.com/sns/v3/home。
2. 在导航窗格中，选择 Subscriptions。
3. 选择 Create subscription。
4. 对于主题 ARN，选择您在步骤 1 中创建的主题。在本教程中，选择 eventbridge-IT-test。
5. 对于协议，选择电子邮件。
6. 对于 Endpoint (终端节点)，输入您的电子邮件地址。
7. 选择 Create subscription（创建订阅）。
8. 在收到的 AWS 通知电子邮件中选择确认订阅，以确认订阅。

步骤 3：创建规则

创建规则，使用输入转换器自定义传送到目标的实例状态信息。

创建 规则

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Rules (规则)。
3. 选择 Create rule (创建规则)。
4. 为规则输入名称和描述。例如，将规则命名为 `ARTestRule`
5. 对于 Event bus（事件总线），请选择要与此规则关联的事件总线。如果您希望此规则匹配来自您账户的事件，请选择默认。当您账户中的某个 AWS 服务发出一个事件时，它始终会发送到您账户的默认事件总线。
6. 对于 Rule type（规则类型），选择 Rule with an event pattern（具有事件模式的规则）。
7. 选择 Next（下一步）。

8. 对于 Event source（事件源），选择 Other（其他）。
9. 对于事件模式，输入以下代码：

```
{
  "detail-type": [
    "customerCreated"
  ]
}
```

10. 选择 Next（下一步）。
11. 对于 Target types（目标类型），选择 AWS service（服务）。
12. 对于选择目标，从下拉列表中选择 SNS 主题。
13. 对于主题，请选择您在步骤 1 中创建的 Amazon SNS 主题。在本教程中，选择 eventbridge-IT-test。
14. 对于其他设置，执行以下操作：
    a. 对于配置目标输入，从下拉列表中选择输入转换器。
    b. 选择配置输入转换器。
    c. 对于示例事件，输入以下代码：

```
{
  "detail-type": "customerCreated"
}
```

    d. 对于目标输入转换器，执行以下操作：
       i. 对于输入路径，输入以下代码：

```
{"detail-type":"$.detail-type"}
```

       ii. 对于输入模板，输入以下代码：

```
"This event was of <detail-type> type."
```

    e. 选择确认。
15. 选择 Next（下一步）。
16. 选择 Next（下一步）。
17. 查看规则详细信息并选择 Create rule（创建规则）。

步骤 4：发送测试事件

现在，您已经设置了 SNS 主题和规则，我们将发送测试事件，以确保规则可正常运行。

发送测试事件（控制台）

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Event Buses (事件总线)。
3. 在默认事件总线图块中，选择操作、发送事件。
4. 输入事件源。例如，`TestEvent`。
5. 在详细信息类型中，输入 `customerCreated`。
6. 在事件详细信息中，输入 `{}`。
7. 选择 Send (发送)。

步骤 5：确认成功

如果您从 AWS 通知收到一封与预期输出相匹配的电子邮件，则说明您已成功完成本教程。

步骤 6：清除资源

除非您想要保留为本教程创建的资源，否则可立即将其删除。请删除您不再使用的 AWS 资源，这样可防止您的 AWS 账户产生不必要的费用。

删除 SNS 主题

1. 在 SNS 控制台中打开主题页面。
2. 选择您创建的主题。
3. 选择 Delete (删除)。
4. 输入 **`delete me`**。
5. 选择 Delete (删除)。

删除 SNS 订阅

1. 在 SNS 控制台中打开订阅页面。
2. 选择您创建的订阅。
3. 选择 Delete。

4. 选择 Delete。

删除 EventBridge 规则

1. 在 EventBridge 控制台中打开规则页面。
2. 选择您创建的规则。
3. 选择 Delete。
4. 选择 Delete。

Amazon EventBridge 与其他 AWS 服务集成的教程

Amazon EventBridge 与其他 AWS 服务配合使用，可处理事件或调用 AWS 资源，作为规则的目标。以下教程介绍如何将 EventBridge 与其他 AWS 服务集成。

教程:

- 教程：使用 EventBridge 记录自动扩缩组的状态
- 教程：使用 EventBridge 记录 AWS API 调用
- 教程：使用 EventBridge 记录 Amazon EC2 实例的状态
- 教程：使用 EventBridge 记录 Amazon S3 对象级别操作
- 教程：使用 EventBridge 和 aws.events 架构将事件发送到 Amazon Kinesis 流
- 教程：使用 EventBridge 安排自动化 Amazon EBS 快照
- 教程：在创建 Amazon S3 对象时发送通知
- 教程：使用 EventBridge 安排 AWS Lambda 函数

教程：使用 EventBridge 记录自动扩缩组的状态

可以运行 AWS Lambda 函数，只要自动扩缩组启动或终止一个 Amazon EC2 实例，此函数就会记录一个事件，指示事件是否成功。

有关使用 Amazon EC2 Auto Scaling 事件的更多场景信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的使用 EventBridge 处理自动扩缩事件。

在本教程中，您将创建一个 Lambda 函数，然后在 EventBridge 控制台中创建一条规则，当 Amazon EC2 Auto Scaling 组启动或终止一个实例时调用该函数。

步骤：

先决条件

完成本教程需要以下资源：

- 一个自动扩缩组。有关创建自动扩缩组的更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的使用启动配置创建自动扩缩组。

步骤 1：创建 Lambda 函数

创建一个 Lambda 函数，以记录您的 Auto Scaling 组的扩展和缩减事件。

创建 Lambda 函数

1. 打开 AWS Lambda 控制台，地址：https://console.aws.amazon.com/lambda/。
2. 选择创建函数。
3. 选择 Author from scratch (从头开始创作)。
4. 输入 Lambda 函数的名称。例如，将函数命名为 `LogAutoScalingEvent`。
5. 将其余选项保留为默认值，然后选择创建函数。

6. 在函数页面的代码选项卡上，双击 index.js。
7. 使用以下代码替换现有代码。

```
'use strict';

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

8. 选择 Deploy (部署)。

步骤 2：创建规则

创建规则，运行您在步骤 1 中创建的 Lambda 函数。在您的自动扩缩组启动或停止实例时，该规则会运行。

创建 规则

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Rules (规则)。
3. 选择 Create rule (创建规则)。
4. 为规则输入名称和描述。例如，将规则命名为 `TestRule`
5. 对于 Event bus（事件总线），请选择要与此规则关联的事件总线。如果您希望此规则匹配来自您账户的事件，请选择默认。当您账户中的某个 AWS 服务发出一个事件时，它始终会发送到您账户的默认事件总线。
6. 对于 Rule type（规则类型），选择 Rule with an event pattern（具有事件模式的规则）。
7. 选择 Next（下一步）。
8. 对于 Event source（事件源），选择 AWS services（服务）。
9. 对于 Event pattern（事件模式），执行以下操作：
   a. 对于事件源，从下拉列表中选择自动扩缩。
   b. 对于事件类型，从下拉列表中选择实例启动和终止。
   c. 选择任何实例事件和任何组名。
10. 选择 Next（下一步）。

11. 对于 Target types（目标类型），选择 AWS service（服务）。
12. 对于选择目标，从下拉列表中选择 Lambda 函数。
13. 在函数中，选择您在步骤 1：创建 Lambda 函数部分创建的 Lambda 函数。在此示例中，选择 `LogAutoScalingEvent`。
14. 选择 Next（下一步）。
15. 选择 Next（下一步）。
16. 查看规则详细信息并选择 Create rule（创建规则）。

步骤 3：测试规则

您可以通过手动扩展 Auto Scaling 组来测试您的规则，以便其启动一个实例。等待几分钟，在扩展事件发生后，验证是否已调用您的 Lambda 函数。

使用 Auto Scaling 组测试您的规则

1. 要增加您的 Auto Scaling 组的大小，请执行以下操作：
   a. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
   b. 在导航窗格上，依次选择 Auto Scaling 和 Auto Scaling Groups (Auto Scaling 组)。
   c. 选中您的 Auto Scaling 组对应的复选框。
   d. 在 Details 选项卡上，选择 Edit。对于 Desired，将所需容量增加一。例如，如果当前值是 2，请输入 3。理想容量必须小于或等于组的最大容量。如果您的 Desired 新值大于 Max，则必须更新 Max。完成后，选择 Save。
2. 要查看 Lambda 函数的输出，请执行以下操作：
   a. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
   b. 在导航窗格中，选择日志。
   c. 选择您的 Lambda 函数 (`/aws/lambda/`*`function-name`*) 的日志组的名称。
   d. 选择日志流的名称，以查看您启动的实例的函数提供的数据。
3. （可选）完成后，您可以将所需的容量减一，这样自动扩缩组就会返回到之前的大小。

步骤 4：确认成功

如果您在 CloudWatch 日志中看到 Lambda 事件，说明您已成功完成本教程。如果您的 CloudWatch 日志中没有该事件，请开始故障排除，首先验证规则是否成功创建，如果规则看起来正确，再验证 Lambda 函数的代码是否正确。

步骤 5：清理资源

除非您想要保留为本教程创建的资源，否则可立即将其删除。请删除您不再使用的 AWS 资源，这样可防止您的 AWS 账户产生不必要的费用。

删除 EventBridge 规则

1. 在 EventBridge 控制台中打开规则页面。
2. 选择您创建的规则。
3. 选择 Delete。
4. 选择 Delete。

删除 Lambda 函数

1. 打开 Lamba 控制台的函数页面。
2. 选择您创建的函数。
3. 依次选择 Actions 和 Delete。
4. 选择 Delete。

教程：使用 EventBridge 记录 AWS API 调用

您可以使用 Amazon EventBridge 规则对 AWS 服务进行的 API 调用做出反应，这些服务由 AWS CloudTrail 记录。

在本教程中，您将在 EventBridge 控制台中创建 AWS CloudTrail 跟踪、Lambda 函数和规则。当一个 Amazon EC2 实例停止时，该规则会调用 Lambda 函数。

步骤：

- 步骤 1：创建 AWS CloudTrail 跟踪
- 步骤 2：创建 AWS Lambda 函数
- 步骤 3：创建规则
- 步骤 4：测试规则
- 步骤 5：确认成功
- 步骤 6：清除资源

步骤 1：创建 AWS CloudTrail 跟踪

如果您已设置了跟踪，请跳转至步骤 2。

创建跟踪

1. 访问 https://console.aws.amazon.com/cloudtrail/，打开 CloudTrail 控制台。
2. 依次选择 Trails (跟踪)、Create trail (创建跟踪)。
3. 对于 Trail name，键入跟踪的名称。
4. 在存储位置中，选择创建新的 S3 桶。
5. 在 AWS KMS 别名中，为 KMS 密钥键入别名。
6. 选择 Next（下一步）。
7. 选择 Next（下一步）。
8. 选择 Create trail (创建跟踪)。

步骤 2：创建 AWS Lambda 函数

创建 Lambda 函数以记录 API 调用事件。

创建 Lambda 函数

1. 打开 AWS Lambda 控制台，地址：https://console.aws.amazon.com/lambda/。
2. 选择创建函数。
3. 选择 Author from scratch (从头开始创作)。
4. 输入 Lambda 函数的名称和说明。例如，将函数命名为 `LogEC2StopInstance`。
5. 将其余选项保留为默认值，然后选择创建函数。
6. 在函数页面的代码选项卡上，双击 index.js。
7. 使用以下代码替换现有代码。

```
'use strict';

exports.handler = (event, context, callback) => {
    console.log('LogEC2StopInstance');
    console.log('Received event:', JSON.stringify(event, null, 2));
    callback(null, 'Finished');
};
```

8. 选择 Deploy（部署）。

步骤 3：创建规则

创建一条规则，在停止 Amazon EC2 实例时运行您在步骤 2 中创建的 Lambda 函数。

创建 规则

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Rules (规则)。
3. 选择 Create rule (创建规则)。
4. 为规则输入名称和描述。例如，将规则命名为 `TestRule`
5. 对于 Event bus（事件总线），请选择要与此规则关联的事件总线。如果您希望此规则匹配来自您账户的事件，请选择默认。当您账户中的某个 AWS 服务发出一个事件时，它始终会发送到您账户的默认事件总线。
6. 对于 Rule type（规则类型），选择 Rule with an event pattern（具有事件模式的规则）。
7. 选择 Next（下一步）。
8. 对于 Event source（事件源），选择 AWS services（服务）。

9. 对于 Event pattern（事件模式），执行以下操作：
   a. 对于事件源，请从下拉列表中选择 EC2。
   b. 对于事件类型，请从下拉列表中选择通过 CloudTrail 进行的 AWS API 调用。
   c. 选择特定操作并输入 `StopInstances`。
10. 选择 Next（下一步）。
11. 对于 Target types（目标类型），选择 AWS service（服务）。
12. 对于选择目标，从下拉列表中选择 Lambda 函数。
13. 在函数中，选择您在步骤 1：创建 Lambda 函数部分创建的 Lambda 函数。在此示例中，选择 `LogEC2StopInstance`。
14. 选择 Next（下一步）。
15. 选择 Next（下一步）。
16. 查看规则详细信息并选择 Create rule（创建规则）。

步骤 4：测试规则

可以使用 Amazon EC2 控制台停止 Amazon EC2 实例来测试您的规则。等待几分钟以便实例停止，然后检查 CloudWatch 控制台中的 AWS Lambda 指标，验证您的函数是否运行。

通过停止一个实例来测试您的规则

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 启动一个实例。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的启动您的实例。
3. 停止实例。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的停止和启动实例。
4. 要查看 Lambda 函数的输出，请执行以下操作：
   a. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
   b. 在导航窗格中，选择日志。
   c. 选择您的 Lambda 函数 (`/aws/lambda/`*`function-name`*) 的日志组的名称。
   d. 选择日志流的名称，以查看您停止的实例的函数提供的数据。
5. （可选）当您完成后，终止已停止的实例。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的终止实例。

步骤 5：确认成功

如果您在 CloudWatch 日志中看到 Lambda 事件，说明您已成功完成本教程。如果您的 CloudWatch 日志中没有该事件，请开始故障排除，首先验证规则是否成功创建，如果规则看起来正确，再验证 Lambda 函数的代码是否正确。

步骤 6：清除资源

除非您想要保留为本教程创建的资源，否则可立即将其删除。请删除您不再使用的 AWS 资源，这样可防止您的 AWS 账户产生不必要的费用。

删除 EventBridge 规则

1. 在 EventBridge 控制台中打开规则页面。
2. 选择您创建的规则。
3. 选择 Delete。
4. 选择 Delete。

删除 Lambda 函数

1. 打开 Lamba 控制台的函数页面。
2. 选择您创建的函数。
3. 依次选择 Actions 和 Delete。
4. 选择 Delete (删除)。

删除 CloudTrail 跟踪

1. 打开 CloudTrail 控制台的 Trails（跟踪记录）页面。
2. 选择您创建的跟踪。
3. 选择 Delete。
4. 选择 Delete。

教程：使用 EventBridge 记录 Amazon EC2 实例的状态

您可以创建 AWS Lambda 函数来记录 Amazon EC2 实例的状态更改。您可以创建一条规则，以便在状态发生任何转换时或者在状态转换为一个或多个相关状态时运行您的 Lambda 函数。在此教程中，您将记录任何新实例的启动。

步骤：

步骤 1：创建 AWS Lambda 函数

创建 Lambda 函数以记录状态更改事件。在步骤 2 中创建规则时，您可以指定此函数。

创建 Lambda 函数

1. 打开 AWS Lambda 控制台，地址：https://console.aws.amazon.com/lambda/。
2. 选择创建函数。
3. 选择 Author from scratch (从头开始创作)。
4. 输入 Lambda 函数的名称和说明。例如，将函数命名为 `LogEC2InstanceStateChange`。
5. 将其余选项保留为默认值，然后选择创建函数。
6. 在函数页面的代码选项卡上，双击 index.js。
7. 使用以下代码替换现有代码。

```
'use strict';

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

8. 选择 Deploy (部署)。

步骤 2：创建规则

创建规则，运行您在步骤 1 中创建的 Lambda 函数。该规则将在您启动 Amazon EC2 实例时运行。

创建 EventBridge 规则

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Rules (规则)。
3. 选择 Create rule (创建规则)。
4. 为规则输入名称和描述。例如，将规则命名为 `TestRule`
5. 对于 Event bus（事件总线），请选择要与此规则关联的事件总线。如果您希望此规则匹配来自您账户的事件，请选择默认。当您账户中的某个 AWS 服务发出一个事件时，它始终会发送到您账户的默认事件总线。
6. 对于 Rule type（规则类型），选择 Rule with an event pattern（具有事件模式的规则）。
7. 选择 Next（下一步）。
8. 对于 Event source（事件源），选择 AWS services（服务）。
9. 对于 Event pattern（事件模式），执行以下操作：
   a. 对于事件源，请从下拉列表中选择 EC2。
   b. 对于事件类型，请从下拉列表中选择 EC2 实例状态更改通知。
   c. 选择特定状态，然后从下拉列表中选择运行。
   d. 选择任何实例
10. 选择 Next（下一步）。
11. 对于 Target types（目标类型），选择 AWS service（服务）。
12. 对于选择目标，从下拉列表中选择 Lambda 函数。
13. 在函数中，选择您在步骤 1：创建 Lambda 函数部分创建的 Lambda 函数。在此示例中，选择 `LogEC2InstanceStateChange`。
14. 选择 Next（下一步）。
15. 选择 Next（下一步）。
16. 查看规则详细信息并选择 Create rule（创建规则）。

步骤 3：测试规则

可以使用 Amazon EC2 控制台停止 Amazon EC2 实例来测试您的规则。等待几分钟以便实例停止，然后检查 CloudWatch 控制台中的 AWS Lambda 指标，验证您的函数是否运行。

通过停止一个实例来测试您的规则

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 启动一个实例。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的启动您的实例。
3. 停止实例。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的停止和启动实例。
4. 要查看 Lambda 函数的输出，请执行以下操作：
   a. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
   b. 在导航窗格中，选择日志。
   c. 选择您的 Lambda 函数 (`/aws/lambda/function-name`) 的日志组的名称。
   d. 选择日志流的名称，以查看您停止的实例的函数提供的数据。
5. （可选）当您完成后，终止已停止的实例。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的终止实例。

步骤 4：确认成功

如果您在 CloudWatch 日志中看到 Lambda 事件，说明您已成功完成本教程。如果您的 CloudWatch 日志中没有该事件，请开始故障排除，首先验证规则是否成功创建，如果规则看起来正确，再验证 Lambda 函数的代码是否正确。

步骤 5：清理资源

除非您想要保留为本教程创建的资源，否则可立即将其删除。请删除您不再使用的 AWS 资源，这样可防止您的 AWS 账户产生不必要的费用。

删除 EventBridge 规则

1. 在 EventBridge 控制台中打开规则页面。
2. 选择您创建的规则。
3. 选择 Delete。

4. 选择 Delete。

删除 Lambda 函数

1. 打开 Lamba 控制台的函数页面。
2. 选择您创建的函数。
3. 依次选择 Actions 和 Delete。
4. 选择 Delete。

教程：使用 EventBridge 记录 Amazon S3 对象级别操作

您可以在 Amazon S3 桶中记录对象级别 API 操作。必须使用 AWS CloudTrail 设置跟踪并配置为接收这些事件，Amazon EventBridge 才能匹配这些事件。

在本教程中，您将创建 CloudTrail 跟踪、创建 AWS Lambda 函数，然后在 EventBridge 控制台中创建规则，用于调用该函数以响应 S3 数据事件。

步骤：

- 步骤 1：配置您的 AWS CloudTrail 跟踪
- 步骤 2：创建 AWS Lambda 函数
- 步骤 3：创建 规则
- 步骤 4：测试 规则
- 步骤 5：确认成功
- 步骤 6：清除资源

步骤 1：配置您的 AWS CloudTrail 跟踪

要在 AWS CloudTrail 和 EventBridge 的 S3 桶中记录数据事件，首先要创建一个跟踪。跟踪 会捕获您账户中的 API 调用和相关事件，并将日志文件传输到您指定的 S3 桶。您可以更新现有跟踪或创建一个新跟踪。

有关更多信息，请参阅《AWS CloudTrail 用户指南》中的数据事件。

创建跟踪

1. 访问 https://console.aws.amazon.com/cloudtrail/，打开 CloudTrail 控制台。
2. 依次选择 Trails (跟踪)、Create trail (创建跟踪)。
3. 对于 Trail name，键入跟踪的名称。
4. 在存储位置中，选择创建新的 S3 桶。
5. 在 AWS KMS 别名中，为 KMS 密钥键入别名。
6. 选择 Next（下一步）。
7. 对于事件类型，选择数据事件。
8. 对于数据事件，执行以下操作之一：

- 要记录存储桶中所有 Amazon S3 对象的数据事件，请指定一个 S3 存储桶和一个空前缀。当事件在该存储桶中的对象上发生时，跟踪将处理和记录事件。
- 要记录桶中特定 Amazon S3 对象的数据事件，请指定 S3 桶和对象前缀。当事件在该 存储桶中的对象上发生且对象以指定前缀开头时，跟踪将处理和记录事件。

9. 对于每个资源，选择是要记录读取事件、写入事件，还是同时记录这两类事件。
10. 选择 Next（下一步）。
11. 选择 Create trail (创建跟踪)。

步骤 2：创建 AWS Lambda 函数

创建一个 Lambda 函数，以记录 S3 存储桶的数据事件。

创建 Lambda 函数

1. 打开 AWS Lambda 控制台，地址：https://console.aws.amazon.com/lambda/。
2. 选择创建函数。
3. 选择 Author from scratch (从头开始创作)。
4. 输入 Lambda 函数的名称和说明。例如，将函数命名为 `LogS3DataEvents`。
5. 将其余选项保留为默认值，然后选择创建函数。
6. 在函数页面的代码选项卡上，双击 index.js。
7. 使用以下代码替换现有代码。

```
'use strict';

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

8. 选择 Deploy（部署）。

步骤 3：创建 规则

创建规则，运行您在步骤 2 中创建的 Lambda 函数。此规则运行是为了响应 Amazon S3 数据事件。

创建 规则

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Rules (规则)。
3. 选择 Create rule (创建规则)。
4. 为规则输入名称和描述。例如，将规则命名为 `TestRule`
5. 对于 Event bus（事件总线），请选择要与此规则关联的事件总线。如果您希望此规则匹配来自您账户的事件，请选择默认。当您账户中的某个 AWS 服务发出一个事件时，它始终会发送到您账户的默认事件总线。
6. 对于 Rule type（规则类型），选择 Rule with an event pattern（具有事件模式的规则）。
7. 选择 Next（下一步）。
8. 对于 Event source（事件源），选择 AWS services（服务）。
9. 对于 Event pattern（事件模式），执行以下操作：
   a. 对于事件源，从下拉列表中选择 Simple Storage Service (S3)。
   b. 对于事件类型，从下拉列表中选择通过 CloudTrail 进行的对象级 API 调用。
   c. 选择 Specific operation(s) (特定操作)，然后选择 PutObject。
   d. 默认情况下，该规则与区域中所有存储桶的数据事件匹配。要匹配特定存储桶的数据事件，请选择 Specify bucket(s) by name (按名称匹配特定存储桶)，然后指定一个或多个存储桶。
10. 选择 Next（下一步）。
11. 对于 Target types（目标类型），选择 AWS service（服务）。
12. 对于选择目标，从下拉列表中选择 Lambda 函数。
13. 对于函数，选择您在步骤 1 中创建的 Lambda 函数。
14. 选择 Next（下一步）。
15. 选择 Next（下一步）。
16. 查看规则详细信息并选择 Create rule（创建规则）。

步骤 4：测试 规则

为了测试规则，将一个对象置于 S3 存储桶中。您可以验证您的 Lambda 函数是否已调用。

要查看您的 Lambda 函数的日志

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。

2. 在导航窗格中，选择日志。
3. 选择您的 Lambda 函数 (`/aws/lambda/function-name`) 的日志组的名称。
4. 选择日志流的名称，以查看您启动的实例的函数提供的数据。

您还可以在 S3 桶中检查您为跟踪指定的 CloudTrail 日志。有关更多信息，请参阅 AWS CloudTrail 用户指南中的获取和查看 CloudTrail 日志文件。

步骤 5：确认成功

如果您在 CloudWatch 日志中看到 Lambda 事件，说明您已成功完成本教程。如果您的 CloudWatch 日志中没有该事件，请开始故障排除，首先验证规则是否成功创建，如果规则看起来正确，再验证 Lambda 函数的代码是否正确。

步骤 6：清除资源

除非您想要保留为本教程创建的资源，否则可立即将其删除。请删除您不再使用的 AWS 资源，这样可防止您的 AWS 账户产生不必要的费用。

删除 EventBridge 规则

1. 在 EventBridge 控制台中打开规则页面。
2. 选择您创建的规则。
3. 选择 Delete。
4. 选择 Delete。

删除 Lambda 函数

1. 打开 Lamba 控制台的函数页面。
2. 选择您创建的函数。
3. 依次选择 Actions 和 Delete。
4. 选择 Delete (删除)。

删除 CloudTrail 跟踪

1. 打开 CloudTrail 控制台的 Trails（跟踪记录）页面。
2. 选择您创建的跟踪。

3. 选择 Delete。
4. 选择 Delete。

教程：使用 EventBridge 和 `aws.events` 架构将事件发送到 Amazon Kinesis 流

您可以将 EventBridge 中的 AWS API 调用事件发送到 Amazon Kinesis 流、创建 Kinesis Data Streams 应用程序并处理大量数据。在本教程中，您将创建一个 Kinesis 流，然后在 EventBridge 控制台中创建一条规则，在 Amazon EC2 实例停止时向该流发送事件。

步骤：

先决条件

在本教程中，您将使用以下工具：

- 使用 AWS CLI 来处理 Kinesis 流。

  要安装 AWS CLI，请参阅安装、更新和卸载 AWS CLI 版本 2。

> **Note**
>
> 本教程使用 AWS 事件和内置 `aws.events` 架构注册表。您也可以根据自定义事件的架构创建 EventBridge 规则，方法是手动将事件添加到自定义架构注册表中，或者使用架构发现。有关架构的更多信息，请参阅 ???。有关使用其他事件模式选项创建规则的更多信息，请参阅 ???。

步骤 1：创建 Amazon Kinesis 流

要创建流，请在命令提示符下使用 `create-stream` AWS CLI 命令。

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

当流状态为 ACTIVE 时，表示流已就绪。要检查流状态，请使用 describe-stream 命令。

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

步骤 2：创建规则

创建一条规则，在您停止 Amazon EC2 实例时将事件发送到您的流。

创建 规则

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Rules (规则)。
3. 选择 Create rule (创建规则)。
4. 为规则输入名称和描述。例如，将规则命名为 TestRule
5. 对于事件总线，选择默认。
6. 对于 Rule type（规则类型），选择 Rule with an event pattern（具有事件模式的规则）。
7. 选择 Next（下一步）。
8. 对于 Event source（事件源），选择 AWS 事件或 EventBridge 合作伙伴事件。
9. 对于创建方法，选择使用架构。
10. 对于 Event pattern（事件模式），执行以下操作：

    a. 对于架构类型，选择从架构注册表中选择架构。

    b. 对于架构注册表，请从下拉列表中选择 aws.events。

    c. 对于架构，请从下拉列表中选择 aws.ec2@EC2InstanceStateChangeNotification。

       EventBridge 在模型下显示事件架构。

       EventBridge 在事件（而不是事件模式）必需的所有属性旁显示一个红色星号。

    d. 在模型中，设置以下事件筛选属性：

       i. 选择 state 属性旁的 + 编辑。

          将关系留空。对于 Value（值），输入 running。选择设置。

       ii. 选择 source 属性旁的 + 编辑。

          将关系留空。对于 Value（值），输入 aws.ec2。选择设置。

       iii. 选择 detail-type 属性旁的 + 编辑。

将关系留空。对于 Value（值），输入 `EC2 Instance State-change Notification`。选择设置。

e. 要查看您构造的事件模式，请选择以 JSON 格式生成事件模式

EventBridge 以 JSON 格式显示事件模式：

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

11. 选择 Next（下一步）。
12. 对于 Target types（目标类型），选择 AWS service（服务）。
13. 对于选择目标，从下拉列表中选择 Kinesis 流。
14. 对于流，请选择您在步骤 1：创建 Amazon Kinesis 流部分创建的 Kinesis 流。在此示例中，选择 `test`。
15. 对于执行角色，选择为此特定资源创建新角色。
16. 选择 Next（下一步）。
17. 选择 Next（下一步）。
18. 查看规则详细信息并选择 Create rule（创建规则）。

步骤 3：测试规则

为了测试规则，停用一个 Amazon EC2 实例。等待几分钟以便实例停止，然后检查 CloudWatch 指标，验证您的函数是否运行。

通过停止一个实例来测试您的规则

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 启动一个实例。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的启动您的实例。
3. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
4. 在导航窗格中，选择 Rules (规则)。

选择所创建规则的名称，然后选择 Metrics for the rule (规则的指标)。

5. （可选）当您完成后，终止该实例。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的终止实例。

步骤 4：验证事件是否已发送

您可以使用 AWS CLI 从流中获取记录，验证事件是否已发送。

获取记录

1. 要开始从 Kinesis 流中读取，请在命令提示符下使用 `get-shard-iterator` 命令。

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

下面是示例输出。

```
{
    "ShardIterator": "AAAAAAAAAAHSywljv0zEgPX4NyKdZ5wryMzP9yALs8NeKbUjp1IxtZs1Sp
+KEd9I6AJ9ZG4lNR1EMi+9Md/nHvtLyxpfhEzYvkTZ4D9DQVz/mBYWRO6OTZRKnW9gd
+efGN2aHFdkH1rJl4BL9Wyrk+ghYG22D2T1Da2EyNSH1+LAbK33gQweTJADBdyMwlo5r6PqcP2dzhg="
}
```

2. 要获取记录，请使用以下 `get-records` 命令。使用上一步输出中的分片迭代器。

```
aws kinesis get-records --shard-
iterator AAAAAAAAAAHSywljv0zEgPX4NyKdZ5wryMzP9yALs8NeKbUjp1IxtZs1Sp
+KEd9I6AJ9ZG4lNR1EMi+9Md/nHvtLyxpfhEzYvkTZ4D9DQVz/mBYWRO6OTZRKnW9gd
+efGN2aHFdkH1rJl4BL9Wyrk+ghYG22D2T1Da2EyNSH1+LAbK33gQweTJADBdyMwlo5r6PqcP2dzhg=
```

如果命令成功，它将从指定分片的流中请求记录。您可能会收到零个或多个记录。返回的任何记录都不能表示流中的所有记录。如果您未收到预期的数据，请继续调用 `get-records`。

3. Kinesis 中的记录是以 Base64 编码的。使用 Base64 解码器对数据进行解码，这样您就可以验证这是不是以 JSON 格式发送到流的事件。

步骤 5：清理资源

除非您想要保留为本教程创建的资源，否则可立即将其删除。请删除您不再使用的 AWS 资源，这样可防止您的 AWS 账户产生不必要的费用。

删除 EventBridge 规则

1. 在 EventBridge 控制台中打开规则页面。
2. 选择您创建的规则。
3. 选择 Delete。
4. 选择 Delete。

删除 Kinesis 流

1. 在 Kinesis 控制台中打开数据流页面。
2. 选择您创建的流。
3. 依次选择操作和删除。
4. 在字段中输入删除，并选择删除。

教程：使用 EventBridge 安排自动化 Amazon EBS 快照

您可以按计划运行 EventBridge 规则。在此教程中，您按照计划为现有 Amazon Elastic Block Store (Amazon EBS) 卷创建快照。您可以选择一个固定速度，每隔几分钟创建一个快照；或者使用 cron 表达式，在每天的特定时间创建快照。

> **Important**
>
> 您必须使用 AWS Management Console 来创建具有内置目标的规则。

步骤：

步骤 1：创建规则

创建按照计划拍摄快照的规则。可以使用 rate 表达式或 Cron 表达式来指定计划。有关更多信息，请参阅创建按计划运行的 Amazon EventBridge 规则。

创建 规则

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Rules (规则)。
3. 选择 Create rule (创建规则)。
4. 为规则输入名称和描述。

   规则不能与同一区域中的另一个规则和同一事件总线上的名称相同。
5. 对于 Event bus（事件总线），请选择要与此规则关联的事件总线。如果您希望此规则对来自您自己的账户的匹配事件触发，请选择 AWS 原定设置事件总线。当您账户中的某个 AWS 服务发出一个事件时，它始终会发送到您账户的默认事件总线。
6. 对于 Rule type（规则类型），选择 Schedule（计划）。
7. 选择 Next（下一步）。

8. 在计划模式中，选择以固定频率运行的计划，例如每 10 分钟运行一次，然后输入 **5** 并从下拉列表中选择分钟。
9. 选择 Next（下一步）。
10. 对于 Target types（目标类型），选择 AWS service（服务）。
11. 对于选择目标，从下拉列表中选择 EBS 创建快照。
12. 对于卷 ID，输入 Amazon EBS 卷的卷 ID。
13. 对于执行角色，选择为此特定资源创建新角色。
14. 选择 Next（下一步）。
15. 选择 Next（下一步）。
16. 查看规则详细信息并选择 Create rule（创建规则）。

步骤 2：测试规则

在生成第一个快照后，您可以查看这个快照来验证您的规则是否运行正常。

测试您的规则

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 在导航窗格中，依次选择 Elastic Block Store 和 Snapshots。
3. 验证第一张快照是否在列表中显示。

步骤 3：确认成功

如果您在列表中看到快照，则说明您已成功完成本教程。如果快照不在列表中，请开始故障排除，验证规则是否成功创建。

步骤 4：清理资源

除非您想要保留为本教程创建的资源，否则可立即将其删除。请删除您不再使用的 AWS 资源，这样可防止您的 AWS 账户产生不必要的费用。

删除 EventBridge 规则

1. 在 EventBridge 控制台中打开规则页面。
2. 选择您创建的规则。
3. 选择 Delete。

4. 选择 Delete。

教程：在创建 Amazon S3 对象时发送通知

使用 Amazon EventBridge 和 Amazon SNS 创建 Amazon Simple Storage Service (Amazon S3) 对象后，您可以发送电子邮件通知。在本教程中，您将创建 SNS 主题和订阅。然后，您将在 EventBridge 控制台中创建一条规则，在收到 Amazon S3 `Object Created` 事件时向该主题发送事件。

步骤：

先决条件

要在 EventBridge 中接收 Amazon S3 事件，您必须在 Amazon S3 控制台中启用 EventBridge。本教程假设 EventBridge 已启用。有关更多信息，请参阅在 S3 控制台中启用 Amazon EventBridge

步骤 1：创建一个 Amazon SNS 主题

创建一个主题，接收来自 EventBridge 的事件。

要创建主题，请执行以下操作

1. 通过以下网址打开 Amazon SNS 控制台：https://console.aws.amazon.com/sns/v3/home。
2. 在导航窗格中，选择 Topics（主题）。
3. 选择 Create topic（创建主题）。
4. 对于 Type（类型），选择 Standard（标准）。
5. 输入 **`eventbridge-test`** 作为主题名称。
6. 选择 Create topic（创建主题）。

步骤 2：创建 Amazon SNS 订阅

创建订阅，以便在主题收到事件时接收来自 Amazon S3 的电子邮件通知。

创建订阅

1. 通过以下网址打开 Amazon SNS 控制台：https://console.aws.amazon.com/sns/v3/home。
2. 在导航窗格中，选择 Subscriptions。
3. 选择 Create subscription。
4. 对于主题 ARN，选择您在步骤 1 中创建的主题。在本教程中，选择 eventbridge-test。
5. 对于协议，选择电子邮件。
6. 对于 Endpoint (终端节点)，输入您的电子邮件地址。
7. 选择 Create subscription（创建订阅）。
8. 在收到的 AWS 通知电子邮件中选择确认订阅，以确认订阅。

步骤 3：创建规则

创建一条规则，在创建 Amazon S3 对象后将事件发送到您的主题。

创建 规则

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Rules (规则)。
3. 选择 Create rule (创建规则)。
4. 为规则输入名称和描述。例如，将规则命名为 `s3-test`
5. 对于事件总线，选择默认。
6. 对于 Rule type（规则类型），选择 Rule with an event pattern（具有事件模式的规则）。
7. 选择 Next（下一步）。
8. 对于 Event source（事件源），选择 AWS 事件或 EventBridge 合作伙伴事件。
9. 对于创建方法，选择使用模式表单。
10. 对于 Event pattern（事件模式），执行以下操作：
    a. 对于事件源，从下拉列表中选择 AWS 服务。
    b. 对于 AWS 服务，从下拉列表中选择 Simple Storage Service (S3)。
    c. 对于事件类型，从下拉列表中选择 Amazon S3 事件通知。
    d. 选择特定事件，然后从下拉列表中选择创建对象。
    e. 选择任意存储桶

11. 选择 Next（下一步）。
12. 对于 Target types（目标类型），选择 AWS service（服务）。
13. 对于选择目标，从下拉列表中选择 SNS 主题。
14. 对于主题，选择您在步骤 1：创建 SNS 主题部分创建的 Amazon SNS 主题。在此示例中，选择 `eventbridge-test`。
15. 选择 Next（下一步）。
16. 选择 Next（下一步）。
17. 查看规则详细信息并选择 Create rule（创建规则）。

步骤 4：测试规则

要测试您的规则，请将文件上传到支持 EventBridge 的桶，创建一个 Amazon S3 对象。然后，等待几分钟，并验证您是否收到 AWS 通知电子邮件。

步骤 5：清理资源

除非您想要保留为本教程创建的资源，否则可立即将其删除。请删除您不再使用的 AWS 资源，这样可防止您的 AWS 账户产生不必要的费用。

删除 SNS 主题

1. 在 SNS 控制台中打开主题页面。
2. 选择您创建的主题。
3. 选择 Delete (删除)。
4. 输入 **`delete me`**。
5. 选择 Delete (删除)。

删除 SNS 订阅

1. 在 SNS 控制台中打开订阅页面。
2. 选择您创建的订阅。
3. 选择 Delete。
4. 选择 Delete。

删除 EventBridge 规则

1. 在 EventBridge 控制台中打开规则页面。
2. 选择您创建的规则。
3. 选择 Delete。
4. 选择 Delete。

教程：使用 EventBridge 安排 AWS Lambda 函数

您可以设置规则，按计划运行 AWS Lambda 函数。本教程演示如何使用 AWS Management Console 或 AWS CLI 创建规则。如果要使用 AWS CLI 但尚未安装，请参阅安装、更新和卸载 AWS CLI 版本 2。

对于计划，EventBridge 不在计划表达式中提供第二级精度。使用 cron 表达式的最高解析精度是一分钟。由于 EventBridge 和目标服务的分布式特性，触发计划规则的时间与目标服务运行目标资源的时间之间，有几秒钟的延迟。

步骤：

- 步骤 1：创建 Lambda 函数
- 步骤 2：创建规则
- 步骤 3：验证规则
- 步骤 4：确认成功
- 步骤 5：清理资源

步骤 1：创建 Lambda 函数

创建 Lambda 函数来记录计划的事件。

创建 Lambda 函数

1. 打开 AWS Lambda 控制台，地址：https://console.aws.amazon.com/lambda/。
2. 选择创建函数。
3. 选择 Author from scratch (从头开始创作)。
4. 输入 Lambda 函数的名称和说明。例如，将函数命名为 `LogScheduledEvent`。
5. 将其余选项保留为默认值，然后选择创建函数。
6. 在函数页面的代码选项卡上，双击 index.js。
7. 使用以下代码替换现有代码。

```
'use strict';

exports.handler = (event, context, callback) => {
    console.log('LogScheduledEvent');
    console.log('Received event:', JSON.stringify(event, null, 2));
```

```
    callback(null, 'Finished');
};
```

8. 选择 Deploy (部署)。

步骤 2：创建规则

创建规则，按计划运行在步骤 1 中创建的 Lambda 函数。

您可以使用控制台或 AWS CLI 来创建规则。要使用 AWS CLI，您首先要向该规则授予调用 Lambda 函数的权限。然后，您可以创建规则并将该 Lambda 函数添加为目标。

创建规则（控制台）

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Rules (规则)。
3. 选择 Create rule (创建规则)。
4. 为规则输入名称和描述。

   规则不能与同一区域中的另一个规则和同一事件总线上的名称相同。
5. 对于 Event bus（事件总线），请选择要与此规则关联的事件总线。如果您希望此规则对来自您自己的账户的匹配事件触发，请选择 AWS 原定设置事件总线。当您账户中的某个 AWS 服务发出一个事件时，它始终会发送到您账户的默认事件总线。
6. 对于 Rule type（规则类型），选择 Schedule（计划）。
7. 选择 Next（下一步）。
8. 在计划模式中，选择以固定频率运行的计划，例如每 10 分钟运行一次，然后输入 **5** 并从下拉列表中选择分钟。
9. 选择 Next（下一步）。
10. 对于 Target types（目标类型），选择 AWS service（服务）。
11. 对于选择目标，从下拉列表中选择 Lambda 函数。
12. 在函数中，选择您在步骤 1：创建 Lambda 函数部分创建的 Lambda 函数。在此示例中，选择 `LogScheduledEvent`。
13. 选择 Next（下一步）。
14. 选择 Next（下一步）。
15. 查看规则详细信息并选择 Create rule（创建规则）。

创建规则 (AWS CLI)

1. 要创建按计划运行的规则，请使用 `put-rule` 命令。

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

此规则运行时，会创建一个事件，然后将其发送到目标。以下是示例事件。

```
{
    "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. 要授予 EventBridge 服务主体 (`events.amazonaws.com`) 权限以运行规则，请使用 `add-permission` 命令。

```
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. 创建文件 `targets.json` 并输入以下内容。

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

```
]
```

4. 要将您在步骤 1 中创建的 Lambda 函数添加到规则中，请使用 `put-targets` 命令。

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

步骤 3：验证规则

完成步骤 2 后至少等待五分钟，然后您可以验证是否已调用 Lambda 函数。

查看 Lambda 函数的输出

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择日志。
3. 选择您的 Lambda 函数 (`/aws/lambda/function-name`) 的日志组的名称。
4. 选择日志流的名称，以查看您启动的实例的函数提供的数据。

步骤 4：确认成功

如果您在 CloudWatch 日志中看到 Lambda 事件，说明您已成功完成本教程。如果您的 CloudWatch 日志中没有该事件，请开始故障排除，首先验证规则是否成功创建，如果规则看起来正确，再验证 Lambda 函数的代码是否正确。

步骤 5：清理资源

除非您想要保留为本教程创建的资源，否则可立即将其删除。请删除您不再使用的 AWS 资源，这样可防止您的 AWS 账户产生不必要的费用。

删除 EventBridge 规则

1. 在 EventBridge 控制台中打开规则页面。
2. 选择您创建的规则。
3. 选择 Delete。
4. 选择 Delete。

删除 Lambda 函数

1. 打开 Lamba 控制台的函数页面。
2. 选择您创建的函数。
3. 依次选择 Actions 和 Delete。
4. 选择 Delete。

用于与 SaaS 提供商集成的 Amazon EventBridge 教程

EventBridge 可以直接与 SaaS 合作伙伴应用程序和服务配合使用，发送和接收事件。以下教程介绍如何将 EventBridge 与 SaaS 合作伙伴集成。

教程:

- 教程：创建连接，将 Datadog 作为 API 目标
- 教程：创建连接，将 Salesforce 作为 API 目标
- 教程：创建连接，将 Zendesk 作为 API 目标

教程：创建连接，将 Datadog 作为 API 目标

您可以使用 EventBridge 将事件路由到第三方服务，例如 Datadog。

在本教程中，您将使用 EventBridge 控制台创建与 Datadog 的连接、指向 Datadog 的 API 目标以及将事件路由到 Datadog 的规则。

步骤：

- 先决条件
- 步骤 1：创建连接
- 步骤 2：创建 API 目标
- 步骤 3：创建规则
- 步骤 4：测试规则
- 步骤 5：清理资源

先决条件

完成本教程需要以下资源：

- 一个 Datadog 账户。
- 一个 Datadog API 密钥。
- 一个启用了 EventBridge 的 Amazon Simple Storage Service (Amazon S3) 桶。

步骤 1：创建连接

要向 Datadog 发送事件，您必须先与 Datadog API 建立连接。

创建连接

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 API 目标。
3. 选择连接选项卡，然后选择创建连接。
4. 为连接输入名称和描述。例如，输入 **`Datadog`** 作为名称，输入 **`Datadog API Connection`** 作为描述。
5. 对于授权类型，选择 API 密钥。

6. 对于 API 密钥名称，请输入 **DD-API-KEY**。
7. 对于值，请粘贴您的 Datadog API 密钥。
8. 选择 Create（创建）。

步骤 2：创建 API 目标

现在，您已经创建了连接，接下来您将创建用作规则目标的 API 目标。

创建 API 目标

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 API 目标。
3. 选择创建 API 目标。
4. 为 API 目标输入名称和描述。例如，输入 **DatadogAD** 作为名称，**Datadog API Destination** 作为描述。
5. 对于 API 目标端点，输入 **https://http-intake.logs.datadoghq.com/api/v2/logs**。
6. 对于 HTTP 方法，选择 POST。
7. 对于调用速率限制，输入 **300**。
8. 对于连接，选择使用现有连接，然后选择您在步骤 1 中创建的 Datadog 连接。
9. 选择 Create（创建）。

步骤 3：创建规则

接下来，您将创建一条规则，在创建 Amazon S3 对象后将事件发送到 Datadog。

创建 规则

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Rules (规则)。
3. 选择 Create rule (创建规则)。
4. 为规则输入名称和描述。例如，输入 **DatadogRule** 作为名称，**Rule to send events to Datadog for S3 object creation** 作为描述。
5. 对于 Event bus（事件总线），选择 default（默认）。

6. 对于 Rule type（规则类型），选择 Rule with an event pattern（具有事件模式的规则）。
7. 选择 Next（下一步）。
8. 对于 Event source（事件源），选择 Other（其他）。
9. 对于事件模式，输入以下代码：

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

10. 选择 Next（下一步）。
11. 对于目标类型，选择 EventBridge API 目标。
12. 对于 API 目标，选择使用现有 API 目标，然后选择您在步骤 2 中创建的 DatadogAD 目标。
13. 对于执行角色，选择为此特定资源创建新角色。
14. 对于其他设置，执行以下操作：
    a. 对于配置目标输入，从下拉列表中选择输入转换器。
    b. 选择配置输入转换器。
    c. 对于示例事件，输入以下代码：

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

    d. 对于目标输入转换器，执行以下操作：
        i. 对于输入路径，输入以下代码：

```
{"detail":"$.detail"}
```

        ii. 对于输入模板，输入以下代码：

```
{"message": <detail>}
```

    e. 选择确认。
15. 选择 Next（下一步）。
16. 选择 Next（下一步）。
17. 查看规则详细信息并选择 Create rule（创建规则）。

步骤 4：测试规则

要测试您的规则，请将文件上传到支持 EventBridge 的桶，创建一个 Amazon S3 对象。将在 Datadog 日志控制台中记录创建的对象。

步骤 5：清理资源

除非您想要保留为本教程创建的资源，否则可立即将其删除。请删除您不再使用的 AWS 资源，这样可防止您的 AWS 账户产生不必要的费用。

删除 EventBridge 连接

1. 在 EventBridge 控制台中打开 API 目标页面。
2. 选择连接选项卡。
3. 选择您创建的连接。
4. 选择 Delete (删除)。
5. 输入连接的名称，然后选择删除。

删除 EventBridge API 目标

1. 在 EventBridge 控制台中打开 API 目标页面。
2. 选择您创建的 API 目标。
3. 选择 Delete (删除)。
4. 输入 API 目标的名称，然后选择删除。

删除 EventBridge 规则

1. 在 EventBridge 控制台中打开规则页面。
2. 选择您创建的规则。
3. 选择 Delete。
4. 选择 Delete。

教程：创建连接，将 Salesforce 作为 API 目标

您可以使用 EventBridge 将事件路由到第三方服务，例如Salesforce。

在本教程中，您将使用 EventBridge 控制台创建指向的连接Salesforce、指向Salesforce的 API 目标以及将事件路由到的规则Salesforce。

步骤：

- 先决条件
- 步骤 1：创建连接
- 步骤 2：创建 API 目标
- 步骤 3：创建规则
- 步骤 4：测试规则
- 步骤 5：清理资源

先决条件

完成本教程需要以下资源：

- 一个 Salesforce 账户。
- 连接 Salesforce 的应用。
- 一个 Salesforce 安全令牌。
- 一个 Salesforce 自定义平台事件。
- EventBridge已启用亚马逊简单存储服务 (Amazon S3) Service 的存储桶。

步骤 1：创建连接

要向 Salesforce 发送事件，您必须先与 Salesforce API 建立连接。

创建连接

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择 API 目标。
3. 选择连接选项卡，然后选择创建连接。
4. 为连接输入名称和描述。例如，输入 **`Salesforce`** 作为名称，输入 **`Salesforce API Connection`** 作为描述。

5. 对于目标类型，选择合作伙伴；对于合作伙伴目标，从下拉列表中选择 Salesforce。
6. 对于授权端点，请输入以下内容之一：
   - 如果您使用的是生产组织，请输入 **https://*MyDomainName*.my.salesforce.com./services/oauth2/token**
   - 如果您使用的是没有增强域的沙盒，请输入 **https://*MyDomainName*--*SandboxName*.my.salesforce.com/services /oauth2/token**
   - 如果您使用的是有增强域的沙盒，请输入 **https://*MyDomainName*--*SandboxName*.sandbox.my.salesforce.com/services/oauth2/token**
7. 对于 HTTP 方法，从下拉列表中选择 POST。
8. 在客户端 ID 中，输入连接的 Salesforce 应用的客户端 ID。
9. 在客户端密钥中，输入连接的 Salesforce 应用的客户端密钥。
10. 对于 OAuth Http 参数，请输入以下键/值对：

密钥	值
grant_type	client_credentials

11. 选择 创建。

步骤 2：创建 API 目标

现在，您已经创建了连接，接下来您将创建用作规则目标的 API 目标。

创建 API 目标

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择 API 目标。
3. 选择创建 API 目标。
4. 为 API 目标输入名称和描述。例如，输入 **SalesforceAD** 作为名称，**Salesforce API Destination** 作为描述。
5. 对于 API 目标端点，请输入 **https://*MyDomainName*.my.salesforce.com/services/data/v54.0/sobjects/*MyEvent__e***，其中 Myevent__e 是您要将信息发送到的平台事件。
6. 对于 HTTP 方法，从下拉列表中选择 POST。
7. 对于调用速率限制，输入 **300**。

8. 对于连接，选择使用现有连接，然后选择您在步骤 1 中创建的 Salesforce 连接。
9. 选择 创建。

步骤 3：创建规则

接下来，您将创建一条规则，在创建 Amazon S3 对象后将事件发送到 Salesforce。

创建 规则

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择规则。
3. 选择创建规则。
4. 为规则输入名称和描述。例如，输入 **SalesforceRule** 作为名称，**Rule to send events to Salesforce for S3 object creation** 作为描述。
5. 对于事件总线，选择默认。
6. 对于规则类型，选择具有事件模式的规则。
7. 选择下一步。
8. 对于事件源，选择其他。
9. 对于事件模式，输入以下代码：

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

10. 选择下一步。
11. 对于目标类型，请选择 EventBridge API 目标。
12. 对于 API 目标，选择使用现有 API 目标，然后选择您在步骤 2 中创建的 SalesforceAD 目标。
13. 对于执行角色，选择为此特定资源创建新角色。
14. 对于其他设置，执行以下操作：

    a. 对于配置目标输入，从下拉列表中选择输入转换器。
    b. 选择配置输入转换器。
    c. 对于示例事件，输入以下代码：

```
{
  "detail":[]
```

```
}
```

d. 对于目标输入转换器，执行以下操作：

i. 对于输入路径，输入以下代码：

```
{"detail":"$.detail"}
```

ii. 对于输入模板，输入以下代码：

```
{"message": <detail>}
```

e. 选择确认。

15. 选择下一步。
16. 选择下一步。
17. 查看规则详细信息并选择创建规则。

步骤 4：测试规则

要测试您的规则，请通过将文件上传到 EventBridge已启用的存储桶来创建 Amazon S3 对象。所创建对象的相关信息将发送到 Salesforce 平台事件。

步骤 5：清理资源

除非您想要保留为本教程创建的资源，否则可立即将其删除。通过删除不再使用的 AWS 资源，可以防止向您的 AWS 账户收取不必要的费用。

删除 EventBridge 连接

1. 打开 EventBridge 控制台的 API 目标页面。
2. 选择连接选项卡。
3. 选择您创建的连接。
4. 选择 Delete（删除)。
5. 输入连接的名称，然后选择删除。

删除 EventBridge API 目的地

1. 打开 EventBridge 控制台的 API 目标页面。

2. 选择您创建的 API 目标。
3. 选择 Delete （ 删除)。
4. 输入 API 目标的名称，然后选择删除。

删除 EventBridge 规则

1. 打开 EventBridge 控制台的 “规则” 页面。
2. 选择您创建的规则。
3. 选择 Delete。
4. 选择 删除。

教程：创建连接，将 Zendesk 作为 API 目标

您可以使用 EventBridge 将事件路由到第三方服务，例如 Zendesk。

在本教程中，您将使用 EventBridge 控制台创建与 Zendesk 的连接、指向 Zendesk 的 API 目标以及将事件路由到 Zendesk 的规则。

步骤：

- 先决条件
- 步骤 1：创建连接
- 步骤 2：创建 API 目标
- 步骤 3：创建规则
- 步骤 4：测试规则
- 步骤 5：清理资源

先决条件

完成本教程需要以下资源：

- 一个 Zendesk 账户。
- 一个启用了 EventBridge 的 Amazon Simple Storage Service (Amazon S3) 桶。

步骤 1：创建连接

要向 Zendesk 发送事件，您必须先与 Zendesk API 建立连接。

创建连接

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 API 目标。
3. 选择连接选项卡，然后选择创建连接。
4. 为连接输入名称和描述。例如，输入 **Zendesk** 作为名称，**Connection to Zendesk API** 作为描述。
5. 对于授权类型，选择基本(用户名/密码)。
6. 对于用户名，输入您的 Zendesk 用户名。

7. 对于密码，输入您的 Zendesk 密码。
8. 选择 Create（创建）。

步骤 2：创建 API 目标

现在，您已经创建了连接，接下来您将创建用作规则目标的 API 目标。

创建 API 目标

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 API 目标。
3. 选择创建 API 目标。
4. 为 API 目标输入名称和描述。例如，输入 **ZendeskAD** 作为名称，**Zendesk API destination** 作为描述。
5. 对于 API 目标端点，请输入 **https://*your-subdomain*.zendesk.com/api/v2/tickets.json**，其中 *your-subdomain* 是与您的 Zendesk 账户关联的子域名。
6. 对于 HTTP 方法，选择 POST。
7. 对于调用速率限制，输入 **10**。
8. 对于连接，选择使用现有连接，然后选择您在步骤 1 中创建的 Zendesk 连接。
9. 选择 Create（创建）。

步骤 3：创建规则

接下来创建一条规则，在创建 Amazon S3 对象后将事件发送到 Zendesk。

创建 规则

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Rules (规则)。
3. 选择 Create rule (创建规则)。
4. 为规则输入名称和描述。例如，输入 **ZendeskRule** 作为名称，**Rule to send events to Zendesk when S3 objects are created** 作为描述。
5. 对于 Event bus（事件总线），选择 default（默认）。
6. 对于 Rule type（规则类型），选择 Rule with an event pattern（具有事件模式的规则）。

7. 选择 Next（下一步）。
8. 对于 Event source（事件源），选择 Other（其他）。
9. 对于事件模式，输入以下代码：

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

10. 选择 Next（下一步）。
11. 对于目标类型，选择 EventBridge API 目标。
12. 对于 API 目标，选择使用现有 API 目标，然后选择您在步骤 2 中创建的 ZendeskAD 目标。
13. 对于执行角色，选择为此特定资源创建新角色。
14. 对于其他设置，执行以下操作：
    a. 对于配置目标输入，从下拉列表中选择输入转换器。
    b. 选择配置输入转换器。
    c. 对于示例事件，输入以下代码：

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

    d. 对于目标输入转换器，执行以下操作：
        i. 对于输入路径，输入以下代码：

```
{"detail":"$.detail"}
```

        ii. 对于输入模板，输入以下代码：

```
{"message": <detail>}
```

    e. 选择确认。
15. 选择 Next（下一步）。
16. 选择 Next（下一步）。
17. 查看规则详细信息并选择 Create rule（创建规则）。

步骤 4：测试规则

要测试您的规则，请将文件上传到支持 EventBridge 的桶，创建一个 Amazon S3 对象。如果事件与规则匹配，EventBridge 将调用 Zendesk Create Ticket API。新工单将显示在 Zendesk 控制面板中。

步骤 5：清理资源

除非您想要保留为本教程创建的资源，否则可立即将其删除。请删除您不再使用的 AWS 资源，这样可防止您的 AWS 账户产生不必要的费用。

删除 EventBridge 连接

1. 在 EventBridge 控制台中打开 API 目标页面。
2. 选择连接选项卡。
3. 选择您创建的连接。
4. 选择 Delete (删除)。
5. 输入连接的名称，然后选择删除。

删除 EventBridge API 目标

1. 在 EventBridge 控制台中打开 API 目标页面。
2. 选择您创建的 API 目标。
3. 选择 Delete (删除)。
4. 输入 API 目标的名称，然后选择删除。

删除 EventBridge 规则

1. 在 EventBridge 控制台中打开规则页面。
2. 选择您创建的规则。
3. 选择 Delete。
4. 选择 Delete。

EventBridge 与 AWS SDK 一起使用

AWS 软件开发套件 (SDK) 可用于许多流行的编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地以其首选语言构建应用程序。

软件开发工具包文档	代码示例
AWS SDK for C++	AWS SDK for C++ 代码示例
AWS SDK for Go	AWS SDK for Go 代码示例
AWS SDK for Java	AWS SDK for Java 代码示例
AWS SDK for JavaScript	AWS SDK for JavaScript 代码示例
AWS SDK for Kotlin	AWS SDK for Kotlin 代码示例
AWS SDK for .NET	AWS SDK for .NET 代码示例
AWS SDK for PHP	AWS SDK for PHP 代码示例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 代码示例
AWS SDK for Ruby	AWS SDK for Ruby 代码示例
AWS SDK for Rust	AWS SDK for Rust 代码示例
适用于 SAP ABAP 的 AWS SDK	适用于 SAP ABAP 的 AWS SDK 代码示例
AWS SDK for Swift	AWS SDK for Swift 代码示例

有关特定于的示例 EventBridge，请参阅 EventBridge 使用 AWS SDK 的代码示例。

> **示例可用性**
>
> 找不到所需的内容？ 通过使用此页面底部的提供反馈链接请求代码示例。

EventBridge 使用 AWS SDK 的代码示例

以下代码示例说明如何 EventBridge 使用 AWS 软件开发套件 (SDK)。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过在同一服务中调用多个函数来完成特定任务的代码示例。

跨服务示例是指跨多个 AWS 服务工作的示例应用程序。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅 EventBridge 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

开始使用

你好 EventBridge

以下代码示例显示了如何开始使用 EventBridge。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
using Amazon.EventBridge;
using Amazon.EventBridge.Model;

namespace EventBridgeActions;

public static class HelloEventBridge
{
    static async Task Main(string[] args)
    {
        var eventBridgeClient = new AmazonEventBridgeClient();
```

```
        Console.WriteLine($"Hello Amazon EventBridge! Following are some of your
 EventBuses:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first five event buses.
        var response = await eventBridgeClient.ListEventBusesAsync(
            new ListEventBusesRequest()
            {
                Limit = 5
            });

        foreach (var eventBus in response.EventBuses)
        {
            Console.WriteLine($"\tEventBus: {eventBus.Name}");
            Console.WriteLine($"\tArn: {eventBus.Arn}");
            Console.WriteLine($"\tPolicy: {eventBus.Policy}");
            Console.WriteLine();
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考[ListEventBuses](#)中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#)中查找完整示例，了解如何进行设置和运行。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 */
public class HelloEventBridge {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
                .region(region)
                .build();

        listBuses(eventBrClient);
        eventBrClient.close();
    }

    public static void listBuses(EventBridgeClient eventBrClient) {
        try {
            ListEventBusesRequest busesRequest = ListEventBusesRequest.builder()
                    .limit(10)
                    .build();

            ListEventBusesResponse response =
 eventBrClient.listEventBuses(busesRequest);
            List<EventBus> buses = response.eventBuses();
            for (EventBus bus : buses) {
                System.out.println("The name of the event bus is: " +
 bus.name());
                System.out.println("The ARN of the event bus is: " + bus.arn());
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListEventBuses中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import aws.sdk.kotlin.services.eventbridge.EventBridgeClient
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesRequest
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesResponse

suspend fun main() {
    listBusesHello()
}

suspend fun listBusesHello() {
    val request = ListEventBusesRequest {
        limit = 10
    }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val response: ListEventBusesResponse =
 eventBrClient.listEventBuses(request)
        response.eventBuses?.forEach { bus ->
            println("The name of the event bus is ${bus.name}")
            println("The ARN of the event bus is ${bus.arn}")
        }
    }
}
```

- 有关 API 的详细信息，请参阅适用ListEventBuses于 K otlin 的AWS SDK API 参考。

代码示例

- EventBridge 使用 AWS SDK 的操作
  - DeleteRule与 S AWS DK 或命令行工具配合使用
  - DescribeRule与 S AWS DK 或命令行工具配合使用

- DisableRule与 S AWS DK 或命令行工具配合使用
- EnableRule与 S AWS DK 或命令行工具配合使用
- ListRuleNamesByTarget与 S AWS DK 或命令行工具配合使用
- ListRules与 S AWS DK 或命令行工具配合使用
- ListTargetsByRule与 S AWS DK 或命令行工具配合使用
- PutEvents与 S AWS DK 或命令行工具配合使用
- PutRule与 S AWS DK 或命令行工具配合使用
- PutTargets与 S AWS DK 或命令行工具配合使用
- RemoveTargets与 S AWS DK 或命令行工具配合使用

- EventBridge 使用 AWS SDK 的场景
  - EventBridge 使用 AWS 软件开发工具包在 Amazon 中创建和触发规则
  - 使用 AWS SDK 开始使用 EventBridge 规则和目标
- EventBridge 使用 AWS SDK 的跨服务示例
  - 使用计划的事件调用 Lambda 函数

EventBridge 使用 AWS SDK 的操作

以下代码示例演示如何使用 AWS 软件开发工具包执行单个 EventBridge操作。这些摘录调用 EventBridge API，是大型程序的代码摘录，这些程序必须在上下文中运行。每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

以下示例仅包括最常用的操作。如需完整列表，请参阅 Amazon EventBridge API 参考。

示例

- DeleteRule与 S AWS DK 或命令行工具配合使用
- DescribeRule与 S AWS DK 或命令行工具配合使用
- DisableRule与 S AWS DK 或命令行工具配合使用
- EnableRule与 S AWS DK 或命令行工具配合使用
- ListRuleNamesByTarget与 S AWS DK 或命令行工具配合使用
- ListRules与 S AWS DK 或命令行工具配合使用
- ListTargetsByRule与 S AWS DK 或命令行工具配合使用
- PutEvents与 S AWS DK 或命令行工具配合使用

- PutRule与 S AWS DK 或命令行工具配合使用
- PutTargets与 S AWS DK 或命令行工具配合使用
- RemoveTargets与 S AWS DK 或命令行工具配合使用

`DeleteRule`与 S AWS DK 或命令行工具配合使用

以下代码示例显示了如何使用`DeleteRule`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用规则和目标

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

按名称删除规则。

```
/// <summary>
/// Delete an event rule by name.
/// </summary>
/// <param name="ruleName">The name of the event rule.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteRuleByName(string ruleName)
{
    var response = await _amazonEventBridge.DeleteRuleAsync(
        new DeleteRuleRequest()
        {
            Name = ruleName
        });

    return response.HttpStatusCode == HttpStatusCode.OK;
```

```
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DeleteRule中的。

CLI

AWS CLI

删除 CloudWatch 事件规则

此示例删除名为 EC2 的规则InstanceStateChanges：

```
aws events delete-rule --name "EC2InstanceStateChanges"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteRule中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
       DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
               .name(ruleName)
               .build();

       eventBrClient.deleteRule(ruleRequest);
       System.out.println("Successfully deleted the rule");
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteRule中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest = DeleteRuleRequest {
        name = ruleName
    }
    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}
```

- 有关 API 的详细信息，请参阅适用DeleteRule于 K otlin 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅 EventBridge 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeRule`与 S AWS DK 或命令行工具配合使用

以下代码示例显示了如何使用`DescribeRule`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用规则和目标

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

使用规则描述获取规则的状态。

```
    /// <summary>
    /// Get the state for a rule by the rule name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="eventBusName">The optional name of the event bus. If empty,
uses the default event bus.</param>
    /// <returns>The state of the rule.</returns>
    public async Task<RuleState> GetRuleStateByRuleName(string ruleName, string?
eventBusName = null)
    {
        var ruleResponse = await _amazonEventBridge.DescribeRuleAsync(
            new DescribeRuleRequest()
            {
                Name = ruleName,
                EventBusName = eventBusName
            });
        return ruleResponse.State;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DescribeRule中的。

CLI

AWS CLI

显示有关 CloudWatch 事件规则的信息

此示例显示有关名为 DailyLambdaFunction：的规则的信息：

```
aws events describe-rule --name "DailyLambdaFunction"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeRule中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
       try {
           DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
                   .name(eventRuleName)
                   .build();

           DescribeRuleResponse response =
eventBrClient.describeRule(ruleRequest);
           System.out.println("The state of the rule is " +
response.stateAsString());

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeRule中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest = DescribeRuleRequest {
        name = eventRuleName
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}
```

- 有关 API 的详细信息，请参阅适用DescribeRule于 K otlin 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅 EventBridge 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DisableRule`与 S AWS DK 或命令行工具配合使用

以下代码示例显示了如何使用`DisableRule`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用规则和目标

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

按规则名称禁用规则。

```
/// <summary>
/// Disable a particular rule on an event bus.
/// </summary
/// <param name="ruleName">The name of the rule.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DisableRuleByName(string ruleName)
{
    var ruleResponse = await _amazonEventBridge.DisableRuleAsync(
        new DisableRuleRequest()
        {
            Name = ruleName
        });
    return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DisableRule中的。

CLI

AWS CLI

禁用 CloudWatch 事件规则

此示例禁用名 DailyLambdaFunction为的规则。该规则未删除：

```
aws events disable-rule --name "DailyLambdaFunction"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DisableRule中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

使用规则名称禁用规则。

```
   public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
       try {
           if (!isEnabled) {
               System.out.println("Disabling the rule: " + eventRuleName);
               DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                       .name(eventRuleName)
                       .build();

               eventBrClient.disableRule(ruleRequest);
           } else {
               System.out.println("Enabling the rule: " + eventRuleName);
               EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                       .name(eventRuleName)
                       .build();
               eventBrClient.enableRule(ruleRequest);
           }

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DisableRule中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun changeRuleState(eventRuleName: String, isEnabled: Boolean?) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest = DisableRuleRequest {
            name = eventRuleName
        }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest = EnableRuleRequest {
            name = eventRuleName
        }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- 有关 API 的详细信息，请参阅适用DisableRule于 K otlin 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅 EventBridge 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`EnableRule`与 S AWS DK 或命令行工具配合使用

以下代码示例显示了如何使用`EnableRule`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用规则和目标

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

按规则名称启用规则。

```
/// <summary>
/// Enable a particular rule on an event bus.
/// </summary>
/// <param name="ruleName">The name of the rule.</param>
/// <returns>True if successful.</returns>
public async Task<bool> EnableRuleByName(string ruleName)
{
    var ruleResponse = await _amazonEventBridge.EnableRuleAsync(
        new EnableRuleRequest()
        {
            Name = ruleName
        });
    return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考EnableRule中的。

CLI

AWS CLI

启用 CloudWatch 事件规则

此示例启用名为的规则 DailyLambdaFunction，该规则以前已被禁用：

```
aws events enable-rule --name "DailyLambdaFunction"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考EnableRule中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

使用规则名称启用规则。

```
   public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
       try {
           if (!isEnabled) {
               System.out.println("Disabling the rule: " + eventRuleName);
               DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                       .name(eventRuleName)
                       .build();

               eventBrClient.disableRule(ruleRequest);
           } else {
               System.out.println("Enabling the rule: " + eventRuleName);
               EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                       .name(eventRuleName)
                       .build();
               eventBrClient.enableRule(ruleRequest);
           }

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考EnableRule中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun changeRuleState(eventRuleName: String, isEnabled: Boolean?) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest = DisableRuleRequest {
            name = eventRuleName
        }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest = EnableRuleRequest {
            name = eventRuleName
        }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- 有关 API 的详细信息，请参阅适用EnableRule于 K otlin 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅 EventBridge 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

ListRuleNamesByTarget与 S AWS DK 或命令行工具配合使用

以下代码示例显示了如何使用ListRuleNamesByTarget。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用规则和目标

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

使用目标列出所有规则名称。

```
    /// <summary>
    /// List names of all rules matching a target.
    /// </summary>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <returns>The list of rule names.</returns>
    public async Task<List<string>> ListAllRuleNamesByTarget(string targetArn)
    {
        var results = new List<string>();
        var request = new ListRuleNamesByTargetRequest()
        {
            TargetArn = targetArn
        };
        ListRuleNamesByTargetResponse response;
        do
        {
            response = await
_amazonEventBridge.ListRuleNamesByTargetAsync(request);
            results.AddRange(response.RuleNames);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);
```

```
        return results;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考ListRuleNamesByTarget中的。

CLI

AWS CLI

显示具有指定目标的所有规则

此示例显示了所有以名为 “MyFunctionName” 的 Lambda 函数作为目标的规则：

```
aws events list-rule-names-by-target --target-arn "arn:aws:lambda:us-
east-1:123456789012:function:MyFunctionName"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ListRuleNamesByTarget中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出使用此目标的所有规则名称。

```
   public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
       ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
               .targetArn(topicArn)
               .build();

       ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
```

```
        List<String> rules = response.ruleNames();
        for (String rule : rules) {
            System.out.println("The rule name is " + rule);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListRuleNamesByTarget中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest = ListRuleNamesByTargetRequest {
        targetArn = topicArnVal
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response =
 eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}
```

- 有关 API 的详细信息，请参阅适用ListRuleNamesByTarget于 K otlin 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅 EventBridge 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListRules`与 S AWS DK 或命令行工具配合使用

以下代码示例显示了如何使用`ListRules`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用规则和目标

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出某事件总线的所有规则。

```
    /// <summary>
    /// List the rules on an event bus.
    /// </summary>
    /// <param name="eventBusArn">The optional ARN of the event bus. If empty,
uses the default event bus.</param>
    /// <returns>The list of rules.</returns>
    public async Task<List<Rule>> ListAllRulesForEventBus(string? eventBusArn =
null)
    {
        var results = new List<Rule>();
        var request = new ListRulesRequest()
        {
            EventBusName = eventBusArn
        };
        // Get all of the pages of rules.
        ListRulesResponse response;
        do
        {
            response = await _amazonEventBridge.ListRulesAsync(request);
            results.AddRange(response.Rules);
            request.NextToken = response.NextToken;
```

```
        } while (response.NextToken is not null);

        return results;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考ListRules中的。

CLI

AWS CLI

显示所有 CloudWatch 事件规则的列表

此示例显示该区域的所有 CloudWatch 事件规则：

```
aws events list-rules
```

显示以特定字符串开头 CloudWatch 的事件规则列表。

此示例显示该区域中名称以 “Daily” 开头的所有 CloudWatch 事件规则：

```
aws events list-rules --name-prefix "Daily"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ListRules中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

使用规则名称启用规则。

```
    public static void listRules(EventBridgeClient eventBrClient) {
```

```
        try {
            ListRulesRequest rulesRequest = ListRulesRequest.builder()
                    .eventBusName("default")
                    .limit(10)
                    .build();

            ListRulesResponse response = eventBrClient.listRules(rulesRequest);
            List<Rule> rules = response.rules();
            for (Rule rule : rules) {
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
 rule.description());
                System.out.println("The rule state is : " +
 rule.stateAsString());
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListRules中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun listRules() {
    val rulesRequest = ListRulesRequest {
        eventBusName = "default"
        limit = 10
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
```

```
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}
```

- 有关 API 的详细信息，请参阅适用ListRules于 K otlin 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅 EventBridge 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListTargetsByRule`与 S AWS DK 或命令行工具配合使用

以下代码示例显示了如何使用`ListTargetsByRule`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用规则和目标

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

使用规则名称列出规则的所有目标。

```
    /// <summary>
    /// List all of the targets matching a rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>The list of targets.</returns>
    public async Task<List<Target>> ListAllTargetsOnRule(string ruleName)
```

```
    {
        var results = new List<Target>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse response;
        do
        {
            response = await _amazonEventBridge.ListTargetsByRuleAsync(request);
            results.AddRange(response.Targets);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考ListTargetsByRule中的。

CLI

AWS CLI

显示 CloudWatch 事件规则的所有目标

此示例显示名为 DailyLambdaFunction：的规则的所有目标：

```
aws events list-targets-by-rule --rule  "DailyLambdaFunction"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ListTargetsByRule中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

使用规则名称列出规则的所有目标。

```
    public static void listTargets(EventBridgeClient eventBrClient, String
 ruleName) {
        ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
                .rule(ruleName)
                .build();

        ListTargetsByRuleResponse res =
 eventBrClient.listTargetsByRule(ruleRequest);
        List<Target> targetsList = res.targets();
        for (Target target: targetsList) {
            System.out.println("Target ARN: "+target.arn());
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListTargetsByRule中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun listTargets(ruleName: String?) {
    val ruleRequest = ListTargetsByRuleRequest {
        rule = ruleName
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}
```

- 有关 API 的详细信息，请参阅适用ListTargetsByRule于 K otlin 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅 EventBridge 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

PutEvents与 S AWS DK 或命令行工具配合使用

以下代码示例显示了如何使用PutEvents。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 创建并触发规则
- 开始使用规则和目标

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

发送与规则的自定义模式相匹配的事件。

```
    /// <summary>
    /// Add an event to the event bus that includes an email, message, and time.
    /// </summary>
    /// <param name="email">The email to use in the event detail of the custom
event.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutCustomEmailEvent(string email)
    {
        var eventDetail = new
        {
```

```
            UserEmail = email,
            Message = "This event was generated by example code.",
            UtcTime = DateTime.UtcNow.ToString("g")
        };
        var response = await _amazonEventBridge.PutEventsAsync(
            new PutEventsRequest()
            {
                Entries = new List<PutEventsRequestEntry>()
                {
                    new PutEventsRequestEntry()
                    {
                        Source = "ExampleSource",
                        Detail = JsonSerializer.Serialize(eventDetail),
                        DetailType = "ExampleType"
                    }
                }
            });

        return response.FailedEntryCount == 0;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考PutEvents中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

包含所需的文件。

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutEventsRequest.h>
#include <aws/events/model/PutEventsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

发送事件。

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;

        Aws::CloudWatchEvents::Model::PutEventsRequestEntry event_entry;
        event_entry.SetDetail(MakeDetails(event_key, event_value));
        event_entry.SetDetailType("sampleSubmitted");
        event_entry.AddResources(resource_arn);
        event_entry.SetSource("aws-sdk-cpp-cloudwatch-example");

        Aws::CloudWatchEvents::Model::PutEventsRequest request;
        request.AddEntries(event_entry);

        auto outcome = cwe.PutEvents(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to post CloudWatch event: " <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully posted CloudWatch event" << std::endl;
        }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考PutEvents中的。

CLI

AWS CLI

向 Events 发送自定义 CloudWatch 事件

此示例向 Events 发送自定义 CloudWatch 事件。该事件包含在 putevents.json 文件中：

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

以下是 putevents.json 文件的内容：

```
[
```

```
  {
    "Source": "com.mycompany.myapp",
    "Detail": "{ \"key1\": \"value1\", \"key2\": \"value2\" }",
    "Resources": [
      "resource1",
      "resource2"
    ],
    "DetailType": "myDetailType"
  },
  {
    "Source": "com.mycompany.myapp",
    "Detail": "{ \"key1\": \"value3\", \"key2\": \"value4\" }",
    "Resources": [
      "resource1",
      "resource2"
    ],
    "DetailType": "myDetailType"
   }
]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考PutEvents中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
       String json = "{" +
               "\"UserEmail\": \"" + email + "\"," +
               "\"Message\": \"This event was generated by example code.\"," +
               "\"UtcTime\": \"Now.\"" +
               "}";

       PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
               .source("ExampleSource")
```

```
                .detail(json)
                .detailType("ExampleType")
                .build();

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

        eventBrClient.putEvents(eventsRequest);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutEvents中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

导入 SDK 和客户端模块，然后调用 API。

```
import {
  EventBridgeClient,
  PutEventsCommand,
} from "@aws-sdk/client-eventbridge";

export const putEvents = async (
  source = "eventbridge.integration.test",
  detailType = "greeting",
  resources = [],
) => {
  const client = new EventBridgeClient({});

  const response = await client.send(
    new PutEventsCommand({
      Entries: [
        {
```

```
          Detail: JSON.stringify({ greeting: "Hello there." }),
          DetailType: detailType,
          Resources: resources,
          Source: source,
        },
      ],
    }),
  );

  console.log("PutEvents response:");
  console.log(response);
  // PutEvents response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3d0df73d-dcea-4a23-ae0d-f5556a3ac109',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Entries: [ { EventId: '51620841-5af4-6402-d9bc-b77734991eb5' } ],
  //   FailedEntryCount: 0
  // }

  return response;
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考PutEvents中的。

适用于 JavaScript (v2) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });
```

```
// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Entries: [
    {
      Detail: '{ "key1": "value1", "key2": "value2" }',
      DetailType: "appRequestSubmitted",
      Resources: ["RESOURCE_ARN"],
      Source: "com.company.app",
    },
  ],
};

ebevents.putEvents(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Entries);
  }
});
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考PutEvents中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun triggerCustomRule(email: String) {
    val json = "{" +
        "\"UserEmail\": \"" + email + "\"," +
        "\"Message\": \"This event was generated by example code.\"" +
        "\"UtcTime\": \"Now.\"" +
        "}"
```

```
    val entry = PutEventsRequestEntry {
        source = "ExampleSource"
        detail = json
        detailType = "ExampleType"
    }

    val eventsRequest = PutEventsRequest {
        this.entries = listOf(entry)
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putEvents(eventsRequest)
    }
}
```

- 有关 API 的详细信息，请参阅适用PutEvents于 K otlin 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅 EventBridge 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

PutRule与 S AWS DK 或命令行工具配合使用

以下代码示例显示了如何使用PutRule。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 创建并触发规则
- 开始使用规则和目标

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建规则，将对象添加到 Amazon Simple Storage Service 桶时触发。

```
    /// <summary>
    /// Create a new event rule that triggers when an Amazon S3 object is created
 in a bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role.</param>
    /// <param name="ruleName">The name to give the rule.</param>
    /// <param name="bucketName">The name of the bucket to trigger the event.</
param>
    /// <returns>The ARN of the new rule.</returns>
    public async Task<string> PutS3UploadRule(string roleArn, string ruleName,
 string bucketName)
    {
        string eventPattern = "{" +
                                "\"source\": [\"aws.s3\"]," +
                                    "\"detail-type\": [\"Object Created\"]," +
                                    "\"detail\": {" +
                                        "\"bucket\": {" +
                                            "\"name\": [\"" + bucketName + "\"]"
 +
                                        "}" +
                                    "}" +
                               "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Example S3 upload rule for EventBridge",
                RoleArn = roleArn,
                EventPattern = eventPattern
            });

        return response.RuleArn;
    }
```

创建使用自定义模式的规则。

```
    /// <summary>
    /// Update a rule to use a custom defined event pattern.
    /// </summary>
```

```
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <returns>The ARN of the updated rule.</returns>
    public async Task<string> UpdateCustomEventPattern(string ruleName)
    {
        string customEventsPattern = "{" +
                                     "\"source\": [\"ExampleSource\"]," +
                                     "\"detail-type\": [\"ExampleType\"]" +
                                     "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Custom test rule",
                EventPattern = customEventsPattern
            });

        return response.RuleArn;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考PutRule中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

包含所需的文件。

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutRuleRequest.h>
#include <aws/events/model/PutRuleResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

创建 规则。

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;
        Aws::CloudWatchEvents::Model::PutRuleRequest request;
        request.SetName(rule_name);
        request.SetRoleArn(role_arn);
        request.SetScheduleExpression("rate(5 minutes)");
        request.SetState(Aws::CloudWatchEvents::Model::RuleState::ENABLED);

        auto outcome = cwe.PutRule(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch events rule " <<
                rule_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch events rule " <<
                rule_name << " with resulting Arn " <<
                outcome.GetResult().GetRuleArn() << std::endl;
        }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考PutRule中的。

CLI

AWS CLI

创建 CloudWatch 活动规则

该示例创建一条可在协调世界时每天上午 9:00 触发的规则。如果您使用 put-targets 将 Lambda 函数添加为该规则的目标，则可以在每天的指定时间运行该 Lambda 函数：

```
aws events put-rule --name "DailyLambdaFunction" --schedule-expression "cron(0 9
 * * ? *)"
```

以下示例创建一条规则，将在区域中的任何 EC2 实例更改状态时触发该规则：

```
aws events put-rule --name "EC2InstanceStateChanges" --event-pattern "{\"source
\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance State-change Notification\"]}"
 --role-arn "arn:aws:iam::123456789012:role/MyRoleForThisRule"
```

以下示例创建一条规则，将在区域中的任何 EC2 实例停止或终止时触发该规则：

```
aws events put-rule --name "EC2InstanceStateChangeStopOrTerminate" --event-
pattern "{\"source\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance State-change
 Notification\"],\"detail\":{\"state\":[\"stopped\",\"terminated\"]}}" --role-arn
 "arn:aws:iam::123456789012:role/MyRoleForThisRule"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考PutRule中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建计划规则。

```
   public static void createEBRule(EventBridgeClient eventBrClient, String
ruleName, String cronExpression) {
       try {
           PutRuleRequest ruleRequest = PutRuleRequest.builder()
                   .name(ruleName)
                   .eventBusName("default")
                   .scheduleExpression(cronExpression)
                   .state("ENABLED")
                   .description("A test rule that runs on a schedule created by
the Java API")
                   .build();

           PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
           System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());
```

```
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

创建规则，将对象添加到 Amazon Simple Storage Service 桶时触发。

```
    // Create a new event rule that triggers when an Amazon S3 object is created
in
    // a bucket.
    public static void addEventRule(EventBridgeClient eventBrClient, String
roleArn, String bucketName,
            String eventRuleName) {
        String pattern = "{\n" +
                "  \"source\": [\"aws.s3\"],\n" +
                "  \"detail-type\": [\"Object Created\"],\n" +
                "  \"detail\": {\n" +
                "    \"bucket\": {\n" +
                "      \"name\": [\"" + bucketName + "\"]\n" +
                "    }\n" +
                "  }\n" +
                "}";

        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .description("Created by using the AWS SDK for Java v2")
                    .name(eventRuleName)
                    .eventPattern(pattern)
                    .roleArn(roleArn)
                    .build();

            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutRule中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

导入 SDK 和客户端模块，然后调用 API。

```
import { EventBridgeClient, PutRuleCommand } from "@aws-sdk/client-eventbridge";

export const putRule = async (
  ruleName = "some-rule",
  source = "some-source",
) => {
  const client = new EventBridgeClient({});

  const response = await client.send(
    new PutRuleCommand({
      Name: ruleName,
      EventPattern: JSON.stringify({ source: [source] }),
      State: "ENABLED",
      EventBusName: "default",
    }),
  );

  console.log("PutRule response:");
  console.log(response);
  // PutRule response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd7292ced-1544-421b-842f-596326bc7072',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
```

```
  //   },
  //   RuleArn: 'arn:aws:events:us-east-1:xxxxxxxxxxxx:rule/
EventBridgeTestRule-1696280037720'
  // }
  return response;
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考PutRule中的。

适用于 JavaScript (v2) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Name: "DEMO_EVENT",
  RoleArn: "IAM_ROLE_ARN",
  ScheduleExpression: "rate(5 minutes)",
  State: "ENABLED",
};

ebevents.putRule(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.RuleArn);
  }
});
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考PutRule中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建计划规则。

```
suspend fun createScRule(ruleName: String?, cronExpression: String?) {
    val ruleRequest = PutRuleRequest {
        name = ruleName
        eventBusName = "default"
        scheduleExpression = cronExpression
        state = RuleState.Enabled
        description = "A test rule that runs on a schedule created by the Kotlin
 API"
    }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

创建规则，将对象添加到 Amazon Simple Storage Service 桶时触发。

```
// Create a new event rule that triggers when an Amazon S3 object is created in a
 bucket.
suspend fun addEventRule(roleArnVal: String?, bucketName: String, eventRuleName:
 String?) {
    val pattern = """{
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
        "bucket": {
            "name": ["$bucketName"]
            }
```

```
        }
    }"""

    val ruleRequest = PutRuleRequest {
        description = "Created by using the AWS SDK for Kotlin"
        name = eventRuleName
        eventPattern = pattern
        roleArn = roleArnVal
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

- 有关 API 的详细信息，请参阅适用PutRule于 K otlin 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅 EventBridge 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

PutTargets与 S AWS DK 或命令行工具配合使用

以下代码示例显示了如何使用PutTargets。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用规则和目标

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

添加 Amazon SNS 主题，作为规则的目标。

```
   /// <summary>
   /// Add an Amazon SNS target topic to a rule.
   /// </summary>
   /// <param name="ruleName">The name of the rule to update.</param>
   /// <param name="targetArn">The ARN of the Amazon SNS target.</param>
   /// <param name="eventBusArn">The optional event bus name, uses default if
empty.</param>
   /// <returns>The ID of the target.</returns>
   public async Task<string> AddSnsTargetToRule(string ruleName, string
targetArn, string? eventBusArn = null)
   {
       var targetID = Guid.NewGuid().ToString();

       // Create the list of targets and add a new target.
       var targets = new List<Target>
       {
           new Target()
           {
               Arn = targetArn,
               Id = targetID
           }
       };

       // Add the targets to the rule.
       var response = await _amazonEventBridge.PutTargetsAsync(
           new PutTargetsRequest()
           {
               EventBusName = eventBusArn,
               Rule = ruleName,
               Targets = targets,
           });

       if (response.FailedEntryCount > 0)
       {
           response.FailedEntries.ForEach(e =>
           {
               _logger.LogError(
                   $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
           });
       }
```

```
        return targetID;
    }
```

将输入转换器添加到规则的目标。

```
    /// <summary>
    /// Update an Amazon S3 object created rule with a transform on the target.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the
default event bus.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateS3UploadRuleTargetWithTransform(string
ruleName, string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputPathsMap = new Dictionary<string, string>()
                    {
                        {"bucket", "$.detail.bucket.name"},
                        {"time", "$.time"}
                    },
                    InputTemplate = "\"Notification: an object was uploaded to
bucket <bucket> at <time>.\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
```

```
            });
        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }
        return targetID;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考PutTargets中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

包含所需的文件。

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutTargetsRequest.h>
#include <aws/events/model/PutTargetsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

添加目标。

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;

        Aws::CloudWatchEvents::Model::Target target;
```

```
        target.SetArn(lambda_arn);
        target.SetId(target_id);

        Aws::CloudWatchEvents::Model::PutTargetsRequest request;
        request.SetRule(rule_name);
        request.AddTargets(target);

        auto putTargetsOutcome = cwe.PutTargets(request);
        if (!putTargetsOutcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch events target for rule "
                << rule_name << ": " <<
                putTargetsOutcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout <<
                "Successfully created CloudWatch events target for rule "
                << rule_name << std::endl;
        }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考[PutTargets](#)中的。

CLI

AWS CLI

为 CloudWatch 事件规则添加目标

以下示例添加了一个 Lambda 函数作为规则的目标：

```
aws events put-targets --rule DailyLambdaFunction --targets
 "Id"="1","Arn"="arn:aws:lambda:us-east-1:123456789012:function:MyFunctionName"
```

以下示例将 Amazon Kinesis 流设置为目标，以便将按此规则捕获的事件中继到该流：

```
aws events put-targets --rule EC2InstanceStateChanges --targets
 "Id"="1","Arn"="arn:aws:kinesis:us-east-1:123456789012:stream/
MyStream","RoleArn"="arn:aws:iam::123456789012:role/MyRoleForThisRule"
```

以下示例将两个 Amazon Kinesis 流设置为一条规则的目标：

```
aws events put-targets --rule DailyLambdaFunction --targets
 "Id"="Target1","Arn"="arn:aws:kinesis:us-east-1:379642911888:stream/
MyStream1","RoleArn"="arn:aws:iam::379642911888:role/ MyRoleToAccessLambda"
  "Id"="Target2"," Arn"="arn:aws:kinesis:us-east-1:379642911888:stream/
MyStream2","RoleArn"="arn:aws:iam::379642911888:role/MyRoleToAccessLambda"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考PutTargets中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

添加 Amazon SNS 主题，作为规则的目标。

```
    // Add a rule which triggers an SNS target when a file is uploaded to an S3
    // bucket.
    public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
            String topicName, String eventRuleName, String bucketName) {
        String targetID = java.util.UUID.randomUUID().toString();
        Target myTarget = Target.builder()
                .id(targetID)
                .arn(topicArn)
                .build();

        List<Target> targets = new ArrayList<>();
        targets.add(myTarget);
        PutTargetsRequest request = PutTargetsRequest.builder()
                .eventBusName(null)
                .targets(targets)
                .rule(ruleName)
                .build();

        eventBrClient.putTargets(request);
```

```
        System.out.println("Added event rule " + eventRuleName + " with Amazon
SNS target " + topicName + " for bucket "
                + bucketName + ".");
    }
```

将输入转换器添加到规则的目标。

```
    public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
            String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: sample event was received.\"")
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutTargets中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

导入 SDK 和客户端模块，然后调用 API。

```
import {
  EventBridgeClient,
  PutTargetsCommand,
} from "@aws-sdk/client-eventbridge";

export const putTarget = async (
  existingRuleName = "some-rule",
  targetArn = "arn:aws:lambda:us-east-1:000000000000:function:test-func",
  uniqueId = Date.now().toString(),
) => {
  const client = new EventBridgeClient({});
  const response = await client.send(
    new PutTargetsCommand({
      Rule: existingRuleName,
      Targets: [
        {
          Arn: targetArn,
          Id: uniqueId,
        },
      ],
    }),
  );

  console.log("PutTargets response:");
  console.log(response);
  // PutTargets response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'f5b23b9a-2c17-45c1-ad5c-f926c3692e3d',
  //     extendedRequestId: undefined,
```

```
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   FailedEntries: [],
  //   FailedEntryCount: 0
  // }

  return response;
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考PutTargets中的。

适用于 JavaScript (v2) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Rule: "DEMO_EVENT",
  Targets: [
    {
      Arn: "LAMBDA_FUNCTION_ARN",
      Id: "myEventBridgeTarget",
    },
  ],
};

ebevents.putTargets(params, function (err, data) {
  if (err) {
    console.log("Error", err);
```

```
  } else {
    console.log("Success", data);
  }
});
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考PutTargets中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
// Add a rule that triggers an SNS target when a file is uploaded to an S3
 bucket.
suspend fun addSnsEventRule(ruleName: String?, topicArn: String?, topicName:
 String, eventRuleName: String, bucketName: String) {
    val targetID = UUID.randomUUID().toString()
    val myTarget = Target {
        id = targetID
        arn = topicArn
    }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request = PutTargetsRequest {
        eventBusName = null
        targets = targetsOb
        rule = ruleName
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target
 $topicName for bucket $bucketName.")
    }
```

```
}
```

将输入转换器添加到规则的目标。

```
suspend fun updateCustomRuleTargetWithTransform(topicArn: String?, ruleName:
 String?) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb = InputTransformer {
        inputTemplate = "\"Notification: sample event was received.\""
    }

    val target = Target {
        id = targetId
        arn = topicArn
        inputTransformer = inputTransformerOb
    }

    val targetsRequest = PutTargetsRequest {
        rule = ruleName
        targets = listOf(target)
        eventBusName = null
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}
```

- 有关 API 的详细信息，请参阅适用PutTargets于 K otlin 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅 EventBridge 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

RemoveTargets与 S AWS DK 或命令行工具配合使用

以下代码示例显示了如何使用RemoveTargets。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

使用规则名称删除规则的所有目标。

```
    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> RemoveAllTargetsFromRule(string ruleName)
    {
        var targetIds = new List<string>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse targetsResponse;
        do
        {
            targetsResponse = await
_amazonEventBridge.ListTargetsByRuleAsync(request);
            targetIds.AddRange(targetsResponse.Targets.Select(t => t.Id));
            request.NextToken = targetsResponse.NextToken;

        } while (targetsResponse.NextToken is not null);

        var removeResponse = await _amazonEventBridge.RemoveTargetsAsync(
            new RemoveTargetsRequest()
            {
                Rule = ruleName,
                Ids = targetIds
            });

        if (removeResponse.FailedEntryCount > 0)
        {
```

```
            removeResponse.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to remove target {e.TargetId}: {e.ErrorMessage},
code {e.ErrorCode}");
            });
        }

        return removeResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考RemoveTargets中的。

CLI

AWS CLI

移除事件的目标

此示例将名为 MyStream 1 的 Amazon Kinesis 直播从规则的目标中移除。DailyLambdaFunction创建时 DailyLambdaFunction ，此直播被设置为目标，ID 为 Target1：

```
aws events remove-targets --rule "DailyLambdaFunction" --ids "Target1"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考RemoveTargets中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

使用规则名称删除该规则的所有目标。

```
    public static void deleteTargetsFromRule(EventBridgeClient eventBrClient,
String eventRuleName) {
```

```
        // First, get all targets that will be deleted.
        ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
                .rule(eventRuleName)
                .build();

        ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
        List<Target> allTargets = response.targets();

        // Get all targets and delete them.
        for (Target myTarget : allTargets) {
            RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
                    .rule(eventRuleName)
                    .ids(myTarget.id())
                    .build();

            eventBrClient.removeTargets(removeTargetsRequest);
            System.out.println("Successfully removed the target");
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[RemoveTargets](#)中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request = ListTargetsByRuleRequest {
        rule = eventRuleName
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
```

```
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest = RemoveTargetsRequest {
                    rule = eventRuleName
                    ids = listOf(myTarget.id.toString())
                }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
}
```

- 有关 API 的详细信息，请参阅适用RemoveTargets于 K otlin 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅 EventBridge 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

EventBridge 使用 AWS SDK 的场景

以下代码示例向您展示了如何 EventBridge使用 AWS 软件开发工具包实现常见场景。这些场景向您展示了如何通过在其中调用多个函数来完成特定任务 EventBridge。每个场景都包含一个指向的链接 GitHub，您可以在其中找到有关如何设置和运行代码的说明。

示例

- EventBridge 使用 AWS 软件开发工具包在 Amazon 中创建和触发规则
- 使用 AWS SDK 开始使用 EventBridge 规则和目标

EventBridge 使用 AWS 软件开发工具包在 Amazon 中创建和触发规则

以下代码示例展示了如何在 Amazon 中创建和触发规则 EventBridge。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

按照正确的顺序调用函数。

```
require "aws-sdk-sns"
require "aws-sdk-iam"
require "aws-sdk-cloudwatchevents"
require "aws-sdk-ec2"
require "aws-sdk-cloudwatch"
require "aws-sdk-cloudwatchlogs"
require "securerandom"
```

检查为该函数提供的主题中是否存在指定的 Amazon Simple Notification Service (Amazon SNS) 主题。

```
# Checks whether the specified Amazon SNS
# topic exists among those provided to this function.
# This is a helper function that is called by the topic_exists? function.
#
# @param topics [Array] An array of Aws::SNS::Types::Topic objects.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   sns_client = Aws::SNS::Client.new(region: 'us-east-1')
#   response = sns_client.list_topics
#   if topic_found?(
#     response.topics,
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
#     puts 'Topic found.'
#   end

def topic_found?(topics, topic_arn)
```

```
  topics.each do |topic|
    return true if topic.topic_arn == topic_arn
  end
  return false
end
```

检查 Amazon SNS 中可供调用方使用的主题中，是否存在指定主题。

```
# Checks whether the specified topic exists among those available to the
# caller in Amazon SNS.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   exit 1 unless topic_exists?(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def topic_exists?(sns_client, topic_arn)
  puts "Searching for topic with ARN '#{topic_arn}'..."
  response = sns_client.list_topics
  if response.topics.count.positive?
    if topic_found?(response.topics, topic_arn)
      puts "Topic found."
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.topics.count.positive?
        if topic_found?(response.topics, topic_arn)
          puts "Topic found."
          return true
        end
      end
    end
  end
  puts "Topic not found."
  return false
rescue StandardError => e
  puts "Topic not found: #{e.message}"
  return false
```

```
end
```

在 Amazon SNS 中创建主题，然后订阅一个电子邮件地址，以接收有关该主题的通知。

```
# Creates a topic in Amazon SNS
# and then subscribes an email address to receive notifications to that topic.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_name [String] The name of the topic to create.
# @param email_address [String] The email address of the recipient to notify.
# @return [String] The ARN of the topic that was created.
# @example
#   puts create_topic(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-topic',
#     'mary@example.com'
#   )
def create_topic(sns_client, topic_name, email_address)
  puts "Creating the topic named '#{topic_name}'..."
  topic_response = sns_client.create_topic(name: topic_name)
  puts "Topic created with ARN '#{topic_response.topic_arn}'."
  subscription_response = sns_client.subscribe(
    topic_arn: topic_response.topic_arn,
    protocol: "email",
    endpoint: email_address,
    return_subscription_arn: true
  )
  puts "Subscription created with ARN " \
    "'#{subscription_response.subscription_arn}'. Have the owner of the " \
    "email address '#{email_address}' check their inbox in a few minutes " \
    "and confirm the subscription to start receiving notification emails."
  return topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  return "Error"
end
```

检查为该函数提供的角色中是否存在指定的 AWS Identity and Access Management (IAM) 角色。

```
# Checks whether the specified AWS Identity and Access Management (IAM)
```

```
# role exists among those provided to this function.
# This is a helper function that is called by the role_exists? function.
#
# @param roles [Array] An array of Aws::IAM::Role objects.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-east-1')
#   response = iam_client.list_roles
#   if role_found?(
#     response.roles,
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
#     puts 'Role found.'
#   end
def role_found?(roles, role_arn)
  roles.each do |role|
    return true if role.arn == role_arn
  end
  return false
end
```

检查 IAM 中可供调用方使用的角色中，是否存在指定角色。

```
# Checks whether the specified role exists among those available to the
# caller in AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   exit 1 unless role_exists?(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
def role_exists?(iam_client, role_arn)
  puts "Searching for role with ARN '#{role_arn}'..."
  response = iam_client.list_roles
  if response.roles.count.positive?
    if role_found?(response.roles, role_arn)
      puts "Role found."
      return true
```

```
    end
    while response.next_page? do
      response = response.next_page
      if response.roles.count.positive?
        if role_found?(response.roles, role_arn)
          puts "Role found."
          return true
        end
      end
    end
  end
  puts "Role not found."
  return false
rescue StandardError => e
  puts "Role not found: #{e.message}"
  return false
end
```

在 IAM 中创建一个角色。

```
# Creates a role in AWS Identity and Access Management (IAM).
# This role is used by a rule in Amazon EventBridge to allow
# that rule to operate within the caller's account.
# This role is designed to be used specifically by this code example.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to create.
# @return [String] The ARN of the role that was created.
# @example
#   puts create_role(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def create_role(iam_client, role_name)
  puts "Creating the role named '#{role_name}'..."
  response = iam_client.create_role(
    assume_role_policy_document: {
      'Version': "2012-10-17",
      'Statement': [
        {
          'Sid': "",
          'Effect': "Allow",
```

```
          'Principal': {
            'Service': "events.amazonaws.com"
          },
          'Action': "sts:AssumeRole"
        }
      ]
    }.to_json,
    path: "/",
    role_name: role_name
  )
  puts "Role created with ARN '#{response.role.arn}'."
  puts "Adding access policy to role..."
  iam_client.put_role_policy(
    policy_document: {
      'Version': "2012-10-17",
      'Statement': [
        {
          'Sid': "CloudWatchEventsFullAccess",
          'Effect': "Allow",
          'Resource': "*",
          'Action': "events:*"
        },
        {
          'Sid': "IAMPassRoleForCloudWatchEvents",
          'Effect': "Allow",
          'Resource': "arn:aws:iam::*:role/AWS_Events_Invoke_Targets",
          'Action': "iam:PassRole"
        }
      ]
    }.to_json,
    policy_name: "CloudWatchEventsPolicy",
    role_name: role_name
  )
  puts "Access policy added to role."
  return response.role.arn
rescue StandardError => e
  puts "Error creating role or adding policy to it: #{e.message}"
  puts "If the role was created, you must add the access policy " \
    "to the role yourself, or delete the role yourself and try again."
  return "Error"
end
```

检查为该函数提供的 EventBridge 规则中是否存在指定的规则。

```
# Checks whether the specified Amazon EventBridge rule exists among
# those provided to this function.
# This is a helper function that is called by the rule_exists? function.
#
# @param rules [Array] An array of Aws::CloudWatchEvents::Types::Rule objects.
# @param rule_arn [String] The name of the rule to find.
# @return [Boolean] true if the name of the rule was found; otherwise, false.
# @example
#   cloudwatchevents_client = Aws::CloudWatch::Client.new(region: 'us-east-1')
#   response = cloudwatchevents_client.list_rules
#   if rule_found?(response.rules, 'aws-doc-sdk-examples-ec2-state-change')
#     puts 'Rule found.'
#   end
def rule_found?(rules, rule_name)
  rules.each do |rule|
    return true if rule.name == rule_name
  end
  return false
end
```

检查指定规则是否存在于中调用者可用的规则中 EventBridge。

```
# Checks whether the specified rule exists among those available to the
# caller in Amazon EventBridge.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to find.
# @return [Boolean] true if the rule name was found; otherwise, false.
# @example
#   exit 1 unless rule_exists?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1')
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def rule_exists?(cloudwatchevents_client, rule_name)
  puts "Searching for rule with name '#{rule_name}'..."
  response = cloudwatchevents_client.list_rules
  if response.rules.count.positive?
    if rule_found?(response.rules, rule_name)
      puts "Rule found."
```

```
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.rules.count.positive?
        if rule_found?(response.rules, rule_name)
          puts "Rule found."
          return true
        end
      end
    end
  end
  puts "Rule not found."
  return false
rescue StandardError => e
  puts "Rule not found: #{e.message}"
  return false
end
```

在中创建规则 EventBridge。

```
# Creates a rule in Amazon EventBridge.
# This rule is triggered whenever an available instance in
# Amazon EC2 changes to the specified state.
# This rule is designed to be used specifically by this code example.
#
# Prerequisites:
#
# - A role in AWS Identity and Access Management (IAM) that is designed
#   to be used specifically by this code example.
# - A topic in Amazon SNS.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to create.
# @param rule_description [String] Some description for this rule.
# @param instance_state [String] The state that available instances in
#   Amazon EC2 must change to, to
#   trigger this rule.
# @param role_arn [String] The Amazon Resource Name (ARN) of the IAM role.
# @param target_id [String] Some identifying string for the rule's target.
# @param topic_arn [String] The ARN of the Amazon SNS topic.
```

```
# @return [Boolean] true if the rule was created; otherwise, false.
# @example
#   exit 1 unless rule_created?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     'Triggers when any available EC2 instance starts.',
#     'running',
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change',
#     'sns-topic',
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def rule_created?(
  cloudwatchevents_client,
  rule_name,
  rule_description,
  instance_state,
  role_arn,
  target_id,
  topic_arn
)
  puts "Creating rule with name '#{rule_name}'..."
  put_rule_response = cloudwatchevents_client.put_rule(
    name: rule_name,
    description: rule_description,
    event_pattern: {
      'source': [
        "aws.ec2"
      ],
      'detail-type': [
        "EC2 Instance State-change Notification"
      ],
      'detail': {
        'state': [
          instance_state
        ]
      }
    }.to_json,
    state: "ENABLED",
    role_arn: role_arn
  )
  puts "Rule created with ARN '#{put_rule_response.rule_arn}'."

  put_targets_response = cloudwatchevents_client.put_targets(
    rule: rule_name,
```

```
    targets: [
      {
        id: target_id,
        arn: topic_arn
      }
    ]
  )
  if put_targets_response.key?(:failed_entry_count) &&
      put_targets_response.failed_entry_count > 0
    puts "Error(s) adding target to rule:"
    put_targets_response.failed_entries.each do |failure|
      puts failure.error_message
    end
    return false
  else
    return true
  end
rescue StandardError => e
  puts "Error creating rule or adding target to rule: #{e.message}"
  puts "If the rule was created, you must add the target " \
    "to the rule yourself, or delete the rule yourself and try again."
  return false
end
```

在 Amazon Logs 中查看调用者可用的日志组中是否存在指定的 CloudWatch 日志组。

```
# Checks to see whether the specified log group exists among those available
# to the caller in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to find.
# @return [Boolean] true if the log group name was found; otherwise, false.
# @example
#   exit 1 unless log_group_exists?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_exists?(cloudwatchlogs_client, log_group_name)
  puts "Searching for log group with name '#{log_group_name}'..."
  response = cloudwatchlogs_client.describe_log_groups(
    log_group_name_prefix: log_group_name
```

```
  )
  if response.log_groups.count.positive?
    response.log_groups.each do |log_group|
      if log_group.log_group_name == log_group_name
        puts "Log group found."
        return true
      end
    end
  end
  puts "Log group not found."
  return false
rescue StandardError => e
  puts "Log group not found: #{e.message}"
  return false
end
```

在 “日志” 中创建 CloudWatch 日志组。

```
# Creates a log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to create.
# @return [Boolean] true if the log group name was created; otherwise, false.
# @example
#   exit 1 unless log_group_created?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_created?(cloudwatchlogs_client, log_group_name)
  puts "Attempting to create log group with the name '#{log_group_name}'..."
  cloudwatchlogs_client.create_log_group(log_group_name: log_group_name)
  puts "Log group created."
  return true
rescue StandardError => e
  puts "Error creating log group: #{e.message}"
  return false
end
```

在 Logs 中将事件写入 CloudWatch 日志流。

```
# Writes an event to a log stream in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
# - A log stream within the log group.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @param log_stream_name [String] The name of the log stream within
#   the log group.
# @param message [String] The message to write to the log stream.
# @param sequence_token [String] If available, the sequence token from the
#   message that was written immediately before this message. This sequence
#   token is returned by Amazon CloudWatch Logs whenever you programmatically
#   write a message to the log stream.
# @return [String] The sequence token that is returned by
#   Amazon CloudWatch Logs after successfully writing the message to the
#   log stream.
# @example
#   puts log_event(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#     '2020/11/19/53f985be-199f-408e-9a45-fc242df41fEX',
#     "Instance 'i-033c48ef067af3dEX' restarted.",
#     '495426724868310740095796045676567882148068632824696073EX'
#   )
def log_event(
  cloudwatchlogs_client,
  log_group_name,
  log_stream_name,
  message,
  sequence_token
)
  puts "Attempting to log '#{message}' to log stream '#{log_stream_name}'..."
  event = {
    log_group_name: log_group_name,
    log_stream_name: log_stream_name,
    log_events: [
      {
        timestamp: (Time.now.utc.to_f.round(3) * 1_000).to_i,
        message: message
```

```
      }
    ]
  }
  unless sequence_token.empty?
    event[:sequence_token] = sequence_token
  end

  response = cloudwatchlogs_client.put_log_events(event)
  puts "Message logged."
  return response.next_sequence_token
rescue StandardError => e
  puts "Message not logged: #{e.message}"
end
```

重启亚马逊弹性计算云 (Amazon EC2) 实例，并将有关相关活动的信息添加到日志中的日志流 CloudWatch 中。

```
# Restarts an Amazon EC2 instance
# and adds information about the related activity to a log stream
# in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - The Amazon EC2 instance to restart.
# - The log group in Amazon CloudWatch Logs to add related activity
#   information to.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client]
#   An initialized Amazon CloudWatch Logs client.
# @param instance_id [String] The ID of the instance.
# @param log_group_name [String] The name of the log group.
# @return [Boolean] true if the instance was restarted and the information
#   was written to the log stream; otherwise, false.
# @example
#   exit 1 unless instance_restarted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX',
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
```

```
def instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  log_stream_name = "#{Time.now.year}/#{Time.now.month}/#{Time.now.day}/" \
    "#{SecureRandom.uuid}"
  cloudwatchlogs_client.create_log_stream(
    log_group_name: log_group_name,
    log_stream_name: log_stream_name
  )
  sequence_token = ""

  puts "Attempting to stop the instance with the ID '#{instance_id}'. " \
    "This might take a few minutes..."
  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts "Instance stopped."
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' stopped.",
    sequence_token
  )

  puts "Attempting to restart the instance. This might take a few minutes..."
  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts "Instance restarted."
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' restarted.",
    sequence_token
  )

  return true
rescue StandardError => e
  puts "Error creating log stream or stopping or restarting the instance: " \
    "#{e.message}"
  log_event(
```

```
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Error stopping or starting instance '#{instance_id}': #{e.message}",
    sequence_token
  )
  return false
end
```

显示有关中某条规则的活动信息 EventBridge。

```
# Displays information about activity for a rule in Amazon EventBridge.
#
# Prerequisites:
#
# - A rule in Amazon EventBridge.
#
# @param cloudwatch_client [Amazon::CloudWatch::Client] An initialized
#   Amazon CloudWatch client.
# @param rule_name [String] The name of the rule.
# @param start_time [Time] The timestamp that determines the first datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param end_time [Time] The timestamp that determines the last datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param period [Integer] The interval, in seconds, to check for activity.
# @example
#   display_rule_activity(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     Time.now - 600, # Start checking from 10 minutes ago.
#     Time.now, # Check up until now.
#     60 # Check every minute during those 10 minutes.
#   )
def display_rule_activity(
  cloudwatch_client,
  rule_name,
  start_time,
  end_time,
  period
)
  puts "Attempting to display rule activity..."
  response = cloudwatch_client.get_metric_statistics(
```

```
    namespace: "AWS/Events",
    metric_name: "Invocations",
    dimensions: [
      {
        name: "RuleName",
        value: rule_name
      }
    ],
    start_time: start_time,
    end_time: end_time,
    period: period,
    statistics: ["Sum"],
    unit: "Count"
  )

  if response.key?(:datapoints) && response.datapoints.count.positive?
    puts "The event rule '#{rule_name}' was triggered:"
    response.datapoints.each do |datapoint|
      puts "  #{datapoint.sum} time(s) at #{datapoint.timestamp}"
    end
  else
    puts "The event rule '#{rule_name}' was not triggered during the " \
      "specified time period."
  end
rescue StandardError => e
  puts "Error getting information about event rule activity: #{e.message}"
end
```

显示日志组中所有日志流的 CloudWatch 日志信息。

```
# Displays log information for all of the log streams in a log group in
# Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Amazon::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @example
#   display_log_data(
```

```
#     Amazon::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def display_log_data(cloudwatchlogs_client, log_group_name)
  puts "Attempting to display log stream data for the log group " \
    "named '#{log_group_name}'..."
  describe_log_streams_response = cloudwatchlogs_client.describe_log_streams(
    log_group_name: log_group_name,
    order_by: "LastEventTime",
    descending: true
  )
  if describe_log_streams_response.key?(:log_streams) &&
      describe_log_streams_response.log_streams.count.positive?
    describe_log_streams_response.log_streams.each do |log_stream|
      get_log_events_response = cloudwatchlogs_client.get_log_events(
        log_group_name: log_group_name,
        log_stream_name: log_stream.log_stream_name
      )
      puts "\nLog messages for '#{log_stream.log_stream_name}':"
      puts "-" * (log_stream.log_stream_name.length + 20)
      if get_log_events_response.key?(:events) &&
          get_log_events_response.events.count.positive?
        get_log_events_response.events.each do |event|
          puts event.message
        end
      else
        puts "No log messages for this log stream."
      end
    end
  end
rescue StandardError => e
  puts "Error getting information about the log streams or their messages: " \
    "#{e.message}"
end
```

向来电者显示提醒，提醒他们手动清理他们不再需要的任何关联 AWS 资源。

```
# Displays a reminder to the caller to manually clean up any associated
# AWS resources that they no longer need.
#
# @param topic_name [String] The name of the Amazon SNS topic.
```

```
# @param role_name [String] The name of the IAM role.
# @param rule_name [String] The name of the Amazon EventBridge rule.
# @param log_group_name [String] The name of the Amazon CloudWatch Logs log
 group.
# @param instance_id [String] The ID of the Amazon EC2 instance.
# @example
#   manual_cleanup_notice(
#     'aws-doc-sdk-examples-topic',
#     'aws-doc-sdk-examples-cloudwatch-events-rule-role',
#     'aws-doc-sdk-examples-ec2-state-change',
#     'aws-doc-sdk-examples-cloudwatch-log',
#     'i-033c48ef067af3dEX'
#   )
def manual_cleanup_notice(
  topic_name, role_name, rule_name, log_group_name, instance_id
)
  puts "-" * 10
  puts "Some of the following AWS resources might still exist in your account."
  puts "If you no longer want to use this code example, then to clean up"
  puts "your AWS account and avoid unexpected costs, you might want to"
  puts "manually delete any of the following resources if they exist:"
  puts "- The Amazon SNS topic named '#{topic_name}'."
  puts "- The IAM role named '#{role_name}'."
  puts "- The Amazon EventBridge rule named '#{rule_name}'."
  puts "- The Amazon CloudWatch Logs log group named '#{log_group_name}'."
  puts "- The Amazon EC2 instance with the ID '#{instance_id}'."
end

# Example usage:
def run_me
  # Properties for the Amazon SNS topic.
  topic_name = "aws-doc-sdk-examples-topic"
  email_address = "mary@example.com"
  # Properties for the IAM role.
  role_name = "aws-doc-sdk-examples-cloudwatch-events-rule-role"
  # Properties for the Amazon EventBridge rule.
  rule_name = "aws-doc-sdk-examples-ec2-state-change"
  rule_description = "Triggers when any available EC2 instance starts."
  instance_state = "running"
  target_id = "sns-topic"
  # Properties for the Amazon EC2 instance.
  instance_id = "i-033c48ef067af3dEX"
  # Properties for displaying the event rule's activity.
  start_time = Time.now - 600 # Go back over the past 10 minutes
```

```
                                # (10 minutes * 60 seconds = 600 seconds).
end_time = Time.now
period = 60 # Look back every 60 seconds over the past 10 minutes.
# Properties for the Amazon CloudWatch Logs log group.
log_group_name = "aws-doc-sdk-examples-cloudwatch-log"
# AWS service clients for this code example.
region = "us-east-1"
sts_client = Aws::STS::Client.new(region: region)
sns_client = Aws::SNS::Client.new(region: region)
iam_client = Aws::IAM::Client.new(region: region)
cloudwatchevents_client = Aws::CloudWatchEvents::Client.new(region: region)
ec2_client = Aws::EC2::Client.new(region: region)
cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
cloudwatchlogs_client = Aws::CloudWatchLogs::Client.new(region: region)

# Get the caller's account ID for use in forming
# Amazon Resource Names (ARNs) that this code relies on later.
account_id = sts_client.get_caller_identity.account

# If the Amazon SNS topic doesn't exist, create it.
topic_arn = "arn:aws:sns:#{region}:#{account_id}:#{topic_name}"
unless topic_exists?(sns_client, topic_arn)
  topic_arn = create_topic(sns_client, topic_name, email_address)
  if topic_arn == "Error"
    puts "Could not create the Amazon SNS topic correctly. Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
    exit 1
  end
end

# If the IAM role doesn't exist, create it.
role_arn = "arn:aws:iam::#{account_id}:role/#{role_name}"
unless role_exists?(iam_client, role_arn)
  role_arn = create_role(iam_client, role_name)
  if role_arn == "Error"
    puts "Could not create the IAM role correctly. Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
end
```

```
# If the Amazon EventBridge rule doesn't exist, create it.
unless rule_exists?(cloudwatchevents_client, rule_name)
  unless rule_created?(
    cloudwatchevents_client,
    rule_name,
    rule_description,
    instance_state,
    role_arn,
    target_id,
    topic_arn
  )
    puts "Could not create the Amazon EventBridge rule correctly. " \
      "Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
end

# If the Amazon CloudWatch Logs log group doesn't exist, create it.
unless log_group_exists?(cloudwatchlogs_client, log_group_name)
  unless log_group_created?(cloudwatchlogs_client, log_group_name)
    puts "Could not create the Amazon CloudWatch Logs log group " \
    "correctly. Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
end

# Restart the Amazon EC2 instance, which triggers the rule.
unless instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  puts "Could not restart the instance to trigger the rule. " \
    "Continuing anyway to show information about the rule and logs..."
end

# Display how many times the rule was triggered over the past 10 minutes.
display_rule_activity(
  cloudwatch_client,
```

```
    rule_name,
    start_time,
    end_time,
    period
  )

  # Display related log data in Amazon CloudWatch Logs.
  display_log_data(cloudwatchlogs_client, log_group_name)

  # Reminder the caller to clean up any AWS resources that are used
  # by this code example and are no longer needed.
  manual_cleanup_notice(
    topic_name, role_name, rule_name, log_group_name, instance_id
  )
end

run_me if $PROGRAM_NAME == __FILE__
```

- 有关 API 详细信息，请参阅 AWS SDK for Ruby API 参考中的以下主题。
  - PutEvents
  - PutRule

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅 EventBridge 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 开始使用 EventBridge 规则和目标

以下代码示例显示了如何：

- 创建规则并为其添加目标。
- 启用和禁用规则。
- 列出并更新规则和目标。
- 发送事件，然后清理资源。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
public class EventBridgeScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks with Amazon EventBridge:
    - Create a rule.
    - Add a target to a rule.
    - Enable and disable rules.
    - List rules and targets.
    - Update rules and targets.
    - Send events.
    - Delete the rule.
    */

    private static ILogger logger = null!;
    private static EventBridgeWrapper _eventBridgeWrapper = null!;
    private static IConfiguration _configuration = null!;

    private static IAmazonIdentityManagementService? _iamClient = null!;
    private static IAmazonSimpleNotificationService? _snsClient = null!;
    private static IAmazonS3 _s3Client = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
```

```
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonEventBridge>()
            .AddAWSService<IAmazonIdentityManagementService>()
            .AddAWSService<IAmazonS3>()
            .AddAWSService<IAmazonSimpleNotificationService>()
            .AddTransient<EventBridgeWrapper>()
            )
            .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<EventBridgeScenario>();

        ServicesSetup(host);

        string topicArn = "";
        string roleArn = "";

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon EventBridge example scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            roleArn = await CreateRole();

            await CreateBucketWithEventBridgeEvents();

            await AddEventRule(roleArn);

            await ListEventRules();

            topicArn = await CreateSnsTopic();
```

```
            var email = await SubscribeToSnsTopic(topicArn);

            await AddSnsTarget(topicArn);

            await ListTargets();

            await ListRulesForTarget(topicArn);

            await UploadS3File(_s3Client);

            await ChangeRuleState(false);

            await GetRuleState();

            await UpdateSnsEventRule(topicArn);

            await ChangeRuleState(true);

            await UploadS3File(_s3Client);

            await UpdateToCustomRule(topicArn);

            await TriggerCustomRule(email);

            await CleanupResources(topicArn);
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
            await CleanupResources(topicArn);
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The Amazon EventBridge example scenario is
complete.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
```

```
        _eventBridgeWrapper =
host.Services.GetRequiredService<EventBridgeWrapper>();
        _snsClient =
host.Services.GetRequiredService<IAmazonSimpleNotificationService>();
        _s3Client = host.Services.GetRequiredService<IAmazonS3>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
    }

    /// <summary>
    /// Create a role to be used by EventBridge.
    /// </summary>
    /// <returns>The role Amazon Resource Name (ARN).</returns>
    public static async Task<string> CreateRole()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating a role to use with EventBridge and attaching
managed policy AmazonEventBridgeFullAccess.");
        Console.WriteLine(new string('-', 80));

        var roleName = _configuration["roleName"];

        var assumeRolePolicy = "{" +
                               "\"Version\": \"2012-10-17\"," +
                               "\"Statement\": [{" +
                               "\"Effect\": \"Allow\"," +
                               "\"Principal\": {" +
                               $"\"Service\": \"events.amazonaws.com\"" +
                               "}," +
                               "\"Action\": \"sts:AssumeRole\"" +
                               "}]" +
                               "}";

        var roleResult = await _iamClient!.CreateRoleAsync(
            new CreateRoleRequest()
            {
                AssumeRolePolicyDocument = assumeRolePolicy,
                Path = "/",
                RoleName = roleName
            });

        await _iamClient.AttachRolePolicyAsync(
            new AttachRolePolicyRequest()
            {
```

```
                PolicyArn = "arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess",
                RoleName = roleName
            });
        // Allow time for the role to be ready.
        Thread.Sleep(10000);
        return roleResult.Role.Arn;
    }

    /// <summary>
    /// Create an Amazon Simple Storage Service (Amazon S3) bucket with
 EventBridge events enabled.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateBucketWithEventBridgeEvents()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating an S3 bucket with EventBridge events
 enabled.");

        var testBucketName = _configuration["testBucketName"];

        var bucketExists = await
 Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_s3Client,
            testBucketName);

        if (!bucketExists)
        {
            await _s3Client.PutBucketAsync(new PutBucketRequest()
            {
                BucketName = testBucketName,
                UseClientRegion = true
            });
        }

        await _s3Client.PutBucketNotificationAsync(new
 PutBucketNotificationRequest()
        {
            BucketName = testBucketName,
            EventBridgeConfiguration = new EventBridgeConfiguration()
        });

        Console.WriteLine($"\tAdded bucket {testBucketName} with EventBridge
 events enabled.");
```

```
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create and upload a file to an S3 bucket to trigger an event.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task UploadS3File(IAmazonS3 s3Client)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Uploading a file to the test bucket. This will trigger
a subscription email.");

        var testBucketName = _configuration["testBucketName"];

        var fileName = $"example_upload_{DateTime.UtcNow.Ticks}.txt";

        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for testing uploads.");
        }

        await s3Client.PutObjectAsync(new PutObjectRequest()
        {
            FilePath = fileName,
            BucketName = testBucketName
        });

        Console.WriteLine($"\tPress Enter to continue.");
        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create an Amazon Simple Notification Service (Amazon SNS) topic to use as
an EventBridge target.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> CreateSnsTopic()
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "Creating an Amazon Simple Notification Service (Amazon SNS) topic
for email subscriptions.");

        var topicName = _configuration["topicName"];

        string topicPolicy = "{" +
                             "\"Version\": \"2012-10-17\"," +
                             "\"Statement\": [{" +
                             "\"Sid\": \"EventBridgePublishTopic\"," +
                             "\"Effect\": \"Allow\"," +
                             "\"Principal\": {" +
                             $"\"Service\": \"events.amazonaws.com\"" +
                             "}," +
                             "\"Resource\": \"*\"," +
                             "\"Action\": \"sns:Publish\"" +
                             "}]" +
                             "}";

        var topicAttributes = new Dictionary<string, string>()
        {
            { "Policy", topicPolicy }
        };

        var topicResponse = await _snsClient!.CreateTopicAsync(new
CreateTopicRequest()
        {
            Name = topicName,
            Attributes = topicAttributes

        });

        Console.WriteLine($"\tAdded topic {topicName} for email subscriptions.");

        Console.WriteLine(new string('-', 80));

        return topicResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a user email to an SNS topic.
    /// </summary>
```

```
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>The user's email.</returns>
    private static async Task<string> SubscribeToSnsTopic(string topicArn)
    {
        Console.WriteLine(new string('-', 80));


        string email = "";
        while (string.IsNullOrEmpty(email))
        {
            Console.WriteLine("Enter your email to subscribe to the Amazon SNS
topic:");
            email = Console.ReadLine()!;
        }

        var subscriptions = new List<string>();
        var paginatedSubscriptions =
_snsClient!.Paginators.ListSubscriptionsByTopic(
            new ListSubscriptionsByTopicRequest()
            {
                TopicArn = topicArn
            });

        // Get the entire list using the paginator.
        await foreach (var subscription in paginatedSubscriptions.Subscriptions)
        {
            subscriptions.Add(subscription.Endpoint);
        }

        if (subscriptions.Contains(email))
        {
            Console.WriteLine($"\tYour email is already subscribed.");
            Console.WriteLine(new string('-', 80));
            return email;
        }

        await _snsClient.SubscribeAsync(new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "email",
            Endpoint = email
        });
```

```
        Console.WriteLine($"Use the link in the email you received to confirm
your subscription, then press Enter to continue.");

        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
        return email;
    }

    /// <summary>
    /// Add a rule which triggers when a file is uploaded to an S3 bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role used by EventBridge.</param>
    /// <returns>Async task.</returns>
    private static async Task AddEventRule(string roleArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating an EventBridge event that sends an email when
an Amazon S3 object is created.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];

        await _eventBridgeWrapper.PutS3UploadRule(roleArn, eventRuleName,
testBucketName);
        Console.WriteLine($"\tAdded event rule {eventRuleName} for bucket
{testBucketName}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add an SNS target to the rule.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>Async task.</returns>
    private static async Task AddSnsTarget(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Adding a target to the rule to that sends an email
when the rule is triggered.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];
```

```
        var topicName = _configuration["topicName"];
        await _eventBridgeWrapper.AddSnsTargetToRule(eventRuleName, topicArn);
        Console.WriteLine($"\tAdded event rule {eventRuleName} with Amazon SNS
target {topicName} for bucket {testBucketName}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List the event rules on the default event bus.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListEventRules()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Current event rules:");

        var rules = await _eventBridgeWrapper.ListAllRulesForEventBus();
        rules.ForEach(r => Console.WriteLine($"\tRule: {r.Name} Description:
{r.Description} State: {r.State}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Update the event target to use a transform.
    /// </summary>
    /// <param name="topicArn">The SNS topic ARN target to update.</param>
    /// <returns>Async task.</returns>
    private static async Task UpdateSnsEventRule(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Let's update the event target with a transform.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];

        await
_eventBridgeWrapper.UpdateS3UploadRuleTargetWithTransform(eventRuleName,
topicArn);
        Console.WriteLine($"\tUpdated event rule {eventRuleName} with Amazon SNS
target {topicArn} for bucket {testBucketName}.");

        Console.WriteLine(new string('-', 80));
```

```
    }

    /// <summary>
    /// Update the rule to use a custom event pattern.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task UpdateToCustomRule(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Updating the event pattern to be triggered by a custom
event instead.");

        var eventRuleName = _configuration["eventRuleName"];

        await _eventBridgeWrapper.UpdateCustomEventPattern(eventRuleName);

        Console.WriteLine($"\tUpdated event rule {eventRuleName} to custom
pattern.");
        await
_eventBridgeWrapper.UpdateCustomRuleTargetWithTransform(eventRuleName,
            topicArn);

        Console.WriteLine($"\tUpdated event target {topicArn}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Send rule events for a custom rule using the user's email address.
    /// </summary>
    /// <param name="email">The email address to include.</param>
    /// <returns>Async task.</returns>
    private static async Task TriggerCustomRule(string email)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Sending an event to trigger the rule. This will
trigger a subscription email.");

        await _eventBridgeWrapper.PutCustomEmailEvent(email);

        Console.WriteLine($"\tEvents have been sent. Press Enter to continue.");
        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
```

```
    }

    /// <summary>
    /// List all of the targets for a rule.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListTargets()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("List all of the targets for a particular rule.");

        var eventRuleName = _configuration["eventRuleName"];
        var targets = await
_eventBridgeWrapper.ListAllTargetsOnRule(eventRuleName);
        targets.ForEach(t => Console.WriteLine($"\tTarget: {t.Arn} Id: {t.Id}
Input: {t.Input}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List all of the rules for a particular target.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>Async task.</returns>
    private static async Task ListRulesForTarget(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("List all of the rules for a particular target.");

        var rules = await _eventBridgeWrapper.ListAllRuleNamesByTarget(topicArn);
        rules.ForEach(r => Console.WriteLine($"\tRule: {r}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Enable or disable a particular rule.
    /// </summary>
    /// <param name="isEnabled">True to enable the rule, otherwise false.</param>
    /// <returns>Async task.</returns>
    private static async Task ChangeRuleState(bool isEnabled)
    {
        Console.WriteLine(new string('-', 80));
```

```
        var eventRuleName = _configuration["eventRuleName"];

        if (!isEnabled)
        {
            Console.WriteLine($"Disabling the rule: {eventRuleName}");
            await _eventBridgeWrapper.DisableRuleByName(eventRuleName);
        }
        else
        {
            Console.WriteLine($"Enabling the rule: {eventRuleName}");
            await _eventBridgeWrapper.EnableRuleByName(eventRuleName);
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get the current state of the rule.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetRuleState()
    {
        Console.WriteLine(new string('-', 80));
        var eventRuleName = _configuration["eventRuleName"];

        var state = await
_eventBridgeWrapper.GetRuleStateByRuleName(eventRuleName);
        Console.WriteLine($"Rule {eventRuleName} is in current state {state}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic to clean up.</param>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        var eventRuleName = _configuration["eventRuleName"];
```

```
        if (GetYesNoResponse($"\tDelete all targets and event rule
{eventRuleName}? (y/n)"))
        {
            Console.WriteLine($"\tRemoving all targets from the event rule.");
            await _eventBridgeWrapper.RemoveAllTargetsFromRule(eventRuleName);

            Console.WriteLine($"\tDeleting event rule.");
            await _eventBridgeWrapper.DeleteRuleByName(eventRuleName);
        }

        var topicName = _configuration["topicName"];
        if (GetYesNoResponse($"\tDelete Amazon SNS subscription topic
{topicName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting topic.");
            await _snsClient!.DeleteTopicAsync(new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        }

        var bucketName = _configuration["testBucketName"];
        if (GetYesNoResponse($"\tDelete Amazon S3 bucket {bucketName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting bucket.");
            // Delete all objects in the bucket.
            var deleteList = await _s3Client.ListObjectsV2Async(new
ListObjectsV2Request()
            {
                BucketName = bucketName
            });
            await _s3Client.DeleteObjectsAsync(new DeleteObjectsRequest()
            {
                BucketName = bucketName,
                Objects = deleteList.S3Objects
                    .Select(o => new KeyVersion { Key = o.Key }).ToList()
            });
            // Now delete the bucket.
            await _s3Client.DeleteBucketAsync(new DeleteBucketRequest()
            {
                BucketName = bucketName
            });
        }
```

```
        var roleName = _configuration["roleName"];
        if (GetYesNoResponse($"\tDelete role {roleName}? (y/n)"))
        {
            Console.WriteLine($"\tDetaching policy and deleting role.");

            await _iamClient!.DetachRolePolicyAsync(new DetachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = "arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess",
            });

            await _iamClient!.DeleteRoleAsync(new DeleteRoleRequest()
            {
                RoleName = roleName
            });
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

创建一个封装 EventBridge 操作的类。

```
/// <summary>
```

```
/// Wrapper for Amazon EventBridge operations.
/// </summary>
public class EventBridgeWrapper
{
    private readonly IAmazonEventBridge _amazonEventBridge;
    private readonly ILogger<EventBridgeWrapper> _logger;

    /// <summary>
    /// Constructor for the EventBridge wrapper.
    /// </summary>
    /// <param name="amazonEventBridge">The injected EventBridge client.</param>
    /// <param name="logger">The injected logger for the wrapper.</param>
    public EventBridgeWrapper(IAmazonEventBridge amazonEventBridge,
 ILogger<EventBridgeWrapper> logger)

    {
        _amazonEventBridge = amazonEventBridge;
        _logger = logger;
    }

    /// <summary>
    /// Get the state for a rule by the rule name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="eventBusName">The optional name of the event bus. If empty,
 uses the default event bus.</param>
    /// <returns>The state of the rule.</returns>
    public async Task<RuleState> GetRuleStateByRuleName(string ruleName, string?
 eventBusName = null)
    {
        var ruleResponse = await _amazonEventBridge.DescribeRuleAsync(
            new DescribeRuleRequest()
            {
                Name = ruleName,
                EventBusName = eventBusName
            });
        return ruleResponse.State;
    }

    /// <summary>
    /// Enable a particular rule on an event bus.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful.</returns>
```

```
    public async Task<bool> EnableRuleByName(string ruleName)
    {
        var ruleResponse = await _amazonEventBridge.EnableRuleAsync(
            new EnableRuleRequest()
            {
                Name = ruleName
            });
        return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Disable a particular rule on an event bus.
    /// </summary
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableRuleByName(string ruleName)
    {
        var ruleResponse = await _amazonEventBridge.DisableRuleAsync(
            new DisableRuleRequest()
            {
                Name = ruleName
            });
        return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the rules on an event bus.
    /// </summary>
    /// <param name="eventBusArn">The optional ARN of the event bus. If empty,
uses the default event bus.</param>
    /// <returns>The list of rules.</returns>
    public async Task<List<Rule>> ListAllRulesForEventBus(string? eventBusArn =
null)
    {
        var results = new List<Rule>();
        var request = new ListRulesRequest()
        {
            EventBusName = eventBusArn
        };
        // Get all of the pages of rules.
        ListRulesResponse response;
        do
        {
            response = await _amazonEventBridge.ListRulesAsync(request);
```

```
            results.AddRange(response.Rules);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// List all of the targets matching a rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>The list of targets.</returns>
    public async Task<List<Target>> ListAllTargetsOnRule(string ruleName)
    {
        var results = new List<Target>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse response;
        do
        {
            response = await _amazonEventBridge.ListTargetsByRuleAsync(request);
            results.AddRange(response.Targets);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// List names of all rules matching a target.
    /// </summary>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <returns>The list of rule names.</returns>
    public async Task<List<string>> ListAllRuleNamesByTarget(string targetArn)
    {
        var results = new List<string>();
        var request = new ListRuleNamesByTargetRequest()
        {
            TargetArn = targetArn
        };
```

```
        ListRuleNamesByTargetResponse response;
        do
        {
            response = await
 _amazonEventBridge.ListRuleNamesByTargetAsync(request);
            results.AddRange(response.RuleNames);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// Create a new event rule that triggers when an Amazon S3 object is created
 in a bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role.</param>
    /// <param name="ruleName">The name to give the rule.</param>
    /// <param name="bucketName">The name of the bucket to trigger the event.</
param>
    /// <returns>The ARN of the new rule.</returns>
    public async Task<string> PutS3UploadRule(string roleArn, string ruleName,
 string bucketName)
    {
        string eventPattern = "{" +
                                "\"source\": [\"aws.s3\"]," +
                                    "\"detail-type\": [\"Object Created\"]," +
                                    "\"detail\": {" +
                                        "\"bucket\": {" +
                                            "\"name\": [\"" + bucketName + "\"]"
 +
                                        "}" +
                                    "}" +
                               "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Example S3 upload rule for EventBridge",
                RoleArn = roleArn,
                EventPattern = eventPattern
            });
```

```
        return response.RuleArn;
    }

    /// <summary>
    /// Update an Amazon S3 object created rule with a transform on the target.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the
default event bus.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateS3UploadRuleTargetWithTransform(string
ruleName, string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputPathsMap = new Dictionary<string, string>()
                    {
                        {"bucket", "$.detail.bucket.name"},
                        {"time", "$.time"}
                    },
                    InputTemplate = "\"Notification: an object was uploaded to
bucket <bucket> at <time>.\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
        if (response.FailedEntryCount > 0)
        {
```

```
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
 {e.ErrorCode}");
            });
        }
        return targetID;
    }

    /// <summary>
    /// Update a custom rule with a transform on the target.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the
 default event bus.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateCustomRuleTargetWithTransform(string
 ruleName, string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputTemplate = "\"Notification: sample event was received.
\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
        if (response.FailedEntryCount > 0)
```

```
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }
        return targetID;
    }

    /// <summary>
    /// Add an event to the event bus that includes an email, message, and time.
    /// </summary>
    /// <param name="email">The email to use in the event detail of the custom
event.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutCustomEmailEvent(string email)
    {
        var eventDetail = new
        {
            UserEmail = email,
            Message = "This event was generated by example code.",
            UtcTime = DateTime.UtcNow.ToString("g")
        };
        var response = await _amazonEventBridge.PutEventsAsync(
            new PutEventsRequest()
            {
                Entries = new List<PutEventsRequestEntry>()
                {
                    new PutEventsRequestEntry()
                    {
                        Source = "ExampleSource",
                        Detail = JsonSerializer.Serialize(eventDetail),
                        DetailType = "ExampleType"
                    }
                }
            });

        return response.FailedEntryCount == 0;
    }

    /// <summary>
    /// Update a rule to use a custom defined event pattern.
```

```
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <returns>The ARN of the updated rule.</returns>
    public async Task<string> UpdateCustomEventPattern(string ruleName)
    {
        string customEventsPattern = "{" +
                                     "\"source\": [\"ExampleSource\"]," +
                                     "\"detail-type\": [\"ExampleType\"]" +
                                     "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Custom test rule",
                EventPattern = customEventsPattern
            });

        return response.RuleArn;
    }

    /// <summary>
    /// Add an Amazon SNS target topic to a rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <param name="targetArn">The ARN of the Amazon SNS target.</param>
    /// <param name="eventBusArn">The optional event bus name, uses default if
empty.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> AddSnsTargetToRule(string ruleName, string
targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        // Create the list of targets and add a new target.
        var targets = new List<Target>
        {
            new Target()
            {
                Arn = targetArn,
                Id = targetID
            }
        };
```

```
        // Add the targets to the rule.
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });

        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }

        return targetID;
    }

    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> RemoveAllTargetsFromRule(string ruleName)
    {
        var targetIds = new List<string>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse targetsResponse;
        do
        {
            targetsResponse = await
_amazonEventBridge.ListTargetsByRuleAsync(request);
            targetIds.AddRange(targetsResponse.Targets.Select(t => t.Id));
            request.NextToken = targetsResponse.NextToken;

        } while (targetsResponse.NextToken is not null);
```

```
        var removeResponse = await _amazonEventBridge.RemoveTargetsAsync(
            new RemoveTargetsRequest()
            {
                Rule = ruleName,
                Ids = targetIds
            });

        if (removeResponse.FailedEntryCount > 0)
        {
            removeResponse.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to remove target {e.TargetId}: {e.ErrorMessage},
 code {e.ErrorCode}");
            });
        }

        return removeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteRuleByName(string ruleName)
    {
        var response = await _amazonEventBridge.DeleteRuleAsync(
            new DeleteRuleRequest()
            {
                Name = ruleName
            });

        return response.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- 有关 API 详细信息，请参阅 AWS SDK for .NET API 参考中的以下主题。
  - DeleteRule
  - DescribeRule
  - DisableRule

- EnableRule
- ListRuleNamesByTarget
- ListRules
- ListTargetsByRule
- PutEvents
- PutRule
- PutTargets

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java code example performs the following tasks:
 *
 * This Java V2 example performs the following tasks with Amazon EventBridge:
 *
 * 1. Creates an AWS Identity and Access Management (IAM) role to use with
 * Amazon EventBridge.
 * 2. Amazon Simple Storage Service (Amazon S3) bucket with EventBridge events
 * enabled.
 * 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 * 4. Lists rules on the event bus.
 * 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and
 * lets the user subscribe to it.
```

```
 * 6. Adds a target to the rule that sends an email to the specified topic.
 * 7. Creates an EventBridge event that sends an email when an Amazon S3 object
 * is created.
 * 8. Lists Targets.
 * 9. Lists the rules for the same target.
 * 10. Triggers the rule by uploading a file to the Amazon S3 bucket.
 * 11. Disables a specific rule.
 * 12. Checks and print the state of the rule.
 * 13. Adds a transform to the rule to change the text of the email.
 * 14. Enables a specific rule.
 * 15. Triggers the updated rule by uploading a file to the Amazon S3 bucket.
 * 16. Updates the rule to be a custom rule pattern.
 * 17. Sending an event to trigger the rule.
 * 18. Cleans up resources.
 *
 */
public class EventbridgeMVP {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException,
 IOException {
        final String usage = """

                Usage:
                    <roleName> <bucketName> <topicName> <eventRuleName>

                Where:
                    roleName - The name of the role to create.
                    bucketName - The Amazon Simple Storage Service (Amazon S3)
 bucket name to create.
                    topicName - The name of the Amazon Simple Notification
 Service (Amazon SNS) topic to create.
                    eventRuleName - The Amazon EventBridge rule name to create.
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String polJSON = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
```

```
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        Scanner sc = new Scanner(System.in);
        String roleName = args[0];
        String bucketName = args[1];
        String topicName = args[2];
        String eventRuleName = args[3];

        Region region = Region.US_EAST_1;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
                .region(region)
                .build();

        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();

        Region regionGl = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(regionGl)
                .build();

        SnsClient snsClient = SnsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EventBridge example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out
                .println("1. Create an AWS Identity and Access Management (IAM)
role to use with Amazon EventBridge.");
        String roleArn = createIAMRole(iam, roleName, polJSON);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("2. Create an S3 bucket with EventBridge events
enabled.");
        if (checkBucket(s3Client, bucketName)) {
            System.out.println("Bucket " + bucketName + " already exists. Ending
this scenario.");
            System.exit(1);
        }

        createBucket(s3Client, bucketName);
        Thread.sleep(3000);
        setBucketNotification(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a rule that triggers when an object is
uploaded to Amazon S3.");
        Thread.sleep(10000);
        addEventRule(eventBrClient, roleArn, bucketName, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. List rules on the event bus.");
        listRules(eventBrClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create a new SNS topic for testing and let the
user subscribe to the topic.");
        String topicArn = createSnsTopic(snsClient, topicName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add a target to the rule that sends an email to
the specified topic.");
        System.out.println("Enter your email to subscribe to the Amazon SNS
topic:");
        String email = sc.nextLine();
        subEmail(snsClient, topicArn, email);
        System.out.println(
                "Use the link in the email you received to confirm your
subscription. Then, press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("7. Create an EventBridge event that sends an email
when an Amazon S3 object is created.");
        addSnsEventRule(eventBrClient, eventRuleName, topicArn, topicName,
eventRuleName, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 8. List Targets.");
        listTargets(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 9. List the rules for the same target.");
        listTargetRules(eventBrClient, topicArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 10. Trigger the rule by uploading a file to the S3
bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Disable a specific rule.");
        changeRuleState(eventBrClient, eventRuleName, false);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Check and print the state of the rule.");
        checkRule(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Add a transform to the rule to change the text of
the email.");
        updateSnsEventRule(eventBrClient, topicArn, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Enable a specific rule.");
```

```
        changeRuleState(eventBrClient, eventRuleName, true);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 15. Trigger the updated rule by uploading a file to
the S3 bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 16. Update the rule to be a custom rule pattern.");
        updateToCustomRule(eventBrClient, eventRuleName);
        System.out.println("Updated event rule " + eventRuleName + " to use a
custom pattern.");
        updateCustomRuleTargetWithTransform(eventBrClient, topicArn,
eventRuleName);
        System.out.println("Updated event target " + topicArn + ".");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Sending an event to trigger the rule. This will
trigger a subscription email.");
        triggerCustomRule(eventBrClient, email);
        System.out.println("Events have been sent. Press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up resources.");
        System.out.println("Do you want to clean up resources (y/n)");
        String ans = sc.nextLine();
        if (ans.compareTo("y") == 0) {
            cleanupResources(eventBrClient, snsClient, s3Client, iam, topicArn,
eventRuleName, bucketName, roleName);
        } else {
            System.out.println("The resources will not be cleaned up. ");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon EventBridge example scenario has
successfully completed.");
```

```
        System.out.println(DASHES);
    }

    public static void cleanupResources(EventBridgeClient eventBrClient,
SnsClient snsClient, S3Client s3Client,
            IamClient iam, String topicArn, String eventRuleName, String
bucketName, String roleName) {
        System.out.println("Removing all targets from the event rule.");
        deleteTargetsFromRule(eventBrClient, eventRuleName);
        deleteRuleByName(eventBrClient, eventRuleName);
        deleteSNSTopic(snsClient, topicArn);
        deleteS3Bucket(s3Client, bucketName);
        deleteRole(iam, roleName);
    }

    public static void deleteRole(IamClient iam, String roleName) {
        String policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess";
        DetachRolePolicyRequest policyRequest = DetachRolePolicyRequest.builder()
                .policyArn(policyArn)
                .roleName(roleName)
                .build();

        iam.detachRolePolicy(policyRequest);
        System.out.println("Successfully detached policy " + policyArn + " from
role " + roleName);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);
    }

    public static void deleteS3Bucket(S3Client s3Client, String bucketName) {
        // Remove all the objects from the S3 bucket.
        ListObjectsRequest listObjects = ListObjectsRequest.builder()
                .bucket(bucketName)
                .build();

        ListObjectsResponse res = s3Client.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
```

```
        for (S3Object myValue : objects) {
            toDelete.add(ObjectIdentifier.builder()
                    .key(myValue.key())
                    .build());
        }

        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
                .bucket(bucketName)
                .delete(Delete.builder()
                        .objects(toDelete).build())
                .build();

        s3Client.deleteObjects(dor);

        // Delete the S3 bucket.
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

        s3Client.deleteBucket(deleteBucketRequest);
        System.out.println("You have deleted the bucket and the objects");
    }

    // Delete the SNS topic.
    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
        DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
```

```
                .name(ruleName)
                .build();

        eventBrClient.deleteRule(ruleRequest);
        System.out.println("Successfully deleted the rule");
    }

    public static void deleteTargetsFromRule(EventBridgeClient eventBrClient,
String eventRuleName) {
        // First, get all targets that will be deleted.
        ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
                .rule(eventRuleName)
                .build();

        ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
        List<Target> allTargets = response.targets();

        // Get all targets and delete them.
        for (Target myTarget : allTargets) {
            RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
                    .rule(eventRuleName)
                    .ids(myTarget.id())
                    .build();

            eventBrClient.removeTargets(removeTargetsRequest);
            System.out.println("Successfully removed the target");
        }
    }

    public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
        String json = "{" +
                "\"UserEmail\": \"" + email + "\"," +
                "\"Message\": \"This event was generated by example code.\"," +
                "\"UtcTime\": \"Now.\"" +
                "}";

        PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
                .source("ExampleSource")
                .detail(json)
                .detailType("ExampleType")
                .build();
```

```
        PutEventsRequest eventsRequest = PutEventsRequest.builder()
                .entries(entry)
                .build();

        eventBrClient.putEvents(eventsRequest);
    }

    public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
            String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: sample event was received.\"")
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void updateToCustomRule(EventBridgeClient eventBrClient, String
ruleName) {
        String customEventsPattern = "{" +
                "\"source\": [\"ExampleSource\"]," +
                "\"detail-type\": [\"ExampleType\"]" +
                "}";

        PutRuleRequest request = PutRuleRequest.builder()
```

```
                .name(ruleName)
                .description("Custom test rule")
                .eventPattern(customEventsPattern)
                .build();

        eventBrClient.putRule(request);
    }

    // Update an Amazon S3 object created rule with a transform on the target.
    public static void updateSnsEventRule(EventBridgeClient eventBrClient, String
topicArn, String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        Map<String, String> myMap = new HashMap<>();
        myMap.put("bucket", "$.detail.bucket.name");
        myMap.put("time", "$.time");

        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: an object was uploaded to bucket
<bucket> at <time>.\"")
                .inputPathsMap(myMap)
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
       try {
           DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
                   .name(eventRuleName)
                   .build();

           DescribeRuleResponse response =
eventBrClient.describeRule(ruleRequest);
           System.out.println("The state of the rule is " +
response.stateAsString());

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
       try {
           if (!isEnabled) {
               System.out.println("Disabling the rule: " + eventRuleName);
               DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                       .name(eventRuleName)
                       .build();

               eventBrClient.disableRule(ruleRequest);
           } else {
               System.out.println("Enabling the rule: " + eventRuleName);
               EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                       .name(eventRuleName)
                       .build();
               eventBrClient.enableRule(ruleRequest);
           }

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   // Create and upload a file to an S3 bucket to trigger an event.
```

```
   public static void uploadTextFiletoS3(S3Client s3Client, String bucketName)
throws IOException {
       // Create a unique file name.
       String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new
Date());
       String fileName = "TextFile" + fileSuffix + ".txt";

       File myFile = new File(fileName);
       FileWriter fw = new FileWriter(myFile.getAbsoluteFile());
       BufferedWriter bw = new BufferedWriter(fw);
       bw.write("This is a sample file for testing uploads.");
       bw.close();

       try {
           PutObjectRequest putOb = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(fileName)
                   .build();

           s3Client.putObject(putOb, RequestBody.fromFile(myFile));

       } catch (S3Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
       ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
               .targetArn(topicArn)
               .build();

       ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
       List<String> rules = response.ruleNames();
       for (String rule : rules) {
           System.out.println("The rule name is " + rule);
       }
   }

   public static void listTargets(EventBridgeClient eventBrClient, String
ruleName) {
```

```
        ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
                .rule(ruleName)
                .build();

        ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
        List<Target> targetsList = res.targets();
        for (Target target: targetsList) {
            System.out.println("Target ARN: "+target.arn());
        }
    }

    // Add a rule which triggers an SNS target when a file is uploaded to an S3
    // bucket.
    public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
            String topicName, String eventRuleName, String bucketName) {
        String targetID = java.util.UUID.randomUUID().toString();
        Target myTarget = Target.builder()
                .id(targetID)
                .arn(topicArn)
                .build();

        List<Target> targets = new ArrayList<>();
        targets.add(myTarget);
        PutTargetsRequest request = PutTargetsRequest.builder()
                .eventBusName(null)
                .targets(targets)
                .rule(ruleName)
                .build();

        eventBrClient.putTargets(request);
        System.out.println("Added event rule " + eventRuleName + " with Amazon
SNS target " + topicName + " for bucket "
                + bucketName + ".");
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String
email) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
```

```
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
"\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listRules(EventBridgeClient eventBrClient) {
        try {
            ListRulesRequest rulesRequest = ListRulesRequest.builder()
                    .eventBusName("default")
                    .limit(10)
                    .build();

            ListRulesResponse response = eventBrClient.listRules(rulesRequest);
            List<Rule> rules = response.rules();
            for (Rule rule : rules) {
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
rule.description());
                System.out.println("The rule state is : " +
rule.stateAsString());
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createSnsTopic(SnsClient snsClient, String topicName) {
        String topicPolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Sid\": \"EventBridgePublishTopic\"," +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
```

```
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Resource\": \"*\"," +
                "\"Action\": \"sns:Publish\"" +
                "}]" +
                "}";

        Map<String, String> topicAttributes = new HashMap<>();
        topicAttributes.put("Policy", topicPolicy);
        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        System.out.println("Added topic " + topicName + " for email
subscriptions.");
        return response.topicArn();
    }

    // Create a new event rule that triggers when an Amazon S3 object is created
in
    // a bucket.
    public static void addEventRule(EventBridgeClient eventBrClient, String
roleArn, String bucketName,
            String eventRuleName) {
        String pattern = "{\n" +
                "  \"source\": [\"aws.s3\"],\n" +
                "  \"detail-type\": [\"Object Created\"],\n" +
                "  \"detail\": {\n" +
                "    \"bucket\": {\n" +
                "      \"name\": [\"" + bucketName + "\"]\n" +
                "    }\n" +
                "  }\n" +
                "}";

        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .description("Created by using the AWS SDK for Java v2")
                    .name(eventRuleName)
                    .eventPattern(pattern)
                    .roleArn(roleArn)
                    .build();
```

```
            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Determine if the S3 bucket exists.
    public static Boolean checkBucket(S3Client s3Client, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.headBucket(headBucketRequest);
            return true;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
    }

    // Set the S3 bucket notification configuration.
    public static void setBucketNotification(S3Client s3Client, String
bucketName) {
        try {
            EventBridgeConfiguration eventBridgeConfiguration =
EventBridgeConfiguration.builder()
                    .build();

            NotificationConfiguration configuration =
NotificationConfiguration.builder()
                    .eventBridgeConfiguration(eventBridgeConfiguration)
                    .build();

            PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
                    .builder()
                    .bucket(bucketName)
                    .notificationConfiguration(configuration)
                    .skipDestinationValidation(true)
```

```
                    .build();

            s3Client.putBucketNotificationConfiguration(configurationRequest);
            System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(polJSON)
                    .description("Created using the AWS SDK for Java")
                    .build();
```

```
            CreateRoleResponse response = iam.createRole(request);
            AttachRolePolicyRequest rolePolicyRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(rolename)
                    .policyArn("arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess")
                    .build();

            iam.attachRolePolicy(rolePolicyRequest);
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 详细信息，请参阅 AWS SDK for Java 2.x API 参考中的以下主题。
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
/*
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin example performs the following tasks with Amazon EventBridge:

 1. Creates an AWS Identity and Access Management (IAM) role to use with Amazon
 EventBridge.
 2. Creates an Amazon Simple Storage Service (Amazon S3) bucket with EventBridge
 events enabled.
 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 4. Lists rules on the event bus.
 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and lets
 the user subscribe to it.
 6. Adds a target to the rule that sends an email to the specified topic.
 7. Creates an EventBridge event that sends an email when an Amazon S3 object is
 created.
 8. Lists targets.
 9. Lists the rules for the same target.
 10. Triggers the rule by uploading a file to the S3 bucket.
 11. Disables a specific rule.
 12. Checks and prints the state of the rule.
 13. Adds a transform to the rule to change the text of the email.
 14. Enables a specific rule.
 15. Triggers the updated rule by uploading a file to the S3 bucket.
 16. Updates the rule to a custom rule pattern.
 17. Sends an event to trigger the rule.
 18. Cleans up resources.
*/
```

```
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <roleName> <bucketName> <topicName> <eventRuleName>

    Where:
        roleName - The name of the role to create.
        bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name to
 create.
        topicName - The name of the Amazon Simple Notification Service (Amazon
 SNS) topic to create.
        eventRuleName - The Amazon EventBridge rule name to create.
    """
    val polJSON = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": \"events.amazonaws.com\"" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]" +
        "}"

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val sc = Scanner(System.`in`)
    val roleName = args[0]
    val bucketName = args[1]
    val topicName = args[2]
    val eventRuleName = args[3]

    println(DASHES)
    println("Welcome to the Amazon EventBridge example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS Identity and Access Management (IAM) role to use
 with Amazon EventBridge.")
    val roleArn = createIAMRole(roleName, polJSON)
```

```
    println(DASHES)

    println(DASHES)
    println("2. Create an S3 bucket with EventBridge events enabled.")
    if (checkBucket(bucketName)) {
        println("$bucketName already exists. Ending this scenario.")
        exitProcess(1)
    }

    createBucket(bucketName)
    delay(3000)
    setBucketNotification(bucketName)
    println(DASHES)

    println(DASHES)
    println("3. Create a rule that triggers when an object is uploaded to Amazon
S3.")
    delay(10000)
    addEventRule(roleArn, bucketName, eventRuleName)
    println(DASHES)

    println(DASHES)
    println("4. List rules on the event bus.")
    listRules()
    println(DASHES)

    println(DASHES)
    println("5. Create a new SNS topic for testing and let the user subscribe to
the topic.")
    val topicArn = createSnsTopic(topicName)
    println(DASHES)

    println(DASHES)
    println("6. Add a target to the rule that sends an email to the specified
topic.")
    println("Enter your email to subscribe to the Amazon SNS topic:")
    val email = sc.nextLine()
    subEmail(topicArn, email)
    println("Use the link in the email you received to confirm your subscription.
Then press Enter to continue.")
    sc.nextLine()
    println(DASHES)

    println(DASHES)
```

```
    println("7. Create an EventBridge event that sends an email when an Amazon S3
object is created.")
    addSnsEventRule(eventRuleName, topicArn, topicName, eventRuleName,
bucketName)
    println(DASHES)

    println(DASHES)
    println("8. List targets.")
    listTargets(eventRuleName)
    println(DASHES)

    println(DASHES)
    println(" 9. List the rules for the same target.")
    listTargetRules(topicArn)
    println(DASHES)

    println(DASHES)
    println("10. Trigger the rule by uploading a file to the S3 bucket.")
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
    println("11. Disable a specific rule.")
    changeRuleState(eventRuleName, false)
    println(DASHES)

    println(DASHES)
    println("12. Check and print the state of the rule.")
    checkRule(eventRuleName)
    println(DASHES)

    println(DASHES)
    println("13. Add a transform to the rule to change the text of the email.")
    updateSnsEventRule(topicArn, eventRuleName)
    println(DASHES)

    println(DASHES)
    println("14. Enable a specific rule.")
    changeRuleState(eventRuleName, true)
    println(DASHES)

    println(DASHES)
```

```
    println("15. Trigger the updated rule by uploading a file to the S3 bucket.")
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
    println("16. Update the rule to a custom rule pattern.")
    updateToCustomRule(eventRuleName)
    println("Updated event rule $eventRuleName to use a custom pattern.")
    updateCustomRuleTargetWithTransform(topicArn, eventRuleName)
    println("Updated event target $topicArn.")
    println(DASHES)

    println(DASHES)
    println("17. Send an event to trigger the rule. This will trigger a
 subscription email.")
    triggerCustomRule(email)
    println("Events have been sent. Press Enter to continue.")
    sc.nextLine()
    println(DASHES)

    println(DASHES)
    println("18. Clean up resources.")
    println("Do you want to clean up resources (y/n)")
    val ans = sc.nextLine()
    if (ans.compareTo("y") == 0) {
        cleanupResources(topicArn, eventRuleName, bucketName, roleName)
    } else {
        println("The resources will not be cleaned up. ")
    }
    println(DASHES)

    println(DASHES)
    println("The Amazon EventBridge example scenario has successfully
 completed.")
    println(DASHES)
}

suspend fun cleanupResources(topicArn: String?, eventRuleName: String?,
 bucketName: String?, roleName: String?) {
    println("Removing all targets from the event rule.")
    deleteTargetsFromRule(eventRuleName)
    deleteRuleByName(eventRuleName)
```

```
    deleteSNSTopic(topicArn)
    deleteS3Bucket(bucketName)
    deleteRole(roleName)
}

suspend fun deleteRole(roleNameVal: String?) {
    val policyArnVal = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
    val policyRequest = DetachRolePolicyRequest {
        policyArn = policyArnVal
        roleName = roleNameVal
    }
    IamClient { region = "us-east-1" }.use { iam ->
        iam.detachRolePolicy(policyRequest)
        println("Successfully detached policy $policyArnVal from role
 $roleNameVal")

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }

        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteS3Bucket(bucketName: String?) {
    // Remove all the objects from the S3 bucket.
    val listObjects = ListObjectsRequest {
        bucket = bucketName
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val myObjects = res.contents
        val toDelete = mutableListOf<ObjectIdentifier>()

        if (myObjects != null) {
            for (myValue in myObjects) {
                toDelete.add(
                    ObjectIdentifier {
                        key = myValue.key
                    }
                )
            }
```

```
        }

        val delOb = Delete {
            objects = toDelete
        }

        val dor = DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }
        s3Client.deleteObjects(dor)

        // Delete the S3 bucket.
        val deleteBucketRequest = DeleteBucketRequest {
            bucket = bucketName
        }
        s3Client.deleteBucket(deleteBucketRequest)
        println("You have deleted the bucket and the objects")
    }
}

// Delete the SNS topic.
suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println(" $topicArnVal was deleted.")
    }
}

suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest = DeleteRuleRequest {
        name = ruleName
    }
    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}

suspend fun deleteTargetsFromRule(eventRuleName: String?) {
```

```
    // First, get all targets that will be deleted.
    val request = ListTargetsByRuleRequest {
        rule = eventRuleName
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest = RemoveTargetsRequest {
                    rule = eventRuleName
                    ids = listOf(myTarget.id.toString())
                }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
}

suspend fun triggerCustomRule(email: String) {
    val json = "{" +
        "\"UserEmail\": \"" + email + "\"," +
        "\"Message\": \"This event was generated by example code.\"" +
        "\"UtcTime\": \"Now.\"" +
        "}"

    val entry = PutEventsRequestEntry {
        source = "ExampleSource"
        detail = json
        detailType = "ExampleType"
    }

    val eventsRequest = PutEventsRequest {
        this.entries = listOf(entry)
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putEvents(eventsRequest)
    }
}
```

```
suspend fun updateCustomRuleTargetWithTransform(topicArn: String?, ruleName:
 String?) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb = InputTransformer {
        inputTemplate = "\"Notification: sample event was received.\""
    }

    val target = Target {
        id = targetId
        arn = topicArn
        inputTransformer = inputTransformerOb
    }

    val targetsRequest = PutTargetsRequest {
        rule = ruleName
        targets = listOf(target)
        eventBusName = null
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun updateToCustomRule(ruleName: String?) {
    val customEventsPattern = "{" +
        "\"source\": [\"ExampleSource\"]," +
        "\"detail-type\": [\"ExampleType\"]" +
        "}"
    val request = PutRuleRequest {
        name = ruleName
        description = "Custom test rule"
        eventPattern = customEventsPattern
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putRule(request)
    }
}

// Update an Amazon S3 object created rule with a transform on the target.
suspend fun updateSnsEventRule(topicArn: String?, ruleName: String?) {
```

```
    val targetId = UUID.randomUUID().toString()
    val myMap = mutableMapOf<String, String>()
    myMap["bucket"] = "$.detail.bucket.name"
    myMap["time"] = "$.time"

    val inputTransOb = InputTransformer {
        inputTemplate = "\"Notification: an object was uploaded to bucket
 <bucket> at <time>.\""
        inputPathsMap = myMap
    }
    val targetOb = Target {
        id = targetId
        arn = topicArn
        inputTransformer = inputTransOb
    }

    val targetsRequest = PutTargetsRequest {
        rule = ruleName
        targets = listOf(targetOb)
        eventBusName = null
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest = DescribeRuleRequest {
        name = eventRuleName
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}

suspend fun changeRuleState(eventRuleName: String, isEnabled: Boolean?) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest = DisableRuleRequest {
            name = eventRuleName
        }
```

```
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest = EnableRuleRequest {
            name = eventRuleName
        }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}

// Create and upload a file to an S3 bucket to trigger an event.
@Throws(IOException::class)
suspend fun uploadTextFiletoS3(bucketName: String?) {
    val fileSuffix = SimpleDateFormat("yyyyMMddHHmmss").format(Date())
    val fileName = "TextFile$fileSuffix.txt"
    val myFile = File(fileName)
    val fw = FileWriter(myFile.absoluteFile)
    val bw = BufferedWriter(fw)
    bw.write("This is a sample file for testing uploads.")
    bw.close()

    val putOb = PutObjectRequest {
        bucket = bucketName
        key = fileName
        body = myFile.asByteStream()
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.putObject(putOb)
    }
}

suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest = ListRuleNamesByTargetRequest {
        targetArn = topicArnVal
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response =
 eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
```

```
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}

suspend fun listTargets(ruleName: String?) {
    val ruleRequest = ListTargetsByRuleRequest {
        rule = ruleName
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}

// Add a rule that triggers an SNS target when a file is uploaded to an S3
 bucket.
suspend fun addSnsEventRule(ruleName: String?, topicArn: String?, topicName:
 String, eventRuleName: String, bucketName: String) {
    val targetID = UUID.randomUUID().toString()
    val myTarget = Target {
        id = targetID
        arn = topicArn
    }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request = PutTargetsRequest {
        eventBusName = null
        targets = targetsOb
        rule = ruleName
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target
 $topicName for bucket $bucketName.")
    }
}
```

```
suspend fun subEmail(topicArnVal: String?, email: String?) {
    val request = SubscribeRequest {
        protocol = "email"
        endpoint = email
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" Subscription ARN: ${result.subscriptionArn}")
    }
}

suspend fun createSnsTopic(topicName: String): String? {
    val topicPolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Sid\": \"EventBridgePublishTopic\"," +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": \"events.amazonaws.com\"" +
        "}," +
        "\"Resource\": \"*\"," +
        "\"Action\": \"sns:Publish\"" +
        "}]" +
        "}"

    val topicAttributes = mutableMapOf<String, String>()
    topicAttributes["Policy"] = topicPolicy

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        println("Added topic $topicName for email subscriptions.")
        return response.topicArn
    }
}
```

```
suspend fun listRules() {
    val rulesRequest = ListRulesRequest {
        eventBusName = "default"
        limit = 10
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}

// Create a new event rule that triggers when an Amazon S3 object is created in a
 bucket.
suspend fun addEventRule(roleArnVal: String?, bucketName: String, eventRuleName:
 String?) {
    val pattern = """{
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
        "bucket": {
            "name": ["$bucketName"]
            }
        }
    }"""

    val ruleRequest = PutRuleRequest {
        description = "Created by using the AWS SDK for Kotlin"
        name = eventRuleName
        eventPattern = pattern
        roleArn = roleArnVal
    }

    EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}

// Set the Amazon S3 bucket notification configuration.
suspend fun setBucketNotification(bucketName: String) {
```

```
    val eventBridgeConfig = EventBridgeConfiguration {
    }

    val configuration = NotificationConfiguration {
        eventBridgeConfiguration = eventBridgeConfig
    }

    val configurationRequest = PutBucketNotificationConfigurationRequest {
        bucket = bucketName
        notificationConfiguration = configuration
        skipDestinationValidation = true
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.putBucketNotificationConfiguration(configurationRequest)
        println("Added bucket $bucketName with EventBridge events enabled.")
    }
}

// Create an S3 bucket using a waiter.
suspend fun createBucket(bucketName: String) {
    val request = CreateBucketRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        s3.waitUntilBucketExists {
            bucket = bucketName
        }
        println("$bucketName is ready")
    }
}

suspend fun checkBucket(bucketName: String?): Boolean {
    try {
        // Determine if the S3 bucket exists.
        val headBucketRequest = HeadBucketRequest {
            bucket = bucketName
        }

        S3Client { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            return true
```

```
        }
    } catch (e: S3Exception) {
        System.err.println(e.message)
    }
    return false
}

suspend fun createIAMRole(rolenameVal: String?, polJSON: String?): String? {
    val request = CreateRoleRequest {
        roleName = rolenameVal
        assumeRolePolicyDocument = polJSON
        description = "Created using the AWS SDK for Kotlin"
    }

    val rolePolicyRequest = AttachRolePolicyRequest {
        roleName = rolenameVal
        policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
    }

    IamClient { region = "us-east-1" }.use { iam ->
        val response = iam.createRole(request)
        iam.attachRolePolicy(rolePolicyRequest)
        return response.role?.arn
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的以下主题。
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅 EventBridge 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

EventBridge 使用 AWS SDK 的跨服务示例

以下示例应用程序使用 AWS SDK 与其他 AWS 服务应用程序组 EventBridge合。每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何设置和运行应用程序的说明。

示例

- 使用计划的事件调用 Lambda 函数

使用计划的事件调用 Lambda 函数

以下代码示例说明如何创建由 Amazon EventBridge 计划事件调用的 AWS Lambda 函数。

Java

适用于 Java 2.x 的 SDK

演示如何创建调用函数的 Amazon EventBridge 计划事件。 AWS Lambda 配置 EventBridge 为使用 cron 表达式来调度 Lambda 函数的调用时间。在本示例中，您使用 Lambda Java 运行时 API 创建 Lambda 函数。此示例调用不同的 AWS 服务来执行特定的用例。此示例展示了如何创建一个应用程序，在其一周年纪念日时向员工发送移动短信表示祝贺。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

演示如何创建调用函数的 Amazon EventBridge 计划事件。 AWS Lambda 配置 EventBridge 为使用 cron 表达式来调度 Lambda 函数的调用时间。在此示例中，您将使用 Lambda 运行时 API

创建一个 Lambda 函数。 JavaScript 此示例调用不同的 AWS 服务来执行特定的用例。此示例展示了如何创建一个应用程序，在其一周年纪念日时向员工发送移动短信表示祝贺。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

该示例也可在 AWS SDK for JavaScript v3 开发人员指南中找到。

本示例中使用的服务

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK for Python (Boto3)

此示例说明如何将 AWS Lambda 函数注册为计划的 Amazon EventBridge 事件的目标。Lambda 处理程序将友好的消息和完整的事件数据写入 Amazon CloudWatch 日志，以供日后检索。

- 部署 Lambda 函数。
- 创建 EventBridge 计划事件并将 Lambda 函数设为目标。
- 授予允许 EventBridge 调用 Lambda 函数的权限。
- 打印来自 CloudWatch Logs 的最新数据以显示计划调用的结果。
- 清理演示期间创建的所有资源。

最好在上查看此示例 GitHub。有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- CloudWatch 日志
- EventBridge
- Lambda

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅 EventBridge 与 AWS SDK 一起使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

亚马逊 EventBridge 安全

Amazon EventBridge AWS Identity and Access Management 用于控制对其他 AWS 服务和资源的访问。有关 IAM 工作原理的概述，请参阅《IAM 用户指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction_access-management.html中的访问管理概述。有关安全凭证的概述，请参阅中的AWS 安全凭证Amazon Web Services 一般参考。

主题

Amazon EventBridge 中的数据保护

AWS 责任共担模式适用于 Amazon EventBridge 中的数据保护。如该模式中所述，AWS 负责保护运行所有 AWS Cloud 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS 安全性博客 上的博客文章 AWS Shared Responsibility Model and GDPR。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置单个用户。这样，每个用户只获得履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 使用 SSL/TLS 与 AWS 资源进行通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用 AWS CloudTrail 设置 API 和用户活动日志记录。
- 使用 AWS 加密解决方案以及 AWS 服务 中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在通过命令行界面或 API 访问 AWS 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准 (FIPS) 第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括使用控制台、API、AWS 服务 或 AWS CLI SDK 处理 EventBridge 或其他 AWS时。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，我们强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

静态加密

EventBridge 会对其存储的事件元数据和消息数据进行加密。默认情况下，EventBridge 使用 AWS 拥有的密钥，使用 256 位高级加密标准 (AES-256) 加密数据，这样有助于保护您的数据，防止未经授权的访问。使用 AWS 拥有的密钥加密数据，不会产生额外费用。

传输中加密

对于 EventBridge 和其他服务之间传递的数据，EventBridge 使用传输层安全性协议 (TLS) 进行加密。

基于标签的策略

在 Amazon EventBridge 中，您可以根据标签使用策略，来控制对资源的访问。

例如，您可能限制对下面这类资源的访问：在这些资源包含的标签中，具有键 `environment` 和值 `production`。以下示例策略将拒绝具有此标签的任何资源，对于标记为 `environment/production` 的资源，无法创建、删除或修改标签、规则或事件总线。

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

有关标记的更多信息，请参阅以下内容。

- 亚马逊 EventBridge 标签
- 使用 IAM 标签控制访问

亚马逊 EventBridge 和 AWS Identity and Access Management

要访问 Amazon EventBridge，您需要AWS可用于验证您的请求的凭证。您的凭证必须有权访问 AWS 资源，例如从其他 AWS 资源检索事件数据。以下各节详细介绍了如何使用 AWS Identity and Access Management(IAM)，以及 EventBridge 如何通过控制谁可以访问资源来帮助保护您的资源。

主题

身份验证

您可以以下面任一类型的身份访问AWS：

- AWS 账户根用户 - 注册 AWS 时，您需要提供与您的账户关联的电子邮件地址和密码。这些是您的根凭证，它们提供对您所有 AWS 资源的完全访问权限。

  > **Important**
  >
  > 出于安全考虑，我们建议您仅使用根凭证创建管理员，它是对您的账户具有完全访问权限的 IAM 用户。随后，您可以使用此管理员来创建具有有限权限的其他 用户和角色。有关更多信息，请参阅 IAM 用户指南中的 IAM 最佳实践和创建管理员用户和组。

- IAM 用户 — IAM 用户是您的账户中具有特定权限的身份，例如，向中的目标发送事件数据的权限 EventBridge。您可以使用 IAM 登录凭证登录安全的 AWS 网页，如 AWS Management Console、AWS 论坛或 AWS Support 中心等。

除了登录凭证之外，还可以为每个用户生成访问密钥。在通过几个开发工具包之一或使用 AWS Command Line Interface (AWS CLI) 以编程方式访问 AWS 服务，以加密签署您的请求时，可以使用这些密钥。如果您不使用 AWS 工具，则必须使用签名版本 4 自行对请求进行签名。这是一种用于对入站 API 请求进行身份验证的协议。有关验证请求的更多信息，请参阅《Amazon Web Services 一般参考》 中的签名版本 4 签名流程。

- IAM 角色 - IAM 角色是可在账户中创建的另一种具有特定权限的 IAM 身份。它类似于 IAM 用户，但未与特定人员相关联。利用 IAM 角色，您可以获得可用于访问 AWS 服务和资源的临时访问密钥。具有临时凭证的 IAM 角色在以下情况下很有用：
  - 联合用户访问 - 您可以不创建用户，而是使用来自 AWS Directory Service、您的企业用户目录或 Web 身份提供者 (IdP) 的既有身份。这些用户被称为联合用户。当用户通过身份提供者请求访问权限时，AWS 将为联合用户分配角色。有关联合用户的更多信息，请参阅《IAM 用户指南》中的联合用户和角色。
  - 跨账户存取 - 您可以使用您账户中的 IAM 角色，向另一个账户授予对您账户中资源的访问权限。有关示例，请参阅 IAM 用户指南中的教程：使用 IAM 角色委派跨 AWS 账户的访问权限。
  - AWS 服务访问 - 您可以使用您账户中的 IAM 角色，向 AWS 服务授予对您账户中资源的访问权。例如，您可以创建一个角色，允许 Amazon Redshift 将存储在一个 Amazon S3 桶中的数据加载到一个 Amazon Redshift 集群中。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。
  - 在 Amazon EC2 上运行的应用程序 — 对于需要访问的 Amazon EC2 应用程序 EventBridge，您可以将访问密钥存储在 EC2 实例中，也可以使用 IAM 角色来管理临时证书。要向 EC2 实例分配 AWS 角色，可创建实例配置文件并附加到实例。实例配置文件包含角色，并为 EC2 实例上运行的应用程序提供临时凭证。有关更多信息，请参阅 IAM 用户指南中的对 Amazon EC2 上的应用程序使用角色。

访问控制

要创建或访问 EventBridge 资源，您需要有效的证书和权限。例如，要调用 AWS Lambda、Amazon Simple Notification Service (Amazon SNS) 和 Amazon Simple Queue Service (Amazon SQS) 目标，您必须拥有这些服务的权限。

管理对 Amazon EventBridge 资源的访问权限

您可以使用基于身份或基于资源的策略，管理对 EventBridge 资源（例如规则或事件）的访问权限。

EventBridge 资源

EventBridge 资源和子资源具有与其关联的唯一 Amazon 资源名称 (ARN)。在 EventBridge 中可以使用 ARN 来创建事件模式。有关 ARN 的详细信息，请参阅 AWS 中的 Amazon 资源名称 (ARN) 和 Amazon Web Services 一般参考 服务命名空间。

有关 EventBridge 提供的处理资源的操作列表，请参阅 Amazon EventBridge 权限参考。

> **Note**
>
> AWS 中的大多数服务将 ARN 中的冒号 (:) 或正斜杠 (/) 视为相同的字符。不过，EventBridge 在事件模式和规则中使用精确匹配。请在创建事件模式时务必使用正确的 ARN 字符，以使其匹配需要匹配的事件中的 ARN 语法。

下表展示了 EventBridge 中的资源。

资源类型	ARN 格式
存档	`arn:aws:events:` *`region`*`:`*`account`*`:archive/` *`archive-name`*
回放	`arn:aws:events:` *`region`*`:`*`account`*`:replay/`*`replay-name`*
规则	`arn:aws:events:` *`region`*`:`*`account`*`:rule/`*`[event-bus-name]/rule-name`*
事件总线	`arn:aws:events:` *`region`*`:`*`account`*`:event-bus/` *`event-bus-name`*
所有 EventBridge 资源	`arn:aws:events:*`
指定账户在指定区域拥有的所有 EventBridge 资源	`arn:aws:events:` *`region`*`:`*`account`*`:*`

以下示例展示如何使用某个特定规则的 ARN 在语句中指定该规则 (*myRule*)。

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

可以使用星号 (*) 通配符指定属于特定账户的所有规则，如下所示。

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

要指定所有资源，或者如果特定 API 操作不支持 ARN，请在 `Resource` 元素中使用星号 (*) 通配符，如下所示。

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

要在单个语句中指定多个资源或 `PutTargets`，请使用逗号分隔其 ARN，如下所示。

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

资源所有权

账户对账户中的资源具有所有权，无论创建资源的人是谁。资源拥有者是主体实体 的账户，即账户根用户、IAM 用户或对创建资源的请求进行身份验证的角色。以下示例说明了它的工作原理：

- 如果使用您账户的根用户凭证创建规则，则您的账户即为该 EventBridge 资源的拥有者。
- 如果您在您的账户中创建用户，并对该用户授予创建 EventBridge 资源的权限，则该用户可以创建 EventBridge 资源。但是，这些 EventBridge 资源归该用户所属的您的账户所有。
- 如果您在您的账户中创建具有创建 EventBridge 资源权限的 IAM 角色，则能够担任该角色的任何人都可以创建 EventBridge 资源。该角色所属的您的账户拥有这些 EventBridge 资源。

管理对资源的访问

权限策略规定谁可以访问哪些内容。下一节介绍创建权限策略时的可用选项。

> **Note**
>
> 本节讨论如何在 EventBridge 场景中使用 IAM。这里不提供有关 IAM 服务的详细信息。有关完整的 IAM 文档，请参阅 IAM 用户指南中的什么是 IAM？。有关 IAM policy 语法和说明的信息，请参阅 IAM 用户指南中的 IAM policy 参考。

附加到 IAM 身份的策略称作基于身份 的策略（IAM policy），附加到资源的策略称作基于资源 的策略。在 EventBridge 中，您可以同时使用基于身份（IAM 策略）和基于资源的策略。

主题

- 基于身份的策略（IAM policy）
- 基于资源的策略（IAM 策略)

基于身份的策略（IAM policy）

您可以向 IAM 身份附加策略。例如，您可以执行以下操作：

- 将权限策略附加到您账户中的用户或组 - 要授予用户在 Amazon CloudWatch 控制台中查看规则的权限，可以将权限策略附加到用户或用户所属的组。
- 向角色附加权限策略（授予跨账户权限）– 您可以向 IAM 角色附加基于身份的权限策略，以授予跨账户的权限。例如，账户 A 中的管理员可以创建一个角色，向另一账户 B 或某项 AWS 服务授予跨账户权限，如下所述：
  1. 账户 A 管理员可以创建一个 IAM 角色，然后向该角色附加授予其访问账户 A 中资源的权限策略。
  2. 账户 A 管理员可以向角色挂载信任策略，将账户 B 标识为能够担任该角色的委托人。
  3. 之后，账户 B 管理员可以委派权限，指派账户 B 中的任何用户代入该角色。这样，账户 B 中的用户就可以创建或访问账户 A 中的资源了。如果您需要授予 AWS 服务代入该角色所需的权限，则信任策略中的主体也可以是 AWS 服务主体。

  有关使用 IAM 委托权限的更多信息，请参阅 IAM 用户指南中的访问权限管理。

可以创建特定的 IAM 策略，来限制您账户中的用户有权访问的调用和资源，然后将这些策略与用户关联。有关创建 IAM 角色和探索适用于 EventBridge 的 IAM 策略语句示例的更多信息，请参阅 管理对 Amazon EventBridge 资源的访问权限。

基于资源的策略（IAM 策略)

当规则在 EventBridge 中运行时，所有与该规则关联的目标都会被调用，这意味着调用 AWS Lambda 函数、发布到 Amazon SNS 主题或将事件中继到 Amazon Kinesis 流。为了对您拥有的资源执行 API 调用，EventBridge 需要相应权限。对于 Lambda、Amazon SNS 和 Amazon SQS 资源，EventBridge 使用基于资源的策略。对于 Kinesis 流，EventBridge 使用 IAM 角色。

有关创建 IAM 角色和探索适用于 EventBridge 的基于资源的策略语句示例的更多信息，请参阅 针对 Amazon EventBridge 使用基于资源的策略。

指定策略元素：操作、效果和主体

对于每种 EventBridge 资源，EventBridge 都定义了一组 API 操作。要为这些 API 操作授予权限，EventBridge 定义了一组您可以在策略中指定的操作。某些 API 操作需要多个操作的权限，才能执行 API 操作。有关资源和 API 操作的更多信息，请参阅 EventBridge 资源 和 Amazon EventBridge 权限参考。

以下是基本的策略元素：

- Resource – 使用 Amazon Resource Name (ARN) 指定策略应用到的资源。有关更多信息，请参阅EventBridge 资源。
- 操作 - 使用关键字指定要允许或拒绝的资源操作。例如，`events:Describe` 权限允许用户执行 `Describe` 操作。
- 效果 - 指定是允许还是拒绝。如果没有明确授予（允许）对资源的访问权限，则拒绝访问。您也可明确拒绝对资源的访问，这样可确保用户无法访问该资源，即使有其他策略授予了访问权限的情况下也是如此。
- 主体 – 在基于身份的策略（IAM policy）中，附加了策略的用户是隐式主体。对于基于资源的策略，您可以指定要接收权限的用户、账户、服务或其他实体（仅适用于基于资源的策略）。

有关 IAM 策略语法和说明的更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略参考。

额关 EventBridge API 操作及其适用资源的信息，请参阅 Amazon EventBridge 权限参考。

在策略中指定条件

当您授予权限时，可使用访问策略语言来指定规定策略何时生效的条件。例如，您可能希望策略仅在特定日期后应用。有关使用策略语言指定条件的更多信息，请参阅 IAM 用户指南中的条件。

要定义条件，可以使用条件键。有 AWS 条件键和 EventBridge 特定键，您可以根据需要使用。有关 AWS 键的完整列表，请参阅《IAM 用户指南》中的条件的可用键。有关 EventBridge 特定键的完整列表，请参阅使用 IAM 策略条件进行精细访问控制。

将基于身份的策略（IAM 策略）用于 Amazon EventBridge

基于身份的策略是附加到 IAM 身份的权限策略。

主题

适用于 EventBridge 的 AWS 托管策略

AWS通过提供由AWS创建和管理的独立 IAM policy 来满足许多常用案例的要求。托管 策略也称为预定义策略，可针对常见使用场景授予必要的权限，让您不必调查需要哪些权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

以下 AWS 托管策略是特定于 EventBridge 的，可将它们附加到账户中的用户：

- AmazonEventBridgeFullAccess - 对 EventBridge 授予完全访问权限，包括 EventBridge Pipes、EventBridge 架构和 EventBridge 调度器。
- AmazonEventBridgeReadOnlyAccess - 对 EventBridge 授予只读访问权限，包括 EventBridge Pipes、EventBridge 架构和 EventBridge 调度器。

AmazonEventBridgeFullAccess 策略

AmazonEventBridgeFullAccess 策略授予使用所有 EventBridge 操作的权限，以及以下权限：

- `iam:CreateServiceLinkedRole` - EventBridge 需要此权限在您的账户中为 API 目标创建服务角色。此权限仅授予 IAM 服务在您的账户中专门为 API 目标创建角色的权限。
- `iam:PassRole` - EventBridge 需要此权限才能将调用角色传递给 EventBridge，以调用规则的目标。
- Secrets Manager 权限 - 当您通过连接资源提供凭证，对 API 目标进行授权时，EventBridge 需要这些权限来管理您账户中的密钥。

以下 JSON 展示 AmazonEventBridgeFullAccess 策略。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EventBridgeActions",
            "Effect": "Allow",
            "Action": [
                "events:*",
                "schemas:*",
                "scheduler:*",
                "pipes:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "IAMCreateServiceLinkedRoleForApiDestinations",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
AmazonEventBridgeApiDestinationsServiceRolePolicy",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "apidestinations.events.amazonaws.com"
                }
            }
        },
        {
            "Sid": "IAMCreateServiceLinkedRoleForAmazonEventBridgeSchemas",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/schemas.amazonaws.com/
AWSServiceRoleForSchemas",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "schemas.amazonaws.com"
                }
            }
        },
        {
            "Sid": "SecretsManagerAccessForApiDestinations",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:CreateSecret",
                "secretsmanager:UpdateSecret",
                "secretsmanager:DeleteSecret",
```

```
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:events!*"
        },
        {
            "Sid": "IAMPassRoleAccessForEventBridge",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "events.amazonaws.com"
                }
            }
        },
        {
            "Sid": "IAMPassRoleAccessForScheduler",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "scheduler.amazonaws.com"
                }
            }
        },
        {
            "Sid": "IAMPassRoleAccessForPipes",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "pipes.amazonaws.com"
                }
            }
        }
    ]
}
```

> **Note**
>
> 本节中的信息也适用于 `CloudWatchEventsFullAccess` 策略。但是，强烈建议您使用 Amazon EventBridge 来代替 Amazon CloudWatch Events。

AmazonEventBridgeReadOnlyAccess 策略

AmazonEventBridgeReadOnlyAccess 策略授予使用所有读取 EventBridge 操作的权限。

以下 JSON 展示 AmazonEventBridgeReadOnlyAccess 策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "events:DescribeRule",
                "events:DescribeEventBus",
                "events:DescribeEventSource",
                "events:ListEventBuses",
                "events:ListEventSources",
                "events:ListRuleNamesByTarget",
                "events:ListRules",
                "events:ListTargetsByRule",
                "events:TestEventPattern",
                "events:DescribeArchive",
                "events:ListArchives",
                "events:DescribeReplay",
                "events:ListReplays",
                "events:DescribeConnection",
                "events:ListConnections",
                "events:DescribeApiDestination",
                "events:ListApiDestinations",
                "events:DescribeEndpoint",
                "events:ListEndpoints",
                "schemas:DescribeCodeBinding",
                "schemas:DescribeDiscoverer",
                "schemas:DescribeRegistry",
                "schemas:DescribeSchema",
                "schemas:ExportSchema",
                "schemas:GetCodeBindingSource",
```

```
                "schemas:GetDiscoveredSchema",
                "schemas:GetResourcePolicy",
                "schemas:ListDiscoverers",
                "schemas:ListRegistries",
                "schemas:ListSchemas",
                "schemas:ListSchemaVersions",

                "schemas:ListTagsForResource",
                "schemas:SearchSchemas",
                "scheduler:GetSchedule",
                "scheduler:GetScheduleGroup",
                "scheduler:ListSchedules",
                "scheduler:ListScheduleGroups",
                "scheduler:ListTagsForResource",
                "pipes:DescribePipe",
                "pipes:ListPipes",
                "pipes:ListTagsForResource"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 本节中的信息也适用于 `CloudWatchEventsReadOnlyAccess` 策略。但是，强烈建议您使用 Amazon EventBridge 来代替 Amazon CloudWatch Events。

特定于 EventBridge 架构的托管策略

架构定义了发送到 EventBridge 的事件的结构。EventBridge 为 AWS 服务生成的所有事件提供架构。以下是可用的 EventBridge 架构专用 AWS 托管策略：

- AmazonEventBridgeSchemasServiceRolePolicy
- AmazonEventBridgeSchemasFullAccess
- AmazonEventBridgeSchemasReadOnlyAccess

特定于 EventBridge 调度器的托管策略

Amazon EventBridge 调度器是一个无服务器调度器，使您能够从一个中央托管服务创建、运行和管理任务。有关特定于 EventBridge 调度器的 AWS 托管策略，请参阅《EventBridge 调度器用户指南》中的适用于 EventBridge 调度器的 AWS 托管策略。

特定于 EventBridge Pipes 的托管策略

Amazon EventBridge Pipes 可将事件源与目标连接起来。在开发事件驱动型架构时，Pipes 可减少对专业知识和集成代码的需求。这有助于确保贵公司应用程序的一致性。以下是可用的 EventBridge Pipes 专用 AWS 托管策略：

- AmazonEventBridgePipesFullAccess

  提供对 Amazon EventBridge Pipes 的完全访问权限。

  > **Note**
  >
  > 本策略提供 `iam:PassRole` - EventBridge Pipes 需要此权限才能将调用角色传递给 EventBridge，用于创建和启动管道。

- AmazonEventBridgePipesReadOnlyAccess

  提供对 Amazon EventBridge Pipes 的只读访问权限。

- AmazonEventBridgePipesOperatorAccess

  提供对 Amazon EventBridge Pipes 的只读访问权限和操作员访问权限（能够停止和开始运行 Pipes）。

用于发送事件的 IAM 角色

为了将事件中继到目标，EventBridge 需要一个 IAM 角色。

创建 IAM 角色，向 EventBridge 发送事件

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 要创建 IAM 角色，请按照《IAM 用户指南》中的创建角色，向 AWS 服务委派权限中的步骤操作。在执行这些步骤时，请注意以下操作：

   - 在角色名称中，使用账户中的唯一名称。

- 在选择角色类型中，选择 AWS 服务角色，然后选择 Amazon EventBridge。这会授予 EventBridge 代入该角色的权限。
- 在附加策略中，选择 AmazonEventBridgeFullAccess。

此外，您还可以创建您自己的自定义 IAM 策略，以授予 EventBridge 操作和资源的相关权限。您可以将这些自定义策略附加到需要这些权限的 IAM 用户或组。有关 IAM 策略的更多信息，请参阅《IAM 用户指南》中的 IAM 策略概述。有关管理和创建自定义 IAM 策略的更多信息，请参阅《IAM 用户指南》中的管理 IAM 策略。

EventBridge 使用 IAM 角色访问目标所需的权限

EventBridge 目标通常需要 IAM 角色，来向 EventBridge 授予调用目标的权限。以下是各种 AWS 服务和目标的一些示例。对于其他目标，请使用 EventBridge 控制台创建一条规则并创建一个新角色，该角色将包含一条策略，具有预先配置、范围明确的权限。

Amazon SQS、Amazon SNS、Lambda、CloudWatch Logs 和 EventBridge 总线目标不使用角色，必须通过资源策略对 EventBridge 授予权限。API Gateway 目标可以使用资源策略或 IAM 角色。

如果目标是 API 目标，您指定的角色必须包含以下策略。

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

如果目标是 Kinesis 流，则用于将事件数据发送到该目标的角色必须包含以下策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:PutRecord"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

如果目标是 Systems Manager Run Command 且您为命令指定了一个或多个 `InstanceIds` 值，则您指定的角色必须包含以下策略。

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

如果目标是 Systems Manager Run Command 且您为命令指定了一个或多个标签，则您指定的角色必须包含以下策略。

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

```
            },
            {
                 "Action": "ssm:SendCommand",
                 "Effect": "Allow",
                 "Resource": [
                      "arn:aws:ssm:region:*:document/documentName"
                 ]
            }
     ]
}
```

如果目标是 AWS Step Functions 状态机，则您指定的角色必须包含以下策略。

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

如果目标是 Amazon ECS 任务，则您指定的角色必须包含以下策略。

```
{
    "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": [
            "*"
        ],
        "Condition": {
            "StringLike": {
                "iam:PassedToService": "ecs-tasks.amazonaws.com"
            }
        }
    }]
}
```

以下策略允许 EventBridge 中的内置目标代表您执行 Amazon EC2 操作。您需要使用 AWS Management Console 来创建具有内置目标的规则。

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

以下策略允许 EventBridge 将事件中继到您账户中的 Kinesis 流。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "KinesisAccess",
            "Effect": "Allow",
            "Action": [
```

```
                "kinesis:PutRecord"
            ],
            "Resource": "*"
        }
    ]
}
```

客户管理型策略示例：使用标记来控制对规则的访问权限

以下示例展示向 EventBridge 操作授予权限的用户策略。当您使用 EventBridge API、AWS 开发工具包或 AWS CLI 时，此政策生效。

您可以授予用户访问特定 EventBridge 规则的权限，但不允许他们访问其他规则。要实现此目标，请标记这两组规则，并使用 IAM 策略引用这些标签。有关标记 EventBridge 资源的更多信息，请参阅 亚马逊 EventBridge 标签。

您可以向用户授予 IAM 策略，仅允许其访问带有特定标签的规则。您可以用这个特定标签来标记所选规则，为它们授予访问权限。例如，以下策略仅向某用户授予对标签键 `Stack` 值为 `Prod` 的规则的访问权限。

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

有关使用 IAM policy 语句的更多信息，请参阅 IAM 用户指南中的使用策略控制访问。

AWS 托管策略的 Amazon EventBridge 更新

查看有关 EventBridge 的 AWS 托管策略更新的详细信息（从该服务开始跟踪这些更改开始）。有关此页面更改的自动提示，请订阅 EventBridge 文档历史记录页面的 RSS 源。

更改	说明	日期
AmazonEventBridgePipesFullAccess - 新添加的策略	EventBridge 添加了托管策略，提供使用 EventBridge Pipes 的全部权限。	2022 年 12 月 1 日
AmazonEventBridgePipesReadOnlyAccess - 新添加的策略	EventBridge 添加了托管策略，提供查看 EventBridge Pipes 信息资源的权限。	2022 年 12 月 1 日
AmazonEventBridgePipesOperatorAccess - 新添加的策略	EventBridge 添加了托管策略，提供查看 EventBridge Pipes 信息，以及开始和停止运行管道的权限。	2022 年 12 月 1 日
AmazonEventBridgeFullAccess - 更新现有策略	EventBridge 更新了策略，增加了使用 EventBridge Pipes 功能所需的权限。	2022 年 12 月 1 日
AmazonEventBridgeReadOnlyAccess - 更新现有策略	EventBridge 添加了查看 EventBridge Pipes 信息资源所需的权限。 添加了以下操作： • `pipes:DescribePipe` • `pipes:ListPipes` • `pipes:ListTagsForResource`	2022 年 12 月 1 日
CloudWatchEventsReadOnlyAccess - 更新现有策略	更新以匹配 AmazonEventBridgeReadOnlyAccess。	2022 年 12 月 1 日
CloudWatchEventsFullAccess - 更新现有策略	更新以匹配 AmazonEventBridgeFullAccess。	2022 年 12 月 1 日
AmazonEventBridgeFullAccess - 更新现有策略	EventBridge 更新了策略，增加了使用架构和调度器功能所需的权限。	2022 年 11 月 10 日

更改	说明	日期
	添加了以下权限： • EventBridge 架构注册表操作 • EventBridge 调度器操作 • 适用于 EventBridge 架构注册表的 `iam:CreateServiceLinkedRole` 权限 • 适用于 EventBridge 调度器的 `iam:PassRole` 权限	

更改	说明	日期
AmazonEventBridgeReadOnlyAccess - 更新现有策略	EventBridge 添加了查看架构和调度器信息资源所需的权限。 添加了以下操作： • schemas:DescribeCodeBinding • schemas:DescribeDiscoverer • schemas:DescribeRegistry • schemas:DescribeSchema • schemas:ExportSchema • schemas:GetCodeBindingSource • schemas:GetDiscoveredSchema • schemas:GetResourcePolicy • schemas:ListDiscoverers • schemas:ListRegistries • schemas:ListSchemas • schemas:ListSchemaVersions • schemas:ListTagsForResource	2022 年 11 月 10 日

更改	说明	日期
	• schemas:SearchSchemas • scheduler:GetSchedule • scheduler:GetScheduleGroup • scheduler:ListSchedules • scheduler:ListScheduleGroups • scheduler:ListTagsForResource	
AmazonEventBridgeReadOnlyAccess - 更新现有策略	EventBridge 添加了查看端点信息所需的权限。 添加了以下操作： • events:ListEndpoints • events:DescribeEndpoint	2022 年 4 月 7 日

更改	说明	日期
AmazonEventBridgeReadOnlyAccess - 更新现有策略	EventBridge 添加了查看连接和 API 目标信息所需的权限。 添加了以下操作： • events:DescribeConnection • events:ListConnections • events:DescribeApiDestination • events:ListApiDestinations	2021 年 3 月 4 日
AmazonEventBridgeFullAccess - 更新现有策略	EventBridge 更新了策略，增加了使用 API 目标所需的 iam:CreateServiceLinkedRole 和 AWS Secrets Manager 权限。 添加了以下操作： • secretsmanager:CreateSecret • secretsmanager:UpdateSecret • secretsmanager:DeleteSecret • secretsmanager:GetSecretValue • secretsmanager:PutSecretValue	2021 年 3 月 4 日

更改	说明	日期
EventBridge 已开始跟踪更改	EventBridge 为其 AWS 托管策略开启了改动跟踪。	2021 年 3 月 4 日

针对 Amazon EventBridge 使用基于资源的策略

当规则在 EventBridge 中运行时，将调用与该规则关联的所有目标。规则可以调用 AWS Lambda 函数、发布到 Amazon SNS 主题或将事件中继到 Kinesis 流。为了对您拥有的资源执行 API 调用，EventBridge 需要相应权限。对于 Lambda、Amazon SNS、Amazon SQS 和 Amazon CloudWatch Logs 资源，EventBridge 使用基于资源的策略。对于 Kinesis 流，EventBridge 使用基于身份的策略。

您可以使用 AWS CLI 向目标添加权限。有关如何安装和配置 AWS CLI 的信息，请参阅 AWS Command Line Interface 用户指南中的使用 AWS Command Line Interface 进行设置。

主题

- Amazon API Gateway 权限
- CloudWatch Logs 权限
- AWS Lambda 权限
- Amazon SNS 权限
- Amazon SQS 权限
- EventBridge Pipes 具体信息

Amazon API Gateway 权限

要使用 EventBridge 规则调用您的 Amazon API Gateway 端点，请在 API Gateway 端点的策略中添加以下权限。

```
{
  "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 权限

如果 CloudWatch Logs 是规则的目标，EventBridge 会创建日志流，并且 CloudWatch Logs 会将事件中的文本存储为日志条目。要允许 EventBridge 创建日志流并记录事件，CloudWatch Logs 必须包含基于资源的策略，允许 EventBridge 对 CloudWatch Logs 进行写入。

如果您使用 AWS Management Console将 CloudWatch Logs 作为规则目标添加，则会自动创建基于资源的策略。如果您使用 AWS CLI 添加目标，并且策略尚不存在，则必须创建此策略。

以下示例允许 EventBridge 写入名称以 `/aws/events/` 开头的所有日志组。如果您对这些类型的日志使用其他命名策略，请对示例进行相应调整。

```
{
    "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"
}
```

有关更多信息，请参阅《CloudWatch Logs API 参考指南》中的 PutResourcePolicy。

AWS Lambda 权限

要使用 EventBridge 规则调用您的 AWS Lambda 函数，可将以下权限添加到您的 Lambda 函数的策略中。

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

要添加以上权限，允许 EventBridge 使用 AWS CLI 调用 Lambda 函数

- 在命令提示符处，输入以下命令：

```
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"
```

有关如何设置权限，以使 EventBridge 能够调用 Lambda 函数的更多信息，请参阅《AWS Lambda 开发人员指南》中的 AddPermission 和将 Lambda 与计划事件配合使用。

Amazon SNS 权限

要允许 EventBridge 发布到 Amazon SNS 主题，请使用 `aws sns get-topic-attributes` 和 `aws sns set-topic-attributes` 命令。

> **Note**
>
> 您不能在适用于 EventBridge 的 Amazon SNS 主题策略中使用 `Condition` 块。

添加权限，允许 EventBridge 发布 SNS 主题

1. 要列出某个 SNS 主题的属性，请使用以下命令。

```
aws sns get-topic-attributes --topic-arn "arn:aws:sns:region:account-id:topic-name"
```

以下示例显示一个新 SNS 主题的结果。

```
{
    "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
\":{\"AWS\":\"*\"},\"Action\":[\"SNS:GetTopicAttributes\",\"SNS:SetTopicAttributes
\",\"SNS:AddPermission\",\"SNS:RemovePermission\",\"SNS:DeleteTopic\",
\"SNS:Subscribe\",\"SNS:ListSubscriptionsByTopic\",\"SNS:Publish\"],\"Resource
\":\"arn:aws:sns:region:account-id:topic-name\",\"Condition\":{\"StringEquals\":
{\"AWS:SourceOwner\":\"account-id\"}}}]}",
        "TopicArn": "arn:aws:sns:region:account-id:topic-name",
        "SubscriptionsPending": "0"
    }
}
```

2. 使用 JSON 转字符串转换器，将以下语句转换为字符串。

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

将语句转换为字符串后，如以下示例所示。

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

3. 将您在上一步中创建的字符串添加到 "Policy" 属性中的 "Statement" 集合中。
4. 使用 aws sns set-topic-attributes 命令设置新策略。

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

有关更多信息，请参阅《Amazon Simple Notification Service API 参考》中的 SetTopicAttributes。

Amazon SQS 权限

要允许 EventBridge 规则调用 Amazon SQS 队列，请使用 aws sqs get-queue-attributes 和 aws sqs set-queue-attributes 命令。

如果 SQS 队列的策略为空，您首先需要创建一个策略，然后可以向其中添加权限语句。新 SQS 队列的策略为空。

如果 SQS 队列已有策略，您需要复制原始策略，并将其与新语句组合，向其中添加权限语句。

添加权限，允许 EventBridge 规则调用 SQS 队列

1. 列出 SQS 队列属性。在命令提示符处，输入以下命令：

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

```
--attribute-names Policy
```

2. 添加以下语句。

```
{
      "Sid": "AWSEvents_custom-eventbus-ack-sqs-rule_dlq_sqs-rule-target",
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Action": "sqs:SendMessage",
      "Resource": "arn:aws:sqs:region:account-id:queue-name",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:events:region:account-id:rule/bus-name/rule-
name"
        }
      }
    }
```

3. 使用 JSON 转字符串转换器，将上一语句转换为字符串。在将策略转换为字符串后，如下所示。

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

4. 使用以下内容创建名为 set-queue-attributes.json 的文件。

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

5. 使用您刚刚创建的 set-queue-attributes.json 文件作为输入，设置策略属性，如以下命令所示。

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

```
--attributes file://set-queue-attributes.json
```

有关更多信息，请参阅《Amazon Simple Queue Service 开发人员指南》中的 Amazon SQS 策略示例。

EventBridge Pipes 具体信息

EventBridge Pipes 不支持基于资源的策略，也没有 API 支持基于资源的策略条件。

跨服务混淆代理问题防范

混淆代理问题是一个安全性问题，即不具有操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在 AWS 中，跨服务模拟可能会导致混淆代理问题。一个服务（呼叫服务）调用另一项服务（所谓的服务)时，可能会发生跨服务模拟。可以操纵调用服务，使用其权限以在其他情况下该服务不应有访问权限的方式对另一个客户的资源进行操作。为了防止这种情况，AWS 提供可帮助您保护所有服务的服务委托人数据的工具，这些服务委托人有权限访问账户中的资源。

我们建议在资源策略中使用 `aws:SourceArn` 或 `aws:SourceAccount` 全局条件上下文键，以限制 Amazon EventBridge 为其他服务提供的资源访问权限。如果您只希望将一个资源与跨服务访问相关联，请使用 `aws:SourceArn`。如果您想允许该账户中的任何资源与跨服务使用操作相关联，请使用 `aws:SourceAccount`。

防范混淆代理问题最有效的方法是使用 `aws:SourceArn` 全局条件上下文键和资源的完整 ARN。如果不知道资源的完整 ARN，或者正在指定多个资源，请针对 ARN 未知部分使用带有通配符字符 (*) 的 `aws:SourceArn` 全局上下文条件键。例如，`arn:aws:`*`servicename`*`:*:`*`123456789012`*`:*`。

如果 `aws:SourceArn` 值不包含账户 ID，例如 Amazon S3 存储桶 ARN，您必须使用两个全局条件上下文密钥来限制权限。

事件总线

对于 EventBridge 事件总线规则目标，`aws:SourceArn` 的值必须是规则 ARN。

以下示例演示如何使用 EventBridge 中的 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件上下文键来防范混淆代理问题。此示例用于角色信任策略，适用于 EventBridge 规则所使用的角色。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
```

```
    "Effect": "Allow",
    "Principal": {
      "Service": "events.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:events:*:123456789012:rule/myRule"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

EventBridge Pipes

对于 EventBridge Pipes，`aws:SourceArn` 的值必须是管道 ARN。

以下示例演示如何使用 EventBridge 中的 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件上下文键来防范混淆代理问题。此示例用于角色信任策略，适用于 EventBridge Pipes 所使用的角色。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "events.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:pipe:*:123456789012::pipe/example"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
```

```
}
```

适用于 Amazon EventBridge 架构的基于资源的策略

EventBridge 架构注册表支持基于资源的策略。基于资源的策略是附加到资源而不是 IAM 身份的策略。例如，在 Amazon Simple Storage Service (Amazon S3) 中，资源策略附加到 Amazon S3 桶。

有关 EventBridge 架构与基于资源的策略的更多信息，请参阅以下内容。

- Amazon EventBridge 架构 REST API 参考
- 《IAM 用户指南》中的基于身份的策略和基于资源的策略。

基于资源的策略支持的 API

对于 EventBridge 架构注册表，您可以将以下 API 与基于资源的策略配合使用。

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

向 AWS 账户授予所有支持的操作的策略示例

对于 EventBridge 架构注册表，您必须始终将基于资源的策略附加到注册表。要对架构授予访问权限，请在策略中指定架构 ARN 和注册表 ARN。

要向用户授予对 EventBridge 架构所有可用 API 的访问权限，请使用与以下代码类似的策略，将 "Principal" 替换为您要授予访问权限的账户的账户 ID。

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

向 AWS 账户授予只读操作权限的策略示例

以下示例仅向账户授予对 EventBridge 架构只读 API 的访问权限。

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

向组织授予所有操作权限的策略示例

您可以将基于资源的策略与 EventBridge 架构注册表配合使用，向组织授予访问权限。有关更多信息，请参阅 AWS Organizations 用户指南。以下示例向 ID 为 o-a1b2c3d4e5 的组织授予对架构注册表的访问权限。

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

```
}
```

Amazon EventBridge 权限参考

要在 EventBridge 策略中指定操作，请在 API 操作名称之前使用 `events:` 前缀，如以下示例所示。

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

要在单个语句中指定多项 操作，请使用逗号将它们隔开，如下所示。

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

要指定多个操作，可以使用通配符。例如，您可以指定以 "Put" 开头的所有操作，如下所示。

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

要指定所有 EventBridge API 操作，请使用 * 通配符，如下所示。

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

下表列出了您可以在 IAM 策略中指定的 EventBridge API 操作和相应行为。

EventBridge API 操作	所需的权限	描述
DeleteRule	events:DeleteRule	删除规则所必需的。
DescribeEventBus	events:DescribeEventBus	列出可以将事件写入当前账户的事件总线的账户所必需的。
DescribeRule	events:DescribeRule	列出有关规则的详细信息所必需的。
DisableRule	events:DisableRule	禁用规则所必需的。
EnableRule	events:EnableRule	启用规则所必需的。
ListRuleNamesByTarget	events:ListRuleNamesByTarget	列出与目标关联的规则所必需的。

EventBridge API 操作	所需的权限	描述
ListRules	events:ListRules	列出您账户中的所有规则所必需的。
ListTagsForResource	events:ListTagsForResource	列出与 EventBridge 资源关联的所有标记所必需的。目前，只能标记规则。
ListTargetsByRule	events:ListTargetsByRule	列出与规则关联的所有目标所必需的。
PutEvents	events:PutEvents	添加可匹配到规则的自定义活动所必需的。
PutPermission	events:PutPermission	向另一个账户授予将事件写入此账户的默认事件总线的权限所必需的。
PutRule	events:PutRule	创建或更新规则所必需的。
PutTargets	events:PutTargets	将目标添加到规则所必需的。
RemovePermission	events:RemovePermission	撤销另一个账户拥有的将事件写入此账户的默认事件总线的权限所必需的。
RemoveTargets	events:RemoveTargets	从规则中删除目标所必需的。
TestEventPattern	events:TestEventPattern	针对给定事件测试事件模式所必需的。

使用 IAM 策略条件进行精细访问控制

要授予权限，可在策略语句中使用 IAM 策略语言指定条件，规定策略何时生效。例如，您可以指定仅在特定日期后应用的策略。

策略中的条件由键值对组成。条件键不区分大小写。

如果您指定了多个条件，或在单一条件中指定了多个键，必须满足所有条件或键，EventBridge 才会授予权限。如果您在单一条件中指定了具有多个值的键，只要满足其中一个值 EventBridge 就会授予权限。

指定条件时，您也可以使用占位符或策略变量。有关更多信息，请参阅《IAM 用户指南》中的策略变量。有关使用 IAM 策略语言指定条件的更多信息，请参阅《IAM 用户指南》中的条件。

默认情况下，IAM 用户和角色无法访问您账户中的事件。要访问事件，用户必须获得 `PutRule` API 操作的授权。如果授权 IAM 用户或角色执行 `events:PutRule` 操作，他们可以创建匹配特定事件的规则。但是，要使该规则发挥作用，用户还必须拥有 `events:PutTargets` 操作的权限，因为如果您希望规则的作用不仅仅是发布 CloudWatch 指标，还必须向规则中添加目标。

可以在 IAM 用户或角色的策略语句中提供条件，允许该用户或角色创建仅匹配一组特定源和事件类型的规则。要对特定源和事件类型授予访问权限，请使用 `events:source` 和 `events:detail-type` 条件键。

同样，您可以在 IAM 用户或角色的策略语句中提供条件，允许该用户或角色创建规则，仅匹配您账户中的特定资源。要对特定资源授予访问权限，请使用 `events:TargetArn` 条件键。

以下示例是一个策略，允许用户使用对 `PutRule` API 操作的拒绝声明，访问除 EventBridge 中的 Amazon EC2 事件之外的所有事件。

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

```
            }
        }
    ]
}
```

EventBridge 条件键

下表显示了可以在 EventBridge 的策略中使用的条件键和键值对。

条件键	键值对	评估类型
aws:SourceAccount	`aws:SourceArn` 指定的规则所在的账户。	Account Id、Null
aws:SourceArn	发送事件的规则的 ARN。	ARN，Null
events:creatorAccount	`"events:creatorAccount":"` *`creatorAccount`* `"` 对于 *`creatorAccount`* ，请使用创建规则的账户的账户 ID。使用此条件授权 API 调用来自特定账户的规则。	creatorAccount、Null
events:detail-type	`"events:detail-type":"` *`detail-type`* `"` 其中，*`detail-type`* 为事件的 detail-type 字段的文字字符串，例如 `"AWS API Call via CloudTrail"` 和 `"EC2 Instance State-change Notification"` 。	Detail Type，Null
events: detail.eventTypeCode	`"events:detail.eventTypeCode":"` *`eventTypeCode`* `"` 对于 *`eventTypeCode`* ，针对事件的 detail.eventTypeCo	eventTypeCode，Null

条件键	键值对	评估类型
	de 字段使用文字字符串，例如 "AWS_ABUSE_DOS_REPORT" 。	
events: detail.service	"events:detail.service":" *service*" 对于 *service*，针对事件的 detail.service 字段使用文字字符串，例如 "ABUSE"。	service，Null
events: detail.userIdentity.principalId	"events:detail.userIdentity.principalId":" *principal-id* " 对于 *principal-id* ，针对事件的 detail.userIdentity.principalId 字段使用文字字符串，而 detail-type 为 "AWS API Call via CloudTrail" ，例如 "AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName." 。	Principal Id，Null
events:eventBusInvocation	"events:eventBusInvocation":" *boolean*" 对于 *boolean*，如果规则将事件发送到目标，而该目标是另一个账户中的事件总线，请使用 true。使用 PutEvents API 调用时，使用 false。	eventBusInvocation, Null
events:ManagedBy	由AWS服务内部使用。如果某个规则由 AWS 服务代表您创建，此值是创建规则的服务主体名称。	不要在客户策略中使用。

条件键	键值对	评估类型
events:source	"events:source":" *source* " 使用 *source* 作为事件源字段的文字字符串，例如 "aws.ec2" 或 "aws.s3"。有关 *source* 的更多可能值，请参阅 来自 AWS 服务的事件中的示例事件。	Source，Null
events:TargetArn	"events:TargetArn" :" *target-arn* " 对于 *target-arn* ，使用规则目标的 ARN，例如 "arn:aws:lambda:*:*:function:*" 。	ArrayOfARN, Null

有关适用于 EventBridge 的策略语句示例，请参阅管理对 Amazon EventBridge 资源的访问权限。

主题

- EventBridge Pipes 具体信息
- 示例：使用 creatorAccount 条件
- 示例：使用 eventBusInvocation 条件
- 示例：限制对特定源的访问
- 示例：定义可在事件模式中分别使用的多个源
- 示例：定义可在事件模式中使用的 Source 和 DetailType
- 示例：确保在事件模式中定义源
- 示例：在包含多个源的事件模式中定义允许的源的列表
- 示例：由 detail.service 限制 PutRule 访问权限
- 示例：由 detail.eventTypeCode 限制 PutRule 访问权限
- 示例：确保仅允许来自特定 PrincipalId 的 API 调用 AWS CloudTrail 事件
- 示例：限制对目标的访问

EventBridge Pipes 具体信息

EventBridge Pipes 不支持任何额外的 IAM 策略条件键。

示例：使用 **creatorAccount** 条件

以下示例策略语句说明如何在策略中使用 creatorAccount 条件，仅当指定为 creatorAccount 的账户是创建规则的账户时，才允许创建规则。

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

示例：使用 **eventBusInvocation** 条件

eventBusInvocation 指示调用是来自跨账户目标还是 PutEvents API 请求。如果调用源于包含跨账户目标的规则，例如目标为另一账户中的事件总线，则该值为 true。如果调用源于 PutEvents API 请求，该值为 false。以下示例表示来自跨账户目标的调用。

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

```
        "BoolIfExists": {
          "events:eventBusInvocation": "true"
        }
      }
    }
  ]
}
```

示例：限制对特定源的访问

以下示例策略可附加到一个 IAM 用户。策略 A 允许所有事件的 PutRule API 操作，而策略 B 仅在所创建规则的事件模式与 Amazon EC2 事件匹配时才允许 PutRule 操作。

策略 A：允许所有事件

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

策略 B：仅允许来自 Amazon EC2 的事件

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

```
        }
    ]
}
```

EventPattern 是 PutRule 的必需参数。因此，如果具有策略 B 的用户使用类似于下面的事件模式调用 PutRule：

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

将创建规则，因为策略允许此特定源，即 "aws.ec2"。但是，如果具有策略 B 的用户使用类似下面的事件模式调用 PutRule，则将拒绝该规则创建操作，因为策略不允许此特定源：也就是 "aws.s3"。

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

实质上，仅允许具有策略 B 的用户创建与源自 Amazon EC2 的事件匹配的规则，因此他们只能访问 Amazon EC2 中的事件。

有关策略 A 和策略 B 的比较，请参阅下表。

事件模式	策略 A 允许的	策略 B 允许的
`{ "source": [ "aws.ec2" ] }`	是	是
`{ "source": [ "aws.ec2", "aws.s3" ] }`	是	否（不允许源 aws.s3）
`{ "source": [ "aws.ec2" ],`	是	是

事件模式	策略 A 允许的	策略 B 允许的

<pre>"detail-type": ["EC2 Instance State-change Notification"]
}</pre>		

<pre>{
"detail-type": ["EC2 Instance State-change Notification"]
}</pre>
	是	否 (必须指定源)

示例：定义可在事件模式中分别使用的多个源

以下策略允许 IAM 用户或角色创建一条规则，EventPattern 中的源是 Amazon EC2 或 Amazon ECS。

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

下表显示此策略允许或拒绝的一些事件模式的示例。

事件模式	策略允许
`{ "source": [ "aws.ec2" ] }`	是
`{ "source": [ "aws.ecs" ] }`	是
`{ "source": [ "aws.s3" ] }`	否
`{ "source": [ "aws.ec2", "aws.ecs" ] }`	否
`{ "detail-type": [ "AWS API Call via CloudTrail" ] }`	否

示例：定义可在事件模式中使用的 Source 和 DetailType

以下策略仅允许来自 aws.ec2 源，且 DetailType 等于 EC2 instance state change notification 的事件。

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

下表显示此策略允许或拒绝的一些事件模式的示例。

事件模式	策略允许
`{ "source": [ "aws.ec2" ] }`	否
`{ "source": [ "aws.ecs" ] }`	否
`{ "source": [ "aws.ec2" ], "detail-type": [ "EC2 Instance State-change Notification" ] }`	是
`{ "source": [ "aws.ec2" ], "detail-type": [ "EC2 Instance Health Failed" ] }`	否
`{`	否

事件模式	策略允许

<pre>"detail-type": ["EC2 Instance State-change Notification"]
}</pre>	

示例：确保在事件模式中定义源

以下策略仅允许用户创建 EventPatterns 具有源字段的规则。使用此策略，IAM 用户或角色不能创建 EventPattern 不提供特定源的规则。

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

下表显示此策略允许或拒绝的一些事件模式的示例。

事件模式	策略允许

<pre>{
"source": ["aws.ec2"],
"detail-type": ["EC2 Instance State-change Notification"]
}</pre>
	是

事件模式	策略允许

<pre>{
"source": ["aws.ecs",
"aws.ec2"]
}</pre>
	是

<pre>{
"detail-type": ["EC2
Instance State-change Notificat
ion"]
}</pre>
	否

示例：在包含多个源的事件模式中定义允许的源的列表

以下策略允许用户创建 EventPatterns 具有多个源的规则。事件模式中的每个源必须是条件中提供的列表的成员。在使用 ForAllValues 条件时，请确保定义条件列表中的至少一个项。

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

下表显示此策略允许或拒绝的一些事件模式的示例。

事件模式	策略允许
`{ "source": [ "aws.ec2" ] }`	是
`{ "source": [ "aws.ec2", "aws.s3" ] }`	是
`{ "source": [ "aws.ec2", "aws.autoscaling" ] }`	否
`{ "detail-type": [ "EC2 Instance State-change Notification" ] }`	否

示例：由 **detail.service** 限制 **PutRule** 访问权限

您可以限制 IAM 用户或角色仅为 events:details.service 字段中具有特定值的事件创建规则。events:details.service 的值并非一定是 AWS 服务的名称。

当处理 AWS Health 中与安全性或滥用相关的事件时，此策略条件有所帮助。通过使用此策略条件，您可以限制只有需要查看敏感警报的那些用户才能访问它们。

例如，以下策略允许仅为 events:details.service 的值为 ABUSE 的事件创建规则。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutRuleEventsWithDetailServiceEC2",
            "Effect": "Allow",
```

```
            "Action": "events:PutRule",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "events:detail.service": "ABUSE"
                }
            }
        }
    ]
}
```

示例：由 **detail.eventTypeCode** 限制 **PutRule** 访问权限

您可以限制 IAM 用户或角色仅为 events:details.eventTypeCode 字段中具有特定值的事件创建规则。当处理 AWS Health 中与安全性或滥用相关的事件时，此策略条件有所帮助。通过使用此策略条件，您可以限制只有需要查看敏感警报的那些用户才能访问它们。

例如，以下策略允许仅为 events:details.eventTypeCode 的值为 AWS_ABUSE_DOS_REPORT 的事件创建规则。

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

示例：确保仅允许来自特定 **PrincipalId** 的 API 调用 AWS CloudTrail 事件

所有 AWS CloudTrail 事件的 detail.userIdentity.principalId 路径中都有执行 API 调用的用户的 PrincipalId。使用 events:detail.userIdentity.principalId 条件键，您可以仅允许 IAM 用户或角色访问来自特定账户的 CloudTrail 事件。

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutRuleOnlyForCloudTrailEventsWhereUserIsASpecificIAMUser",
            "Effect": "Allow",
            "Action": "events:PutRule",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "events:detail-type": [ "AWS API Call via CloudTrail" ],
                    "events:detail.userIdentity.principalId":
 [ "AIDAJ45Q7YFFAREXAMPLE" ]
                }
            }
        }
    ]
}
```

下表显示此策略允许或拒绝的一些事件模式的示例。

事件模式	策略允许
`{ "detail-type": [ "AWS API Call via CloudTrail" ] }`	否
`{ "detail-type": [ "AWS API Call via CloudTrail" ], "detail.userIdentity.principalId": [ "AIDAJ45Q7YFFAREXAMPLE" ] }`	是
`{ "detail-type": [ "AWS API Call via CloudTrail" ],`	否

事件模式	策略允许

<pre>"detail.userIdentity.principalId": ["AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName"]
}</pre>	

示例：限制对目标的访问

如果 IAM 用户或角色具有 events:PutTargets 权限，他们就可以在相同账户下将任何目标添加到他们有权访问的规则。以下策略仅限用户将目标添加到特定规则：账户 123456789012 下的 MyRule。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutTargetsOnASpecificRule",
            "Effect": "Allow",
            "Action": "events:PutTargets",
            "Resource": "arn:aws:events:us-east-1:123456789012:rule/MyRule"
        }
    ]
}
```

为了限制可以添加到规则的目标，请使用 events:TargetArn 条件密钥。您可以将目标限制为 Lambda 函数，如以下示例中所示。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutTargetsOnASpecificRuleAndOnlyLambdaFunctions",
            "Effect": "Allow",
            "Action": "events:PutTargets",
            "Resource": "arn:aws:events:us-east-1:123456789012:rule/MyRule",
            "Condition": {
                "ArnLike": {
                    "events:TargetArn": "arn:aws:lambda:*:*:function:*"
                }
```

```
            }
        }
    ]
}
```

将服务相关角色用于 EventBridge

Amazon EventBridge 使用 AWS Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种独特类型的 IAM 角色，它与 EventBridge 直接相关。服务相关角色由 EventBridge 预定义，并包含该服务代表您调用其他 AWS 服务所需的一切权限。

主题

- 使用角色为 API 目标创建密钥
- 使用角色进行架构发现

使用角色为 API 目标创建密钥

Amazon EventBridge 使用 AWS Identity and Access Management（IAM）服务相关角色。服务相关角色是一种独特类型的 IAM 角色，它与 EventBridge 直接相关。服务相关角色由 EventBridge 预定义，并包含该服务代表您调用其他 AWS 服务所需的一切权限。

服务相关角色使 EventBridge 的设置更轻松，因为您不必手动添加必要的权限。EventBridge 定义其服务相关角色的权限，除非另行定义，否则仅 EventBridge 可以担任其角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其他 IAM 实体的权限策略。

只有在首先删除相关资源后，您才能删除服务相关角色。这将保护您的 EventBridge 资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅与 IAM 配合使用的 AWS 服务，并查找 Service-linked roles（服务相关角色）列中显示为 Yes（是）的服务。请选择是与查看该服务的服务相关角色文档的链接。

EventBridge 的服务相关角色权限

EventBridge使用名为的服务相关角色 AWSServiceRoleForAmazonEventBridgeApiDestinations— 允许访问由EventBridge创建的 Secrets Manager 密钥。

AWSServiceRoleForAmazonEventBridgeApiDestinations 服务相关角色信任以下服务代入该角色：

- `apidestinations.events.amazonaws.com`

名为 Policy 的角色权限AmazonEventBridgeApiDestinationsServiceRole策略EventBridge允许对指定资源完成以下操作：

- 操作：`secrets created for all connections by EventBridge` 上的 `create, describe, update and delete secrets; get and put secret values`

您必须配置允许用户、组或角色创建、编辑或删除服务相关角色的权限。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

为 EventBridge 创建服务相关角色

您无需手动创建服务相关角色。当您在AWS Management Console、或 AWS API 中创建连接时，EventBridge会为您创建服务相关角色。AWS CLI

> **Important**
>
> 如果您在其他使用此角色支持的功能的服务中完成某个操作，此服务相关角色可以出现在您的账户中。如果您在 2021 年 2 月 11 日EventBridge服务开始支持服务相关角色之前使用该服务，则在您的账户中EventBridge创建了该AWSServiceRoleForAmazonEventBridgeApiDestinations角色。要了解更多信息，请参阅我的 AWS 账户中出现新角色。

如果您删除该服务相关角色，然后需要再次创建，您可以使用相同流程在账户中重新创建此角色。创建连接时，EventBridge会再次为您创建服务相关角色。

为 EventBridge 编辑服务相关角色

EventBridge 不允许您编辑 AWSServiceRoleForAmazonEventBridgeApiDestinations 服务相关角色。创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。有关更多信息，请参阅《IAM 用户指南》中的编辑服务相关角色。

删除 EventBridge 的服务相关角色

如果不再需要使用某个需要服务相关角色的特征或服务，建议您删除该角色。这样就没有未被主动监控或维护的未使用实体。但是，您必须先清除您的服务相关角色，然后才能手动删除它。

清除服务相关角色

必须先删除服务相关角色使用的所有资源，然后才能使用 IAM 删除该角色。

> **Note**
>
> 如果在您试图删除资源时 EventBridge 服务正在使用该角色，则删除操作可能会失败。如果发生这种情况，请等待几分钟后重试。

删除 AWSServiceRoleForAmazonEventBridgeApiDestinations 所用的 EventBridge 资源（控制台）

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在 “集成” 下选择 “API 目的地”，然后选择 “连接” 选项卡。
3. 选择连接，然后选择删除。

删除 AWSServiceRoleForAmazonEventBridgeApiDestinations 所用的 EventBridge 资源 (AWS CLI)

- 使用以下命令：`delete-connection`。

删除 AWSServiceRoleForAmazonEventBridgeApiDestinations 所用的 EventBridge 资源 (API)

- 使用以下命令：`DeleteConnection`。

手动删除服务相关角色

使用 IAM 控制台、AWS CLI 或 AWS API 删除 AWSServiceRoleForAmazonEventBridgeApiDestinations 服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的删除服务相关角色。

EventBridge 服务相关角色的受支持区域

EventBridge 支持在服务可用的所有区域中使用服务相关角色。有关更多信息，请参阅 AWS 区域和端点。

使用角色进行架构发现

Amazon EventBridge 使用 AWS Identity and Access Management（IAM）服务相关角色。服务相关角色是一种独特类型的 IAM 角色，它与 EventBridge 直接相关。服务相关角色由 EventBridge 预定义，并包含该服务代表您调用其他 AWS 服务所需的一切权限。

服务相关角色使 EventBridge 的设置更轻松，因为您不必手动添加必要的权限。EventBridge 定义其服务相关角色的权限，除非另行定义，否则仅 EventBridge 可以担任其角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其他 IAM 实体的权限策略。

只有在首先删除相关资源后，您才能删除服务相关角色。这将保护您的 EventBridge 资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅与 IAM 配合使用的 AWS 服务，并查找 Service-linked roles（服务相关角色）列中显示为 Yes（是）的服务。请选择是与查看该服务的服务相关角色文档的链接。

EventBridge 的服务相关角色权限

EventBridge使用名为的服务相关角色 AWSServiceRoleForSchemas— 向Amazon EventBridge架构创建的托管规则授予权限。

AWSServiceRoleForSchemas 服务相关角色信任以下服务代入该角色：

- `schemas.amazonaws.com`

名为的角色权限策略AmazonEventBridgeSchemasServiceRolePolicyEventBridge允许对指定资源完成以下操作：

- 操作：`all managed rules created by EventBridge` 上的 `put, enable, disable, and delete rules; put and remove targets; list targets per rule`

您必须配置允许用户、组或角色创建、编辑或删除服务相关角色的权限。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

为 EventBridge 创建服务相关角色

您无需手动创建服务相关角色。当您在AWS Management Console、或 AWS API 中进行架构发现时，EventBridge会为您创建与服务相关的角色。AWS CLI

> **Important**
>
> 如果您在其他使用此角色支持的功能的服务中完成某个操作，此服务相关角色可以出现在您的账户中。如果您在 2019 年 11 月 27 日EventBridge服务开始支持服务相关角色之前使用该服务，则在您的账户中EventBridge创建了该AWSServiceRoleForSchemas角色。要了解更多信息，请参阅我的 AWS 账户中出现新角色。

如果您删除该服务相关角色，然后需要再次创建，您可以使用相同流程在账户中重新创建此角色。当您进行架构发现时，EventBridge会再次为您创建服务相关角色。

为 EventBridge 编辑服务相关角色

EventBridge 不允许您编辑 AWSServiceRoleForSchemas 服务相关角色。创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。有关更多信息，请参阅《IAM 用户指南》中的编辑服务相关角色。

删除 EventBridge 的服务相关角色

如果不再需要使用某个需要服务相关角色的特征或服务，建议您删除该角色。这样就没有未被主动监控或维护的未使用实体。但是，您必须先清除您的服务相关角色，然后才能手动删除它。

清除服务相关角色

必须先删除服务相关角色使用的所有资源，然后才能使用 IAM 删除该角色。

> **Note**
>
> 如果在您试图删除资源时 EventBridge 服务正在使用该角色，则删除操作可能会失败。如果发生这种情况，请等待几分钟后重试。

删除 AWSServiceRoleForSchemas 所用的 EventBridge 资源（控制台）

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在 “总线” 下，选择 “事件总线”，然后选择事件总线。
3. 选择 “停止发现”。

删除 AWSServiceRoleForSchemas 所用的 EventBridge 资源 (AWS CLI)

- 使用以下命令：`delete-discoverer`。

删除 AWSServiceRoleForSchemas 所用的 EventBridge 资源 (API)

- 使用以下命令：`DeleteDiscoverer`。

手动删除服务相关角色

使用 IAM 控制台、AWS CLI 或 AWS API 删除 AWSServiceRoleForSchemas 服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的删除服务相关角色。

EventBridge 服务相关角色的受支持区域

EventBridge 支持在服务可用的所有区域中使用服务相关角色。有关更多信息，请参阅 AWS 区域和端点。

使用记录 Amazon EventBridge API 调用 AWS CloudTrail

Amazon EventBridge 与AWS CloudTrail一项服务集成，该服务提供用户、角色或. 所执行操作的记录 AWS 服务。 CloudTrail 将所有 API 调用捕获 EventBridge 为事件。捕获的调用包括来自 EventBridge 控制台的调用和对 EventBridge API 操作的代码调用。使用收集的信息 CloudTrail，您可以确定向哪个请求发出 EventBridge、发出请求的 IP 地址、发出请求的时间以及其他详细信息。

每个事件或日记账条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是用户凭证发出的。
- 请求是否代表 IAM Identity Center 用户发出。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

CloudTrail 在您创建账户 AWS 账户 时在您的账户中处于活动状态，并且您可以自动访问 CloudTrail 活动历史记录。 CloudTrail 事件历史记录提供了过去 90 天中记录的管理事件的可查看、可搜索、可下载且不可变的记录。 AWS 区域有关更多信息，请参阅《AWS CloudTrail 用户指南》中的 “使用 CloudTrail 事件历史记录”。查看活动历史记录不 CloudTrail收取任何费用。

要持续记录 AWS 账户 过去 90 天内的事件，请创建跟踪或 CloudTrailLake 事件数据存储。

CloudTrail 步道

跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。使用创建的所有跟踪 AWS Management Console 都是多区域的。您可以通过使用 AWS CLI创建单区域或多区域跟踪。建议创建多区域跟踪，因为您可以捕获账户 AWS 区域 中的所有活动。如果您创建单区域跟踪，则只能查看跟踪的 AWS 区域中记录的事件。有关跟踪的更多信息，请参阅《AWS CloudTrail 用户指南》中的为您的 AWS 账户创建跟踪和为组织创建跟踪。

通过创建跟踪，您可以免费将正在进行的管理事件的一份副本传送到您的 Amazon S3 存储桶，但会收取 Amazon S3 存储费用。 CloudTrail 有关 CloudTrail 定价的更多信息，请参阅AWS CloudTrail 定价。有关 Amazon S3 定价的信息，请参阅 Amazon S3 定价。

CloudTrail 湖泊事件数据存储

CloudTrail L@@ ak e 允许您对事件运行基于 SQL 的查询。 CloudTrail Lake 将基于行的 JSON 格式的现有事件转换为 Apache ORC 格式。ORC 是一种针对快速检索数据进行优化的列式存储格式。事件将被聚合到事件数据存储中，它是基于您通过应用高级事件选择器选择的条件的不可

变的事件集合。应用于事件数据存储的选择器用于控制哪些事件持续存在并可供您查询。有关 CloudTrail Lake 的更多信息，请参阅AWS CloudTrail 用户指南中的使用 AWS CloudTrail Lake。

CloudTrail 湖泊事件数据存储和查询会产生费用。创建事件数据存储时，您可以选择要用于事件数据存储的定价选项。定价选项决定了摄取和存储事件的成本，以及事件数据存储的默认和最长保留期。有关 CloudTrail 定价的更多信息，请参阅AWS CloudTrail 定价。

EventBridge 中的数据事件 CloudTrail

数据事件可提供对资源或在资源中所执行资源操作（例如，读取或写入 Amazon S3 对象）的相关信息。这些也称为数据层面操作。数据事件通常是高容量活动。默认情况下， CloudTrail 不记录数据事件。 CloudTrail 事件历史记录不记录数据事件。

记录数据事件将收取额外费用。有关 CloudTrail 定价的更多信息，请参阅AWS CloudTrail 定价。

您可以使用 CloudTrail 控制台或 CloudTrail API 操作记录 EventBridge 资源类型的数据事件。AWS CLI有关如何记录数据事件的更多信息，请参阅《AWS CloudTrail 用户指南》中的使用 AWS Management Console记录数据事件和使用 AWS Command Line Interface记录数据事件。

下表列出了您可以记录数据事件的 EventBridge 资源类型。数据事件类型（控制台）列显示要从控制 CloudTrail 台上的数据事件类型列表中选择的值。res ources.type 值列显示该`resources.type`值，您将在使用或 API 配置高级事件选择器时指定该值。 AWS CLI CloudTrail “记录到的数据 API CloudTrail” 列显示了 CloudTrail 针对该资源类型记录的 API 调用。

数据事件类型（控制台）	resources.type 值	数据 API 已登录到 CloudTrail
活动巴士	`AWS::Events::Event Bus`	• DescribeEventBus
事件总线规则	`AWS::Events::Rule`	• DeleteRule • DescribeRule • DisableRule • EnableRule • ListRuleNamesByTarget • ListRules • ListTargetsByRule • PutRule

数据事件类型（控制台）	resources.type 值	数据 API 已登录到 CloudTrail
		• PutTargets • RemoveTargets • TestEventPattern
管道	AWS::Pipes::Pipe	• CreatePipe • DeletePipe • DescribePipe • ListPipes • StartPipe • StopPipe • UpdatePipe

您可以将高级事件选择器配置为在 eventName、readOnly 和 resources.ARN 字段上进行筛选，从而仅记录那些对您很重要的事件。有关这些字段的更多信息，请参阅《AWS CloudTrail API 参考》中的 AdvancedFieldSelector。

EventBridge 中的管理事件 CloudTrail

管理事件提供有关对中的资源执行的管理操作的信息 AWS 账户。这些也称为控制层面操作。默认情况下， CloudTrail 记录管理事件。

Amazon EventBridge 将所有 EventBridge 控制平面操作记录为管理事件。有关 EventBridge 记录到的 Amazon EventBridge 控制平面操作的列表 CloudTrail，请参阅 Amazon EventBridge API 参考。

EventBridge 事件示例

事件代表来自任何来源的单个请求，包括有关所请求的 API 操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此事件不会按任何特定顺序出现。

以下示例显示了一个演示该PutRule操作 CloudTrail 的事件。

```
{
  "eventVersion":"1.03",
  "userIdentity":{
    "type":"Root",
    "principalId":"123456789012",
```

```
    "arn":"arn:aws:iam::123456789012:root",
    "accountId":"123456789012",
    "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
    "sessionContext":{
      "attributes":{
      "mfaAuthenticated":"false",
      "creationDate":"2015-11-17T23:56:15Z"
      }
    }
  },
  "eventTime":"2015-11-18T00:11:28Z",
  "eventSource":"events.amazonaws.com",
  "eventName":"PutRule",
  "awsRegion":"us-east-1",
  "sourceIPAddress":"AWS Internal",
  "userAgent":"AWS CloudWatch Console",
  "requestParameters":{
    "description":"",
    "name":"cttest2",
    "state":"ENABLED",
    "eventPattern":"{\"source\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance State-
change Notification\"]}",
    "scheduleExpression":""
  },
  "responseElements":{
    "ruleArn":"arn:aws:events:us-east-1:123456789012:rule/cttest2"
  },
  "requestID":"e9caf887-8d88-11e5-a331-3332aa445952",
  "eventID":"49d14f36-6450-44a5-a501-b0fdcdfaeb98",
  "eventType":"AwsApiCall",
  "apiVersion":"2015-10-07",
  "recipientAccountId":"123456789012"
}
```


有关 CloudTrail 录音内容的信息，请参阅《AWS CloudTrail 用户指南》中的CloudTrail录制内容。

CloudTrail Pip EventBridge es 采取的操作的日志条目

EventBridge 从源读取事件、调用丰富内容或调用目标时，Pipes 会扮演提供的 IAM 角色。对于与您的账户中针对所有充实内容、目标以及亚马逊 SQS、Kinesis 和 DynamoDB 来源采取的操作相关的 CloudTrail 条目，和字段将包括。`sourceIPAddress` `invokedBy` `pipes.amazonaws.com`

所有扩充内容、目标以及 Amazon SQS、Kinesis 和 DynamoDB 来源的示例 CloudTrail 日志条目

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "...",
    "arn": "arn:aws:sts::111222333444:assumed-role/...",
    "accountId": "111222333444",
    "accessKeyId": "...",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "...",
        "arn": "...",
        "accountId": "111222333444",
        "userName": "userName"
      },
      "webIdFederationData": {},
      "attributes": {
        "creationDate": "2022-09-22T21:41:15Z",
        "mfaAuthenticated": "false"
      }
    },
    "invokedBy": "pipes.amazonaws.com"
  },
  "eventTime": ",,,",
  "eventName": "...",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "pipes.amazonaws.com",
  "userAgent": "pipes.amazonaws.com",
  "requestParameters": {
    ...
  },
  "responseElements": null,
  "requestID": "...",
  "eventID": "...",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "...",
  "eventCategory": "Management"
}
```

对于所有其他来源， CloudTrail 日志条目的sourceIPAddress字段将具有动态 IP 地址，不应依赖该字段进行任何集成或事件分类。此外，这些条目没有 invokedBy 字段。

所有其他来源的示例 CloudTrail 日志条目

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    ...
  },
  "eventTime": ",,,",
  "eventName": "...",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "Python-httplib2/0.8 (gzip)",
}
```

Amazon EventBridge 中的合规性验证

作为多个 AWS 合规性计划的一部分，第三方审核机构（例如 SOC、PCI、FedRAMP 和 HIPAA）将评测 AWS 服务的安全性与合规性。

有关特定合规性计划范围内的 AWS 服务列表，请参阅合规性计划范围内的 AWS 服务。有关常规信息，请参阅AWS合规性计划。

您可以使用 AWS Artifact 下载第三方审计报告。有关更多信息，请参阅下载 AWS Artifact 中的报告。

您使用 EventBridge 的合规性责任取决于您数据的敏感度、公司的合规性目标以及适用的法律法规。AWS 提供以下资源来帮助满足合规性：

- 安全性与合规性 Quick Start 指南 - 在 AWS 上部署基于安全性和合规性的基准环境的架构注意事项和步骤。
- 设计符合 HIPAA 安全性和合规性要求的架构白皮书 - 公司如何使用 AWS 创建符合 HIPAA 标准的应用程序。
- AWS 合规性资源 - 一组业务手册和指南。
- 《AWS Config 开发人员指南》中的利用规则评估资源 - AWS Config 评估您的资源配置遵循内部实践、行业指南和法规的情况的相关信息。
- AWS Security Hub – AWS中安全状况的全面视图，可帮助您检查是否符合安全行业标准和最佳实践。

Amazon EventBridge 恢复能力

AWS全球基础设施围绕AWS区域和可用区构建。AWS区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关AWS区域和可用区的更多信息，请参阅AWS全球基础设施。

Amazon EventBridge 中的基础设施安全性

作为一项托管式服务，Amazon EventBridge 受 AWS 全球网络安全保护。有关 AWS 安全服务以及 AWS 如何保护基础架构的信息，请参阅 AWS 云安全。要按照基础设施安全最佳实践设计您的 AWS 环境，请参阅《安全性支柱 AWS Well-Architected Framework》中的 基础设施保护。

您可以使用 AWS 发布的 API 调用通过网络访问 EventBridge。客户端必须支持以下内容：

- 传输层安全性协议（TLS）我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密 (PFS) 的密码套件，例如 DHE（Ephemeral Diffie-Hellman）或 ECDHE（Elliptic Curve Ephemeral Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service (AWS STS) 生成临时安全凭证来对请求进行签名。

您可以从任何网络位置调用这些 API 操作，还可以在 EventBridge 中使用基于资源的访问策略，其中可以包含基于源 IP 地址的限制。您还可以使用 EventBridge 策略，控制来自特定 Amazon Virtual Private Cloud (Amazon VPC) 端点或特定 VPC 的访问。事实上，这隔离了在 AWS 网络中仅从特定 VPC 到给定 EventBridge 资源的网络访问。

Amazon EventBridge 中的配置和漏洞分析

配置和 IT 控制是AWS和您（我们的客户）之间的共同责任。有关更多信息，请参阅AWS责任共担模型。

监控亚马逊 EventBridge

EventBridge CloudWatch 每分钟向 Amazon 发送指标，从匹配的事件数到规则调用目标的次数，应有尽有。

以下视频回顾了通过以下方式监控和审计 EventBridge 行为 CloudWatch：监控和审计事件

主题

- EventBridge 指标
- EventBridge 指标的维度

EventBridge 指标

AWS/Events 命名空间包括以下指标。

对于使用计数作为单位的指标，求和 SampleCount 往往是最有用的统计数据。

仅指定RuleName维度的指标指的是默认的事件总线。同时指定EventBusName和RuleName维度的指标指的是自定义事件总线。

指标	描述
DeadLetterInvocations	未作为事件响应而调用规则目标的次数。其中包括将导致再次运行同一规则从而引发无限循环的调用。 有效尺寸： RuleName 单位：计数
Events	摄取的合作伙伴事件的数量。 EventBridge 有效尺寸： EventSourceName 单位：计数
FailedInvocations	永久失败的调用次数。其中不包括重试的调用，或重试尝试后成功的调用。也不会计算计入 DeadLetterInvocations 的失败调用。

指标	描述
	Note EventBridge 只有当该指标不为零时，CloudWatch 才会将其发送到。 有效尺寸： RuleName 单位：计数
Invocations	规则为响应事件而调用目标的次数。其中包括成功和失败的调用，但不包括在永久失败之前被阻止或重试的尝试。不包括 DeadLetterInvocations 。 Note EventBridge 只有当该指标不为零时，CloudWatch 才会将其发送到。 有效尺寸：无，RuleName 单位：计数
InvocationAttempts	EventBridge 尝试调用目标的次数。 有效维度：无 单位：计数
InvocationsCreated	为响应每个事件而创建的调用总数。 此指标通常用于监控每秒EventBridge 事务处理服务配额中调用次数限制的利用率。 有效维度：无 单位：计数

指标	描述
InvocationsFailedToBeSentToDlq	无法移动到死信队列中的调用数量。权限错误、资源不可用或大小限制可能会致发生死信队列错误。 **Note** EventBridge 只有当该指标不为零时， CloudWatch 才会将其发送到。 有效尺寸： RuleName 单位：计数
IngestiontoInvocationCompleteLatency	从事件接收到完成第一次成功调用尝试所花费的时间。 有效尺寸： EventBusName，无， RuleName 单位：毫秒
IngestiontoInvocationStartLatency	处理事件的时间，从事件被摄取到目标的第一次调用 EventBridge 来衡量。 有效尺寸： EventBusName，无， RuleName 单位：毫秒
InvocationsSentToDlq	移动到死信队列的调用数量。 **Note** EventBridge 只有当该指标不为零时， CloudWatch 才会将其发送到。 有效尺寸： RuleName 单位：计数

指标	描述
MatchedEvents	如果指定了 EventBusName 或 EventSourceName ，则为与任何规则匹配的事件数。如果 RuleName 指定，则为与特定规则匹配的事件数。 有效尺寸： EventBusName、 RuleName、 EventSourceName 单位：计数
RetryInvocationAttempts	重试目标调用的次数。 Note EventBridge 只有当该指标不为零时， CloudWatch 才会将其发送到。 有效维度：无 单位：计数
SuccessfulInvocationAttempts	成功调用目标的次数。 有效维度：无 单位：计数
ThrottledRules	规则执行被节流的次数。对这些规则的调用可能会延迟。 有关更多信息，请参阅 ???中的调用每秒事务数节流限制。 有效尺寸： EventBusName，无， RuleName 单位：计数

指标	描述
TriggeredRules	已运行并与任何事件匹配的规则数量。 在触发规则 CloudWatch 之前，您不会在中看到此指标。 有效尺寸： EventBusName，无， RuleName 单位：计数

EventBridge PutEvents 指标

AWS/Events 命名空间包括以下指标，它们与 PutEvents API 请求相关。

对于使用计数作为单位的指标，求和 SampleCount 往往是最有用的统计数据。

指标	描述
PutEvents ApproximateCallCount	收到的 PutEvents 请求的大致数量。 有效维度：无 单位：计数
PutEvents ApproximateFailedCount	失败 PutEvents 请求的大致数量。 有效维度：无 单位：计数
PutEvents ApproximateSuccessCount	成功 PutEvents 请求的大致数量。 有效维度：无 单位：计数
PutEvents ApproximateThrottledCount	由于节流而被拒绝的 PutEvents 请求数量。 有效维度：无

指标	描述
	单位：计数
PutEvents EntriesCount	PutEvents 请求中包含的事件条目数。 有效维度：无 单位：计数
PutEvents FailedEntriesCount	PutEvents 请求中包含但未能提取的事件条目数。 有效维度：无 单位：计数
PutEvents Latency	每个 PutEvents 请求所花费的时间。 有效维度：无 单位：毫秒
PutEvents RequestSize	PutEvents 请求的大小。 有效维度：无 单位：字节

EventBridge PutPartnerEvents 指标

AWS/Events 命名空间包括以下指标，它们与 PutPartnerEvents API 请求相关。

> **Note**
>
> EventBridge 仅包括与 SaaS 合作伙伴账户中发送事件的PutPartnerEvents请求相关的指标。有关更多信息，请参阅 ???。

对于使用计数作为单位的指标，求和 SampleCount 往往是最有用的统计数据。

指标	描述
PutPartnerEventsApproximateCallCount	收到的 PutPartnerEvents 请求的大致数量。 有效维度：无 单位：计数
PutPartnerEventsApproximateFailedCount	失败 PutPartnerEvents 请求的大致数量。 有效维度：无 单位：计数
PutPartnerEventsApproximateThrottledCount	由于节流而被拒绝的 PutPartnerEvents 请求数量。 有效维度：无 单位：计数
PutPartnerEventsApproximateSuccessCount	成功 PutPartnerEvents 请求的大致数量。 有效维度：无 单位：计数
PutPartnerEventsEntriesCount	PutPartnerEvents 请求中包含的事件条目数。 有效维度：无 单位：计数
PutPartnerEventsFailedEntriesCount	PutPartnerEvents 请求中包含但未能提取的事件条目数。 有效维度：无 单位：计数
PutPartnerEventsLatency	每个 PutPartnerEvents 请求所花费的时间。 有效维度：无

指标	描述
	单位：毫秒

EventBridge 指标的维度

EventBridge 指标具有维度或可排序的属性，如下所示。

维度	描述
EventBusName	按事件总线名称筛选可用指标。
EventSourceName	按合作伙伴事件源名称筛选可用指标。
RuleName	按规则名称筛选可用指标。

对亚马逊进行故障排除 EventBridge

您可以使用本节中的步骤对 Amazon 进行故障排除 EventBridge。

主题

我的规则已运行，但 Lambda 函数没有被调用

您的 Lambda 函数无法运行的原因之一是您没有适当的权限。

检查您对 Lambda 函数的权限

1. 使用 AWS CLI，对您的函数和 AWS 区域运行以下命令：

```
aws lambda get-policy --function-name MyFunction --region us-east-1
```

您应当看到如下输出。

```
{
    "Policy": "{\"Version\":\"2012-10-17\",
    \"Statement\":[
        {\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":\"arn:aws:events:us-
east-1:123456789012:rule/MyRule\"}},
        \"Action\":\"lambda:InvokeFunction\",
        \"Resource\":\"arn:aws:lambda:us-east-1:123456789012:function:MyFunction\",
        \"Effect\":\"Allow\",
        \"Principal\":{\"Service\":\"events.amazonaws.com\"},
        \"Sid\":\"MyId\"}
    ],
    \"Id\":\"default\"}"
}
```

2. 如果您看到以下消息：

```
A client error (ResourceNotFoundException) occurred when calling the GetPolicy
 operation: The resource you requested does not exist.
```

或者，您看到输出，但无法将 events.amazonaws.com 定位为策略中的受信任实体，请运行以下命令：

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

3. 如果输出包含 SourceAccount 字段，则需要删除该字段。SourceAccount设置会 EventBridge 阻止调用该函数。

> **Note**
>
> 如果策略不正确，则可以在 EventBridge 控制台中编辑该规则，方法是将其删除，然后将其重新添加到规则中。然后， EventBridge 控制台会为目标设置正确的权限。
> 如果您使用特定的 Lambda 别名或版本，则必须在 aws lambda get-policy 和 aws lambda add-permission 命令中添加 --qualifier 参数，如以下命令所示

```
aws lambda add-permission \
--function-name MyFunction \
--statement-id MyId \
--action 'lambda:InvokeFunction' \
--principal events.amazonaws.com \
--source-arn arn:aws:events:us-east-1:123456789012:rule/MyRule
--qualifier alias or version
```

我刚刚创建或修改了规则，但它未能匹配测试事件

在更改规则或其目标后，传入事件可能无法立即开始或停止与新的或更新后的规则的匹配。请稍等片刻，以便更改生效。

如果事件在短时间后仍然不匹配，请检查 CloudWatch 指标`TriggeredRulesInvocations`、和`FailedInvocations`是否符合您的规则。有关这些指标的更多信息，请参阅监控 Amazon EventBridge。

如果该规则旨在匹配来自 AWS 服务的事件，请执行以下操作之一：

- 使用 `TestEventPattern` 操作来测试您的规则的事件模式是否与测试事件匹配。有关更多信息，请参阅 Amazon EventBridge API 参考TestEventPattern中的。
- 在EventBridge 主机上使用沙盒。

我在 **ScheduleExpression** 中指定了时间，但我的规则没有运行

确保您在 UTC+0 时区为规则设置计划。如果 `ScheduleExpression` 正确，则按照我刚刚创建或修改了规则，但它未能匹配测试事件中的步骤操作。

我的规则未在我期望的时间运行

EventBridge 在您设置的开始时间后的一分钟内运行规则。规则一旦创建，倒计时立即开始。

> **Note**
>
> 计划规则的传送类型为 `guaranteed`，这意味着事件将在每个预期时间至少触发一次。

您可以使用 cron 表达式在指定时间调用目标。要创建每四小时的第 0 分钟运行一次的规则，请执行以下操作之一：

- 在 EventBridge 控制台中，您可以使用 cron 表达式`0 0/4 * * ? *`。
- 使用 AWS CLI，您可以使用表达式`cron(0 0/4 * * ? *)`。

例如，要使用创建每隔 4 小时运行一次的名为`TestRule`的规则 AWS CLI，请使用以下命令。

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

要每五分钟运行一次规则，请使用以下 cron 表达式。

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

使用 cron 表达式的 EventBridge 规则的最佳分辨率为一分钟。您的计划规则会在这一分钟内运行，但不会精确到第 0 秒。

由于 EventBridge 目标服务是分布式的，因此从计划规则运行到目标服务对目标资源执行操作之间可能会有几秒钟的延迟。

我的规则与 AWS 全局服务 API 调用相匹配，但它没有运行

AWS 全球服务；例如 IAM 和 Amazon Route 53 仅在美国东部（弗吉尼亚北部）地区可用，因此来自全球服务的 AWS API 调用所产生的事件仅在该地区可用。有关更多信息，请参阅 来自 AWS 服务的事件。

规则运行时，与我的规则关联的 IAM 角色被忽略

EventBridge 仅将 IAM 角色用于向 Kinesis 直播发送事件的规则。对于调用 Lambda 函数或 Amazon SNS 主题的规则，您需要提供基于资源的权限。

确保您的区域 AWS STS 终端节点已启用，以便在担任您提供的 IAM 角色时 EventBridge 可以使用这些终端节点。有关更多信息，请参阅 IAM 用户指南 AWS STS 中的在 AWS 区域激活和停用。

我的规则的事件模式应该与某资源匹配，但未匹配事件

中的大多数服务都 AWS 将冒号 (:) 或斜杠 (/) 视为亚马逊资源名称 (ARN) 中的相同字符。，但在事件模式和规则中 EventBridge 使用完全匹配的字符。请务必在创建事件模式时使用正确的 ARN 字符，以使其与需要匹配的事件中的 ARN 语法相匹配。

有些事件（例如来自 AWS CloudTrail的 API 调用事件）在资源字段中没有任何内容。

向目标传送我的事件时存在延迟

EventBridge 尝试将事件传送到目标最多 24 小时，但目标资源受限的情况除外。事件一旦到达事件流，立即会进行第一次尝试。如果目标服务出现问题，则 EventBridge自动重新安排其他交付。如果自事件到达以来已过去 24 小时，则 EventBridge 停止尝试交付该事件并在中发布`FailedInvocations`指标 CloudWatch。我们建议您设置 DLQ，存储无法成功传送到目标的事件。有关更多信息，请参阅 事件重试策略和使用死信队列。

某些事件从未传送到我的目标

如果 EventBridge 规则的目标长时间受到限制，则 EventBridge 可能无法重试交付。例如，如果未配置目标来处理传入的事件流量，而目标服务正在限制代表您发 EventBridge 出的请求，则 EventBridge 可能不会重试传送。

我的规则在回应一个事件时多次运行

在很少的情况下，同一规则可能会因一个事件或计划时间多次运行，或同一目标可能会因特定的已触发规则而被多次调用。

防止无限循环

在中 EventBridge，可以创建一条导致无限循环的规则，在这种循环中，规则会重复运行。如果您有导致无限循环的规则，请重写，使该规则执行的操作不会与同一规则匹配。

例如，检测到 Amazon S3 桶中的 ACL 更改后，会运行软件将其更改为新状态的规则会导致无限循环。解决该问题的一种方法是重写规则，使其仅匹配处于不良状态的 ACL。

无限循环可能快速导致费用超出预期。我们建议您使用预算功能，以便在费用超出您指定的限制时提醒您。有关更多信息，请参阅通过预算管理成本。

我的事件没有传送到目标 Amazon SQS 队列

如果您的 Amazon SQS 队列已加密，必须创建由客户管理的 KMS 密钥，并在您的 KMS 密钥政策中包含以下权限部分。有关更多信息，请参阅配置 AWS KMS 权限。

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

我的规则可以运行，但未看到任何消息发布到我的 Amazon SNS 主题

方案 1

您需要获得许可，才能将消息发布到 Amazon SNS 主题中。使用以下命令 AWS CLI，将 us-east-1 替换为您所在的地区并使用您的主题 ARN。

```
aws sns get-topic-attributes --region us-east-1 --topic-arn "arn:aws:sns:us-
east-1:123456789012:MyTopic"
```

要获得正确的权限，您的策略属性应类似于以下内容。

```
"{\"Version\":\"2012-10-17\",
\"Id\":\"__default_policy_ID\",
\"Statement\":[{\"Sid\":\"__default_statement_ID\",
\"Effect\":\"Allow\",
\"Principal\":{\"AWS\":\"*\"},
\"Action\":[\"SNS:Subscribe\",
\"SNS:ListSubscriptionsByTopic\",
\"SNS:DeleteTopic\",
```

```
\"SNS:GetTopicAttributes\",
\"SNS:Publish\",
\"SNS:RemovePermission\",
\"SNS:AddPermission\",
\"SNS:SetTopicAttributes\"],
\"Resource\":\"arn:aws:sns:us-east-1:123456789012:MyTopic\",
\"Condition\":{\"StringEquals\":{\"AWS:SourceOwner\":\"123456789012\"}}},{\"Sid\":
\"Allow_Publish_Events\",
\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"events.amazonaws.com\"},
\"Action\":\"sns:Publish\",
\"Resource\":\"arn:aws:sns:us-east-1:123456789012:MyTopic\"}]}"
```

如果您在策略中看不到 events.amazonaws.com 具有 Publish 权限，请先复制当前策略，然后将以下语句添加到语句列表中。

```
{\"Sid\":\"Allow_Publish_Events\",
\"Effect\":\"Allow\",\"Principal\":{\"Service\":\"events.amazonaws.com\"},
\"Action\":\"sns:Publish\",
\"Resource\":\"arn:aws:sns:us-east-1:123456789012:MyTopic\"}
```

然后使用设置主题属性 AWS CLI，使用以下命令。

```
aws sns set-topic-attributes --region us-east-1 --topic-arn "arn:aws:sns:us-
east-1:123456789012:MyTopic" --attribute-name Policy --attribute-
value NEW_POLICY_STRING
```

> **Note**
>
> 如果策略不正确，您也可以在 EventBridge 控制台中编辑该规则，方法是将其删除，然后将其重新添加到规则中。 EventBridge 为目标设置正确的权限。

方案 2

如果您的 SNS 主题已加密，您必须在 KMS 密钥政策中包含以下部分。

```
{
  "Sid": "Allow EventBridge to use the key",
  "Effect": "Allow",
  "Principal": {
```

```
    "Service": "events.amazonaws.com"
  },
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
  ],
  "Resource": "*"
}
```

EventBridge即使我删除了与亚马逊 SNS 主题相关的规则，我的 Amazon SNS 主题仍然具有访问权限

当您创建以亚马逊 SNS 为目标的规则时， EventBridge 会代表您向您的亚马逊 SNS 主题添加权限。如果您在创建规则后不久就将其删除，则 EventBridge 可能无法从您的 Amazon SNS 主题中删除该权限。如果发生此情况，您可以使用 `aws sns set-topic-attributes` 命令从该主题删除权限。有关用于发送事件的基于资源权限的信息，请参阅针对 Amazon EventBridge 使用基于资源的策略。

我可以将哪些 IAM 条件键与之搭配使用 EventBridge？

EventBridge 支持 AWS范围内的条件键（请参阅 IAM 用户指南中的 IAM 和 AWS STS 条件上下文密钥）以及中列出的密钥。使用 IAM 策略条件进行精细访问控制

我怎么知道什么时候 EventBridge 违反了规则？

当您的 EventBridge 规则被违反时，您可以使用以下警报来通知您。

创建警报以在违反规则时发出通知

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 选择创建警报。在 “按类别划分的CloudWatch 指标” 窗格中，选择事件指标。
3. 在指标列表中，选择FailedInvocations。
4. 在图形上方，依次选择 Statistic 和 Sum。
5. 对于 Period，选择一个值，例如 5 minutes。选择下一步。
6. 例如，在 “警报阈值” 下的 “名称” 中，键入警报的唯一名称myFailedRules。对于描述，键入警报的描述，例如 Rules aren't delivering events to targets。
7. 对于 is，依次选择 >= 和 1。对于 for，输入 10。

8. 在 Actions（操作）下，为 Whenever this alarm（每当此告警）选择 State is ALARM（状态为“告警”）。
9. 对于 Send notification to，选择一个现有 Amazon SNS 主题或创建一个新 SNS 主题。要创建新主题，请选择新建列表。为新的 Amazon SNS 主题键入一个名称，例如：。myFailedRules
10. 对于 Email list，请键入警报变为 ALARM 状态时将通知发送到的电子邮件地址列表 (以逗号分隔)。
11. 选择创建警报。

Amazon EventBridge 配额

EventBridge 的大部分组成都有配额。

主题

> **Note**
>
> 有关 EventBridge 调度器配额的列表，请参阅《EventBridge 调度器用户指南》中的 EventBridge 调度器的配额。

EventBridge 配额

EventBridge 具有以下配额。

服务限额控制台提供有关 EventBridge 配额的信息。除了查看默认配额外，还可以使用“服务配额”控制台为可调整的配额请求提高配额。

名称	默认值	是否可调整	描述
API 目标	每个支持的区域：3000 个	可以	每个区域每个账户的 API 目标的最大数量。
连接	每个支持的区域：3000 个	可以	每个区域每个账户的连接的最大数量。

名称	默认值	是否可调整	描述
每秒事务中的 CreateEndpoint 节流限制	每个支持的区域：每秒 5 个	否	每秒请求 CreateEndpoint API 的最大次数。额外的请求将被阻止。
每秒事务中的 DeleteEndpoint 节流限制	每个支持的区域：每秒 5 个	否	每秒请求 DeleteEndpoint API 的最大次数。额外的请求将被阻止。
端点	每个受支持的区域：100 个	可以	每个区域每个账户的端点的最大数量。
事件总线策略大小	每个支持的区域：1.0240 万个	可以	策略大小上限，以字符为单位。每当您授予对另一个账户的访问权限时，此策略大小都会增加。您可以使用 DescribeEventBus API 查看您的当前策略及其大小。
事件总线	每个受支持的区域：100 个	可以	每个账户的事件总线上限。
事件模式大小	每个支持的区域：2048 个	可以	事件模式的大小上限，以字符为单位。

名称	默认值	是否可调整	描述
调用每秒事务数节流限制	us-east-1：每秒 18750 个 us-east-2：每秒 4500 个 us-west-1：每秒 2250 个 us-west-2：每秒 18750 个 af-south-1：每秒 750 个 ap-northeast-1：每秒 2250 个 ap-northeast-3：每秒 750 个 ap-southeast-1：每秒 2250 个 ap-southeast-2：每秒 2250 个 ap-southeast-3：每秒 750 个 eu-central-1：每秒 4500 个	可以	一个调用是一个事件与规则匹配并发送到规则目标上。在达到限制后，调用将被节流；即，调用仍会发生，但会延迟。

名称	默认值	是否可调整	描述
	eu-south-1：每秒 750 个 eu-west-1：每秒 18750 个 eu-west-2：每秒 2250 个 每个其他支持的区域：每秒 1100 个		
规则数量	af-south-1：100 个 eu-south-1：100 个 每个其他支持的区域：300 个	可以	每个事件总线一个账户可以拥有的规则的最大数量

名称	默认值	是否可调整	描述
PutEvents 每秒事务数节流限制	us-east-1：每秒 10000 个 us-east-2：每秒 2400 个 us-west-1：每秒 1200 个 us-west-2：每秒 10000 个 af-south-1：每秒 400 个 ap-northeast-1：每秒 1200 个 ap-northeast-3：每秒 400 个 ap-southeast-1：每秒 1200 个 ap-southeast-2：每秒 1200 个 ap-southeast-3：每秒 400 个 eu-central-1：每秒 2400 个	可以	每秒请求 PutEvents API 的最大次数。额外的请求将被阻止。

名称	默认值	是否可调整	描述
	eu-south-1：每秒 400 个 eu-west-1：每秒 10000 个 eu-west-2：每秒 1200 个 每个其他支持的区域：每秒 600 个		
每个 API 目标的调用率	每个支持的区域：每秒 300 个	可以	每个区域每个账户每秒发送到每个 API 目标端点的调用的最大次数。一旦达到限额，未来对该 API 端点的调用将被节流。调用仍会发生，但会延迟。
每个规则的目标	每个受支持的区域：5 个	否	可与规则关联的目标的最大数量
每秒事务数节流限制	每个支持的区域：每秒 50 个	可以	每秒请求所有 EventBridge API 操作（除 PutEvents 外）的最大次数。额外的请求将被阻止
每秒事务中的 UpdateEndpoint 节流限制	每个支持的区域：每秒 5 个	否	每秒请求 UpdateEndpoint API 的最大次数。额外的请求将被阻止。

此外，EventBridge 还有以下配额，它们并不是通过服务限额控制台进行管理的。

名称	默认值	描述
事件总线	每个受支持的区域：100 个	每个账户的事件总线上限。
事件总线策略大小	每个支持的区域：10240 个	策略大小上限，以字符为单位。每当您授予对另一个账户的访问权限时，此策略大小都会增加。您可以使用 `DescribeEventBus` API 查看您的当前策略及其大小。
事件模式大小	每个支持的区域：2048 个	事件模式的大小上限，以字符为单位。 此值最多可调至 4096 个字符。如果您需要更高的上限，请联系支持人员。
包含通配符的规则	每个支持的区域：每个事件总线 30 条规则	每个账户的每个事件总线中的规则数上限，这些规则可以包含具有通配符的事件筛选器。无法调整此配额。 有关在事件模式中使用通配符的更多信息，请参阅???。
架构发现级别	每个支持的区域：255 个级别	架构发现将推断出嵌套事件的级别数上限。任何超过 255 个级别的事件都将被忽略。

按区域确定的 PutPartnerEvents 配额

如果您需要更高的上限，请联系支持人员。

区域	每秒事务数
• AWS GovCloud（美国西部） • AWS GovCloud（美国东部） • 美国东部（弗吉尼亚州北部） • 美国东部（俄亥俄州） • 美国西部（北加利福尼亚）	默认情况下，PutPartnerEvents 在所有区域的软限制为每秒 1,400 个吞吐量请求和每秒 3,600 个突增请求。

区域	每秒事务数
• 美国西部（俄勒冈州） • 非洲（开普敦） • 亚太地区（香港） • Asia Pacific (Mumbai) • 亚太地区（大阪） • 亚太地区（首尔） • 亚太地区（新加坡） • 亚太地区（悉尼） • 亚太地区（东京） • 加拿大（中部） • 欧洲地区（法兰克福） • 欧洲地区（爱尔兰） • 欧洲地区（伦敦） • 欧洲地区（米兰） • 欧洲地区（巴黎） • Europe (Stockholm) • 欧洲地区（米兰） • 南美洲（圣保罗） • 中国（宁夏） • 中国（北京）	

EventBridge 架构注册表配额

EventBridge 架构注册表具有以下配额。

服务限额控制台提供有关 EventBridge 配额的信息。除了查看默认配额外，还可以使用“服务配额”控制台为可调整的配额请求提高配额。

名称	默认值	是否可调整	描述
DiscoveredSchemas	每个支持的区域：200 个	可以	您可以在当前区域中创建的已发现架构注册表的最大架构数量
发现者	每个受支持的区域：10 个	可以	您可以在当前区域中创建的最大发现者数量。
注册表	每个受支持的区域：10 个	可以	您可以在当前区域中创建的最大注册表数量。
SchemaVersions	每个受支持的区域：100 个	可以	您可以在当前区域中创建的每个架构最大版本数量。
架构	每个受支持的区域：100 个	可以	您可以在当前区域中创建的每个注册表最大架构数量。（不包括发现的架构注册表）

EventBridge Pipes 配额

EventBridge Pipes 具有以下配额。如果您需要更高的上限，请联系支持人员。

资源	区域	默认限制
每个账户的并发管道执行数	• AWS GovCloud（美国西部） • AWS GovCloud（美国东部） • 中国（宁夏）	1000

资源	区域	默认限制
	• 中国（北京） • 亚太地区（大阪） • 非洲（开普敦） • 欧洲地区（米兰） • 美国东部（俄亥俄州） • 欧洲地区（法兰克福） • 美国西部（北加利福尼亚） • 欧洲地区（伦敦） • 亚太地区（悉尼） • 亚太地区（东京） • 亚太地区（新加坡） • 加拿大（中部） • 欧洲地区（巴黎） • 欧洲地区（斯德哥尔摩） • 南美洲（圣保罗） • 亚太地区（首尔） • 亚太地区（孟买） • 亚太地区（香港） • 中东（巴林） • 中国（宁夏） • 中国（北京） • 亚太地区（大阪） • 非洲（开普敦） • 欧洲地区（米兰）	
每个账户的并发管道执行数	• 美国东部（弗吉尼亚州北部） • 美国西部（俄勒冈州） • 欧洲地区（爱尔兰）	3000

资源	区域	默认限制
每个账户的管道数	全部	1000

亚马逊 EventBridge 标签

标签是您或 AWS 分配给 AWS 资源的自定义属性标签。在中 EventBridge，您可以为规则和事件总线分配标签。每个资源最多可以有 50 个标签。

您可以使用标签来识别和组织您的 AWS 资源。许多 AWS 服务都支持标记，因此您可以为来自不同服务的资源分配相同的标签，以表明这些资源是相关的。例如，您可以为分配给 EC2 实例的 EventBridge 规则分配相同的标签。

标签具有两个部分：

- 标签键，例如 `CostCenter`、`Environment` 或 `Project`。
  - 标签键区分大小写。
  - 最大标签键长度为 128 个 Unicode 字符 (采用 UTF-8 格式)。
  - 每个资源的每个标签键都必须是唯一的。
  - 允许使用的字符包括可用 UTF-8 格式表示的字母、数字和空格，以及以下字符：. : + = @ _ / - (连字符)。
  - 禁止在标签中使用`aws:`前缀，因为它是保留供 AWS 使用的。无法编辑或删除带此前缀的标签键或值。具有此前缀的标签不计入每个资源的标签数限制。
- 一个可选的标签值 字段，例如 `111122223333` 或 `Production`。
  - 每个标签键只能有一个值。
  - 标签值区分大小写。
  - 省略标签值与使用空字符串效果相同。
  - 最大标签值长度为 256 个 Unicode 字符 (采用 UTF-8 格式)。
  - 允许使用的字符包括可用 UTF-8 格式表示的字母、数字和空格，以及以下字符：. : + = @ _ / - (连字符)。

> **Tip**
>
> 最佳实践是，决定利用标签的策略并在所有资源类型中一致地实施该策略。例如，决定是否使用 `Costcenter`、`costcenter` 或 `CostCenter`，以及是否对所有标签使用相同的约定。

您可以使用 EventBridge 控制台、 EventBridge API 或 AWS CLI 添加、编辑或删除标签。有关更多信息，请参阅下列内容：

- TagResourceUntagResource、，以及ListTagsForResource《亚马逊 EventBridge API 参考》
- tag-resourc e、untag-resource 和 “参考” list-tags-for-resourceAWS CLI
- Resource Groups 用户指南中的使用标签编辑器

文档历史记录

下表列出了从 2019 年 7 月开始的每个版本的《Amazon EventBridge 用户指南》中的重要更改。如需对此文档更新的通知，您可以订阅 RSS 源。

更改	描述	发行日期
根据事件总线和规则生成 AWS CloudFormation 模板。	现在，您可以根据现有 Amazon EventBridge 事件总线和规则生成 AWS CloudFormation 模板。 • 从 Amazon EventBridge 事件总线生成 AWS CloudFormation 模板	2022 年 11 月 18 日
推出 EventBridge Pipes 文档。	现在，您可以创建管道，将源连接到目标，并提供可选的筛选和富集功能。 • 管道	2022 年 12 月 1 日
根据事件总线和规则生成 AWS CloudFormation 模板。	现在，您可以根据现有 Amazon EventBridge 事件总线和规则生成 AWS CloudFormation 模板。 • 从 Amazon EventBridge 事件总线生成 AWS CloudFormation 模板	2022 年 11 月 18 日
添加了 AmazonEventBridgePipesFullAccess 策略	提供对 Amazon EventBridge Pipes 的完全访问权限。 • 特定于 EventBridge Pipes 的托管策略	2022 年 12 月 1 日
添加了 AmazonEventBridgePipesReadOnlyAccess 策略。	提供对 Amazon EventBridge Pipes 的只读访问权限。 • 特定于 EventBridge Pipes 的托管策略	2022 年 12 月 1 日

更改	描述	发行日期
添加了 AmazonEventBridgePipesOperatorAccess 策略。	提供对 Amazon EventBridge Pipes 的只读访问权限和操作员访问权限（能够停止和开始运行 Pipes）。 • 特定于 EventBridge Pipes 的托管策略	2022 年 12 月 1 日
更新了 CloudWatchEventsFullAccess 策略。	更新以匹配 `AmazonEventBridgeFullAccess` 。 • AmazonEventBridgeFullAccess 策略	2022 年 12 月 1 日
更新了 CloudWatchEventsReadOnlyAccess 策略。	更新以匹配 `AmazonEventBridgeReadOnlyAccess` 。 • AmazonEventBridgeReadOnlyAccess 策略	2022 年 12 月 1 日
更新了事件模式中的内容筛选。	现在，您可以使用 `suffix`、`equals-ignore-case` 和 `$or` 筛选选项来创建事件模式。 • 在 Amazon EventBridge 事件模式中筛选内容	2022 年 11 月 14 日
更新了 AmazonEventBridgeFullAccess 策略。	增加了使用 EventBridge 架构注册表和 EventBridge 调度器所需的权限。 • AmazonEventBridgeFullAccess 策略	2022 年 11 月 10 日
更新了 AmazonEventBridgeReadOnlyAccess 策略。	现在，您可以查看 EventBridge 架构注册表和 EventBridge 调度器信息。 • AmazonEventBridgeReadOnlyAccess 策略	2022 年 11 月 10 日
更新了事件模式中的内容筛选。	现在，您可以使用 `suffix`、`equals-ignore-case` 和 `$or` 筛选选项来创建事件模式。 • 在 Amazon EventBridge 事件模式中筛选内容	2022 年 11 月 14 日

更改	描述	发行日期
更新了 AmazonEventBridgeFullAccess 策略。	增加了使用 EventBridge 架构注册表和 EventBridge 调度器所需的权限。 • AmazonEventBridgeFullAccess 策略	2022 年 11 月 10 日
更新了 AmazonEventBridgeReadOnlyAccess 策略。	现在，您可以查看 EventBridge 架构注册表和 EventBridge 调度器信息。 • AmazonEventBridgeReadOnlyAccess 策略	2022 年 11 月 10 日
更新了 AmazonEventBridgeReadOnlyAccess 策略。	现在，您可以查看端点信息。 • AmazonEventBridgeReadOnlyAccess 策略	2022 年 4 月 7 日
添加了对全局端点的支持。	Amazon EventBridge 现已支持使用全局端点帮助您的应用程序具备区域容错能力，且无需支付额外费用。要了解更多信息，请参阅以下内容： • 通过全局端点和事件复制使应用程序具备区域容错能力 • `CreateEndpoint`	2022 年 4 月 7 日
增加了对存档和事件重放的支持。	Amazon EventBridge 现已支持使用存档来存储事件，并使用事件重放来重放存档中的事件。要了解更多信息，请参阅以下内容： • 存档 Amazon EventBridge 事件. • `CreateArchive` • `StartReplay`	2020 年 11 月 5 日

更改	描述	发行日期
增加了对目标的死信队列和重试策略的支持。	Amazon EventBridge 现已支持使用死信队列，并为目标定义重试策略。要了解更多信息，请参阅以下内容： • 事件重试策略和使用死信队列. • PutTargets	2020 年 10 月 12 日
添加了对 JSONSchema Draft4 格式架构的支持。	Amazon EventBridge 现已支持 JSONSchema Draft 4 格式的架构。现在，您还可以使用 EventBridge API 导出架构。要了解更多信息，请参阅以下内容： • 亚马逊 EventBridge 架构 • EventBridge 架构注册表 API 参考中的 Export。	2020 年 9 月 28 日
适用于 EventBridge 架构注册表的基于资源的策略。	Amazon EventBridge 架构注册表现已支持基于资源的策略。有关更多信息，请参阅下列内容。 • 适用于 Amazon EventBridge 架构的基于资源的策略 • EventBridge 架构注册表 API 参考中的 Policy • 《AWS CloudFormation 用户指南》中的 RegistryPolicy 资源类型	2020 年 4 月 30 日
事件总线的标签	此版本允许您创建和管理事件总线的标签。您可以在创建事件总线时添加标签，并通过调用相关 API 来添加或管理现有标签。有关更多信息，请参阅下列内容。 • 亚马逊 EventBridge 标签 • 基于标签的策略 • TagResource • UntagResource • ListTagsForResource	2020 年 2 月 24 日

更改	描述	发行日期
提高服务配额	Amazon EventBridge 现已增加了调用和 `PutEvents` 的配额。配额因区域而异，必要时可以增加。	2020 年 2 月 11 日
添加了一个关于转换目标输入的新主题，并添加了指向 Application Auto Scaling 事件的链接。	改进了输入转换器的文档记录。 • 亚马逊 EventBridge 输入转换 • 使用输入转换器从事件中提取数据并将该数据输入到目标 • 教程：使用输入转换器自定义 EventBridge 传递给事件目标的内容 增加了指向 Application Auto Scaling 事件的链接。 • 应用程序 Auto Scaling 事件和 EventBridge • 来自 AWS 服务的事件	2019 年 12 月 20 日
基于内容的筛选		2019 年 12 月 19 日
添加了指向 Amazon Augmented AI 事件示例的链接。	添加了指向《Amazon SageMaker 开发人员指南》中 Amazon Augmented AI 主题的链接，该主题提供 Amazon Augmented AI 的示例事件。有关更多信息，请参阅下列内容。 • 使用 Amazon Augmented AI 中的事件 • 来自 AWS 服务的事件	2019 年 12 月 13 日
添加了指向 Amazon Chime 事件示例的链接。	添加了一个链接，此链接指向提供该服务的示例事件的 Amazon Chime 主题。有关更多信息，请参阅下列内容。 • 使用 EventBridge 实现 Amazon Chime 自动化 • 来自 AWS 服务的事件	2019 年 12 月 12 日

更改	描述	发行日期
Amazon EventBridge 架构	现在，您可以管理架构并为 Amazon EventBridge 中的事件生成代码绑定。有关更多信息，请参阅下列内容。 • 亚马逊 EventBridge 架构 • EventBridge 架构 API 参考 • AWS CloudFormation 中的 EventSchemas 资源类型参考	2019 年 12 月 1 日
AWS CloudFormation 对事件总线的支持	AWS CloudFormation 现在支持 EventBus 资源。它还支持 EventBusPolicy 和 Rule 资源中的 EventBusName 参数。有关更多信息，请参阅 Amazon EventBridge 资源类型参考。	2019 年 10 月 7 日
新增服务	Amazon EventBridge 的初始版本。	2019 年 7 月 11 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。