aws

사용 설명서

아마존 EventBridge

아마존 EventBridge: 사용 설명서

Table of Contents

Amazon EventBridge란 무엇인가요?

EventBridge는 이벤트를 사용하여 애플리케이션 구성 요소를 서로 연결하는 서버리스 서비스로, 확장 가능한 이벤트 기반 애플리케이션을 더 쉽게 구축할 수 있습니다. 이벤트 기반 아키텍처는 이벤트를 내보내고 이에 응답하여 함께 작동하는 느슨하게 결합된 소프트웨어 시스템을 구축하는 스타일입니다. 이벤트 기반 아키텍처는 민첩성을 높이고 안정적이고 확장 가능한 애플리케이션을 구축하는 데 도움이 될 수 있습니다.

EventBridge를 사용하여 자체 개발 애플리케이션, AWS 서비스 및 타사 소프트웨어와 같은 소스의 이벤트를 조직 전체의 소비자 애플리케이션으로 라우팅할 수 있습니다. EventBridge는 이벤트를 수집, 필터링, 변환 및 전달하는 간단하고 일관된 방법을 제공하므로 애플리케이션을 빠르게 구축할 수 있습니다.

다음은 Amazon EventBridge의 기능에 대한 간략한 정보를 제공합니다.

EventBridge에는 이벤트를 처리하는 두 가지 방법, 즉 이벤트 버스와 파이프가 있습니다.

- 이벤트 버스는 이벤트를 수신하여 0개 이상의 대상에 전달하는 라우터입니다. 이벤트 버스는 다양한 소스의 이벤트를 여러 대상으로 라우팅하는 데 적합하며, 대상으로 전달하기 전에 선택적으로 이벤트를 변환할 수 있습니다.

  다음 비디오는 이벤트 버스에 대한 종합적인 개요를 설명하는 동영상입니다.

- 파이프 EventBridge 파이프는 지점 간 통합을 위한 것으로, 파이프마다 단일 소스로부터 이벤트를 수신하여 처리하고 단일 대상으로 전달합니다. 파이프에는 대상에 전달되기 전에 고급 변환 및 이벤트 보강 지원도 포함됩니다.

파이프와 이벤트 버스는 종종 함께 사용됩니다. 일반적인 사용 사례는 이벤트 버스를 대상으로 하는 파이프를 만드는 것입니다. 이 파이프는 이벤트를 이벤트 버스로 보낸 다음 이벤트 버스가 해당 이벤트를 여러 대상으로 전송합니다. 예를 들어, 소스용 DynamoDB 스트림과 대상 이벤트 버스를 사용하여 파이프를 생성할 수 있습니다. 파이프는 DynamoDB 스트림에서 이벤트를 수신하여 이벤트 버스로 전송합니다. 그런 다음 이벤트 버스에서 지정한 규칙에 따라 이벤트를 여러 대상으로 전송합니다.

EventBridge는 Amazon CloudWatch Events의 진화입니다.

이전에는 EventBridge를 Amazon CloudWatch Events라고 했습니다. 기본 이벤트 버스와 CloudWatch Events에서 생성한 규칙은 EventBridge 콘솔에도 표시됩니다. EventBridge는 동일한 CloudWatch Events API를 사용하므로 CloudWatch Events API를 사용하는 코드는 동일하게 유지됩니다.

EventBridge는 CloudWatch Events의 기능을 기반으로 파트너 이벤트, 스키마 레지스트리, EventBridge 파이프와 같은 기능을 제공합니다. EventBridge에 추가된 새 기능은 CloudWatch Events에 추가되지 않습니다. 자세한 내용은 ??? 섹션을 참조하세요.

CloudWatch Events에서 사용하던 모든 기능을 EventBridge에서도 사용할 수 있습니다. 여기에는 다음이 포함됩니다.

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

이벤트 기능을 구축하고 확장하는 EventBridge 기능은 다음과 같습니다.

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

Amazon EventBridge 설정 및 사전 요구 사항

EventBridgeAmazon을 사용하려면 AWS 계정이 필요합니다. 계정을 사용하면 Amazon EC2와 같은 서비스를 사용하여 콘솔에서 볼 수 있는 이벤트를 생성할 수 있습니다. EventBridge 명령줄 인터페이스를 사용하여 이벤트를 볼 수 있도록 AWS Command Line Interface (AWS CLI) 를 설치 및 구성할 수도 있습니다.

주제

가입하여 AWS 계정

계정이 없는 경우 다음 단계를 완료하여 계정을 만드세요. AWS 계정

가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 여세요.
2. 온라인 지시 사항을 따르세요.

   등록 절차 중에는 전화를 받고 키패드로 인증 코드를 입력하는 과정이 있습니다.

   에 AWS 계정가입하면 AWS 계정 루트 사용자a가 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스 액세스 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 가입 절차가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 가서 내 계정(My Account)을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

등록한 AWS 계정후에는 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 보호하고 AWS IAM Identity Center활성화하고 생성하십시오 AWS 계정 루트 사용자.

보안을 유지하세요. AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 AWS Management Console소유자로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면AWS 로그인 사용 설명서의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자 (콘솔) 에 대한 가상 MFA 디바이스 활성화를 참조하십시오.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 ID 소스로 사용하는 방법에 대한 자습서는 사용 설명서의 기본값으로 IAM Identity Center 디렉터리사용자 액세스 구성을 참조하십시오. IAM Identity Center 디렉터리 AWS IAM Identity Center

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM IDentity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 사용 설명서의 AWS 액세스 포털에 로그인을 참조하십시오.AWS 로그인

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.

2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

Amazon EventBridge 콘솔에 로그인

Amazon EventBridge 콘솔에 로그인하려면

- 에 AWS Management Console 로그인하고 https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.

계정 자격 증명

루트 사용자 자격 증명을 사용하여 액세스할 EventBridge 수 있지만 대신 AWS Identity and Access Management (IAM) 계정을 사용하는 것이 좋습니다. IAM 계정을 사용하여 액세스하는 EventBridge 경우 다음 권한이 있어야 합니다.

```
{
  "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"
        }
      }
```

```
    }
  ]
}
```

자세한 정보는 인증을 참조하세요.

설정 AWS Command Line Interface

를 AWS CLI 사용하여 EventBridge 작업을 수행할 수 있습니다.

설치 및 구성 방법에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS CLI를 사용하여 설치하기를 참조하십시오. AWS Command Line Interface

리전 엔드포인트

사용할 EventBridge 기본 지역 엔드포인트를 활성화해야 합니다. 자세한 내용은 IAM 사용 설명서의 AWS 지역 활성화 및 비활성화를 AWS STS 참조하십시오.

아마존 시작하기 EventBridge

EventBridge 기본은 이벤트를 대상으로 라우팅하는 규칙을 만드는 것입니다. 이 섹션에서는 기본 규칙을 생성합니다. 특정 시나리오 및 대상에 대한 자습서는 Amazon EventBridge 자습서 섹션을 참조하세요.

아마존에서 규칙 생성 EventBridge

이벤트에 대한 규칙을 생성하려면 규칙의 이벤트 패턴과 일치하는 이벤트를 EventBridge 수신할 때 취할 조치를 지정합니다. 이벤트가 일치하면 지정된 대상으로 이벤트를 EventBridge 보내고 규칙에 정의된 작업을 트리거합니다.

AWS 계정의 AWS 서비스가 이벤트를 내보내는 경우 해당 이벤트는 항상 계정의 기본 이벤트 버스로 이동합니다. 계정 내 AWS 서비스의 이벤트와 일치하는 규칙을 작성하려면 해당 규칙을 기본 이벤트 버스에 연결해야 합니다.

AWS 서비스에 대한 규칙을 만들려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 규칙을 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙에 대해 이름과 설명을 입력하십시오.

   규칙은 동일한 지역과 동일한 이벤트 버스의 다른 규칙과 동일한 이름을 가질 수 없습니다.
5. 이벤트 버스에서 이 규칙과 연결할 이벤트 버스를 선택합니다. 이 규칙이 자신의 계정에서 발생하는 이벤트와 일치하도록 하려면 AWS 기본 이벤트 버스(default event bus)를 선택합니다. 계정의 AWS 서비스가 이벤트를 출력하면 항상 계정의 기본 이벤트 버스로 이동합니다.
6. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다.
7. 다음을 선택합니다.
8. 이벤트 소스에서 AWS 서비스를 선택합니다.
9. (선택 사항) 샘플 이벤트의 경우 이벤트 유형을 선택합니다.
10. 이벤트 패턴에서 다음을 수행합니다.
    - 템플릿을 사용하여 이벤트 패턴을 생성하려면 이벤트 패턴 양식에서 이벤트 소스 및 이벤트 유형을 선택합니다. 이벤트 유형으로 All Events를 선택하면 이 AWS 서비스에서 발생하는 모든 이벤트가 규칙과 일치합니다.

템플릿을 사용자 정의하려면 사용자 정의 패턴(JSON 편집기)(Custom pattern (JSON editor))을 선택하고 변경 사항을 작성합니다.

- 사용자 정의 이벤트 패턴을 사용하려면 사용자 정의 패턴(JSON 편집기)(Custom pattern (JSON editor))을 선택하고 이벤트 패턴을 만들 수 있습니다.

11. 다음을 선택합니다.
12. 대상 유형에서 AWS 서비스를 선택합니다.
13. 대상 선택에서 이벤트 패턴과 일치하는 이벤트가 EventBridge 감지되었을 때 정보를 보내려는 AWS 서비스를 선택합니다.
14. 표시되는 필드는 선택한 서비스에 따라 달라집니다. 필요에 따라 이 대상 유형과 관련된 정보를 입력합니다.
15. 많은 대상 유형의 경우 대상에 이벤트를 보낼 수 있는 권한이 EventBridge 필요합니다. 이러한 경우 규칙을 실행하는 데 필요한 IAM 역할을 생성할 EventBridge 수 있습니다. 다음 중 하나를 수행하십시오.

    - IAM 역할을 자동으로 생성하려면 이 특정 리소스에 대해 새 역할 생성을 선택합니다.
    - 이전에 생성한 IAM 역할을 사용하려면 기존 역할 사용을 선택하고 드롭다운 목록에서 기존 역할을 선택합니다.

16. (선택 사항)추가 설정에서 다음을 수행합니다.

    a. 최대 이벤트 기간(Maximum age of event)에 1분(00:01)에서 24시간(24:00) 사이의 값을 입력합니다.

    b. 재시도(Retry attempts)에 0에서 185 사이의 숫자를 입력합니다.

    c. 데드레터 대기열의 경우 표준 Amazon SQS 대기열을 데드레터 대기열로 사용할지 여부를 선택합니다. EventBridge 타겟으로 성공적으로 전송되지 않은 경우 이 규칙과 일치하는 이벤트를 데드레터 대기열로 전송합니다. 다음 중 하나를 수행하십시오.

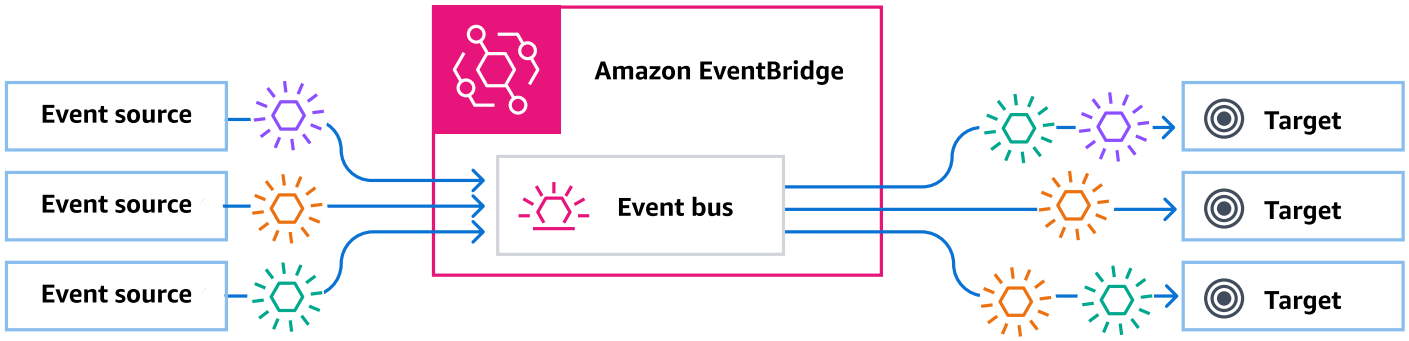
        - 배달 못한 편지 대기열을 사용하지 않으려면 없음(None)을 선택합니다.
        - 현재 AWS 계정에서 배달 못한 편지 대기열로 사용할 대기열 선택(Select an Amazon SQS queue in the current account to use as the dead-letter queue)을 선택하고 드롭다운 목록에서 사용할 대기열을 선택합니다.
        - 다른 AWS 계정의 Amazon SQS 대기열을 데드레터 대기열로 선택을 선택한 다음 사용할 대기열의 ARN을 입력합니다. 메시지를 전송할 EventBridge 권한을 부여하는 리소스 기반 정책을 대기열에 연결해야 합니다. 자세한 정보는 DLQ(Dead Letter Queue)에 권한 부여을 참조하세요.

17. (선택 사항) 이 규칙에 다른 대상을 추가하려면 다른 대상 추가를 선택합니다.

18. 다음을 선택합니다.
19. (선택 사항)규칙에 대해 하나 이상의 태그를 입력하세요. 자세한 정보는 아마존 EventBridge 태그을 참조하세요.
20. 다음을 선택합니다.
21. 규칙의 세부 정보를 검토하고 규칙 생성을 선택합니다.

아마존 EventBridge 이벤트 버스

이벤트 버스는 이벤트를 수신하여 0개 이상의 목적지 또는 대상에 전달하는 라우터입니다. 이벤트 버스는 다양한 소스의 이벤트를 여러 대상으로 라우팅하는 데 적합하며, 대상으로 전달하기 전에 선택적으로 이벤트를 변환할 수 있습니다.



이벤트 버스와 연결된 규칙은 이벤트가 도착할 때 이벤트를 평가합니다. 각 규칙은 이벤트가 규칙의 패턴과 일치하는지 확인합니다. 이벤트가 일치하면 이벤트를 EventBridge 전송합니다.

규칙을 특정 이벤트 버스에 연결하면 해당 이벤트 버스에서 수신한 이벤트에만 규칙이 적용됩니다.

> **Note**
>
> EventBridge 파이프를 사용하여 이벤트를 처리할 수도 있습니다. EventBridge 파이프는 point-to-point 통합을 위한 것으로, 각 파이프는 단일 소스로부터 이벤트를 수신하여 처리하고 단일 대상으로 전달합니다. 파이프에는 대상에 전달되기 전에 고급 변환 및 이벤트 보강 지원도 포함됩니다. 자세한 정보는 ???을 참조하세요.

주제

이벤트 버스 작동 방식

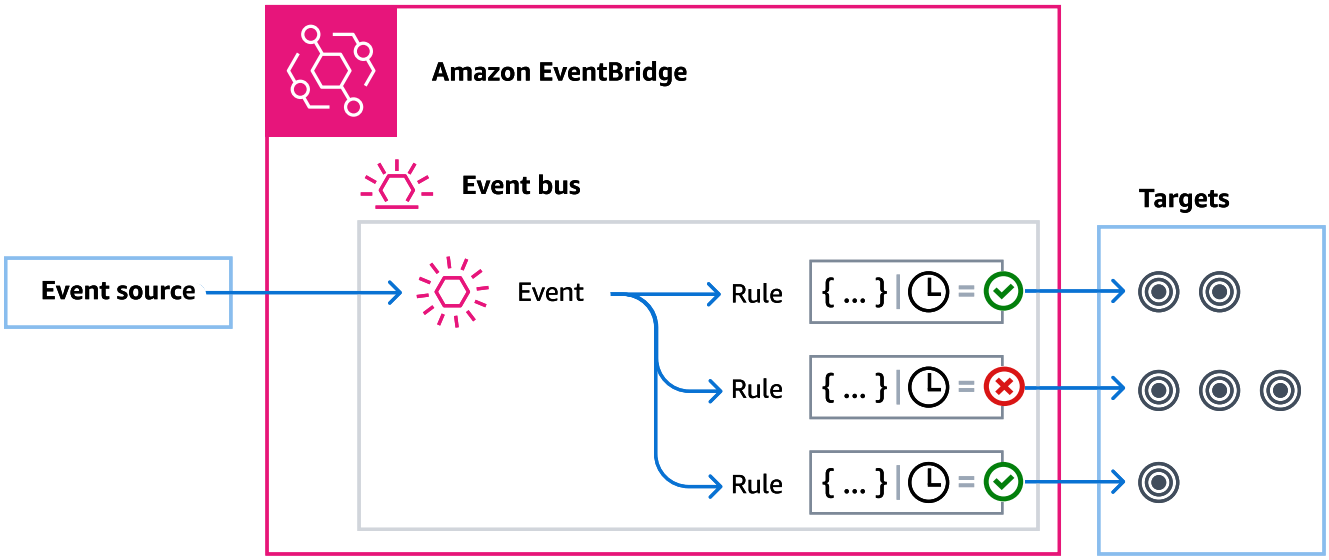
이벤트 버스를 사용하면 다양한 소스의 이벤트를 여러 목적지 또는 대상으로 라우팅할 수 있습니다.

개괄적으로 살펴볼 때 작동 방식은 다음과 같습니다.

1. AWS 서비스, 사용자 지정 애플리케이션 또는 SaaS 제공업체일 수 있는 이벤트 소스는 이벤트 버스로 이벤트를 전송합니다.
2. EventBridge 그런 다음 해당 이벤트 버스에 정의된 각 규칙을 기준으로 이벤트를 평가합니다.

   EventBridge 그런 다음 규칙과 일치하는 각 이벤트에 대해 해당 규칙에 지정된 대상으로 이벤트를 보냅니다. 선택적으로, 이벤트를 대상으로 전송하기 전에 이벤트를 EventBridge 변환해야 하는 방법을 규칙의 일부로 지정할 수도 있습니다.

   이벤트는 여러 규칙과 일치할 수 있으며 규칙마다 최대 5개의 대상을 지정할 수 있습니다. (이벤트는 어떤 규칙과도 일치하지 않을 수 있으며, 이 경우 아무 조치도 EventBridge 취하지 않습니다.)



AWS 서비스로부터 이벤트를 자동으로 수신하는 EventBridge 기본 이벤트 버스를 사용하는 예를 들어 보겠습니다.

1. `EC2 Instance State-change Notification` 이벤트에 대한 기본 이벤트 버스에 규칙을 생성합니다.

- 규칙이 Amazon EC2 인스턴스가 해당 `state`를 `running`으로 변경한 이벤트와 일치하도록 지정합니다.

  이를 위해서는 규칙을 트리거하기 위해 이벤트가 일치해야 하는 속성 및 값을 정의하는 JSON을 지정하면 됩니다. 이를 이벤트 패턴이라고 합니다.

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

- 규칙의 대상을 특정 Lambda 함수가 되도록 지정합니다.

2. Amazon EC2 인스턴스의 상태가 변경될 때마다 Amazon EC2(이벤트 소스)는 자동으로 해당 이벤트를 기본 이벤트 버스로 전송합니다.
3. EventBridge 기본 이벤트 버스로 전송된 모든 이벤트를 사용자가 만든 규칙과 비교하여 평가합니다.

   이벤트가 규칙과 일치하는 경우 (즉, Amazon EC2 인스턴스가 상태를 로 변경하는 이벤트인 경우`running`) 지정된 대상으로 이벤트를 EventBridge 보냅니다. 이 경우에는 Lambda 함수입니다.

다음 동영상에서는 이벤트 버스의 정의와 기능을 설명합니다. 이벤트 버스의 정의

다음 동영상에서는 다양한 이벤트 버스와 사용 시기에 대해 설명합니다. 이벤트 버스의 차이점

Amazon EventBridge 이벤트 버스 컨셉

이벤트 버스에 구축된 이벤트 기반 아키텍처의 주요 구성 요소를 자세히 살펴보겠습니다.

이벤트 버스

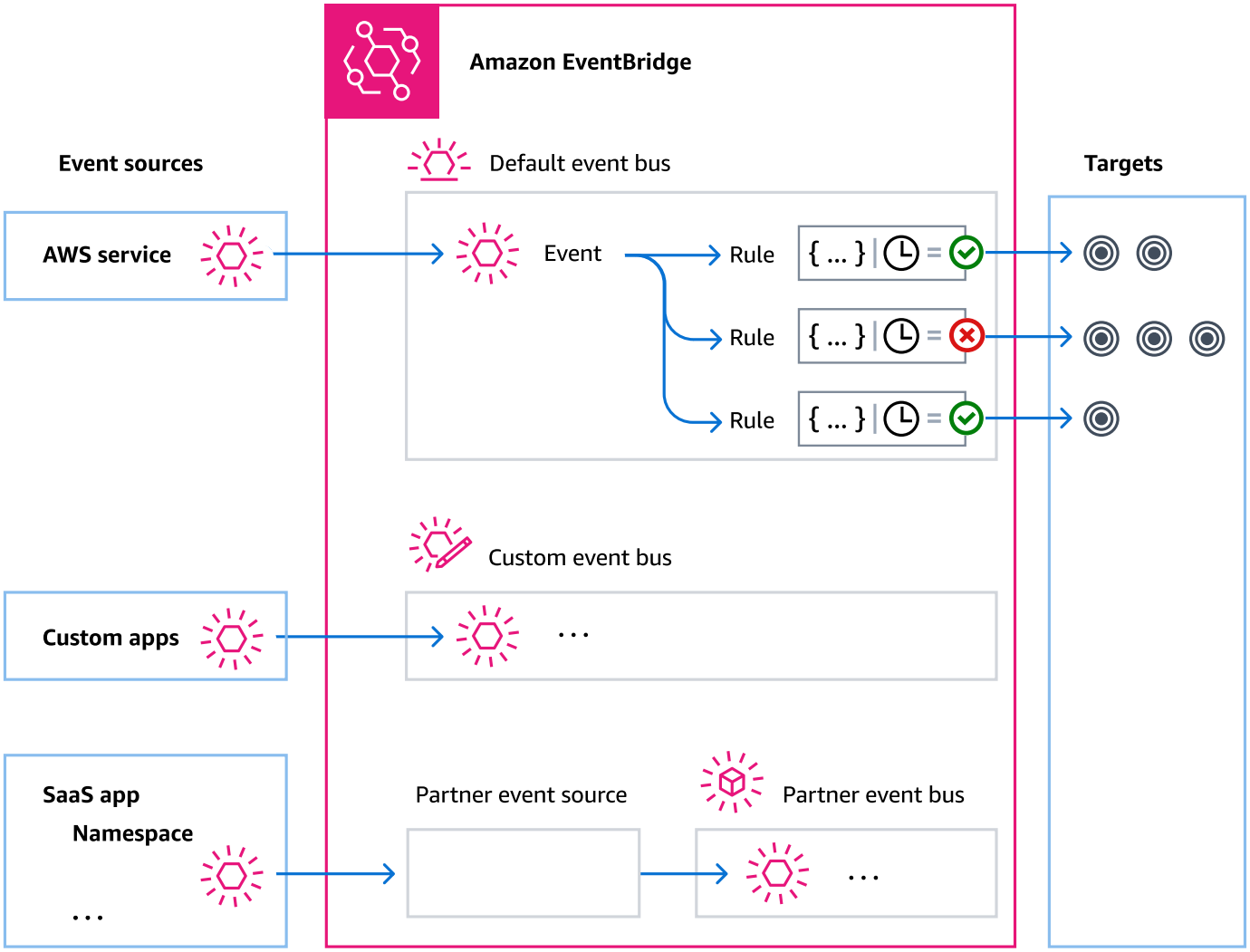
이벤트 버스는 이벤트를 수신하여 0개 이상의 목적지 또는 대상에 전달하는 라우터입니다. 다양한 소스의 이벤트를 여러 대상으로 라우팅해야 할 때 이벤트 버스를 사용하며, 대상으로 전달하기 전에 선택적으로 이벤트를 변환할 수 있습니다.

계정에는 AWS 서비스로부터 이벤트를 자동으로 수신하는 기본 이벤트 버스가 포함되어 있습니다. 다른 방법:

- 사용자 지정 이벤트 버스라고 하는 추가 이벤트 버스를 생성하고 수신할 이벤트를 지정합니다.
- SaaS 파트너로부터 이벤트를 수신하는 파트너 이벤트 버스를 생성합니다.

이벤트 버스의 일반적인 사용 사례는 다음과 같습니다.

- 이벤트 버스를 서로 다른 워크로드, 서비스 또는 시스템 간의 브로커로 사용합니다.
- 애플리케이션에서 여러 이벤트 버스를 사용하여 이벤트 트래픽을 분할합니다. 예를 들어 개인 식별 정보(PII)가 포함된 이벤트를 처리하는 버스를 생성하고, 그렇지 않은 이벤트를 처리하는 또 다른 버스를 생성합니다.
- 여러 이벤트 버스의 이벤트를 중앙 집중식 이벤트 버스로 전송하여 이벤트를 집계합니다. 이 중앙 집중식 버스는 다른 버스와 동일한 계정에 있을 수 있지만 다른 계정 또는 리전에 있을 수도 있습니다.



이벤트

간단히 말해 EventBridge 이벤트는 이벤트 버스 또는 파이프로 전송되는 JSON 객체입니다.

이벤트 기반 아키텍처(EDA)의 맥락에서 이벤트는 주로 리소스 또는 환경의 변화를 나타내는 지표입니다.

자세한 정보는 ???을 참조하세요.

이벤트 소스

EventBridge 다음을 포함한 이벤트 소스에서 이벤트를 수신할 수 있습니다.

- AWS 서비스

- 사용자 정의 애플리케이션
- 서비스형 소프트웨어(SaaS) 파트너

규칙

규칙은 들어오는 이벤트를 수신하고 적절한 이벤트를 대상으로 전송하여 처리합니다. 다음 중 하나를 기반으로 각 규칙이 대상을 간접 호출하는 방법을 지정할 수 있습니다.

- 이벤트와 일치하는 하나 이상의 필터가 포함된 이벤트 패턴입니다. 이벤트 패턴에는 다음과 일치하는 필터가 포함될 수 있습니다.
  - 이벤트 메타데이터 - 이벤트 소스, 이벤트가 발생한 계정 또는 리전 등의 이벤트 관련 데이터입니다.
  - 이벤트 데이터 - 이벤트 자체의 속성입니다. 이러한 속성은 이벤트에 따라 달라집니다.
  - 이벤트 콘텐츠 - 이벤트 데이터의 실제 속성 값입니다.
- 정기적으로 대상을 간접 호출하는 일정입니다.

  EventBridge 스케줄러 내에서 EventBridge 또는 스케줄러를 사용하여 스케줄링된 규칙을 지정할 수 있습니다.

> **Note**
>
> EventBridge 는 하나의 중앙 관리형 서비스에서 작업을 생성, 실행 및 관리할 수 있는 서버리스 스케줄러인 Amazon EventBridge Scheduler를 제공합니다. EventBridge Scheduler는 고도로 사용자 지정이 가능하며, 대상 API 작업 및 서비스의 범위가 더 넓어 EventBridge 예정된 규칙보다 향상된 확장성을 제공합니다. AWS
> EventBridge 스케줄러를 사용하여 일정에 따라 대상을 호출하는 것이 좋습니다. 자세한 정보는 ???을 참조하세요.

각 규칙은 특정 이벤트 버스에 대해 정의되며 해당 이벤트 버스의 이벤트에만 적용됩니다.

단일 규칙은 최대 5개의 대상에 이벤트를 보낼 수 있습니다.

기본적으로 이벤트 버스당 규칙을 최대 300개까지 구성할 수 있습니다. Service Quotas 콘솔에서 이 할당량을 수천 개의 규칙으로 늘릴 수 있습니다. 규칙 제한은 각 버스에 적용되므로 더 많은 규칙이 필요한 경우 계정에서 사용자 지정 이벤트 버스를 추가로 생성할 수 있습니다.

서비스별로 권한이 다른 이벤트 버스를 생성하여 계정에서 이벤트가 수신되는 방식을 사용자 지정할 수 있습니다.

이벤트를 대상으로 EventBridge 전달하기 전에 이벤트의 구조나 날짜를 사용자 지정하려면 정보가 대상으로 전달되기 전에 입력 변환기를 사용하여 정보를 편집하십시오.

자세한 정보는 ???을 참조하세요.

대상

대상은 이벤트가 규칙에 정의된 이벤트 패턴과 일치할 때 이벤트를 EventBridge 보내는 리소스 또는 엔드포인트입니다.

대상은 여러 이벤트 버스에서 다양한 이벤트를 수신할 수 있습니다.

자세한 정보는 ???을 참조하세요.

이벤트 버스의 고급 기능

EventBridge 이벤트 버스를 개발, 관리 및 사용하는 데 도움이 되는 다음 기능이 포함되어 있습니다.

API 대상을 사용하여 서비스 간 REST API 직접 호출 활성화

EventBridge API 대상은 AWS 서비스 또는 리소스에 이벤트 데이터를 보내는 것과 같은 방식으로 규칙의 대상으로 설정할 수 있는 HTTP 엔드포인트입니다. API 대상을 사용하면 API 직접 호출을 사용하여 AWS 서비스, 통합 SaaS 애플리케이션 및 AWS외부 애플리케이션 간에 이벤트를 라우팅할 수 있습니다. API 대상을 생성할 때 사용할 연결을 지정합니다. 각 연결에는 API 대상 엔드포인트에 권한을 부여하는 데 사용할 권한 부여 유형 및 파라미터에 대한 세부 정보가 포함됩니다.

개발 및 재해 복구를 지원하기 위한 이벤트 보관 및 재생

이벤트를 보관하거나 저장한 다음, 아카이브에서 나중에 재생할 수 있습니다. 보관은 다음과 같은 경우에 유용합니다.

- 새 이벤트를 기다릴 필요 없이 사용할 이벤트 스토어가 있으므로 애플리케이션을 테스트합니다.
- 새 서비스를 처음 온라인에 제공할 때 하이드레이트합니다.
- 이벤트 기반 애플리케이션에 내구성을 더합니다.

스키마 레지스트리를 사용하여 빠르게 이벤트 패턴 생성 시작

를 사용하는 EventBridge 서버리스 애플리케이션을 구축할 때는 이벤트를 생성하지 않고도 일반적인 이벤트의 구조를 아는 것이 유용할 수 있습니다. 이벤트 구조는 스키마에 설명되어 있으며, 스키마는 의 AWS 서비스에서 생성되는 모든 이벤트에 사용할 수 있습니다. EventBridge

AWS 서비스에서 전송되지 않는 이벤트의 경우 다음을 수행할 수 있습니다.

- 사용자 지정 스키마를 생성하거나 업로드합니다.
- 스키마 검색을 사용하면 이벤트 버스로 전송되는 이벤트에 대한 스키마를 EventBridge 자동으로 생성할 수 있습니다.

이벤트에 대한 스키마가 있으면 인기 있는 프로그래밍 언어에 대한 코드 바인딩을 다운로드할 수 있습니다.

정책을 통해 리소스 및 액세스 관리

AWS 리소스를 구성하거나 비용을 추적하기 위해 AWS 리소스에 EventBridge 사용자 지정 레이블 또는 태그를 할당할 수 있습니다. 태그 기반 정책을 사용하면 리소스가 내에서 EventBridge 수행할 수 있는 작업과 수행할 수 없는 작업을 제어할 수 있습니다.

태그 기반 정책 외에도 액세스를 제어할 수 있는 ID 기반 및 리소스 기반 정책을 EventBridge 지원합니다. EventBridge 자격 증명 기반 정책을 사용하면 그룹, 역할 또는 사용자의 권한을 제어할 수 있습니다. 리소스 기반 정책을 사용하면 Lambda 함수 또는 Amazon SNS 주제와 같은 각 리소스에 특정 권한을 부여할 수 있습니다.

Amazon EventBridge 이벤트 버스 만들기

애플리케이션에서 이벤트를 수신할 사용자 지정 이벤트 버스를 생성할 수 있습니다. 애플리케이션에서 기본 이벤트 버스로 이벤트를 보낼 수도 있습니다. 이벤트 버스를 생성할 때 리소스 기반 정책을 연결하여 다른 계정에 권한을 부여할 수 있습니다. 그러면 다른 계정이 현재 계정의 이벤트 버스에 이벤트를 보낼 수 있습니다.

다음 동영상에서는 이벤트 버스를 생성하는 과정을 보여줍니다. 이벤트 버스 생성

사용자 지정 이벤트 버스를 생성하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 이벤트 버스를 선택합니다.

3. Create event bus(이벤트 버스 생성)를 선택하십시오.
4. 새 이벤트 버스의 이름을 입력하십시오.
5. 이벤트 KMS key EventBridge 버스에 저장된 이벤트 데이터를 암호화할 때 사용할 형식을 선택합니다.

> **Note**
>
> 고객 관리형 키 a를 사용하여 암호화된 이벤트 버스에는 아카이브 및 스키마 검색이 지원되지 않습니다. 이벤트 버스에서 아카이브 또는 스키마 검색을 활성화하려면 를 사용하십시오 AWS 소유 키. 자세한 정보는 ???을 참조하세요.

- 를 사용하여 데이터를 EventBridge 암호화하려면 [AWS 소유 키 Use AWS 소유 키 for] 를 선택합니다.

  여러 AWS 계정에서 사용할 수 KMS key 있도록 EventBridge 소유하고 관리하는 AWS 소유 키 계정입니다. 일반적으로 리소스를 보호하는 암호화 키를 감사하거나 제어해야 하는 경우가 아니라면 이 방법을 AWS 소유 키 사용하는 것이 좋습니다.

  이 값이 기본값입니다.
- 지정하거나 생성한 데이터를 사용하여 데이터를 EventBridge 암호화하려면 사용 고객 관리형 키 용을 선택합니다. 고객 관리형 키

  고객 관리형 키 만들고, 소유하고, 관리하는 AWS 계정에 있습니다 KMS keys . 이를 완전히 제어할 수 KMS keys있습니다.

  a. 기존 고객 관리형 키항목을 지정하거나 새로 만들기를 선택합니다 KMS key.

     EventBridge 키 상태 및 지정된 고객 관리형 키것과 관련된 모든 키 별칭을 표시합니다.

  b. 이 이벤트 버스의 데드레터 큐 (DLQ) 로 사용할 Amazon SQS 대기열을 선택합니다 (있는 경우).

     EventBridge 성공적으로 암호화되지 않은 이벤트를 DLQ로 전송합니다 (구성된 경우). 그러면 나중에 처리할 수 있습니다.
6. 선택적 이벤트 버스 기능 구성:

- 다음 중 하나를 수행하여 리소스 기반 정책을 지정합니다.

- 이벤트 버스에 부여할 권한이 포함된 정책을 입력합니다. 다른 소스의 정책을 붙여넣거나 정책에 대한 JSON을 입력할 수 있습니다. 예제 정책 중 하나를 사용하여 환경에 맞게 수정할 수 있습니다.
- 정책에 템플릿을 사용하려면 템플릿 로드를 선택합니다. 정책에서 보안 주체에게 사용 권한을 부여하는 추가 작업을 추가하는 등 환경에 맞게 정책을 수정합니다.

리소스 기반 정책을 통해 이벤트 버스에 권한을 부여하는 방법에 대한 자세한 내용은 을 참조하십시오. ???

- 아카이브 활성화 (선택 사항)

  나중에 쉽게 재생할 수 있도록 이벤트 아카이브를 만들 수 있습니다. 예를 들어 이벤트를 재생하여 오류를 복구하거나 애플리케이션의 새 기능을 검증할 수 있습니다. 자세한 내용은 ??? 섹션을 참조하세요.

  a. 아카이브에서 활성화를 선택합니다.

  b. 아카이브의 이름과 설명을 지정합니다.

  > **Note**
  >
  > a를 사용하여 암호화된 이벤트 버스에는 아카이브 및 스키마 검색이 지원되지 않습니다 고객 관리형 키. 이벤트 버스에서 아카이브 또는 스키마 검색을 활성화하려면 를 사용하십시오 AWS 소유 키. 자세한 정보는 ???을 참조하세요.

- 스키마 검색 활성화 (선택 사항)

  이 이벤트 버스에서 실행되는 이벤트에서 직접 스키마를 EventBridge 자동으로 추론하도록 스키마 검색을 활성화합니다. 자세한 내용은 ??? 섹션을 참조하세요.

  a. 스키마 검색에서 활성화를 선택합니다.

  > **Note**
  >
  > a를 사용하여 암호화된 이벤트 버스에는 아카이브 및 스키마 검색이 지원되지 않습니다 고객 관리형 키. 이벤트 버스에서 아카이브 또는 스키마 검색을 활성화하려면 를 사용하십시오 AWS 소유 키. 자세한 정보는 ???을 참조하세요.

- 태그 지정 (선택 사항)

  태그는 AWS 리소스에 할당하는 사용자 지정 속성 레이블입니다. 태그를 사용하여 AWS 리소스를 식별하고 구성하세요. 많은 AWS 서비스가 태그 지정을 지원하므로 서로 다른 서비스의 리

소스에 동일한 태그를 할당하여 리소스가 관련되어 있음을 나타낼 수 있습니다. 자세한 내용은 ??? 섹션을 참조하세요.

a. 태그 아래에서 새 태그 추가를 선택합니다.

b. 새 태그의 키와 값 (선택 사항) 을 지정합니다.

7. 생성을 선택합니다.

Amazon EventBridge 이벤트 버스 업데이트

이벤트 버스를 생성한 후 구성을 업데이트할 수 있습니다. 여기에는 계정에 자동으로 EventBridge 생성되는 기본 이벤트 버스가 포함됩니다.

암호화에 KMS key 사용되는 항목 업데이트

> **Note**
>
> a를 사용하여 암호화된 이벤트 버스에는 아카이브 및 스키마 검색이 지원되지 않습니다 고객 관리형 키. 이벤트 버스에서 아카이브 또는 스키마 검색을 활성화하려면 를 사용하십시오 AWS 소유 키. 자세한 정보는 ???을 참조하세요.

EventBridge 콘솔을 사용하여 이벤트 버스의 저장 중 암호화에 KMS key 사용되는 데이터를 변경하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 이벤트 버스를 선택합니다.
3. 업데이트하려는 이벤트 버스를 선택합니다.
4. 이벤트 버스 세부 정보 페이지에서 암호화 탭을 선택합니다.
5. 이벤트 KMS key EventBridge 버스에 저장된 이벤트 데이터를 암호화할 때 사용할 형식을 선택합니다.
   - 를 사용하여 데이터를 EventBridge 암호화하려면 사용 AWS 소유 키 용을 선택합니다. AWS 소유 키

     여러 AWS 계정에서 사용할 수 KMS key 있도록 EventBridge 소유하고 관리하는 AWS 소유 키 계정입니다. 일반적으로 리소스를 보호하는 암호화 키를 감사하거나 제어해야 하는 경우가 아니라면 이 방법을 AWS 소유 키 사용하는 것이 좋습니다.

이 값이 기본값입니다.

- 지정하거나 생성한 데이터를 사용하여 데이터를 EventBridge 암호화하려면 사용 고객 관리형 키 용을 선택합니다. 고객 관리형 키

  고객 관리형 키 만들고, 소유하고, 관리하는 AWS 계정에 있습니다 KMS keys . 이를 완전히 제어할 수 KMS keys있습니다.

  a. 기존 고객 관리형 키항목을 지정하거나 새로 만들기를 선택합니다 KMS key.

     EventBridge 키 상태 및 지정된 고객 관리형 키것과 관련된 모든 키 별칭을 표시합니다.

  b. 이 이벤트 버스의 데드레터 큐 (DLQ) 로 사용할 Amazon SQS 대기열을 선택합니다 (있는 경우).

     EventBridge 성공적으로 암호화되지 않은 이벤트를 DLQ로 전송합니다 (구성된 경우). 그러면 나중에 처리할 수 있습니다.

이벤트 버스에 대한 권한 업데이트

리소스 기반 정책을 이벤트 버스에 연결하여 추가 권한을 이벤트 버스에 부여할 수 있습니다. 이벤트 버스에 지정된 권한을 업데이트하는 방법에 대한 자세한 지침은 이벤트 버스 권한 관리를 참조하십시오.

이벤트 버스의 아카이브 추가 또는 제거

아카이브를 사용하면 이벤트를 캡처하여 나중에 쉽게 재생할 수 있습니다. 예를 들어 이벤트를 재생하여 오류를 복구하거나 애플리케이션의 새 기능을 검증할 수 있습니다. 자세한 내용은 EventBridge 아카이브 및 재생을 참조하십시오.

> **Note**
>
> 고객 관리형 키 a를 사용하여 암호화된 이벤트 버스에 대해서는 아카이브 및 스키마 검색이 지원되지 않습니다. 이벤트 버스에서 아카이브 또는 스키마 검색을 활성화하려면 를 사용하십시오 AWS 소유 키. 자세한 정보는 ???을 참조하세요.

EventBridge 콘솔을 사용하여 이벤트 버스에 아카이브를 추가하거나 이벤트 버스에서 아카이브를 제거하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 이벤트 버스를 선택합니다.
3. 업데이트하려는 이벤트 버스를 선택합니다.
4. 이벤트 버스 세부 정보 페이지에서 아카이브 탭을 선택합니다.
5. 다음 중 하나를 수행하십시오.
   - 아카이브를 추가하려면:
     a. 아카이브 생성을 선택합니다.
     b. 아카이브의 속성을 지정합니다.
     c. 다음을 선택합니다.
     d. 아카이브의 이벤트에 적용할 이벤트 패턴을 선택합니다.
     e. 아카이브 생성을 선택합니다.
   - 아카이브를 삭제하려면:
     a. 제거하려는 태그의 경우 삭제를 선택합니다.
     b. 아카이브 이름을 입력하고 삭제를 선택합니다.

        아카이브가 영구적으로 삭제됩니다. 이 작업은 실행 취소할 수 없습니다.

를 사용하여 이벤트 버스의 아카이브를 만들거나 삭제하려면 AWS CLI

- 아카이브를 만들려면 생성-아카이브를 사용하십시오.

  아카이브를 영구 삭제하려면 아카이브 삭제를 사용합니다.

이벤트 버스에서 스키마 검색 시작 또는 중지

스키마 검색에 대한 자세한 내용은 EventBridge 스키마를 참조하십시오.

> **Note**
>
> 고객 관리형 키 a를 사용하여 암호화된 이벤트 버스에는 아카이브 및 스키마 검색이 지원되지 않습니다. 이벤트 버스에서 아카이브 또는 스키마 검색을 활성화하려면 를 사용하십시오 AWS 소유 키. 자세한 정보는 ???을 참조하세요.

EventBridge 콘솔을 사용하여 이벤트 버스에서 스키마 검색을 시작하거나 중지하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 이벤트 버스를 선택합니다.
3. 업데이트하려는 이벤트 버스를 선택합니다.
4. 다음 중 하나를 수행하십시오.
   - 스키마 검색을 시작하려면 검색 시작을 선택합니다.
   - 스키마 검색을 중지하려면 검색 삭제를 선택합니다.

를 사용하여 이벤트 버스에서 스키마 검색을 시작하거나 중지하려면 AWS CLI

- 스키마 검색을 시작하려면 create-discoverer를 사용하십시오.

  스키마 검색을 중지하려면 삭제-디스커버러를 사용하십시오.

이벤트 버스의 태그 추가 또는 제거

태그는 사용자가 또는 AWS 리소스에 AWS 할당하는 사용자 지정 속성 레이블입니다. 태그를 사용하여 AWS 리소스를 식별하고 구성하세요. 자세한 내용은 EventBridge 태그를 참조하십시오.

EventBridge 콘솔을 사용하여 이벤트 버스에 태그를 추가하거나 이벤트 버스에서 태그를 제거하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 이벤트 버스를 선택합니다.
3. 업데이트하려는 이벤트 버스를 선택합니다.
4. 이벤트 버스 세부 정보 페이지에서 [태그] 탭을 선택한 다음 [Manage tags] 를 선택합니다.
5. 다음 중 하나를 수행하십시오.
   - 태그를 추가하려면:

a. 새 태그 추가를 선택합니다.

b. 태그의 키와 값을 지정합니다.

c. 업데이트를 선택합니다.

- 태그 삭제하기

a. 제거하려는 태그의 경우 제거를 선택합니다.

b. 업데이트를 선택합니다.

를 사용하여 이벤트 버스에 태그를 추가하거나 이벤트 버스에서 태그를 제거하려면 AWS CLI

- 태그를 추가하려면 tag-resource를 사용하십시오.

  태그를 제거하려면 untag-resource를 사용하세요.

를 사용하여 기본 이벤트 버스 업데이트 AWS CloudFormation

AWS CloudFormation 인프라를 코드로 취급하여 중앙 집중화되고 반복 가능한 방식으로 계정 및 지역 전반의 AWS 리소스를 구성하고 관리할 수 있습니다. CloudFormation 프로비저닝 및 관리하려는 리소스를 정의하는 템플릿을 생성하여 이를 수행할 수 있습니다.

기본 이벤트 버스가 계정에 자동으로 EventBridge 프로비저닝되므로 CloudFormation 스택에 포함하려는 리소스에 대해 일반적으로 사용하는 것처럼 CloudFormation 템플릿을 사용하여 생성할 수 없습니다. 스택에 기본 이벤트 버스를 포함하려면 먼저 CloudFormation 스택으로 기본 이벤트 버스를 가져와야 합니다. 기본 이벤트 버스를 스택으로 가져온 후에는 필요에 따라 이벤트 버스 속성을 업데이트할 수 있습니다.

기존 리소스를 새 스택 또는 기존 CloudFormation 스택으로 가져오려면 다음 정보가 필요합니다.

- 가져올 리소스의 고유 식별자.

  기본 이벤트 버스의 경우 식별자는 이고 `Name` 식별자 값은 입니다`default`.

- 기존 리소스의 현재 속성을 정확하게 설명하는 템플릿입니다.

  아래 템플릿 스니펫에는 기본 이벤트 버스의 현재 속성을 설명하는 `AWS::Events::EventBus` 리소스가 포함되어 있습니다. 이 예제에서 이벤트 버스는 저장 시 암호화에 고객 관리형 키 및 DLQ를 사용하도록 구성되었습니다.

또한 가져오려는 기본 이벤트 버스를 설명하는 AWS::Events::EventBus 리소스에는 DeletionPolicy 속성 설정이 포함되어야 합니다. Retain

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Default event bus import example",
    "Resources": {
        "defaultEventBus": {
            "Type" : "AWS::Events::EventBus",
            "DeletionPolicy": "Retain",
            "Properties" : {
                "Name" : "default",
                "KmsKeyIdentifier" : "KmsKeyArn",
                "DeadLetterConfig" : {
                    "Arn" : "DLQ_ARN"
                }
            }
        }
    }
}
```

자세한 내용은 CloudFormation 사용 설명서의 기존 리소스를 CloudFormation 관리로 가져오기를 참조하십시오.

Amazon EventBridge 이벤트 버스 삭제

커스텀 또는 파트너 이벤트 버스를 삭제할 수 있습니다. 기본 이벤트 버스는 삭제할 수 없습니다. 이벤트 버스를 삭제하면 해당 이벤트 버스와 관련된 규칙이 삭제됩니다.

콘솔을 EventBridge 사용하여 이벤트 버스를 삭제하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 이벤트 버스를 선택합니다.
3. 삭제하려는 이벤트 버스를 선택합니다.
4. 다음 중 하나를 수행하십시오.
   - 삭제를 선택합니다.
   - 이벤트 버스의 이름을 선택합니다.

이벤트 버스 세부 정보 페이지에서 삭제를 선택합니다.

Amazon EventBridge 이벤트 버스에 대한 권한

AWS 계정의 기본 이벤트 버스는 한 계정의 이벤트만 허용합니다. 리소스 기반 정책을 이벤트 버스에 연결하여 추가 권한을 이벤트 버스에 부여할 수 있습니다. 리소스 기반 정책을 사용하면 다른 계정의 `PutEvents`, `PutRule` 및 `PutTargets` API 직접 호출을 허용할 수 있습니다. 또한 정책의 IAM 조건을 사용하여 조직에 권한을 부여하거나, 태그를 적용하거나, 특정 규칙 또는 계정의 조직에만 이벤트를 필터링할 수 있습니다. 이벤트 버스 생성 시 또는 이후에 이벤트 버스에 대한 리소스 기반 정책을 설정할 수 있습니다.

`PutRule`, `PutTargets`, `DeleteRule`, `RemoveTargets`, `DisableRule` 및 `EnableRule` 등의 이벤트 버스 `Name` 파라미터를 허용하는 EventBridge API는 이벤트 버스 ARN도 허용합니다. 이러한 파라미터를 사용하여 API를 통해 교차 계정 또는 교차 리전 이벤트 버스를 참조합니다. 예를 들어 `PutRule`를 호출하면 역할을 맡을 필요 없이 다른 계정의 이벤트 버스에 규칙을 생성할 수 있습니다.

이 주제의 정책 예제를 IAM 역할에 연결하여 이벤트를 다른 계정이나 리전으로 전송할 권한을 부여할 수 있습니다. IAM 역할을 사용하여 사용자 계정에서 다른 계정으로 이벤트를 전송할 수 있는 사람에 대한 조직 제어 정책 및 경계를 설정합니다. 규칙의 대상이 이벤트 버스인 경우 항상 IAM 역할을 사용하는 것이 좋습니다. `PutTarget` 호출을 사용하여 IAM 역할을 연결할 수 있습니다. 이벤트를 다른 계정이나 리전으로 전송하는 규칙을 생성하는 방법에 대한 자세한 내용은 AWS 계정 간 Amazon EventBridge 이벤트 전송 및 수신 섹션을 참조하세요.

주제

- 이벤트 버스 권한 관리
- 예제 정책: 다른 계정의 기본 버스로 이벤트 전송
- 예제 정책: 다른 계정의 사용자 지정 버스로 이벤트 전송
- 예제 정책: 동일한 계정의 이벤트 버스로 이벤트 전송
- 예제 정책: 동일한 계정으로 이벤트를 전송하고 업데이트 제한
- 예제 정책: 특정 규칙의 이벤트만 다른 리전의 버스로 전송
- 예제 정책: 특정 리전의 이벤트만 다른 리전으로 전송
- 예제 정책: 특정 리전의 이벤트 전송 거부

이벤트 버스 권한 관리

기존 이벤트 버스에 대한 권한을 수정하려면 다음 절차를 따릅니다. AWS CloudFormation을 사용해 이벤트 버스 정책을 생성하는 방법에 대한 자세한 내용은 AWS::Events::EventBusPolicy를 참조하세요.

기존 이벤트 버스에 대한 권한을 관리하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 이벤트 버스를 선택합니다.
3. 이름에서 권한을 관리할 이벤트 버스의 이름을 선택합니다.

   리소스 정책이 이벤트 버스에 연결된 경우 해당 정책이 표시됩니다.
4. 권한 관리를 선택하고 다음 중 하나를 수행합니다.
   - 이벤트 버스에 부여할 권한이 포함된 정책을 입력합니다. 다른 소스의 정책을 붙여넣거나 정책에 대한 JSON을 입력할 수 있습니다.
   - 정책에 템플릿을 사용하려면 템플릿 로드를 선택합니다. 환경에 맞게 정책을 수정하고 정책에서 보안 주체에게 사용 권한을 부여하는 추가 작업을 추가합니다.
5. 업데이트를 선택합니다.

템플릿은 계정과 환경에 맞게 사용자 지정할 수 있는 예제 정책 구문을 제공합니다. 템플릿은 유효한 정책이 아닙니다. 사용 사례에 맞게 템플릿을 수정하거나 정책 예제 중 하나를 복사하여 사용자 지정할 수 있습니다.

템플릿은 계정에 PutEvents 작업을 사용할 권한을 부여하는 방법, 조직에 권한을 부여하는 방법, 계정의 규칙을 관리할 수 있는 권한을 계정에 부여하는 방법에 대한 예가 포함된 정책을 로드합니다. 특정 계정에 맞게 템플릿을 사용자 지정하고 템플릿에서 다른 섹션을 삭제할 수 있습니다. 더 많은 정책 예제는 이 주제의 뒷부분에 나와 있습니다.

버스에 대한 권한을 업데이트하려고 시도했지만 정책에 오류가 있는 경우 오류 메시지에 정책의 특정 문제가 나타납니다.

```
 ### 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>"
          }
        }
    }]
  }
```

예제 정책: 다른 계정의 기본 버스로 이벤트 전송

다음 예제 정책은 계정 111122223333에 이벤트를 계정 123456789012의 기본 이벤트 버스에 게시할 수 있는 권한을 부여합니다.

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

예제 정책: 다른 계정의 사용자 지정 버스로 이벤트 전송

다음 예제 정책은 계정 111122223333에 이벤트를 계정 123456789012의 central-event-bus에 게시할 수 있는 권한을 부여합니다. 단, 소스 값이 com.exampleCorp.webStore로 설정되고 detail-type이 newOrderCreated로 설정된 이벤트에만 적용합니다.

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

예제 정책: 동일한 계정의 이벤트 버스로 이벤트 전송

CustomBus1이라는 이벤트 버스에 연결된 다음 예제 정책은 이벤트 버스가 동일한 계정 및 리전에서 이벤트를 수신할 수 있도록 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
```

```
      "Resource": [
        "arn:aws:events:us-east-1:123456789:event-bus/CustomBus1"
      ]
    }
  ]
}
```

예제 정책: 동일한 계정으로 이벤트를 전송하고 업데이트 제한

다음 예제 정책은 계정 123456789012에 규칙을 생성, 삭제, 업데이트, 비활성화 및 활성화하고 대상을 추가 또는 제거할 수 있는 권한을 부여합니다. 이 정책은 소스가 com.exampleCorp.webStore인 이벤트와 일치하는 규칙을 제한하고, 이러한 규칙 및 대상이 생성되면 계정 123456789012만 이를 수정할 수 있도록 "events:creatorAccount": "${aws:PrincipalAccount}"를 사용합니다.

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

예제 정책: 특정 규칙의 이벤트만 다른 리전의 버스로 전송

다음 예제 정책은 계정 123456789012에서 중동(바레인) 및 미국 서부(오레곤) 리전의 SendToUSE1AnotherAccount라는 규칙과 일치하는 이벤트를 미국 동부(버지니아 북부)의 CrossRegionBus라는 이벤트 버스로 전송할 수 있는 권한을 계정 111122223333에 부여합니다. 예제 정책은 계정 123456789012에 CrossRegionBus라는 이벤트 버스에 추가됩니다. 이 정책은 계정 111122223333의 이벤트 버스에 지정된 규칙과 일치하는 경우에만 이벤트를 허용합니다. Condition 구문은 이벤트를 지정된 규칙 ARN이 있는 규칙과 일치하는 이벤트로만 제한합니다.

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

예제 정책: 특정 리전의 이벤트만 다른 리전으로 전송

다음 예제 정책은 중동(바레인) 및 미국 서부(오레곤) 리전에서 생성된 모든 이벤트를 미국 동부(버지니아 북부) 리전의 계정 123456789012에서 CrossRegionBus라는 이벤트 버스로 전송할 수 있는 권한을 계정 111122223333에 부여합니다. 계정 111122223333에는 다른 리전에서 생성된 이벤트를 전송할 권한이 없습니다.

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

예제 정책: 특정 리전의 이벤트 전송 거부

계정 123456789012의 CrossRegionBus라는 이벤트 버스에 연결된 다음 예제 정책은 이벤트 버스가 계정 1111222233333에서 이벤트를 수신할 수 있는 권한을 부여하지만 미국 서부(오레곤) 리전에서 생성된 이벤트는 허용하지 않습니다.

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

Amazon EventBridge 이벤트 버스에서 AWS CloudFormation 템플릿 생성

AWS CloudFormation을 사용하면 인프라를 코드로 취급하여 중앙 집중화되고 반복 가능한 방식으로 계정 및 리전 전반의 AWS 리소스를 구성하고 관리할 수 있습니다. CloudFormation은 프로비저닝 및 관리하려는 리소스를 정의하는 템플릿을 생성하게 하여 이 작업을 수행합니다.

EventBridge를 사용하면 계정의 기존 이벤트 버스에서 템플릿을 생성할 수 있으므로 CloudFormation 템플릿 개발을 바로 시작할 수 있습니다. 또한 EventBridge는 템플릿에 해당 이벤트 버스와 관련된 규칙을 포함하는 옵션을 제공합니다. 따라서 이러한 템플릿을 기반으로 사용해 CloudFormation 관리에 따른 리소스 스택을 생성할 수 있습니다.

CloudFormation에 대한 자세한 내용은 AWS CloudFormation 사용 설명서를 참조하십시오.

> **Note**
>
> EventBridge는 생성된 템플릿에 관리형 규칙을 포함하지 않습니다.

선택한 이벤트 버스에 포함된 하나 이상의 규칙에서 템플릿을 생성할 수도 있습니다.

이벤트 버스에서 CloudFormation 템플릿 생성

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 이벤트 버스를 선택합니다.
3. CloudFormation 템플릿을 생성할 이벤트 버스를 선택합니다.
4. 작업 메뉴에서 CloudFormation 템플릿을 선택한 다음, EventBridge에서 템플릿을 생성할 때 사용할 형식(JSON 또는 YAML)을 선택합니다.

   EventBridge는 선택한 형식으로 생성된 템플릿을 표시합니다. 기본적으로 이벤트 버스와 관련된 모든 규칙이 템플릿에 포함됩니다.

   - 규칙을 포함하지 않고 템플릿을 생성하려면 이 이벤트 버스에 규칙 포함을 선택 취소합니다.
5. EventBridge는 템플릿 파일을 다운로드하거나 템플릿을 클립보드에 복사하는 옵션을 제공합니다.

   - 템플릿 파일을 다운로드하려면 다운로드를 선택합니다.
   - 템플릿을 클립보드에 복사하려면 복사를 선택합니다.
6. 템플릿을 종료하려면 취소를 선택합니다.

사용 사례에 맞게 AWS CloudFormation 템플릿을 사용자 지정한 후에는 이를 사용하여 CloudFormation에 스택을 생성할 수 있습니다.

Amazon EventBridge에서 생성된 CloudFormation 템플릿 사용 시 고려할 사항

이벤트 버스에서 생성한 CloudFormation 템플릿을 사용할 때는 다음 요소를 고려합니다.

- EventBridge는 생성 템플릿에 암호를 포함하지 않습니다.

  템플릿을 사용하여 CloudFormation 스택을 생성하거나 업데이트할 때 사용자가 암호 또는 기타 중요한 정보를 지정할 수 있는 템플릿 파라미터를 포함하도록 템플릿을 편집할 수 있습니다.

  또한 사용자는 Secrets Manager를 이용해 원하는 리전에 보안 암호를 생성한 다음, 생성된 템플릿을 편집하여 동적 파라미터를 적용할 수 있습니다.
- 생성된 템플릿의 대상은 원래 이벤트 버스에 지정된 그대로 유지됩니다. 따라서 템플릿을 사용하여 다른 리전에 스택을 생성하기 전에 템플릿을 적절하게 편집하지 않으면 교차 리전 문제가 발생할 수 있습니다.

  또한 생성된 템플릿은 다운스트림 대상을 자동으로 생성하지 않습니다.

아마존 EventBridge 이벤트

이벤트는 AWS 환경, SaaS 파트너 서비스나 애플리케이션 또는 사용자 애플리케이션이나 서비스 중 하나와 같은 환경의 변화를 나타냅니다. 다음은 이벤트의 예제입니다.

- Amazon EC2는 인스턴스의 상태가 보류 중에서 실행 중으로 변경될 때 이벤트를 생성합니다.
- Amazon EC2 Auto Scaling은 인스턴스를 시작하거나 종료할 때 이벤트를 생성합니다.
- AWS CloudTrail API 호출 시 이벤트를 게시합니다.

또한 정기적으로 생성되는 예약 이벤트를 설정할 수도 있습니다.

각 서비스의 샘플 이벤트를 비롯하여 이벤트를 생성하는 서비스 목록은 서비스의 이벤트 AWS 섹션을 참조하고 표에 있는 링크를 따르세요.

이벤트는 JSON 객체로 표현되고 모두 구조가 비슷하며 최상위 필드가 동일합니다.

detail 최상위 필드의 콘텐츠는 이벤트를 생성한 서비스와 이벤트에 따라 달라집니다. source 필드와 detail-type 필드를 조합하여 detail 필드에 나타나는 필드와 값을 식별할 수 있습니다. AWS 서비스에서 생성된 이벤트의 예는 을 참조하십시오서비스의 이벤트 AWS.

주제

- 이벤트 구조 참조
- 를 사용하여 Amazon EventBridge 이벤트 추가 PutEvents
- 서비스의 이벤트 AWS
- Amazon의 SaaS 파트너로부터 이벤트 받기 EventBridge
- 이벤트 전송 디버깅

다음 동영상에서는 이벤트의 기본 사항을 설명합니다. 이벤트 정의

다음 동영상은 이벤트가 어떻게 진행되는지 설명합니다 EventBridge. 이벤트는 어디서 오는 걸까요?

이벤트 구조 참조

다음 필드는 이벤트 버스로 전달되는 모든 이벤트에 나타나며 이벤트의 메타데이터를 구성합니다.

```
{
  "???": "0",
  "???": "UUID",
  "???": "event name",
  "???": "event source",
  "???": "ARN",
  "???": "timestamp",
  "???": "region",
  "???": [
    "ARN"
  ],
  "???": {
    JSON object
  }
}
```

version

기본적으로 이 필드는 모든 이벤트에서 0으로 설정되어 있습니다.

id

모든 이벤트에 대해 생성되는 버전 4 UUID입니다. id를 사용하여 규칙을 통해 대상으로 이동하는 이벤트를 추적할 수 있습니다.

detail-type

source 필드를 함께 사용하여 detail 필드에 나타나는 필드와 값을 식별합니다.

에서 CloudTrail 전달한 이벤트는 의 AWS API Call via CloudTrail 값으로 사용됩니다. detail-type

소스

이벤트를 생성한 서비스를 식별합니다. AWS 서비스에서 발생하는 모든 이벤트는 'aws'로 시작합니다. 고객이 생성한 이벤트는 "aws"로 시작되지 않는 한 어떤 값이라도 가질 수 있습니다. Java 패키지 이름 스타일을 사용하여 도메인 이름 문자열을 역순으로 만드는 것이 좋습니다.

AWS 서비스의 올바른 값을 source 찾으려면 조건 키 테이블을 참조하고 목록에서 서비스를 선택한 다음 서비스 접두사를 찾으십시오. 예를 들어, 아마존의 source 가치는 CloudFront 다음과 같습니다aws.cloudfront.

account

AWS 계정을 식별하는 12자리 숫자.

시간

이벤트를 호출한 서비스에서 지정할 수 있는 이벤트 타임스탬프입니다. 이벤트가 시간 간격 내에 있으면 서비스는 시작 시간을 보고해서 이 값이 이벤트가 수신되는 시간보다 앞에 있도록 할 수 있습니다.

region

이벤트가 시작된 AWS 지역을 식별합니다.

resources

이벤트에서 간접 호출된 리소스를 식별하는 ARN이 포함된 JSON 배열입니다. 이벤트를 생성하는 서비스에 따라 이러한 ARN을 포함할지 여부가 결정됩니다. 예를 들어 Amazon EC2 인스턴스 상태 변경에는 Amazon EC2 인스턴스 ARN이 포함되어 있고, Auto Scaling 이벤트에는 인스턴스 및 Auto Scaling 그룹 모두에 대한 ARN이 포함되어 있지만, AWS CloudTrail 에서의 API 직접 호출에는 리소스 ARN이 포함되지 않습니다.

detail

이벤트에 대한 정보를 포함하는 JSON 객체입니다. 이벤트를 생성하는 서비스에 따라 이 필드의 내용이 결정됩니다. `"{}"`이(가) 될 수 있습니다.

AWS API 호출 이벤트에는 여러 수준 깊이로 중첩된 약 50개의 필드가 있는 세부 객체가 있습니다.

> **Note**
>
> PutEventsJSON 형식의 데이터를 받아들입니다. JSON 숫자(정수) 데이터 유형의 경우 제약 조건은 최솟값 -9,223,372,036,854,775,808 및 최댓값 9,223,372,036,854,775,807입니다.

Example 예: Amazon EC2 인스턴스 상태 변경 알림

Amazon에서 발생한 다음 이벤트는 Amazon EC2 인스턴스가 EventBridge 종료되었음을 나타냅니다.

```
{
  "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"
  }
}
```

유효한 사용자 지정 이벤트에 필요한 최소 정보

사용자 지정 이벤트를 생성할 때는 다음 필드를 포함해야 합니다.

- `detail`
- `detail-type`
- `source`

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

를 사용하여 Amazon EventBridge 이벤트 추가 **PutEvents**

`PutEvents`액션은 단일 요청으로 EventBridge 여러 이벤트를 전송합니다. 자세한 내용은 Amazon EventBridge API 참조 및 AWS CLI 명령 PutEvents참조의 put-events를 참조하십시오.

각 `PutEvents` 요청은 제한된 수의 항목을 지원할 수 있습니다. 자세한 내용은 Amazon EventBridge 할당량 단원을 참조하십시오. `PutEvents` 작업은 요청의 일반 순서에 따라 모든 항목들을 처리하고자 시도합니다. `PutEvents`호출한 후 각 이벤트에 EventBridge 고유한 ID를 할당합니다.

주제

- PutEvents를 사용하여 실패 처리
- 를 사용하여 이벤트 전송 AWS CLI
- Amazon EventBridge PutEvents 이벤트 항목 크기 계산

다음 예제 Java 코드는 두 개의 동일한 이벤트를 에 EventBridge 전송합니다.

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());
    }
}
```

이 코드를 실행하면 PutEvents 결과에 응답 항목 어레이가 포함됩니다. 응답 어레이의 각 항목은 요청 및 응답이 처음부터 끝까지 순서대로 나열된 요청 어레이의 항목에 해당합니다. 응답의 Entries 어레이에는 항상 요청 어레이와 같은 수의 항목이 포함됩니다.

PutEvents를 사용하여 실패 처리

기본적으로 요청 내의 개별 항목이 실패하는 경우 요청의 나머지 항목을 EventBridge 계속 처리합니다. 응답 Entries 어레이에는 성공한 항목과 실패한 항목이 모두 포함될 수 있습니다. 따라서 실패한 항목들을 찾아서 후속 호출에 이를 포함시켜야 합니다.

성공한 결과 항목에는 Id 값이 포함되고, 실패한 결과 항목에는 ErrorCode 및 ErrorMessage 값이 포함됩니다. ErrorCode는 오류 유형을 설명하며 ErrorMessage는 오류에 대한 자세한 정보를 제공합니다. 다음 예제에서는 하나의 PutEvents 요청에 대해 3개의 결과 항목이 있습니다. 두 번째 항목은 실패했습니다.

```
{
    "FailedEntryCount": 1,
    "Entries": [
        {
            "EventId": "11710aed-b79e-4468-a20b-bb3c0c3b4860"
        },
        {   "ErrorCode": "InternalFailure",
            "ErrorMessage": "Internal Service Failure"
```

```
        },
        {
            "EventId": "d804d26a-88db-4b66-9eaf-9a11c708ae82"
        }
    ]
}
```

> **Note**
>
> 를 PutEvents 사용하여 존재하지 않는 이벤트 버스에 이벤트를 게시하는 경우 EventBridge 이벤트 매칭에서는 해당 규칙을 찾지 못하고 이벤트를 삭제합니다. 200응답을 보내긴 하지만 EventBridge 요청이 실패하거나 요청 응답 FailedEntryCount 값에 이벤트가 포함되지는 않습니다.

후속 PutEvents 요청에 실패한 항목들을 포함할 수 있습니다. 먼저, 요청에 실패한 항목이 있는지 알아보려면 PutEventsResult에서 FailedRecordCount 파라미터를 확인합니다. 값이 0이 아닌 경우 null이 아닌 ErrorCode 값을 가진 각 Entry 항목을 후속 요청에 추가할 수 있습니다. 다음 예제에서는 실패 핸들러를 보여줍니다.

```
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);
    }
```

를 사용하여 이벤트 전송 AWS CLI

를 사용하여 사용자 지정 이벤트를 AWS CLI EventBridge 전송하여 처리할 수 있습니다. 다음 예제에서는 사용자 지정 이벤트 EventBridge 하나를 삽입합니다.

```
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\" }"}]'
```

사용자 지정 이벤트가 포함된 JSON 파일을 생성할 수도 있습니다.

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

그런 다음 를 사용하여 이 파일에서 항목을 읽고 이벤트를 보내려면 명령 프롬프트에 다음을 입력합니다. AWS CLI

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

Amazon EventBridge PutEvents 이벤트 항목 크기 계산

PutEvents작업을 사용하여 사용자 지정 이벤트를 EventBridge 에 보낼 수 있습니다. 효율성을 위해 여러 개의 이벤트 항목을 하나의 요청으로 일괄 처리할 수 있습니다. 총 항목 크기는 256KB 미만이어야 합니다. 이벤트를 전송하기 전에 항목 크기를 계산할 수 있습니다.

> **Note**
>
> 항목에는 크기 제한이 있습니다. 항목이 크기 제한보다 작더라도 이벤트의 JSON 표현에 EventBridge 필요한 문자 및 키 때문에 의 이벤트는 항상 항목 크기보다 큽니다. 자세한 정보는 아마존 EventBridge 이벤트을 참조하세요.

EventBridge 다음과 같이 PutEventsRequestEntry 크기를 계산합니다.

- 지정된 경우 Time 파라미터는 14바이트입니다.
- Source 및 DetailType 파라미터는 UTF-8 인코딩 형식의 바이트 수입니다.
- 지정된 경우 Detail 파라미터는 UTF-8 인코딩 형식의 바이트 수입니다.
- 지정된 경우 각 Resources 파라미터 항목은 UTF-8 인코딩 형식의 바이트 수입니다.

다음의 Java 코드 예제는 지정된 PutEventsRequestEntry 객체의 크기를 계산합니다.

```
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**
>
> 항목 크기가 256KB보다 큰 경우 Amazon S3 버킷에 이벤트를 업로드하고 `PutEvents` 항목에 `Object URL`을 포함하는 것이 좋습니다.

서비스의 이벤트 AWS

많은 AWS 서비스가 EventBridge 수신하는 이벤트를 생성합니다. 계정의 AWS 서비스에서 이벤트가 발생하면 해당 이벤트는 계정의 기본 이벤트 버스로 이동합니다.

서비스를 통한 이벤트 전달 AWS

이벤트를 생성하는 각 AWS 서비스는 최선을 다하거나 지속적인 전송 시도를 위해 EventBridge 이벤트를 전송합니다.

- 최선 전송이란 서비스가 모든 이벤트를 에 전송하려고 시도하지만 드문 경우이긴 하지만 이벤트가 전송되지 않는 경우도 있다는 의미입니다. EventBridge
- 지속 가능한 전송이란 서비스가 EventBridge 적어도 한 번은 이벤트 전송을 성공적으로 시도한다는 의미입니다.

  EventBridge 정상적인 조건에서는 모든 유효한 이벤트를 수락합니다. EventBridge 서비스 중단으로 인해 이벤트를 전달할 수 없는 경우 AWS 서비스는 나중에 최대 24시간 동안 이벤트를 다시 시도합니다.

이벤트가 전달되면 해당 EventBridge 이벤트를 EventBridge 규칙과 대조한 다음 재시도 정책 및 이벤트 대상에 지정된 데드레터 대기열을 따릅니다.

이벤트를 생성하는 AWS 서비스 목록은 을 참조하십시오. ???

를 통해 AWS 서비스 이벤트에 액세스 AWS CloudTrail

AWS CloudTrail AWS API 호출과 같은 이벤트를 자동으로 기록하는 서비스입니다. 의 정보를 사용하는 EventBridge 규칙을 만들 수 있습니다 CloudTrail. 에 대한 자세한 내용은 CloudTrail What is AWS CloudTrail? 를 참조하십시오. .

에서 제공하는 CloudTrail 모든 이벤트는 가치가 `AWS API Call via CloudTrail` 동일합니다 다`detail-type`.

`detail-type`값이 인 이벤트를 기록하려면 로깅이 활성화된 CloudTrail 트레일이 필요합니다. `AWS API Call via CloudTrail`

Amazon CloudTrail S3와 함께 사용하는 경우 데이터 이벤트를 CloudTrail 기록하도록 구성해야 합니다. 자세한 내용은 S3 버킷 및 객체에 대한 CloudTrail 이벤트 로깅 활성화를 참조하십시오.

서비스에서 발생하는 일부 사례는 AWS 서비스 자체와 서비스 측에서 EventBridge 모두 보고할 수 있습니다. CloudTrail 예를 들어, 인스턴스를 시작하거나 중지하는 Amazon EC2 API 호출은 EventBridge 이벤트뿐만 아니라 이벤트를 생성합니다. CloudTrail

CloudTrail 는 API 호출자와 리소스 소유자가 트레일을 생성하여 Amazon S3 버킷에서 이벤트를 수신할 수 있도록 지원하고, 를 통해 API 호출자에게 이벤트를 전송합니다. EventBridge API 호출자뿐 아니라 리소스 소유자도 계정 간 API 호출을 모니터링할 수 있습니다. EventBridge CloudTrail와 통합하면 EventBridge 이벤트에 대한 응답으로 자동화된 규칙 기반 워크플로를 편리하게 설정할 수 있습니다.

AWS Put*Events 요청의 최대 크기가 256KB이므로 크기가 256KB를 초과하는 Put*Events API 호출 이벤트는 이벤트 패턴으로 사용할 수 없습니다. 사용할 수 있는 API 호출에 대한 자세한 내용은 지원되는 서비스 및 통합을 참조하십시오. CloudTrail

서비스로부터 읽기 전용 관리 이벤트 수신 AWS

를 통해 AWS 서비스로부터 읽기 전용 관리 이벤트를 수신하도록 기본 또는 사용자 지정 이벤트 버스에 규칙을 설정할 수 있습니다. CloudTrail 관리 이벤트를 통해 AWS 계정의 리소스에서 수행되는 관리 작업을 파악할 수 있습니다. 이를 제어 영역 작업이라고도 합니다. 자세한 내용은 CloudTrail 사용 설명서의 관리 이벤트 로깅 섹션을 참조하십시오.

기본 또는 사용자 지정 이벤트 버스의 각 규칙에 대해 규칙 상태를 설정하여 수신할 이벤트 유형을 제어할 수 있습니다.

- 이벤트가 규칙과 EventBridge 일치하지 않도록 규칙을 비활성화하십시오.
- 이벤트를 EventBridge 규칙과 일치시키도록 규칙을 활성화합니다. 단, CloudTrail전달된 읽기 전용 AWS 관리 이벤트는 예외입니다.
- 전달된 읽기 전용 관리 이벤트를 포함하여 모든 이벤트가 EventBridge 규칙과 일치하도록 규칙을 활성화하십시오. CloudTrail

파트너 이벤트 버스는 AWS 이벤트를 수신하지 않습니다.

읽기 전용 관리 이벤트를 수신할지 여부를 결정할 때 고려해야 할 몇 가지 사항은 다음과 같습니다.

- 및 또는 IAM AWS Key Management Service `GetKeyPolicy` `GetPolicy` 및 `DescribeKey` 이벤트와 같은 특정 읽기 전용 관리 `GetRole` 이벤트는 일반적인 변경 이벤트보다 훨씬 많은 볼륨으로 발생합니다.
- `Describe`, `Get` 또는 `List`로 시작하지 않는 읽기 전용 관리 이벤트는 이미 수신되고 있을 수 있습니다. 예를 들어 다음 AWS STS API의 이벤트는 동사로 시작한다고 생각해도 변경 이벤트입니다. `Get`
  - `GetFederationToken`
  - `GetSessionToken`

  서비스별 `Describe` `Get` AWS , 또는 `List` 명명 규칙을 준수하지 않는 읽기 전용 관리 이벤트 목록은 을 참조하십시오. ???

CLI를 사용하여 읽기 전용 관리 이벤트를 수신하는 규칙을 만들려면 AWS

- `put-rule` 명령으로 규칙을 만들거나 업데이트하여 파라미터를 사용하여 다음을 수행할 수 있습니다.
  - 규칙이 기본 이벤트 버스 또는 특정 사용자 지정 이벤트 버스에 속하도록 지정합니다.
  - 규칙 상태를 `ENABLED_WITH_ALL_CLOUDTRAIL_MANAGEMENT_EVENTS`로 설정합니다.

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

> **Note**
>
> CloudWatch 관리 이벤트에 대한 규칙 활성화는 AWS CLI 및 AWS CloudFormation 템플릿을 통해서만 지원됩니다.

Example

다음 예시는 특정 이벤트에 일치시키는 방법을 보여줍니다. 모범 관행은 명확하고 편집하기 쉽도록 특정 이벤트에 일치시키기 위한 전용 규칙을 정의하는 것입니다.

이 경우 전용 규칙은 `AssumeRole` 관리 이벤트 AWS Security Token Service원본과 일치합니다.

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

AWS 이벤트를 생성하는 서비스

다음 표에는 이벤트를 생성하는 AWS 서비스가 나와 있습니다. 서비스 이름을 선택하면 해당 서비스와 함께 EventBridge 작동하는 방식에 대한 자세한 내용을 볼 수 있습니다.

이벤트를 생성하는 각 AWS 서비스는 최선의 노력을 다하거나 지속적인 전송 시도를 위해 EventBridge 이벤트를 전송합니다. 자세한 정보는 ???을 참조하세요.

이 표에는 이벤트를 보내는 AWS 서비스가 EventBridge 표시되지만 모든 서비스가 포함되지는 않습니다. 목록에 없는 서비스 중에서 이벤트를 보내는 서비스의 EventBridge 경우 최선을 다해 전달한다고 가정하십시오.

Service	시도 유형
Alexa for Business	최선의 작업
AWS Account Management	최선의 작업
Amazon API Gateway	최선의 작업
AWS AppConfig	최선의 작업
아마존 AppFlow	최선의 작업
Application Auto Scaling	최선의 작업
AWS 애플리케이션 비용 프로파일러	최선의 작업
AWS Application Migration Service	최선의 작업
Amazon Athena	최선의 작업
AWS Backup	최선의 작업

Service	시도 유형
AWS Batch	지속적
Amazon Braket	지속적
AWS Certificate Manager	최선의 작업
Amazon Chime	최선의 작업
Amazon Cloud Directory	최선의 작업
AWS CloudFormation	지속적
아마존 CloudFront	최선의 작업
AWS CloudHSM	최선의 작업
아마존 CloudSearch	최선의 작업
AWS CloudShell	최선의 작업
이벤트 출처 AWS CloudTrail	최선의 작업
아마존 CloudWatch	지속적
Amazon CloudWatch 애플리케이션 인사이트	최선의 작업
아마존 CloudWatch 인터넷 모니터	최선의 작업
아마존 CloudWatch 로그	최선의 작업
아마존 CloudWatch 신세틱스	최선의 작업
AWS CodeArtifact	지속적
AWS CodeBuild	최선의 작업
AWS CodeCommit	최선의 작업
AWS CodeDeploy	최선의 작업

Service	시도 유형
아마존 CodeGuru 프로파일러	최선의 작업
AWS CodePipeline	최선의 작업
AWS CodeStar	최선의 작업
CodeConnections	최선의 작업
Amazon Cognito 자격 증명	최선의 작업
Amazon Cognito 사용자 풀	최선의 작업
Amazon Cognito Sync	최선의 작업
AWS Config	최선의 작업
Amazon Connect	최선의 작업
Amazon Connect Voice ID	최선의 작업
AWS Control Tower	최선의 작업
AWS Database Migration Service	최선의 작업
AWS Data Exchange	최선의 작업
Amazon Data Lifecycle Manager	최선의 작업
AWS Data Pipeline	최선의 작업
AWS DataSync	최선의 작업
AWS Device Farm	최선의 작업
아마존 DevOps 전문가	최선의 작업
AWS Direct Connect	최선의 작업
AWS Directory Service	최선의 작업

Service	시도 유형
Amazon DynamoDB	최선의 작업
AWS Elastic Beanstalk	최선의 작업
Amazon Elastic Block Store	최선의 작업
Amazon Elastic Block Store 볼륨	최선의 작업
아마존 ElastiCache	최선의 작업
Amazon Elastic Compute Cloud(Amazon EC2)	최선의 작업
Amazon EC2 Auto Scaling	최선의 작업
Amazon EC2 플릿	최선의 작업
Amazon EC2 스팟 인스턴스 중단	최선의 작업
Amazon Elastic 컨테이너 레지스트리	최선의 작업
Amazon Elastic Container Service	지속적
AWS Elastic Disaster Recovery	최선의 작업
Amazon Elastic File System	최선의 작업
Amazon Elastic Kubernetes 서비스	최선의 작업
Elastic Load Balancing	최선의 작업
아마존 엘라스틱 MapReduce	최선의 작업
Amazon Elastic Transcoder	최선의 작업
AWS Elemental MediaConnect	최선의 작업
AWS Elemental MediaConvert	지속적
AWS Elemental MediaLive	최선의 작업

Service	시도 유형
AWS Elemental MediaPackage	최선의 작업
AWS Elemental MediaStore	지속적
Amazon EMR	최선의 작업
Amazon EMR on EKS	최선의 작업
Amazon EMR Serverless	최선의 작업
아마존 EventBridge 예약 규칙	지속적
아마존 EventBridge 스키마	최선의 작업
AWS Fault Injection Service	최선의 작업
예측	최선의 작업
아마존 GameLift	최선의 작업
AWS Glue	최선의 작업
AWS Glue DataBrew	최선의 작업
AWS Ground Station	최선의 작업
아마존 GuardDuty	최선의 작업
AWS Health	최선의 작업
AWS HealthLake	지속적
AWS Identity and Access Management (IAM)	최선의 작업
IAM 액세스 분석기	최선의 작업
Amazon Inspector Classic	최선의 작업
Amazon Inspector	최선의 작업

Service	시도 유형
AWS IoT	최선의 작업
AWS IoT Analytics	지속적
AWS IoT Greengrass V1	최선의 작업
AWS IoT Greengrass V2	최선의 작업
Amazon Interactive Video Service	최선의 작업
Amazon Kinesis	최선의 작업
Amazon Data Firehose	최선의 작업
AWS Key Management Service CMK 삭제	지속적
AWS Key Management Service CMK 로테이션	최선의 작업
AWS Key Management Service 수입된 주요 자료의 만료	최선의 작업
AWS Lambda	최선의 작업
Amazon Location Service	지속적
Amazon Machine Learning	최선의 작업
Amazon Macie	최선의 작업
Amazon Managed Blockchain	최선의 작업
AWS Managed Services	최선의 작업
AWS Management Console 로그인	최선의 작업
AWS 미터링 마켓플레이스	최선의 작업
AWS Migration Hub	최선의 작업
AWS Migration Hub Refactor Spaces	최선의 작업

Service	시도 유형
AWS 모니터링	최선의 작업
AWS Network Manager	최선의 작업
아마존 OpenSearch 서비스	최선의 작업
AWS OpsWorks	지속적
AWS OpsWorks CM	최선의 작업
AWS Organizations	최선의 작업
Amazon Polly	최선의 작업
AWS Private Certificate Authority	최선의 작업
AWS Proton	최선의 작업
Amazon QLDB	지속적
아마존 QuickSight	최선의 작업
Amazon RDS	최선의 작업
AWS 휴지통	최선의 작업
Amazon Redshift	지속적
Amazon Redshift 데이터 API	최선의 작업
Amazon Redshift Serverless	최선의 작업
AWS Resource Access Manager	최선의 작업
AWS Resource Groups	최선의 작업
AWS Resource Groups Tagging API	최선의 작업
Amazon Route 53	최선의 작업

Service	시도 유형
Amazon Route 53 Recovery Readiness	최선의 작업
아마존 SageMaker	최선의 작업
Savings Plans	최선의 작업
AWS Secrets Manager	최선의 작업
AWS Security Hub	지속적
AWS Security Token Service	최선의 작업
AWS Server Migration Service	최선의 작업
AWS Service Catalog	최선의 작업
AWS Signer	지속적
Amazon Simple Email Service	최선의 작업
Amazon Simple Storage Service(S3)	지속적
Amazon S3 Glacier	최선의 작업
Outposts에서의 Amazon S3	최선의 작업
Amazon Simple Queue Service	최선의 작업
Amazon Simple Notification Service	최선의 작업
Amazon Simple Workflow Service	최선의 작업
AWS Step Functions	최선의 작업
AWS Storage Gateway	지속적
AWS Support	최선의 작업
AWS Systems Manager	최선의 작업

Service	시도 유형
Amazon Transcribe	최선의 작업
AWS Transfer Family	최선의 작업
AWS Transit Gateway	최선의 작업
Amazon Translate	지속적
AWS Trusted Advisor	최선의 작업
AWS WAF	최선의 작업
AWS WAF 지역	최선의 작업
AWS Well-Architected Tool	최선의 작업
아마존 WorkDocs	최선의 작업
아마존 WorkSpaces	최선의 작업
AWS X-Ray	최선의 작업

AWS 서비스에서 생성된 관리 이벤트

일반적으로 관리(또는 읽기 전용) 이벤트를 생성하는 API는 동사 `Describe`, `Get` 또는 `List`로 시작합니다. 아래 표에는 이 명명 규칙을 따르지 않는 AWS 서비스와 해당 서비스가 생성하는 관리 이벤트가 나열되어 있습니다. 관리 이벤트에 대한 자세한 내용은 ??? 섹션을 참조하십시오.

`Describe`, **`Get`** 또는 **`List`**로 시작하지 않는 관리 이벤트

다음 표에는`Describe`, `Get` 또는 로 시작하는 일반적인 명명 규칙을 따르지 않는 AWS 서비스 및 해당 서비스에서 생성되는 관리 이벤트가 나열되어 있습니다. `List`

Service	이벤트 이름	이벤트 유형
Alexa for Business	ResolveRoom	API 호출
Alexa for Business	SearchAddressBooks	API 호출

Service	이벤트 이름	이벤트 유형
Alexa for Business	SearchContacts	API 호출
Alexa for Business	SearchDevices	API 호출
Alexa for Business	SearchProfiles	API 호출
Alexa for Business	SearchRooms	API 호출
Alexa for Business	SearchSkillGroups	API 호출
Alexa for Business	SearchUsers	API 호출
IAM 액세스 분석기	ValidatePolicy	API 호출
AWS AdSpace 클린룸	BatchGetSchema	API 호출
AWS Amplify UI 빌더	ExportComponents	API 호출
AWS Amplify UI 빌더	ExportForms	API 호출
AWS Amplify UI 빌더	ExportThemes	API 호출
아마존 OpenSearch 서비스	BatchGetCollection	API 호출
Amazon API Gateway	ExportApi	API 호출
AWS AppConfig	ValidateConfiguration	API 호출
아마존 AppFlow	RetrieveConnectorData	API 호출
Amazon CloudWatch 애플리케이션 인사이트	UpdateApplicationDashboardConfiguration	API 호출
Amazon Athena	BatchGetNamedQuery	API 호출
Amazon Athena	BatchGetPreparedStatement	API 호출
Amazon Athena	BatchGetQueryExecution	API 호출
Amazon Athena	CheckQueryCompatibility	API 호출

Service	이벤트 이름	이벤트 유형
Amazon Athena	ExportNotebook	API 호출
AWS Auto Scaling	AreScalableTargetsRegistered	API 호출
AWS Auto Scaling	테스트	API 호출
AWS Marketplace	SearchAgreements	API 호출
AWS Backup	CreateLegalHold	API 호출
AWS Backup	ExportBackupPlanTemplate	API 호출
AWS Backup gateway	TestHypervisorConfiguration	API 호출
AWS Billing and Cost Management	AWSPaymentInstrumentGateway.Get	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.DescribeMakePaymentPage	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.DescribePaymentsDashboard	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetAccountPreferences	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetAdvancePaySummary	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetAsoBulkDownload	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetBillingContactAddress	콘솔 작업

Service	이벤트 이름	이벤트 유형
AWS Billing and Cost Management	AWSPaymentPortalService.GetDocuments	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetEligiblePaymentInstruments	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetEntitiesByIds	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetFundingDocuments	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetKybcValidationStatus	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetOneTimePasswordStatus	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentHistory	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentProfileByArn	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentProfileCurrencies	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentProfiles	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentProfileServiceProviders	콘솔 작업

Service	이벤트 이름	이벤트 유형
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentsDue	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetRemittanceInformation	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetTaxInvoiceMetadata	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetTermsAndConditionsForProgramGroup	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetTransactionsHistory	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetUnappliedFunds	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPortalService.GetUnpaidInvoices	콘솔 작업
AWS Billing and Cost Management	AWSPaymentPreferenceGateway.겟	콘솔 작업
AWS Billing and Cost Management	CancelBulkDownload	콘솔 작업
AWS Billing and Cost Management	DownloadCommercialInvoice	콘솔 작업
AWS Billing and Cost Management	DownloadCsv	콘솔 작업
AWS Billing and Cost Management	DownloadDoc	콘솔 작업

Service	이벤트 이름	이벤트 유형
AWS Billing and Cost Management	CSV 다운로드 ForBillingPeriod	콘솔 작업
AWS Billing and Cost Management	DownloadPaymentHistory	콘솔 작업
AWS Billing and Cost Management	DownloadRegistrationDocument	콘솔 작업
AWS Billing and Cost Management	DownloadTaxInvoice	콘솔 작업
AWS Billing and Cost Management	FindBankRedirectPaymentInstruments	콘솔 작업
AWS Billing and Cost Management	CSV를 찾아보세요 ForBillingPeriod	콘솔 작업
AWS Billing and Cost Management	ValidateReportDestination	콘솔 작업
AWS Billing and Cost Management	VerifyChinaPaymentEligibility	콘솔 작업
Amazon Braket	SearchCompilations	API 호출
Amazon Braket	SearchDevices	API 호출
Amazon Braket	SearchQuantumTasks	API 호출
Amazon Connect Cases	BatchGetField	API 호출
Amazon Connect Cases	SearchCases	API 호출
Amazon Connect Cases	SearchRelatedItems	API 호출
Amazon Chime	RetrieveDataExports	API 호출
Amazon Chime	SearchChannels	API 호출

Service	이벤트 이름	이벤트 유형
Amazon Chime SDK ID	DeleteProfile	서비스 이벤트
Amazon Chime SDK ID	DeleteWorkTalkAccount	서비스 이벤트
AWS 클린 룸	BatchGetSchema	API 호출
Amazon Cloud Directory	BatchRead	API 호출
Amazon Cloud Directory	LookupPolicy	API 호출
AWS CloudFormation	DetectStackDrift	API 호출
AWS CloudFormation	DetectStackResourceDrift	API 호출
AWS CloudFormation	DetectStackSetDrift	API 호출
AWS CloudFormation	EstimateTemplateCost	API 호출
AWS CloudFormation	ValidateTemplate	API 호출
AWS CloudShell	RedeemCode	API 호출
AWS CloudTrail	LookupEvents	API 호출
AWS CodeArtifact	ReadFromRepository	API 호출
AWS CodeArtifact	SearchPackages	API 호출
AWS CodeArtifact	VerifyResourcesExistForTagris	API 호출
AWS CodeBuild	BatchGetBuildBatches	API 호출
AWS CodeBuild	BatchGetBuilds	API 호출
AWS CodeBuild	BatchGetProjects	API 호출
AWS CodeBuild	BatchGetReportGroups	API 호출
AWS CodeBuild	BatchGetReports	API 호출

Service	이벤트 이름	이벤트 유형
AWS CodeBuild	BatchPutCodeCoverages	API 호출
AWS CodeBuild	BatchPutTestCases	API 호출
AWS CodeBuild	RequestBadge	서비스 이벤트
AWS CodeCommit	BatchDescribeMergeConflicts	API 호출
AWS CodeCommit	BatchGetCommits	API 호출
AWS CodeCommit	BatchGetPullRequests	API 호출
AWS CodeCommit	BatchGetRepositories	API 호출
AWS CodeCommit	EvaluatePullRequestApprovalRules	API 호출
AWS CodeCommit	GitPull	API 호출
AWS CodeDeploy	BatchGetApplicationRevisions	API 호출
AWS CodeDeploy	BatchGetApplications	API 호출
AWS CodeDeploy	BatchGetDeploymentGroups	API 호출
AWS CodeDeploy	BatchGetDeploymentInstances	API 호출
AWS CodeDeploy	BatchGetDeployments	API 호출
AWS CodeDeploy	BatchGetDeploymentTargets	API 호출
AWS CodeDeploy	BatchGetOnPremisesInstances	API 호출
아마존 CodeGuru 프로파일러	BatchGetFrameMetricData	API 호출
아마존 CodeGuru 프로파일러	SubmitFeedback	API 호출
AWS CodePipeline	PollForJobs	API 호출

Service	이벤트 이름	이벤트 유형
AWS CodePipeline	PollForThirdPartyJobs	API 호출
CodeConnections	StartAppRegistrationHandshake	API 호출
CodeConnections	스타트 투 AuthHandshake	API 호출
CodeConnections	ValidateHostWebhook	API 호출
아마존 CodeWhisperer	CreateCodeScan	API 호출
아마존 CodeWhisperer	CreateProfile	API 호출
아마존 CodeWhisperer	CreateUploadUrl	API 호출
아마존 CodeWhisperer	GenerateRecommendations	API 호출
아마존 CodeWhisperer	UpdateProfile	API 호출
Amazon Cognito 자격 증명	LookupDeveloperIdentity	API 호출
Amazon Cognito 사용자 풀	AdminGetDevice	API 호출
Amazon Cognito 사용자 풀	AdminGetUser	API 호출
Amazon Cognito 사용자 풀	AdminListDevices	API 호출
Amazon Cognito 사용자 풀	AdminListGroupsForUser	API 호출
Amazon Cognito 사용자 풀	AdminListUserAuthEvents	API 호출
Amazon Cognito 사용자 풀	Beta_Authorize_GET	서비스 이벤트
Amazon Cognito 사용자 풀	Confirm_GET	서비스 이벤트
Amazon Cognito 사용자 풀	ConfirmForgotPassword_GET	서비스 이벤트
Amazon Cognito 사용자 풀	Error_GET	서비스 이벤트
Amazon Cognito 사용자 풀	ForgotPassword_GET	서비스 이벤트

Service	이벤트 이름	이벤트 유형
Amazon Cognito 사용자 풀	IntrospectToken	API 호출
Amazon Cognito 사용자 풀	Login_Error_POST	서비스 이벤트
Amazon Cognito 사용자 풀	Login_GET	서비스 이벤트
Amazon Cognito 사용자 풀	Mfa_GET	서비스 이벤트
Amazon Cognito 사용자 풀	MfaOption_GET	서비스 이벤트
Amazon Cognito 사용자 풀	ResetPassword_GET	서비스 이벤트
Amazon Cognito 사용자 풀	Signup_GET	서비스 이벤트
Amazon Cognito 사용자 풀	UserInfo_GET	서비스 이벤트
Amazon Cognito 사용자 풀	UserInfo_포스트	서비스 이벤트
Amazon Cognito Sync	BulkPublish	API 호출
Amazon Comprehend	BatchContainsPiiEntities	API 호출
Amazon Comprehend	BatchDetectDominantLanguage	API 호출
Amazon Comprehend	BatchDetectEntities	API 호출
Amazon Comprehend	BatchDetectKeyPhrases	API 호출
Amazon Comprehend	BatchDetectPiiEntities	API 호출
Amazon Comprehend	BatchDetectSentiment	API 호출
Amazon Comprehend	BatchDetectSyntax	API 호출
Amazon Comprehend	BatchDetectTargetedSentiment	API 호출
Amazon Comprehend	ClassifyDocument	API 호출

Service	이벤트 이름	이벤트 유형
Amazon Comprehend	ContainsPiiEntities	API 호출
Amazon Comprehend	DetectDominantLanguage	API 호출
Amazon Comprehend	DetectEntities	API 호출
Amazon Comprehend	DetectKeyPhrases	API 호출
Amazon Comprehend	DetectPiiEntities	API 호출
Amazon Comprehend	DetectSentiment	API 호출
Amazon Comprehend	DetectSyntax	API 호출
Amazon Comprehend	DetectTargetedSentiment	API 호출
Amazon Comprehend	DetectToxicContent	API 호출
AWS Compute Optimizer	ExportAutoScalingGroupRecommendations	API 호출
AWS Compute Optimizer	EBS 내보내기 VolumeRecommendations	API 호출
AWS Compute Optimizer	EC를 익스포트하십시오. InstanceRecommendations	API 호출
AWS Compute Optimizer	ECS 익스포트 ServiceRecommendations	API 호출
AWS Compute Optimizer	ExportLambdaFunctionRecommendations	API 호출
AWS Compute Optimizer	RDS 익스포트 InstanceRecommendations	API 호출
AWS Config	BatchGetAggregateResourceConfig	API 호출

Service	이벤트 이름	이벤트 유형
AWS Config	BatchGetResourceConfig	API 호출
AWS Config	SelectAggregateResourceConfig	API 호출
AWS Config	SelectResourceConfig	API 호출
Amazon Connect	AdminGetEmergencyAccessToken	API 호출
Amazon Connect	SearchQueues	API 호출
Amazon Connect	SearchRoutingProfiles	API 호출
Amazon Connect	SearchSecurityProfiles	API 호출
Amazon Connect	SearchUsers	API 호출
AWS Glue DataBrew	SendProjectSessionAction	API 호출
AWS Data Pipeline	EvaluateExpression	API 호출
AWS Data Pipeline	QueryObjects	API 호출
AWS Data Pipeline	ValidatePipelineDefinition	API 호출
AWS DataSync	VerifyResourcesExistForTagris	API 호출
AWS DeepLens	BatchGetDevice	API 호출
AWS DeepLens	BatchGetModel	API 호출
AWS DeepLens	BatchGetProject	API 호출
AWS DeepLens	CreateDeviceCertificates	API 호출
AWS DeepRacer	AdminGetAccountConfig	API 호출
AWS DeepRacer	AdminListAssociatedUsers	API 호출

Service	이벤트 이름	이벤트 유형
AWS DeepRacer	TestRewardFunction	API 호출
AWS DeepRacer	VerifyResourcesExistForTagris	API 호출
Amazon Detective	BatchGetGraphMemberDatasources	API 호출
Amazon Detective	BatchGetMembershipDatasources	API 호출
Amazon Detective	SearchGraph	API 호출
아마존 DevOps 전문가	SearchInsights	API 호출
아마존 DevOps 전문가	SearchOrganizationInsights	API 호출
AWS Database Migration Service	BatchStartRecommendations	API 호출
AWS Database Migration Service	ModifyRecommendation	API 호출
AWS Database Migration Service	StartRecommendations	API 호출
AWS Database Migration Service	VerifyResourcesExistForTagris	API 호출
AWS Directory Service	VerifyTrust	API 호출
Amazon Elastic Compute Cloud	ConfirmProductInstance	API 호출
Amazon Elastic Compute Cloud	ReportInstanceStatus	API 호출

Service	이벤트 이름	이벤트 유형
Amazon Elastic 컨테이너 레지스트리	BatchCheckLayerAvailability	API 호출
Amazon Elastic 컨테이너 레지스트리	BatchGetImage	API 호출
Amazon Elastic 컨테이너 레지스트리	BatchGetImageReferrer	API 호출
Amazon Elastic 컨테이너 레지스트리	BatchGetRepository ScanningConfiguration	API 호출
Amazon Elastic 컨테이너 레지스트리	DryRunEvent	서비스 이벤트
Amazon Elastic 컨테이너 레지스트리	PolicyExecutionEvent	서비스 이벤트
Amazon Elastic Container Registry Public	BatchCheckLayerAvailability	API 호출
Amazon Elastic Container Service	DiscoverPollEndpoint	API 호출
Amazon Elastic Container Service	FindSubfleetRoute	API 호출
Amazon Elastic Container Service	ValidateResources	API 호출
Amazon Elastic Container Service	VerifyTaskSetsExist	API 호출
Amazon Elastic Kubernetes 서비스	AccessKubernetesApi	API 호출
AWS Elastic Beanstalk	CheckDNSAvailability	API 호출

Service	이벤트 이름	이벤트 유형
AWS Elastic Beanstalk	RequestEnvironmentInfo	API 호출
AWS Elastic Beanstalk	RetrieveEnvironmentInfo	API 호출
AWS Elastic Beanstalk	ValidateConfigurationSettings	API 호출
Amazon Elastic File System	NewClientConnection	서비스 이벤트
Amazon Elastic File System	UpdateClientConnection	서비스 이벤트
Amazon Elastic Transcoder	ReadJob	API 호출
Amazon Elastic Transcoder	ReadPipeline	API 호출
Amazon Elastic Transcoder	ReadPreset	API 호출
아마존 EventBridge	TestEventPattern	API 호출
아마존 EventBridge	TestScheduleExpression	API 호출
Amazon FinSpace API	BatchListCatalogNodesByDataset	API 호출
Amazon FinSpace API	BatchListNodesByDataset	API 호출
Amazon FinSpace API	BatchValidateAccess	API 호출
Amazon FinSpace API	CreateAuditRecordsQuery	API 호출
Amazon FinSpace API	SearchDatasets	API 호출
Amazon FinSpace API	SearchDatasetsV	API 호출
Amazon FinSpace API	ValidateIdToken	API 호출
AWS Firewall Manager	DisassociateAdminAccount	API 호출
Amazon Forecast	InvokeForecastEndpoint	API 호출
Amazon Forecast	QueryFeature	API 호출

Service	이벤트 이름	이벤트 유형
Amazon Forecast	QueryForecast	API 호출
Amazon Forecast	QueryWhatIfForecast	API 호출
Amazon Forecast	VerifyResourcesExistForTagris	API 호출
Amazon Fraud Detector	BatchGetVariable	API 호출
Amazon Fraud Detector	VerifyResourcesExistForTagris	API 호출
FreeRTOS	VerifyEmailAddress	API 호출
아마존 GameLift	RequestUploadCredentials	API 호출
아마존 GameLift	ResolveAlias	API 호출
아마존 GameLift	SearchGameSessions	API 호출
아마존 GameLift	ValidateMatchmakingRuleSet	API 호출
아마존 GameSparks	ExportSnapshot	API 호출
Amazon Location Service	BatchGetDevicePosition	API 호출
Amazon Location Service	CalculateRoute	API 호출
Amazon Location Service	CalculateRouteMatrix	API 호출
Amazon Location Service	SearchPlaceIndexForPosition	API 호출
Amazon Location Service	SearchPlaceIndexForSuggestions	API 호출
Amazon Location Service	SearchPlaceIndexForText	API 호출
Amazon S3 Glacier	InitiateJob	API 호출
AWS Glue	BatchGetBlueprints	API 호출

Service	이벤트 이름	이벤트 유형
AWS Glue	BatchGetColumnStatisticsForTable	API 호출
AWS Glue	BatchGetCrawlers	API 호출
AWS Glue	BatchGetCustomEntityTypes	API 호출
AWS Glue	BatchGetDataQualityResult	API 호출
AWS Glue	BatchGetDevEndpoints	API 호출
AWS Glue	BatchGetJobs	API 호출
AWS Glue	BatchGetML 트랜스폼	API 호출
AWS Glue	BatchGetPartition	API 호출
AWS Glue	BatchGetTriggers	API 호출
AWS Glue	BatchGetWorkflows	API 호출
AWS Glue	QueryJobRuns	API 호출
AWS Glue	QueryJobRunsAggregated	API 호출
AWS Glue	QueryJobs	API 호출
AWS Glue	QuerySchemaVersionMetadata	API 호출
AWS Glue	SearchTables	API 호출
AWS HealthLake	ReadResource	API 호출
AWS HealthLake	SearchWithGet	API 호출
AWS HealthLake	SearchWithPost	API 호출
AWS Identity and Access Management	GenerateCredentialReport	API 호출

Service	이벤트 이름	이벤트 유형
AWS Identity and Access Management	GenerateOrganizationsAccessReport	API 호출
AWS Identity and Access Management	GenerateServiceLastAccessedDetails	API 호출
AWS Identity and Access Management	SimulateCustomPolicy	API 호출
AWS Identity and Access Management	SimulatePrincipalPolicy	API 호출
AWS 아이덴티티 스토어	IsMemberInGroups	API 호출
AWS 아이덴티티 스토어 인증	BatchGetSession	API 호출
Amazon Inspector Classic	PreviewAgents	API 호출
Amazon Inspector Classic	BatchGetAccountStatus	API 호출
Amazon Inspector Classic	BatchGetFreeTrialInfo	API 호출
Amazon Inspector Classic	BatchGetMember	API 호출
AWS 인보이스 발행	ValidateDocumentDeliveryS3LocationInfo	API 호출
AWS IoT	SearchIndex	API 호출
AWS IoT	TestAuthorization	API 호출
AWS IoT	TestInvokeAuthorizer	API 호출
AWS IoT	ValidateSecurityProfileBehaviors	API 호출
AWS IoT Analytics	SampleChannelData	API 호출

Service	이벤트 이름	이벤트 유형
AWS IoT SiteWise	GatewaysVerifyResourcesExistForTagrisInternal	API 호출
AWS IoT Things Graph	SearchEntities	API 호출
AWS IoT Things Graph	SearchFlowExecutions	API 호출
AWS IoT Things Graph	SearchFlowTemplates	API 호출
AWS IoT Things Graph	SearchSystemInstances	API 호출
AWS IoT Things Graph	SearchSystemTemplates	API 호출
AWS IoT Things Graph	SearchThings	API 호출
AWS IoT TwinMaker	ExecuteQuery	API 호출
AWS IoT Wireless	CreateNetworkAnalyzerConfiguration	API 호출
AWS IoT Wireless	DeleteNetworkAnalyzerConfiguration	API 호출
AWS IoT Wireless	DeregisterWirelessDevice	API 호출
Amazon Interactive Video Service	BatchGetChannel	API 호출
Amazon Interactive Video Service	BatchGetStreamKey	API 호출
Amazon Kendra	BatchGetDocumentStatus	API 호출
Amazon Kendra	Query	API 호출
Amazon Managed Service for Apache Flink	DiscoverInputSchema	API 호출

Service	이벤트 이름	이벤트 유형
AWS Key Management Service	Decrypt	API 호출
AWS Key Management Service	암호화	API 호출
AWS Key Management Service	GenerateDataKey	API 호출
AWS Key Management Service	GenerateDataKeyPair	API 호출
AWS Key Management Service	GenerateDataKeyPairWithoutPlaintext	API 호출
AWS Key Management Service	GenerateDataKeyWithoutPlaintext	API 호출
AWS Key Management Service	GenerateMac	API 호출
AWS Key Management Service	GenerateRandom	API 호출
AWS Key Management Service	ReEncrypt	API 호출
AWS Key Management Service	Sign	API 호출
AWS Key Management Service	확인	API 호출
AWS Key Management Service	VerifyMac	API 호출
AWS Lake Formation	SearchDatabasesByLF 태그	API 호출

Service	이벤트 이름	이벤트 유형
AWS Lake Formation	SearchTablesByLF 태그	API 호출
AWS Lake Formation	StartQueryPlanning	API 호출
Amazon Lex	BatchCreateCustomVocabularyItem	API 호출
Amazon Lex	BatchDeleteCustomVocabularyItem	API 호출
Amazon Lex	BatchUpdateCustomVocabularyItem	API 호출
Amazon Lex	DeleteCustomVocabulary	API 호출
Amazon Lex	SearchAssociatedTranscripts	API 호출
Amazon Lightsail	GUI 생성 SessionAccessDetails	API 호출
Amazon Lightsail	DownloadDefaultKeyPair	API 호출
Amazon Lightsail	IsVpcPeered	API 호출
아마존 CloudWatch 로그	FilterLogEvents	API 호출
Amazon Macie	BatchGetCustomDataIdentifiers	API 호출
Amazon Macie	UpdateFindingsFilter	API 호출
AWS Elemental MediaConnect	ManagedDescribeFlow	API 호출
AWS Elemental MediaConnect	PrivateDescribeFlowMeta	API 호출
AWS Application Migration Service	OperationalDescribeJobLogItems	API 호출

Service	이벤트 이름	이벤트 유형
AWS Application Migration Service	OperationalDescribeJobs	API 호출
AWS Application Migration Service	OperationalDescribeReplicationConfigurationTemplates	API 호출
AWS Application Migration Service	OperationalDescribeSourceServer	API 호출
AWS Application Migration Service	OperationalGetLaunchConfiguration	API 호출
AWS Application Migration Service	OperationalListSourceServers	API 호출
AWS Application Migration Service	VerifyClientRoleForMgn	API 호출
AWS HealthOmics	VerifyResourceExists	API 호출
AWS HealthOmics	VerifyResourcesExistForTagris	API 호출
Amazon Polly	SynthesizeLongSpeech	API 호출
Amazon Polly	SynthesizeSpeech	API 호출
Amazon Polly	SynthesizeSpeechGet	API 호출
AWS 관리형 사설 네트워크를 제공하는 서비스	Ping	API 호출
AWS Proton	DeleteEnvironmentTemplateVersion	API 호출
AWS Proton	DeleteServiceTemplateVersion	API 호출

Service	이벤트 이름	이벤트 유형
Amazon QLDB	ShowCatalog	API 호출
아마존 QuickSight	GenerateEmbedUrlFo rAnonymousUser	API 호출
아마존 QuickSight	GenerateEmbedUrlFo rRegisteredUser	API 호출
아마존 QuickSight	QueryDatabase	서비스 이벤트
아마존 QuickSight	SearchAnalyses	API 호출
아마존 QuickSight	SearchDashboards	API 호출
아마존 QuickSight	SearchDataSets	API 호출
아마존 QuickSight	SearchDataSources	API 호출
아마존 QuickSight	SearchFolders	API 호출
아마존 QuickSight	SearchGroups	API 호출
아마존 QuickSight	SearchUsers	API 호출
Amazon Relational Database Service	DownloadCompleteDB LogFile	API 호출
Amazon Relational Database Service	DB 다운로드 LogFilePortion	API 호출
Amazon Rekognition	CompareFaces	API 호출
Amazon Rekognition	DetectCustomLabels	API 호출
Amazon Rekognition	DetectFaces	API 호출
Amazon Rekognition	DetectLabels	API 호출
Amazon Rekognition	DetectModerationLabels	API 호출

Service	이벤트 이름	이벤트 유형
Amazon Rekognition	DetectProtectiveEquipment	API 호출
Amazon Rekognition	DetectText	API 호출
Amazon Rekognition	RecognizeCelebrities	API 호출
Amazon Rekognition	SearchFaces	API 호출
Amazon Rekognition	SearchFacesByImage	API 호출
Amazon Rekognition	SearchUsers	API 호출
Amazon Rekognition	SearchUsersByImage	API 호출
AWS 리소스 탐색기	BatchGetView	API 호출
AWS 리소스 탐색기	검색	API 호출
AWS Resource Groups	SearchResources	API 호출
AWS Resource Groups	ValidateResourceSharing	API 호출
AWS RoboMaker	BatchDescribeSimulationJob	API 호출
Amazon Route 53	TestDNSAnswer	API 호출
Amazon Route 53 도메인	checkAvailabilities	API 호출
Amazon Route 53 도메인	CheckDomainAvailability	API 호출
Amazon Route 53 도메인	checkDomainTransferability	API 호출
Amazon Route 53 도메인	CheckDomainTransferability	API 호출
Amazon Route 53 도메인	isEmailReachable	API 호출
Amazon Route 53 도메인	searchDomains	API 호출
Amazon Route 53 도메인	sendVerificationMessage	API 호출

Service	이벤트 이름	이벤트 유형
Amazon Route 53 도메인	ViewBilling	API 호출
Amazon Route 53 도메인	viewBilling	API 호출
아마존 CloudWatch 럼	BatchGetRumMetricDefinitions	API 호출
Amazon Simple Storage Service(S3)	echo	API 호출
Amazon Simple Storage Service(S3)	GenerateInventory	서비스 이벤트
아마존 SageMaker	BatchDescribeModelPackage	API 호출
아마존 SageMaker	DeleteModelCard	API 호출
아마존 SageMaker	QueryLineage	API 호출
아마존 SageMaker	RenderUiTemplate	API 호출
아마존 SageMaker	검색	API 호출
아마존 EventBridge 스키마	ExportSchema	API 호출
아마존 EventBridge 스키마	SearchSchemas	API 호출
Amazon SimpleDB	DomainMetadata	API 호출
AWS Secrets Manager	ValidateResourcePolicy	API 호출
AWS Service Catalog	ScanProvisionedProducts	API 호출
AWS Service Catalog	SearchProducts	API 호출
AWS Service Catalog	SearchProductsAsAdmin	API 호출
AWS Service Catalog	SearchProvisionedProducts	API 호출
Amazon SES	BatchGetMetricData	API 호출

Service	이벤트 이름	이벤트 유형
Amazon SES	TestRenderEmailTemplate	API 호출
Amazon SES	TestRenderTemplate	API 호출
Amazon Simple Notification Service	CheckIfPhoneNumber IsOptedOut	API 호출
AWS SQL Workbench	BatchGetNotebookCell	API 호출
AWS SQL Workbench	ExportNotebook	API 호출
Amazon EC2 Systems Manager	ExecuteApi	API 호출
AWS Systems Manager Incident Manager	DeleteContactChannel	API 호출
AWS IAM Identity Center	IsMemberInGroup	API 호출
AWS IAM Identity Center	SearchGroups	API 호출
AWS IAM Identity Center	SearchUsers	API 호출
AWS STS	AssumeRole	API 호출
AWS STS	AssumeRoleWithSAML	API 호출
AWS STS	AssumeRoleWithWebIdentity	API 호출
AWS STS	DecodeAuthorizationMessage	API 호출
AWS 세금 설정	BatchGetTaxExemptions	API 호출
AWS WAFV2	CheckCapacity	API 호출
AWS WAFV2	GenerateMobileSdkR eleaseUrl	API 호출
AWS Well-Architected Tool	ExportLens	API 호출

Service	이벤트 이름	이벤트 유형
AWS Well-Architected Tool	TagResource	API 호출
AWS Well-Architected Tool	UntagResource	API 호출
AWS Well-Architected Tool	UpdateGlobalSettings	API 호출
Amazon Q Connect	QueryAssistant	API 호출
Amazon Q Connect	SearchContent	API 호출
Amazon Q Connect	SearchSessions	API 호출
아마존 WorkDocs	AbortDocumentVersionUpload	API 호출
아마존 WorkDocs	AddUsersToGroup	API 호출
아마존 WorkDocs	BatchGetUsers	API 호출
아마존 WorkDocs	CheckAlias	API 호출
아마존 WorkDocs	CompleteDocumentVersionUpload	API 호출
아마존 WorkDocs	CreateAnnotation	API 호출
아마존 WorkDocs	CreateComment	API 호출
아마존 WorkDocs	CreateFeedbackRequest	API 호출
아마존 WorkDocs	CreateFolder	API 호출
아마존 WorkDocs	CreateGroup	API 호출
아마존 WorkDocs	CreateShare	API 호출
아마존 WorkDocs	CreateUser	API 호출
아마존 WorkDocs	DeleteAnnotation	API 호출
아마존 WorkDocs	DeleteComment	API 호출

Service	이벤트 이름	이벤트 유형
아마존 WorkDocs	DeleteDocument	API 호출
아마존 WorkDocs	DeleteFeedbackRequest	API 호출
아마존 WorkDocs	DeleteFolder	API 호출
아마존 WorkDocs	DeleteFolderContents	API 호출
아마존 WorkDocs	DeleteGroup	API 호출
아마존 WorkDocs	DeleteOrganizationShare	API 호출
아마존 WorkDocs	DeleteUser	API 호출
아마존 WorkDocs	DownloadDocumentVersion	API 호출
아마존 WorkDocs	DownloadDocumentVersionUnderlays	API 호출
아마존 WorkDocs	InitiateDocumentVersionUpload	API 호출
아마존 WorkDocs	LogoutUser	API 호출
아마존 WorkDocs	PaginatedOrganizationActivity	API 호출
아마존 WorkDocs	PublishAnnotations	API 호출
아마존 WorkDocs	PublishComments	API 호출
아마존 WorkDocs	RestoreDocument	API 호출
아마존 WorkDocs	RestoreFolder	API 호출
아마존 WorkDocs	SearchGroups	API 호출
아마존 WorkDocs	SearchOrganizationUsers	API 호출
아마존 WorkDocs	TransferUserResources	API 호출

Service	이벤트 이름	이벤트 유형
아마존 WorkDocs	UpdateAnnotation	API 호출
아마존 WorkDocs	UpdateComment	API 호출
아마존 WorkDocs	UpdateDocument	API 호출
아마존 WorkDocs	UpdateDocumentVersion	API 호출
아마존 WorkDocs	UpdateFolder	API 호출
아마존 WorkDocs	UpdateGroup	API 호출
아마존 WorkDocs	UpdateOrganization	API 호출
아마존 WorkDocs	UpdateUser	API 호출
아마존 WorkMail	AssumeImpersonationRole	API 호출
아마존 WorkMail	QueryDnsRecords	API 호출
아마존 WorkMail	SearchMembers	API 호출
아마존 WorkMail	TestAvailabilityConfiguration	API 호출
아마존 WorkMail	TestInboundMailFlowRules	API 호출
아마존 WorkMail	TestOutboundMailFlowRules	API 호출

EventBridge 이벤트 세부 정보 참조

EventBridge 자체적으로 다음 이벤트를 내보냅니다. 이러한 이벤트는 다른 AWS 서비스와 마찬가지로 기본 이벤트 버스로 자동 전송됩니다.

모든 이벤트에 포함되는 메타데이터 필드의 정의는 을 참조하십시오the section called “이벤트 구조 참조”.

주제

- 스키마 생성됨
- 스키마 버전 생성됨

예정된 이벤트

다음은 Scheduled Event 이벤트의 세부 정보 필드입니다.

source및 detail-type 필드는 EventBridge 이벤트에 대한 특정 값을 포함하므로 포함됩니다. 모든 이벤트에 포함된 다른 메타데이터 필드의 정의는 을 참조하십시오the section called “이벤트 구조 참조”.

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

`detail-type`

이벤트의 유형을 식별합니다.

이 이벤트의 경우 이 값은 입니다Scheduled Event.

필수 여부: 예

`source`

이벤트를 생성한 서비스를 식별합니다. EventBridge 이벤트의 경우 이 값은 입니다aws.events.

필수 여부: 예

`detail`

이벤트에 대한 정보를 포함하는 JSON 객체입니다. 이벤트를 생성하는 서비스에 따라 이 필드의 내용이 결정됩니다.

필수 여부: 예

이 객체에는 Scheduled Event 이벤트에 대한 필수 필드가 없습니다.

Example 예약된 이벤트 이벤트의 예

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

스키마 생성됨

다음은 Schema Created 이벤트의 세부 정보 필드입니다.

스키마가 생성되면 Schema Created a와 Schema Version Created 이벤트를 모두 EventBridge 보냅니다.

source및 detail-type 필드는 EventBridge 이벤트에 대한 특정 값을 포함하므로 포함됩니다. 모든 이벤트에 포함된 다른 메타데이터 필드의 정의는 을 참조하십시오the section called “이벤트 구조 참조”.

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

`detail-type`

이벤트의 유형을 식별합니다.

이 이벤트의 경우 이 값은 입니다Schema Created.

필수 여부: 예

source

이벤트를 생성한 서비스를 식별합니다. EventBridge 이벤트의 경우 이 값은 입니다aws.schemas.

필수 여부: 예

detail

이벤트에 대한 정보를 포함하는 JSON 객체입니다. 이벤트를 생성하는 서비스에 따라 이 필드의 내용이 결정됩니다.

필수 여부: 예

이 이벤트의 경우 이 데이터에는 다음이 포함됩니다.

SchemaName

스키마의 이름입니다.

필수 여부: 예

SchemaType

스키마의 유형입니다.

유효한 값: OpenApi3 | JSONSchemaDraft4

필수 여부: 예

RegistryName

해당 스키마가 포함된 레지스트리의 이름입니다.

필수 여부: 예

CreationDate

스키마가 생성된 날짜.

필수 여부: 예

Version

스키마의 버전입니다.

Schema Created이벤트의 경우 이 값은 항상 1 유효합니다.

필수 여부: 예

Example 예제: 스키마 생성 이벤트

```
{
  "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"
  }
}
```

스키마 버전 생성됨

다음은 `Schema Version Created` 이벤트의 세부 정보 필드입니다.

스키마가 생성되면 `Schema Created` a와 `Schema Version Created` 이벤트를 모두 EventBridge 보냅니다.

`source`및 `detail-type` 필드는 EventBridge 이벤트에 대한 특정 값을 포함하므로 포함됩니다. 모든 이벤트에 포함된 다른 메타데이터 필드의 정의는 을 참조하십시오the section called “이벤트 구조 참조”.

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

```
    "CreationDate" : "DateTime",
    "Version" : "Number"
  }
}
```

`detail-type`

이벤트의 유형을 식별합니다.

이 이벤트의 경우 이 값은 입니다`Schema Version Created`.

필수 여부: 예

`source`

이벤트를 생성한 서비스를 식별합니다. EventBridge 이벤트의 경우 이 값은 입니다`aws.schemas`.

필수 여부: 예

`detail`

이벤트에 대한 정보를 포함하는 JSON 객체입니다. 이벤트를 생성하는 서비스에 따라 이 필드의 내용이 결정됩니다.

필수 여부: 예

이 이벤트의 경우 이 데이터에는 다음이 포함됩니다.

`SchemaName`

스키마의 이름입니다.

필수 여부: 예

`SchemaType`

스키마의 유형입니다.

유효한 값: `OpenApi3` | `JSONSchemaDraft4`

필수 여부: 예

`RegistryName`

해당 스키마가 포함된 레지스트리의 이름입니다.

필수 여부: 예

`CreationDate`

스키마 버전이 생성된 날짜.

필수 여부: 예

`Version`

스키마의 버전입니다.

필수 여부: 예

Example 예제: 스키마 버전 생성 이벤트

```
{
  "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"
  }
}
```

Amazon의 SaaS 파트너로부터 이벤트 받기 EventBridge

SaaS 파트너 애플리케이션과 서비스에서 이벤트를 수신하려면 파트너의 파트너 이벤트 소스가 필요합니다. 그런 다음, 파트너 이벤트 버스를 생성하여 해당 파트너 이벤트 소스와 매칭할 수 있습니다.

다음 동영상은 SaaS 파트너와의 SaaS 통합을 다룹니다 EventBridge. 서비스형 소프트웨어 (SaaS) 파트너

주제

지원되는 SaaS 파트너 통합

EventBridge 다음 SaaS 파트너 통합을 지원합니다.

- 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 셀러 파트너 API

파트너 이벤트 소스는 다음 리전에서 사용 가능합니다.

코드	명칭
us-east-1	미국 동부(버지니아 북부)
us-east-2	미국 동부(오하이오)
us-west-1	미국 서부(캘리포니아 북부)
us-west-2	미국 서부(오레곤)
ca-central-1	캐나다(중부)
eu-central-1	유럽(프랑크푸르트)
eu-central-2	유럽(취리히)
eu-west-1	유럽(아일랜드)

코드	명칭
eu-west-2	유럽(런던)
eu-west-3	유럽(파리)
eu-north-1	유럽(스톡홀름)
eu-south-1	유럽(밀라노)
eu-south-2	유럽(스페인)
af-south-1	아프리카(케이프타운)
ap-south-1	아시아 태평양(뭄바이)
ap-south-2	아시아 태평양(하이데라바드)
ap-east-1	아시아 태평양(홍콩)
ap-northeast-1	아시아 태평양(도쿄)
ap-northeast-2	아시아 태평양(서울)
ap-northeast-3	아시아 태평양(오사카)
ap-southeast-1	아시아 태평양(싱가포르)
ap-southeast-2	아시아 태평양(시드니)
ap-southeast-3	아시아 태평양(자카르타)
ap-southeast-4	아시아 태평양(멜버른)
cn-north-1	중국(베이징)
cn-northwest-1	중국(닝샤)
me-central-1	중동(UAE)
me-south-1	중동(바레인)

코드	명칭
sa-east-1	남아메리카(상파울루)
il-central-1	이스라엘(텔아비브)

SaaS 통합을 통해 이벤트를 EventBridge 수신하도록 Amazon 구성

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 Partner event sources(파트너 이벤트 소스)를 선택하십시오.
3. 원하는 파트너를 찾은 다음, 해당 파트너에 대해 설정을 선택하세요.
4. 계정 ID를 클립보드에 복사하려면 복사를 선택하세요.
5. 탐색 창에서 Partner event sources(파트너 이벤트 소스)를 선택하십시오.
6. 파트너 웹사이트로 이동하여 지침에 따라 계정 ID를 사용해 파트너 이벤트 소스를 생성하세요. 생성한 이벤트 소스는 본인 계정에서만 사용할 수 있습니다.
7. EventBridge 콘솔로 돌아가서 탐색 창에서 파트너 이벤트 소스를 선택합니다.
8. 파트너 이벤트 소스 옆에 있는 버튼을 선택하고 이벤트 버스에 연결을 선택하세요.

   해당 이벤트 소스의 상태가 `Pending`에서 `Active`로 변경되고 파트너 이벤트 소스 이름과 일치하도록 이벤트 버스 이름이 업데이트됩니다. 이제 파트너 이벤트 소스의 이벤트와 일치하는 규칙 생성을 시작할 수 있습니다. 자세한 정보는 SaaS 파트너 이벤트와 일치하는 규칙 생성을 참조하세요.

   > **Note**
   >
   > 이벤트 버스와 연결되지 않은 파트너 이벤트 소스에 파트너가 게시한 모든 이벤트는 즉시 삭제됩니다. 해당 이벤트는 유휴 상태로 유지되지 않습니다. EventBridge

SaaS 파트너 이벤트와 일치하는 규칙 생성

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 규칙을 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙에 대해 이름과 설명을 입력하십시오.

규칙은 동일한 지역과 동일한 이벤트 버스의 다른 규칙과 동일한 이름을 가질 수 없습니다.

5. 이벤트 버스에서 이 규칙과 연결할 이벤트 버스를 선택합니다. 이 규칙이 자신의 계정에서 발생하는 이벤트와 일치하도록 하려면 AWS 기본 이벤트 버스(default event bus)를 선택합니다. 계정의 AWS 서비스가 이벤트를 출력하면 항상 계정의 기본 이벤트 버스로 이동합니다.
6. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다.
7. 다음을 선택합니다.
8. 이벤트 소스에서 기타를 선택합니다.
9. (선택 사항) 샘플 이벤트의 경우 이벤트 유형을 선택합니다.
10. 이벤트 패턴에는 JSON 이벤트 패턴을 입력합니다.
11. 다음을 선택합니다.
12. 대상 유형에서 AWS 서비스를 선택합니다.
13. 대상 선택에서 이벤트 패턴과 일치하는 이벤트가 EventBridge 감지되었을 때 정보를 보내려는 AWS 서비스를 선택합니다.
14. 표시되는 필드는 선택한 서비스에 따라 달라집니다. 필요에 따라 이 대상 유형과 관련된 정보를 입력합니다.
15. 많은 대상 유형의 경우 대상에 이벤트를 보낼 수 있는 권한이 EventBridge 필요합니다. 이러한 경우 규칙을 실행하는 데 필요한 IAM 역할을 생성할 EventBridge 수 있습니다. 다음 중 하나를 수행하십시오.

    - IAM 역할을 자동으로 생성하려면 이 특정 리소스에 대해 새 역할 생성을 선택합니다.
    - 이전에 생성한 IAM 역할을 사용하려면 기존 역할 사용을 선택하고 드롭다운 목록에서 기존 역할을 선택합니다.

16. (선택 사항)추가 설정에서 다음을 수행합니다.

    a. 최대 이벤트 기간(Maximum age of event)에 1분(00:01)에서 24시간(24:00) 사이의 값을 입력합니다.

    b. 재시도(Retry attempts)에 0에서 185 사이의 숫자를 입력합니다.

    c. 데드레터 대기열의 경우 표준 Amazon SQS 대기열을 데드레터 대기열로 사용할지 여부를 선택합니다. EventBridge 타겟으로 성공적으로 전송되지 않은 경우 이 규칙과 일치하는 이벤트를 데드레터 대기열로 전송합니다. 다음 중 하나를 수행하십시오.

        - 배달 못한 편지 대기열을 사용하지 않으려면 없음(None)을 선택합니다.

- 현재 AWS 계정에서 배달 못한 편지 대기열로 사용할 대기열 선택(Select an Amazon SQS queue in the current account to use as the dead-letter queue)을 선택하고 드롭다운 목록에서 사용할 대기열을 선택합니다.
- 다른 AWS 계정의 Amazon SQS 대기열을 데드레터 대기열로 선택을 선택한 다음 사용할 대기열의 ARN을 입력합니다. 메시지를 전송할 EventBridge 권한을 부여하는 리소스 기반 정책을 대기열에 연결해야 합니다. 자세한 정보는 DLQ(Dead Letter Queue)에 권한 부여을 참조하세요.

17. (선택 사항) 이 규칙에 다른 대상을 추가하려면 다른 대상 추가를 선택합니다.
18. 다음을 선택합니다.
19. (선택 사항)규칙에 대해 하나 이상의 태그를 입력하세요. 자세한 정보는 아마존 EventBridge 태그을 참조하세요.
20. 다음을 선택합니다.
21. 규칙의 세부 정보를 검토하고 규칙 생성을 선택합니다.

함수 URL을 사용하여 AWS Lambda 이벤트를 수신합니다.

> **Note**
>
> 파트너가 인바운드 웹훅에 액세스할 수 있도록 타사 파트너가 보낸 인증 서명을 확인하여 Lambda 애플리케이션 수준에서 보호되는 Open Lambda를 사용자 AWS 계정에 생성하고 있습니다. 보안 팀과 함께 이 구성을 검토하세요. 자세한 내용은 Lambda 함수 URL에 대한 보안 및 인증 모델 섹션을 참조하세요.

Amazon EventBridge 이벤트 버스는 AWS CloudFormation 템플릿으로 생성된 AWS Lambda 함수 URL을 사용하여 지원되는 SaaS 공급자로부터 이벤트를 수신할 수 있습니다. 함수 URL을 사용하면 이벤트 데이터가 Lambda 함수로 전송됩니다. 그런 다음 함수는 이 데이터를 이벤트로 변환하여 이벤트 버스에서 수집한 후 이벤트 버스로 EventBridge 전송하여 처리합니다. 이벤트가 이벤트 버스에 있으면 규칙을 사용하여 이벤트를 필터링하고 구성된 입력 변환을 적용한 후 올바른 대상으로 라우팅할 수 있습니다.

> **Note**
>
> Lambda 함수 URL을 생성하면 월별 비용이 증가합니다. 자세한 내용은 AWS Lambda 요금을 참조하십시오.

연결을 EventBridge 설정하려면 먼저 연결을 설정하려는 SaaS 공급자를 선택합니다. 그런 다음 해당 공급자와 함께 만든 서명 암호를 제공하고 이벤트를 전송할 EventBridge 이벤트 버스를 선택합니다. 마지막으로 AWS CloudFormation 템플릿을 사용하고 연결을 완료하는 데 필요한 리소스를 생성합니다.

현재 EventBridge Lambda 함수 URL을 사용하여 사용할 수 있는 SaaS 공급자는 다음과 같습니다.

- GitHub
- Twilio

주제

- Twilio에 대한 연결 설정
- 웹후크 보안 암호 또는 인증 토큰 업데이트
- Lambda 함수 업데이트
- 사용 가능한 이벤트 유형
- 할당량, 오류 코드, 전송 재시도

GitHub에 대한 연결 설정

1단계: 스택 생성 AWS CloudFormation

먼저 Amazon EventBridge 콘솔을 사용하여 CloudFormation 스택을 생성합니다.

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 빠른 시작을 선택합니다.
3. Lambda fURL을 사용하는 인바운드 웹후크에서 시작하기를 선택합니다.
4. GitHub에서 설정을 선택합니다.
5. 1단계: 이벤트 버스 선택의 드롭다운 목록에서 이벤트 버스를 선택합니다. 이 이벤트 버스는 사용자가 GitHub에 제공하는 Lambda 함수 URL에서 데이터를 수신합니다. 새 이벤트 버스를 선택하여 이벤트 버스를 생성할 수도 있습니다.
6. 2단계: 설정을 사용하여 CloudFormation 아래에서 새 GitHub 웹훅을 선택합니다.
7. 내가 만든 인바운드 웹후크에 공개적으로 액세스할 수 있음을 승인합니다를 선택하고 확인을 선택합니다.
8. 스택의 이름을 입력합니다.
9. 파라미터에서 올바른 이벤트 버스가 나열되어 있는지 확인한 다음, GitHubWebhookSecret에 대한 보안 토큰을 지정합니다. 보안 토큰 생성에 대한 자세한 내용은 GitHub 설명서의 Setting your secret token을 참조하세요.
10. 기능 및 변환에서 다음을 각각 선택합니다.
    - 그러면 IAM 리소스가 AWS CloudFormation 생성될 수 있다는 점을 인정합니다.
    - 사용자 지정 이름을 사용하여 IAM 리소스를 생성할 AWS CloudFormation 수도 있다는 점을 인정합니다.
    - 이를 위해서는 다음과 같은 기능이 AWS CloudFormation 필요할 수 있다는 점을 인정합니다. **`CAPABILITY_AUTO_EXPAND`**

11. 스택 생성을 선택합니다.

2단계: GitHub 웹후크 생성

다음으로 GitHub에 웹후크를 생성합니다. 이 단계를 완료하려면 2단계에서 생성한 Lambda 함수 URL과 보안 토큰이 모두 필요합니다. 자세한 내용은 GitHub 설명서의 Creating webhooks를 참조하세요.

Twilio에 대한 연결 설정

1단계: Twilio 인증 토큰 찾기

Twilio와 사이의 연결을 EventBridge 설정하려면 먼저 Twilio 계정의 인증 토큰 또는 비밀번호를 Twilio 사용하여 연결을 설정하십시오. 자세한 내용은 Twilio 설명서에서 Auth Tokens and How To Change Them을 참조하세요.

2단계: 스택 생성 AWS CloudFormation

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 빠른 시작을 선택합니다.
3. Lambda fURL을 사용하는 인바운드 웹후크에서 시작하기를 선택합니다.
4. Twilio에서 설정을 선택합니다.
5. 1단계: 이벤트 버스 선택에서 드롭다운 목록에서 이벤트 버스를 선택합니다. 이 이벤트 버스는 사용자가 Twilio에 제공하는 Lambda 함수 URL에서 데이터를 수신합니다. 새 이벤트 버스를 선택하여 이벤트 버스를 생성할 수도 있습니다.
6. 2단계: 설정을 사용하여 CloudFormation 아래에서 새 Twilio 웹훅을 선택합니다.
7. 내가 만든 인바운드 웹후크에 공개적으로 액세스할 수 있음을 승인합니다를 선택하고 확인을 선택합니다.
8. 스택의 이름을 입력합니다.
9. 파라미터에서 올바른 이벤트 버스가 나열되어 있는지 확인한 다음, 1단계에서 생성한 TwilioWebhookSecret을(를) 입력합니다.
10. 기능 및 변환에서 다음을 각각 선택합니다.
    - 그러면 IAM 리소스가 AWS CloudFormation 생성될 수 있다는 점을 인정합니다.
    - 사용자 지정 이름을 사용하여 IAM 리소스를 생성할 AWS CloudFormation 수도 있다는 점을 인정합니다.

- 이를 위해서는 다음과 같은 기능이 AWS CloudFormation 필요할 수 있다는 점을 인정합니다. CAPABILITY_AUTO_EXPAND

11. 스택 생성을 선택합니다.

3단계: Twilio 웹후크 생성

Lambda 함수 URL을 설정한 후에는 이벤트 데이터를 전송할 수 있도록 Twilio에 이를 제공해야 합니다. 자세한 내용을 알아보려면 Twilio 설명서의 Configure your public URL with Twilio를 참조하세요.

웹후크 보안 암호 또는 인증 토큰 업데이트

GitHub 보안 암호 업데이트

> **Note**
>
> GitHub는 동시에 두 개의 보안 암호를 사용하는 것을 지원하지 않습니다. 스택의 GitHub 비밀과 비밀이 동기화되지 않는 경우 리소스 다운타임이 발생할 수 있습니다. AWS CloudFormation GitHub비밀이 동기화되지 않은 상태에서 전송된 메시지는 잘못된 서명으로 인해 실패합니다. GitHub및 CloudFormation 비밀이 동기화될 때까지 기다린 다음 다시 시도하세요.

1. 새 GitHub 보안 암호를 생성합니다. 자세한 내용은 GitHub 문서의 Encrypted secrets를 참조하세요.
2. https://console.aws.amazon.com/cloudformation 에서 AWS CloudFormation 콘솔을 여세요.
3. 탐색 창에서 스택을 선택합니다.
4. 업데이트할 보안 암호가 포함된 웹후크 스택을 선택합니다.
5. 업데이트를 선택합니다.
6. 현재 템플릿 사용이 선택되어 있는지 확인하고 다음을 선택합니다.
7. 기존 값 사용에서 선택을 취소하고 1단계에서 만든 새 GitHub 암호를 입력하고 다음을 선택합니다. GitHubWebhookSecret
8. 다음을 선택합니다.
9. 스택 업데이트를 선택합니다.

보안 암호가 전파하는 데 최대 1시간까지 걸릴 수 있습니다. 이 가동 중지 시간을 줄이기 위해 Lambda 실행 컨텍스트를 새로 고칠 수 있습니다.

Twilio 보안 암호 업데이트

> **Note**
>
> Twilio는 동시에 두 개의 보안 암호를 사용하는 것을 지원하지 않습니다. AWS CloudFormation 스택의 Twilio 비밀과 비밀이 동기화되지 않은 경우 리소스 다운타임이 발생할 수 있습니다. Twilio비밀이 동기화되지 않은 상태에서 전송된 메시지는 잘못된 서명으로 인해 실패합니다. Twilio및 CloudFormation 비밀이 동기화될 때까지 기다린 다음 다시 시도하세요.

1. 새 Twilio 보안 암호를 생성합니다. 자세한 내용은 Twilio 설명서에서 Auth Tokens and How To Change Them을 참조하세요.
2. https://console.aws.amazon.com/cloudformation 에서 AWS CloudFormation 콘솔을 여세요.
3. 탐색 창에서 스택을 선택합니다.
4. 업데이트할 보안 암호가 포함된 웹후크 스택을 선택합니다.
5. 업데이트를 선택합니다.
6. 현재 템플릿 사용이 선택되어 있는지 확인하고 다음을 선택합니다.
7. 기존 값 사용에서 선택을 취소하고 1단계에서 만든 새 Twilio 암호를 입력하고 다음을 선택합니다. TwilioWebhookSecret
8. 다음을 선택합니다.
9. 스택 업데이트를 선택합니다.

보안 암호가 전파하는 데 최대 1시간까지 걸릴 수 있습니다. 이 가동 중지 시간을 줄이기 위해 Lambda 실행 컨텍스트를 새로 고칠 수 있습니다.

Lambda 함수 업데이트

스택에서 생성된 Lambda 함수는 기본 CloudFormation 웹후크를 생성합니다. 사용자 지정 로깅과 같은 특정 사용 사례에 맞게 Lambda 함수를 사용자 지정하려면 콘솔을 사용하여 함수에 액세스한 CloudFormation 다음 Lambda 콘솔을 사용하여 Lambda 함수 코드를 업데이트하십시오.

Lambda 함수에 액세스

1. https://console.aws.amazon.com/cloudformation 에서 콘솔을 엽니다. AWS CloudFormation
2. 탐색 창에서 스택을 선택합니다.

3. 업데이트할 Lambda 함수가 포함된 웹후크 스택을 선택합니다.
4. 리소스 탭을 선택합니다.
5. Lambda 콘솔에서 Lambda 함수를 열려면 물리적 ID에서 Lambda 함수의 ID를 선택합니다.

이제 Lambda 함수에 액세스했으므로 Lambda 콘솔을 사용하여 함수 코드를 업데이트하세요.

Lambda 함수 코드 업데이트

1. 작업에서 함수 내보내기를 선택합니다.
2. 배포 패키지 다운로드를 선택하고 파일을 컴퓨터에 저장합니다.
3. 배포 패키지.zip 파일의 압축을 풀고, `app.py` 파일을 업데이트하고, 원본.zip 파일의 모든 파일이 포함되도록 업데이트된 배포 패키지를 압축합니다.
4. Lambda 콘솔에서 코드 탭을 선택합니다.
5. 코드 소스(Code source)에서 업로드(Upload from)를 선택합니다.
6. .zip 파일을 선택한 후 [업로드(Upload)]를 선택합니다.
    - 파일 선택기에서 업데이트한 파일을 선택하고 열기와 저장을 차례로 선택합니다.
7. 작업에서 새 버전 발행을 선택합니다.

사용 가능한 이벤트 유형

현재 이벤트 버스에서 지원되는 CloudFormation 이벤트 유형은 다음과 같습니다.

- GitHub— 모든 이벤트 유형이 지원됩니다.
- Twilio - 이벤트 후 웹후크가 지원됩니다.

할당량, 오류 코드, 전송 재시도

할당량

웹후크로 들어오는 요청 수는 기본 AWS 서비스에 따라 제한됩니다. 다음 표에는 관련 할당량이 포함되어 있습니다.

Service	할당량
AWS Lambda	기본값: 동시 실행 10개

Service	할당량
	할당량 증가 요청을 비롯한 할당량에 대한 자세한 내용은 Lambda 할당량을 참조하세요.
AWS Secrets Manager	기본값: 초당 요청 5,000개 할당량 증가 요청을 비롯한 할당량에 대한 자세한 내용은 AWS Secrets Manager 할당량을 참조하세요. Note AWS Secrets Manager Python 캐싱 클라이언트를 사용하면 초당 요청 수가 최소화됩니다.
아마존 EventBridge	액션의 최대 입력 크기는 256KB입니다 PutEvents . EventBridge 지역 기반 요금 할당량을 적용합니다. 자세한 정보는 ???을 참조하세요.

오류 코드

각 AWS 서비스는 오류 발생 시 특정 오류 코드를 반환합니다. 다음 표에는 관련 오류 코드가 포함되어 있습니다.

Service	오류 코드	설명
AWS Lambda	429 “” TooManyRequestsException	동시 실행 할당량을 초과했습니다.
AWS Secrets Manager	500 “Internal Server Error”	초당 요청 할당량을 초과했습니다.
아마존 EventBridge	500 “Internal Server Error”	해당 리전의 비율 할당량을 초과했습니다.

이벤트 재전송

오류가 발생하면 영향을 받은 이벤트의 전송을 다시 시도할 수 있습니다. SaaS 공급자마다 재시도 절차가 다릅니다.

GitHub

GitHub 웹후크 API를 사용하여 웹후크 호출의 전달 상태를 확인하고 필요한 경우 이벤트를 다시 전송합니다. 자세한 내용은 다음 GitHub 설명서를 참조하세요.

- 조직 - 조직 웹후크에 대한 전송 재전달
- 리포지토리 - 리포지토리 웹후크에 대한 전송 재전달
- 앱 - 앱 웹후크에 대한 전송 재전달

Twilio

Twilio 사용자는 연결 재정의를 통해 이벤트 재시도 옵션을 사용자 지정할 수 있습니다. 자세한 내용은 Twilio 설명서의 Webhooks (HTTP callbacks): Connection Overrides를 참조하세요.

Salesforce에서 이벤트 수신

EventBridge Amazon을 사용하여 다음과 같은 Salesforce 방법으로 이벤트를 수신할 수 있습니다.

- Salesforce's이벤트 버스 릴레이 기능을 사용하여 EventBridge 파트너 이벤트 버스에서 직접 이벤트를 수신합니다.
- Amazon에서 데이터 Salesforce 소스로 AppFlow 사용하는 플로우를 구성함으로써 AppFlow 그러면 Amazon은 파트너 Salesforce 이벤트 EventBridge 버스를 사용하여 이벤트를 에 전송합니다.

API 대상을 사용하여 Salesforce에 이벤트 정보를 전송할 수 있습니다. 이벤트가 Salesforce에 전송되면 Flows 또는 Apex 트리거로 이벤트를 처리할 수 있습니다. Salesforce API 대상 설정에 대한 자세한 내용은 ??? 섹션을 참조하세요.

주제

- 이벤트 버스 릴레이를 사용하여 Salesforce에서 이벤트 수신
- Amazon Salesforce 사용 이벤트 수신 AppFlow

이벤트 버스 릴레이를 사용하여 Salesforce에서 이벤트 수신

1단계: Salesforce 이벤트 버스 릴레이 및 EventBridge 파트너 이벤트 소스 설정

에서 Salesforce 이벤트 릴레이 구성을 만들면 보류 상태의 파트너 이벤트 소스가 Salesforce 생성됩니다. EventBridge

Salesforce 이벤트 버스 릴레이를 구성하려면 다음을 수행하세요.

1. REST API 도구 설정
2. (선택 사항) 플랫폼 이벤트 정의
3. 사용자 지정 플랫폼 이벤트를 위한 채널 생성
4. 채널 멤버를 생성하여 사용자 지정 플랫폼 이벤트 연결
5. 명명된 보안 인증 생성
6. 이벤트 릴레이 구성 생성

2단계: EventBridge 콘솔에서 Salesforce 파트너 이벤트 소스를 활성화하고 이벤트 릴레이를 시작합니다.

1. EventBridge 콘솔에서 파트너 이벤트 소스 페이지를 엽니다.
2. 1단계에서 생성한 Salesforce 파트너 이벤트 소스를 선택합니다.
3. 이벤트 버스에 연결을 선택합니다.
4. 파트너 이벤트 버스의 이름을 확인합니다.
5. Associate(연결)를 선택합니다.
6. 이벤트 릴레이 시작

이제 Event Bus Relay를 설정하고 시작하고 파트너 이벤트 소스를 구성했으므로 이벤트에 반응하여 데이터를 필터링하고 대상에 전송하는 EventBridge 규칙을 만들 수 있습니다.

Amazon Salesforce 사용 이벤트 수신 AppFlow

Amazon은 이벤트 Salesforce 봉투의 EventBridge 이벤트를 AppFlow 캡슐화합니다. 다음 예는 EventBridge 파트너 Salesforce 이벤트 버스에서 수신한 이벤트를 보여줍니다.


```
{
    "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"
    }
}
```

1단계: 파트너 이벤트 AppFlow Salesforce 소스로 사용하도록 Amazon을 구성합니다.

로 이벤트를 보내려면 먼저 AppFlow Amazon을 파트너 이벤트 Salesforce 소스로 사용하도록 구성해야 합니다. EventBridge

1. Amazon AppFlow 콘솔에서 흐름 생성을 선택합니다.
2. 플로우 세부 정보 섹션의 플로우 이름에 해당 플로우의 이름을 입력합니다.
3. (선택 사항) 플로우의 설명을 입력하고 다음을 선택합니다.
4. 소스 세부 정보에서 소스 이름 드롭다운에서 Salesforce를 선택한 다음, 연결을 선택하여 새 연결을 생성합니다.
5. Salesforce에 연결 대화 상자에서 Salesforce 환경에 대한 프로덕션 또는 샌드박스를 선택합니다.
6. 연결 이름 필드에 연결의 고유한 이름을 입력한 다음, 계속을 선택합니다.
7. Salesforce 대화 상자에서 다음을 수행합니다.
    a. Salesforce에 로그인할 Salesforce 로그인 보안 인증 정보를 입력합니다.
    b. AppFlow Amazon에서 처리할 데이터 유형에 맞는 Salesforce 이벤트를 선택합니다.
8. Salesforce이벤트 선택 드롭다운에서 전송할 이벤트 유형을 선택합니다 EventBridge.
9. 목적지로 Amazon을 선택합니다 EventBridge.
10. 새 파트너 이벤트 소스 생성을 선택합니다.
11. (선택 사항) 파트너 이벤트 소스의 고유한 접미사를 지정합니다.
12. 파트너 이벤트 소스 생성을 선택합니다.
13. 256KB보다 큰 이벤트 페이로드 파일을 저장하려면 Amazon S3 버킷을 선택합니다.
14. 플로우 트리거 섹션에서 이벤트 발생 시 플로우 실행이 선택되어 있는지 확인합니다. 이 설정을 사용하면 새 Salesforce 이벤트가 발생할 때 플로우가 실행됩니다.

15. 다음을 선택합니다.
16. 필드 매핑의 경우 모든 필드를 직접 매핑을 선택합니다. 또는 소스 필드 이름 목록에서 관심 있는 필드를 선택할 수도 있습니다.

    필드 매핑에 대한 자세한 내용을 알아보려면 Map data fields를 참조하세요.
17. 다음을 선택합니다.
18. (선택 사항) Amazon에서 데이터 필드에 대한 필터를 구성합니다 AppFlow.
19. 다음을 선택합니다.
20. 설정을 검토한 다음, 플로우 생성을 선택합니다.

플로우를 AppFlow 구성하면 Amazon에서 새 파트너 이벤트 소스를 생성한 다음 이를 계정의 파트너 이벤트 버스와 연결해야 합니다.

2단계: Salesforce 이벤트를 EventBridge 수신하도록 구성

이 섹션의 지침을 따르기 전에 EventBridge 목적지로 설정된 Salesforce 이벤트에서 트리거되는 Amazon AppFlow Flow가 구성되어 있는지 확인하십시오.

Salesforce이벤트를 EventBridge 수신하도록 구성하려면

1. EventBridge 콘솔에서 파트너 이벤트 소스 페이지를 엽니다.
2. 1단계에서 생성한 Salesforce 파트너 이벤트 소스를 선택합니다.
3. 이벤트 버스에 연결을 선택합니다.
4. 파트너 이벤트 버스의 이름을 확인합니다.
5. Associate(연결)를 선택합니다.
6. Amazon AppFlow 콘솔에서 생성한 흐름을 열고 흐름 활성화를 선택합니다.
7. EventBridge 콘솔에서 규칙 페이지를 엽니다.
8. Create rule을 선택합니다.
9. 역할의 고유한 이름을 입력합니다.
10. 패턴 정의 섹션에서 이벤트 패턴을 선택합니다.
11. 이벤트 매칭 패턴에서 서비스에서 제공하는 사전 정의된 패턴을 선택합니다.
12. 서비스 공급자 섹션에서 모든 이벤트를 선택합니다.
13. 이벤트 버스 선택에서 사용자 지정 또는 파트너 이벤트 버스를 선택합니다.

14. Amazon AppFlow 파트너 이벤트 소스에 연결한 이벤트 버스를 선택합니다.
15. 대상 선택에서 규칙 실행 시 작동할 AWS 서비스를 선택합니다. 규칙 하나에 최대 5개의 대상을 사용할 수 있습니다.
16. 생성을 선택합니다.

대상 서비스는 계정에 구성된 모든 Salesforce 이벤트를 수신합니다. 이벤트를 필터링하거나 일부 이벤트를 다른 대상으로 보내려면 이벤트 패턴과 함께 콘텐츠 기반 필터링을 사용할 수 있습니다.

> **Note**
>
> 256KB보다 큰 이벤트의 경우 AppFlow Amazon은 전체 이벤트를 로 전송하지 않습니다. EventBridge 대신 AppFlow Amazon은 이벤트를 사용자 계정의 S3 버킷에 넣은 다음 Amazon S3 버킷에 대한 EventBridge 포인터와 함께 이벤트를 보냅니다. 포인터를 사용하여 버킷에서 전체 이벤트를 가져올 수 있습니다.

이벤트 전송 디버깅

이벤트 전송 문제는 식별하기 어려울 수 있으므로 이벤트 전송 실패를 디버깅하고 복구할 수 있는 몇 가지 방법을 EventBridge 제공합니다.

이벤트 전송을 EventBridge 재시도하는 방법

규칙에 지정된 대상에 이벤트가 성공적으로 전달되지 않는 경우가 있습니다. 이런 일이 발생할 수 있습니다. 예를 들면 다음과 같습니다.

- 대상 리소스를 사용할 수 없는 경우
- 네트워크 상황으로 인해

재시도 가능한 오류로 인해 이벤트가 대상에 성공적으로 전달되지 않으면 이벤트 전송을 EventBridge 재시도합니다. 대상의 재시도 정책 설정에서 시도 시간 및 재시도 횟수를 설정합니다. 기본적으로 이벤트 전송을 24시간 동안 최대 185회까지 EventBridge 재시도합니다. 이때 백오프와 지터가 기하급수적으로 발생하거나 임의 지연이 발생합니다.

모든 재시도 횟수를 모두 사용한 후에도 이벤트가 전달되지 않으면 이벤트가 삭제되고 계속 처리되지 않습니다. EventBridge

데드레터 대기열을 사용하여 전달되지 않은 이벤트를 처리합니다.

대상으로 전달되지 않는 이벤트가 손실되지 않도록 DLQ(Dead Letter Queue)를 구성하고 실패한 모든 이벤트를 이 대기열로 전송하여 나중에 처리할 수 있습니다.

EventBridge DLQ는 대상에 성공적으로 전송되지 못한 이벤트를 저장하는 데 사용하는 표준 Amazon SQS EventBridge 대기열입니다. 규칙을 생성하고 대상을 추가할 때 DLQ 사용 여부를 선택할 수 있습니다. DLQ를 구성하면 성공적으로 전송되지 않은 모든 이벤트를 유지할 수 있습니다. 그러면 이벤트 전송 실패로 이어진 문제를 해결하고 나중에 이벤트를 처리할 수 있습니다.

규칙의 대상에 대해 DLQ를 구성하는 경우, 호출이 실패한 이벤트를 선택한 Amazon SQS 대기열로 EventBridge 전송합니다.

이벤트 오류는 다양한 방식으로 처리됩니다. 재시도 없이 일부 이벤트가 삭제되거나 DLQ로 전송됩니다. 예를 들어 대상에 대한 권한 누락이나 더 이상 존재하지 않는 대상 리소스로 인해 발생한 오류의 경우, 기본 문제를 해결하기 위한 조치가 취해질 때까지 모든 재시도가 실패합니다. DLQ가 있는 경우 다시 시도하지 말고 DLQ로 직접 이벤트를 EventBridge 전송하십시오.

이벤트 전송이 실패하면 대상이 `invocation` 실패했음을 나타내는 이벤트를 Amazon CloudWatch 지표에 EventBridge 게시합니다. DLQ를 사용하는 경우 및 를 CloudWatch 포함하여 `InvocationsSentToDLQ` 추가 지표가 전송됩니다. `InvocationsFailedToBeSentToDLQ`

미사용 이벤트를 암호화하는 데 사용하는 경우 이벤트 버스에 대한 AWS KMS 고객 관리형 키 DLQ를 지정할 수도 있습니다. 자세한 정보는 ???을 참조하세요.

DLQ의 각 메시지에는 다음과 같은 사용자 지정 속성이 포함됩니다.

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

  다음은 DLQ가 반환할 수 있는 오류 코드의 샘플입니다.

  - `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

  다음 조건이 반환될 수 있습니다.

  - MaximumRetryAttempts
  - MaximumEventAgeInSeconds
- RETRY_ATTEMPTS

다음 동영상에서는 DLQ 설정에 대해 설명합니다. DLQ(Dead Letter Queue) 사용

주제

- DLQ(Dead Letter Queue) 사용 시 고려할 사항
- DLQ(Dead Letter Queue)에 권한 부여

- DLQ(Dead Letter Queue)에서 이벤트를 재전송하는 방법

DLQ(Dead Letter Queue) 사용 시 고려할 사항

에 대한 DLQ를 구성할 때는 다음 사항을 고려하십시오. EventBridge

- 표준 대기열만 지원됩니다. DLQ 입력에는 FIFO 대기열을 사용할 수 없습니다. EventBridge
- EventBridge 오류 코드, 오류 메시지, 소진된 재시도 조건, 규칙 ARN, 재시도 시도, 대상 ARN 등 이벤트 메타데이터 및 메시지 속성을 메시지에 포함합니다. 이러한 값을 사용하여 이벤트와 실패 원인을 식별할 수 있습니다.
- 동일한 계정의 DLQ에 대한 권한:
  - 콘솔을 사용하여 규칙에 대상을 추가하고 동일한 계정에서 Amazon SQS 대기열을 선택하면 대기열에 EventBridge 대한 액세스 권한을 부여하는 리소스 기반 정책이 대기열에 자동으로 연결됩니다.
  - EventBridge API `PutTargets` 작업을 사용하여 규칙의 대상을 추가하거나 업데이트하고 동일한 계정에서 Amazon SQS 대기열을 선택하는 경우 선택한 대기열에 권한을 수동으로 부여해야 합니다. 자세한 내용은 DLQ(Dead Letter Queue)에 권한 부여 섹션을 참조하세요.
- 다른 계정에서 Amazon SQS 대기열을 사용할 수 있는 권한. AWS
  - 콘솔에서 규칙을 생성하는 경우 다른 계정의 대기열은 선택할 수 있도록 표시되지 않습니다. 다른 계정의 대기열에 대한 ARN을 제공한 다음, 수동으로 리소스 기반 정책을 연결하여 대기열에 권한을 부여해야 합니다. 자세한 내용은 DLQ(Dead Letter Queue)에 권한 부여 섹션을 참조하세요.
  - API를 사용하여 규칙을 생성하는 경우 DLQ(Dead Letter Queue)로 사용되는 다른 계정의 SQS 대기열에 리소스 기반 정책을 수동으로 연결해야 합니다. 자세한 내용은 DLQ(Dead Letter Queue)에 권한 부여 섹션을 참조하세요.
- 사용하는 Amazon SQS 대기열은 규칙을 생성한 리전과 동일한 리전에 있어야 합니다.

DLQ(Dead Letter Queue)에 권한 부여

대기열에 이벤트를 성공적으로 전송하려면 이벤트를 전송할 수 있는 권한이 EventBridge 있어야 합니다. EventBridge 콘솔을 사용하여 DLQ를 지정하면 권한이 자동으로 추가됩니다. 여기에는 다음이 포함됩니다.

- 규칙의 대상에 대해 DLQ를 구성하는 경우.
- 지정된 이벤트 버스에 DLQ를 구성하면 유휴 상태의 이벤트를 암호화하는 AWS KMS 고객 관리형 키 데 a를 EventBridge 사용합니다.

자세한 정보는 ???을 참조하세요.

API를 사용하여 DLQ를 지정하거나 다른 AWS 계정의 대기열을 사용하는 경우 필요한 권한을 부여하는 리소스 기반 정책을 수동으로 만든 다음 대기열에 연결해야 합니다.

대상 데드레터 대기열 권한 예시

다음 리소스 기반 정책은 Amazon SQS 대기열로 이벤트 메시지를 보내는 EventBridge 데 필요한 권한을 부여하는 방법을 보여줍니다. 정책 예제는 SendMessage 작업을 사용하여 "DLQ"라는 대기열로 메시지를 전송할 수 있는 권한을 EventBridge 서비스에 부여합니다. MyEvent 대기열은 계정 123456789012의 us-west-2 지역에 있어야 합니다. AWS 이 Condition 명령문은 us-west-2 지역의 123456789012 계정에 생성된 MyTestRule "“라는 이름의 규칙에서 오는 요청만 허용합니다. 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"
    }
  }
}
```

이벤트 버스 데드레터 대기열 권한 예제

다음 리소스 기반 정책은 이벤트 버스에 DLQ를 지정할 때 필요한 권한을 부여하는 방법을 보여줍니다. 이 경우 aws:SourceArn DLQ로 이벤트를 보내는 이벤트 버스의 ARN을 지정합니다. 이 예제에서도 마찬가지로 대기열은 이벤트 버스와 동일한 지역에 있어야 합니다.

```
{
  "Sid": "Dead-letter queue permissions",
  "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:event-bus/event-bus-arn"
    }
  }
}
```

정책을 대기열에 연결하려면 Amazon SQS 콘솔을 사용하여 대기열을 열고 액세스 정책을 선택하여 정책을 편집합니다. AWS CLI도 사용할 수 있습니다. 자세한 내용은 Amazon SQS 권한 섹션을 참조하세요.

DLQ(Dead Letter Queue)에서 이벤트를 재전송하는 방법

다음 두 가지 방법을 사용하여 메시지를 DLQ로부터 이동할 수 있습니다.

- Amazon SQS 소비자 로직 작성 방지 – DLQ를 Lambda 함수에 대한 이벤트 소스로 설정하여 DLQ를 비웁니다.
- Amazon SQS 소비자 로직 작성 — Amazon SQS API AWS , SDK를 사용하거나 DLQ에서 메시지를 폴링, AWS CLI 처리 및 삭제하기 위한 사용자 지정 소비자 로직을 작성할 수 있습니다.

아마존 EventBridge 이벤트 패턴

이벤트 패턴은 일치하는 이벤트와 동일한 구조를 갖습니다. 규칙은 이벤트 패턴을 사용하여 이벤트를 선택하고 대상으로 이를 전송합니다. 이벤트 패턴은 이벤트와 일치할 수도 있고 아닐 수도 있습니다.

> **Important**
>
> EventBridge에서는 higher-than-expected 요금 청구 및 스로틀링으로 이어질 수 있는 규칙을 생성할 수 있습니다. 예를 들어 규칙이 끝없이 반복적으로 실행되는 무한 루프로 이어지는 규칙을 본의 아니게 생성할 수 있습니다. Amazon S3 버킷에서 ACL이 바뀐 것을 감지하고 소프트웨어를 트리거하여 ACL을 원하는 상태로 변경하는 규칙을 생성했다고 가정합니다. 이때 규칙이 부주의하게 작성되면 ACL에 대한 변경이 이어져 규칙을 다시 실행하면서 무한 루프에 빠지게 됩니다.
>
> 이러한 예상치 못한 결과를 최소화하기 위해 정확한 규칙 및 이벤트 패턴을 작성하는 방법에 대한 지침은 ??? 및 ??? 섹션을 참조하세요.

다음 동영상에서는 이벤트 패턴의 기본 사항에 대해 설명합니다. 이벤트를 필터링하는 방법

주제

- 이벤트 패턴 생성
- 예제 이벤트 및 이벤트 패턴
- Amazon EventBridge 이벤트 패턴에서 null 값과 빈 문자열 매칭
- Amazon EventBridge 이벤트 패턴의 배열
- Amazon EventBridge 이벤트 패턴의 콘텐츠 필터링
- 샌드박스를 사용하여 이벤트 패턴 테스트 EventBridge
- Amazon EventBridge 이벤트 패턴 정의 모범 사례

다음 이벤트는 Amazon EC2의 간단한 AWS 이벤트를 보여줍니다.

```
{
  "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"
  }
}
```

다음 이벤트 패턴은 모든 Amazon EC2 `instance-termination` 이벤트를 처리합니다.

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

이벤트 패턴 생성

이벤트 패턴을 만들려면 이벤트 패턴에서 매칭할 이벤트의 필드를 지정합니다. 매칭에 사용하는 필드만 지정하세요. 이전 이벤트 패턴 예제는 최상위 필드 `"source"` 및 및 `"detail"` 객체 필드 내 `"state"` 필드 `"detail-type"` 등 세 개의 필드에 대한 값만 제공합니다. EventBridge 규칙을 적용할 때 이벤트의 다른 모든 필드를 무시합니다.

이벤트와 이벤트 패턴을 매칭하려면 이벤트에 이벤트 패턴에 나열된 모든 필드 이름이 포함되어 있어야 합니다. 필드 이름은 같은 중첩 구조를 가진 이벤트에도 나타나야 합니다.

이벤트에 매칭되는 이벤트 패턴을 작성할 때는 `TestEventPattern` API 또는 `test-event-pattern` CLI 명령을 사용하여 패턴이 올바른 이벤트와 매칭되는지 테스트할 수 있습니다. 자세한 내용은 을 참조하십시오 TestEventPattern.

이벤트 값 일치

이벤트 패턴에서 매칭할 값은 대괄호("[", "]")로 묶인 JSON 배열에 있으므로 여러 값을 제공할 수 있습니다. 예를 들어 Amazon EC2 또는 AWS Fargate의 이벤트를 매칭하려면 `"source"` 필드 값이 또는 `"aws.ec2"` 인 이벤트와 일치하는 다음 패턴을 사용할 수 있습니다. `"aws.fargate"`

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

이벤트 패턴 생성 시 고려 사항

다음은 이벤트 패턴을 구성할 때 고려해야 할 몇 가지 사항입니다.

- EventBridge 이벤트 패턴에 포함되지 않은 이벤트의 필드를 무시합니다. 그 결과 이벤트 패턴에 나타나지 않는 필드에 `"*": "*"` 와일드카드가 있습니다.
- 이벤트 패턴이 매칭하는 값은 JSON 규칙을 따릅니다. 따옴표(“)로 묶인 문자열, 숫자 및 키워드(`true`, `false` 및 `null`)를 포함할 수 있습니다.
- 문자열의 경우 EventBridge 대소문자를 구분하거나 다른 문자열 정규화를 사용하지 않고 정확히 character-by-character 일치시킵니다.
- 숫자의 경우 문자열 표현을 EventBridge 사용합니다. 예를 들어 300, 300.0 및 3.0e2는 동일한 것으로 간주되지 않습니다.
- 동일한 JSON 필드에 여러 패턴을 지정하는 경우 마지막 EventBridge 패턴만 사용합니다.
- 사용할 이벤트 패턴을 EventBridge 컴파일할 때는 점 (.) 을 조인 문자로 사용한다는 점에 유의하세요.

  즉, 다음 이벤트 패턴을 동일하게 취급한다는 EventBridge 의미입니다.

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

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

  그리고 두 이벤트 패턴이 모두 다음 두 이벤트와 일치합니다.

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

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

> **Note**
>
> 이것은 현재의 EventBridge 행동을 설명하며, 변하지 않을 것이라고 믿어서는 안 됩니다.

- 중복된 필드를 포함하는 이벤트 패턴은 유효하지 않습니다. 패턴에 중복 필드가 포함된 경우 최종 필드 EventBridge 값만 고려합니다.

  예를 들어 다음 이벤트 패턴은 동일한 이벤트와 일치합니다.

```
## 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"] }
}
```

  그리고 다음 두 이벤트를 동일하게 EventBridge 취급합니다.

```
## 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**
>
> 이것은 현재의 EventBridge 행동을 설명하며, 변하지 않는다고 믿어서는 안 됩니다.

이벤트 패턴에 사용하기 위한 비교 연산

아래에는 에서 EventBridge 사용할 수 있는 모든 비교 연산자가 요약되어 있습니다.

비교 연산자는 `$or` 및 `anything-but`을 제외하고 리프 노드에서만 작동합니다.

비교	예	규칙 구문
및	위치가 “New York”이고 요일이 “Monday”임	`"Location": [ "New York" ], "Day": ["Monday"]`
그 외에는 아무것도 없습니다.	상태는 “초기화 중”을 제외한 모든 값입니다.	`"state": [ { "anything-but": "initializing" } ]`
(로 시작) 을 제외한 모든 항목	지역은 미국에 있지 않습니다.	`"Region": [ { "anything-but": {"prefix": "us-" } } ]`
(로 끝남) 을 제외한 모든 것	FileName .png 확장자로 끝나지 않습니다.	`"FileName": [ { "anything-but": { "suffix": ".png" } } ]`

비교	예	규칙 구문
제외 (대소문자 무시)	상태는 “초기화 중” 또는 “초기화”와 같은 기타 대소문자 변형을 제외한 모든 값입니다.	`"state": : [{ "anything-but": { "equals-ignore-case": "initializing" }}]}`
와일드카드 사용 이외의 모든 것	FileName 를 포함하는 파일 경로가 아닙니다. `/lib/`	`"FilePath" : [{ "anything-but": { "wildcard": "*/lib/*" }}]`
다음으로 시작	지역은 미국입니다.	`"Region": [ {"prefix": "us-" } ]`
로 시작 (대소문자 무시)	서비스 이름은 대소문자를 불문하고 “eventb”로 시작합니다.	`{"service" : [{ "prefix": { "equals-ignore-case": "eventb" }}]}`
비어 있음	LastName 비어 있습니다.	`"LastName": [""]`
같음	이름이 “Alice”임	`"Name": [ "Alice" ]`
같음(대/소문자 무시)	이름이 “Alice”임	`"Name": [ { "equals-ignore-case": "alice" } ]`
다음으로 끝남	FileName .png 확장자로 끝납니다.	`"FileName": [ { "suffix": ".png" } ]`
로 끝남 (대소문자 무시)	서비스 이름은 “tbridge”라는 글자 또는 “TBRIDGE”와 같은 기타 대소문자를 변형하여 끝납니다.	`{"service" : [{ "suffix": { "equals-ignore-case": "tBridge" }}]}`
존재함	ProductName 존재합니다	`"ProductName": [ { "exists": true } ]`
존재하지 않음	ProductName 존재하지 않습니다	`"ProductName": [ { "exists": false } ]`
아님	날씨는 “Raining”이 아님	`"Weather": [ { "anything-but": [ "Raining" ] } ]`

비교	예	규칙 구문
Null	UserID가 null임	`"UserID": [ null ]`
숫자(같음)	가격은 100임	`"Price": [ { "numeric": [ "=", 100 ] } ]`
숫자(범위)	가격이 10을 초과하고 20보다 작거나 같음	`"Price": [ { "numeric": [ ">", 10, "<=", 20 ] } ]`
Or	PaymentType “신용카드” 또는 “직불카드”입니다.	`"PaymentType": [ "Credit", "Debit"]`
또는(여러 필드)	위치가 'New York'이고 요일이 'Monday'임	`"$or": [ { "Location": [ "New York" ] }, { "Day": [ "Monday" ] } ]`
와일드카드	‘dir’ 폴더 내에 있는 확장명이.png인 모든 파일	`"FileName": [ { "wildcard": "dir/*.png" } ]`

예제 이벤트 및 이벤트 패턴

모든 JSON 데이터 유형과 값을 사용하여 이벤트를 매칭할 수 있습니다. 다음 예제에서는 이벤트와 이에 일치하는 이벤트 패턴을 보여줍니다.

필드 일치

필드 값을 기준으로 매칭할 수 있습니다. 다음과 같은 Amazon EC2 Auto Scaling 이벤트를 살펴보세요.

```
{
  "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
  }
}
```

이전 이벤트의 경우 "responseElements" 필드를 사용하여 매칭할 수 있습니다.

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

값 일치

다음과 같은 잘린 Amazon Macie 이벤트를 살펴보세요.

```
{
  "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,
.
.
.
```

다음 이벤트 패턴은 심각도 점수가 1이고 개수가 2인 모든 이벤트와 일치합니다.

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

Amazon EventBridge 이벤트 패턴에서 null 값과 빈 문자열 매칭

> **Important**
>
> EventBridge에서는 higher-than-expected 요금 청구 및 스로틀링으로 이어질 수 있는 규칙을 생성할 수 있습니다. 예를 들어 규칙이 끝없이 반복적으로 실행되는 무한 루프로 이어지는 규칙을 본의 아니게 생성할 수 있습니다. Amazon S3 버킷에서 ACL이 바뀐 것을 감지하고 소프트웨어를 트리거하여 ACL을 원하는 상태로 변경하는 규칙을 생성했다고 가정합니다. 이때 규칙이 부주의하게 작성되면 ACL에 대한 변경이 이어져 규칙을 다시 실행하면서 무한 루프에 빠지게 됩니다.
>
> 이러한 예상치 못한 결과를 최소화하기 위해 정확한 규칙 및 이벤트 패턴을 작성하는 방법에 대한 지침은 ??? 및 ??? 섹션을 참조하세요.

null 값이 있거나 빈 문자열인 이벤트의 필드와 일치하는 이벤트 패턴을 생성할 수 있습니다. 다음 예제 이벤트를 살펴보세요.

예상보다 높은 요금과 제한을 방지하기 위한 모범 사례를 참조하세요.

```
{
  "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
   }
}
```

`eventVersion`의 값이 빈 문자열인 이벤트를 매칭하려면 이전 이벤트와 일치하는 다음 이벤트 패턴을 사용합니다.

```
{
  "detail": {
```

```
    "eventVersion": [""]
  }
}
```

`responseElements`의 값이 null인 이벤트를 매칭하려면 이전 이벤트와 일치하는 다음 이벤트 패턴을 사용합니다.

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

> **Note**
>
> Null 값과 빈 문자열은 패턴 일치 시 서로 바꾸어 사용할 수 없습니다. 빈 문자열과 일치하는 이벤트 패턴은 `null` 값과 일치하지 않습니다.

Amazon EventBridge 이벤트 패턴의 배열

이벤트 패턴의 각 필드 값은 하나 이상의 값을 포함하는 배열입니다. 배열의 값 중 하나라도 이벤트의 값과 일치하면 이벤트 패턴이 이벤트와 일치합니다. 이벤트의 값이 배열일 경우에는 이벤트 패턴 배열과 이벤트 배열에서 교차되는 부분이 없는 경우에만 이벤트 패턴이 일치합니다.

> **Important**
>
> EventBridge에서는 higher-than-expected 요금 청구 및 스로틀링으로 이어질 수 있는 규칙을 생성할 수 있습니다. 예를 들어 규칙이 끝없이 반복적으로 실행되는 무한 루프로 이어지는 규칙을 본의 아니게 생성할 수 있습니다. Amazon S3 버킷에서 ACL이 바뀐 것을 감지하고 소프트웨어를 트리거하여 ACL을 원하는 상태로 변경하는 규칙을 생성했다고 가정합니다. 이때 규칙이 부주의하게 작성되면 ACL에 대한 변경이 이어져 규칙을 다시 실행하면서 무한 루프에 빠지게 됩니다.
>
> 이러한 예상치 못한 결과를 최소화하기 위해 정확한 규칙 및 이벤트 패턴을 작성하는 방법에 대한 지침은 ??? 및 ??? 섹션을 참조하세요.

예를 들어 다음 필드를 포함하는 이벤트 패턴을 살펴보세요.

```
"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",
]
```

이벤트 패턴 배열의 첫 번째 항목이 이벤트 배열의 두 번째 항목과 일치하기 때문에 이전 이벤트 패턴은 다음 필드가 포함된 이벤트와 일치합니다.

```
"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"
]
```

Amazon EventBridge 이벤트 패턴의 콘텐츠 필터링

EventBridge Amazon은 이벤트 패턴을 사용한 선언적 콘텐츠 필터링을 지원합니다. 콘텐츠 필터링을 사용하면 매우 특정한 조건에서만 이벤트와 일치하는 복잡한 이벤트 패턴을 작성할 수 있습니다. 예를 들어 다음과 같은 경우 이벤트와 일치하는 이벤트 패턴을 생성할 수 있습니다.

- 이벤트 필드는 특정 숫자 범위 내에 있습니다.
- 이벤트는 특정 IP 주소에서 발생합니다.
- 이벤트 JSON에는 특정 필드가 없습니다.

> **Important**
>
> EventBridge에서는 higher-than-expected 요금 청구 및 스로틀링으로 이어질 수 있는 규칙을 만들 수 있습니다. 예를 들어 규칙이 끝없이 반복적으로 실행되는 무한 루프로 이어지는 규칙을 본의 아니게 생성할 수 있습니다. Amazon S3 버킷에서 ACL이 바뀐 것을 감지하고 소프트웨어를 트리거하여 ACL을 원하는 상태로 변경하는 규칙을 생성했다고 가정합니다. 이때 규칙이 부주의하게 작성되면 ACL에 대한 변경이 이어져 규칙을 다시 실행하면서 무한 루프에 빠지게 됩니다.
>
> 이러한 예상치 못한 결과를 최소화하기 위해 정확한 규칙 및 이벤트 패턴을 작성하는 방법에 대한 지침은 ??? 및 ??? 섹션을 참조하세요.

필터 유형

- 접두사 일치
- 접미사 일치
- Anything-but 일치
- 숫자 일치
- IP 주소 일치
- Exists 일치
- E quals-ignore-case 매칭
- 와일드카드를 사용한 매칭
- 여러 일치가 있는 복잡한 예제
- $or 일치가 있는 복잡한 예제

접두사 일치

이벤트 소스에 있는 값의 접두사에 따라 이벤트를 매칭할 수 있습니다. 문자열 값에 접두사 일치를 사용할 수 있습니다.

예를 들어 다음 이벤트 패턴은 "time" 필드가 "time": "2017-10-02T18:43:48Z"와 같이 "2017-10-02"로 시작된 모든 이벤트와 일치합니다.

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

대소문자를 무시하고 접두사 매칭

다음과 함께 사용하여 equals-ignore-case 값이 시작하는 문자의 대/소문자에 관계없이 접두사 값을 일치시킬 수도 있습니다. prefix.

예를 들어, 다음 이벤트 패턴은 service 필드가 해당 문자열로 시작하는 모든 이벤트와 일치하지만 EventB EVENTBeventb, 또는 해당 문자의 다른 대/소문자로 시작하는 모든 이벤트와 일치합니다.

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

접미사 일치

이벤트 소스에 있는 값의 접미사에 따라 이벤트를 매칭할 수 있습니다. 문자열 값에 접미사 일치를 사용할 수 있습니다.

예를 들어 다음 이벤트 패턴은 "FileName" 필드가 .png 파일 확장명으로 끝나는 모든 이벤트와 일치합니다.

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

대소문자를 무시하고 접미사 매칭

다음과 함께 사용하면 값이 끝나는 문자의 대소문자에 상관없이 접미사 값을 일치시킬 수도 있습니다. equals-ignore-case suffix.

예를 들어, 다음 이벤트 패턴은 FileName 필드가 문자열로 끝나는 모든 이벤트와 .PNG 일치하거나 해당 문자의 다른 .png 대문자도 일치합니다.

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

Anything-but 일치

일치하는 항목 이외의 모든 항목은 규칙에 지정된 항목 이외의 모든 항목과 일치합니다.

문자열만 포함하거나 숫자만 포함하는 목록을 포함하여 문자열 및 숫자 값과 함께 Anything-but 일치를 사용할 수 있습니다.

다음 이벤트 패턴은 문자열 및 숫자와의 anything-but 일치를 보여줍니다.

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

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

다음 이벤트 패턴은 문자열 목록과의 anything-but 일치를 보여줍니다.

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

다음 이벤트 패턴은 숫자 목록과의 anything-but 일치를 보여줍니다.

```
{
  "detail": {
```

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

대소문자를 무시하고 매칭하는 것 외에는 아무 것도 없습니다.

와 함께 anything-but 사용하여 equals-ignore-case 대/소문자를 구분하지 않고 문자열 값을 일치시킬 수도 있습니다.

다음 이벤트 패턴은 “초기화”, “초기화”, “초기화” 문자열 또는 해당 문자의 기타 대소문자를 포함하지 않는 state 필드와 일치합니다.

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

와 함께 anything-but 사용하여 equals-ignore-case 값 목록과 일치시킬 수도 있습니다.

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

접두사와 일치하는 것 외에는 아무 것도 없습니다.

와 함께 anything-but 사용하여 prefix 지정된 값으로 시작하지 않는 문자열 값을 일치시킬 수 있습니다. 여기에는 단일 값 또는 값 목록이 포함됩니다.

다음 이벤트 패턴은 필드에 "init" 접두사가 없는 모든 이벤트와 일치하는 항목을 제외하고 모두 보여줍니다. "state"

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

다음 이벤트 패턴은 접두사 값 목록에 사용된 일치 항목 이외의 모든 항목을 보여줍니다. 이 이벤트 패턴은 접두사 또는 필드에 접두사가 "init" 없는 모든 이벤트와 일치합니다. "stop" "state"

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

접미사가 일치하는 경우를 제외한 모든 항목

와 함께 anything-but 사용하여 suffix 지정된 값으로 끝나지 않는 문자열 값을 일치시킬 수 있습니다. 여기에는 단일 값 또는 값 목록이 포함됩니다.

다음 이벤트 패턴은 로 끝나지 않는 FileName 필드의 모든 값과 .txt 일치합니다.

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

다음 이벤트 패턴은 접미사 값 목록에 사용된 일치 항목 외에는 모든 항목을 보여줍니다. 이 이벤트 패턴은 또는 로 끝나지 않는 모든 FileName 필드 값과 일치합니다. .txt .rtf

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

와일드카드를 사용한 일치 이외의 모든 항목

지정한 값 내에서 와일드카드 문자 (*) 를 사용하여 매칭을 제외한 모든 항목에 사용할 수 있습니다. 여기에는 단일 값이나 값 목록이 포함됩니다.

다음 이벤트 패턴은 포함하지 않는 FileName 필드의 모든 값과 /lib/ 일치합니다.

```
{
"detail": {
  "FilePath" : [{ "anything-but": { "wildcard": "*/lib/*" }}]
  }
```

```
}
```

다음 이벤트 패턴은 와일드카드를 포함한 값 목록에 사용된 일치 항목 이외의 모든 항목을 보여줍니다. 이 이벤트 패턴은 또는 중 하나를 포함하지 않는 FileName 필드의 모든 값과 일치합니다. /lib/ /bin/

```
{
"detail": {
  "FilePath" : [{ "anything-but": { "wildcard": ["*/lib/*", "*/bin/*"] }}]
  }
}
```

자세한 정보는 ???을 참조하세요.

숫자 일치

숫자 일치는 JSON 번호인 값과 함께 작동합니다. -5.0e9와 +5.0e9 사이의 값으로 제한되며 정밀도는 15자리(소수점 오른쪽의 6자리)입니다.

다음은 모든 필드에 해당하는 이벤트만 매칭되는 이벤트 패턴의 숫자 일치를 보여줍니다.

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

IP 주소 일치

IPv4 및 IPv6 주소에 대한 IP 주소 일치를 사용할 수 있습니다. 다음 이벤트 패턴은 10.0.0으로 시작하고 0에서 255 사이의 숫자로 끝나는 IP 주소와 일치하는 IP 주소를 보여줍니다.

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

Exists 일치

Exists 일치는 이벤트의 JSON에서 필드의 유무에 따라 작동합니다.

Exists 일치는 리프 노드에서만 작동합니다. 중간 노드에서는 작동하지 않습니다.

다음 이벤트 패턴은 `detail.state` 필드가 있는 모든 이벤트와 일치합니다.

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

이전 이벤트 패턴은 다음 이벤트와 일치합니다.

```
{
  "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"
  }
}
```

다음 이벤트에는 `detail.state` 필드가 없으므로 이전 이벤트 패턴은 다음 이벤트와 일치하지 않습니다.

```
{
  "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 매칭

E quals-ignore-case 매칭은 대소문자를 구분하지 않고 문자열 값에 적용됩니다.

다음 이벤트 패턴은 대소문자를 구분하지 않고 지정된 문자열과 일치하는 `detail-type` 필드가 있는 모든 이벤트와 일치합니다.

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

이전 이벤트 패턴은 다음 이벤트와 일치합니다.

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

와일드카드를 사용한 매칭

와일드카드 문자(*)를 사용하여 이벤트 패턴의 문자열 값을 매칭할 수 있습니다.

> **Note**
>
> 현재 와일드카드 문자는 이벤트 버스 규칙에서만 지원됩니다.

이벤트 패턴에서 와일드카드를 사용할 때 고려할 사항:

- 주어진 문자열 값에 와일드카드 문자를 원하는 수만큼 지정할 수 있지만 연속된 와일드카드 문자는 지원되지 않습니다.

- EventBridge 와일드카드 필터에 백슬래시 문자 (\) 를 사용하여 리터럴 * 및\ 문자를 지정할 수 있습니다.
  - 문자열 *은 리터럴 * 문자를 나타냅니다.
  - 문자열 \\은 리터럴 \ 문자를 나타냅니다.

  백슬래시를 사용하여 다른 문자를 이스케이프 처리하는 것은 지원되지 않습니다.

와일드카드 및 이벤트 패턴 복잡성

와일드카드를 사용하는 규칙의 복잡성에는 제한이 있습니다. 규칙이 너무 복잡하면 규칙을 만들려고 `InvalidEventPatternException` 하면 an을 EventBridge 반환합니다. 규칙에서 이러한 오류가 발생하는 경우 아래 지침을 사용하여 이벤트 패턴의 복잡성을 줄이는 것이 좋습니다.

- 와일드카드 문자 사용 횟수 줄이기

  가능한 여러 값과 매칭해야 하는 경우에만 와일드카드 문자를 사용하세요. 예를 들어 동일한 리전의 이벤트 버스와 매칭하려는 다음 이벤트 패턴을 살펴보세요.

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

  위의 경우 ARN의 많은 섹션은 이벤트 버스가 위치한 리전을 직접 기반으로 합니다. 따라서 `us-east-1` 리전을 사용하는 경우 원하는 값과 여전히 일치하는 덜 복잡한 패턴은 다음을 예로 들 수 있습니다.

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

- 와일드카드 문자 뒤에 나오는 반복되는 문자 시퀀스 줄이기

  와일드카드를 사용한 후 동일한 문자 시퀀스가 여러 번 나타나면 이벤트 패턴 처리가 더 복잡해집니다. 이벤트 패턴을 재구성하여 반복되는 시퀀스를 최소화하세요. 예를 들어 모든 사용자의 파일 이름 `doc.txt` 파일과 일치하는 다음 예를 살펴보세요.

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

doc.txt 파일이 지정된 경로에서만 발생한다는 것을 알고 있다면 다음과 같은 방법으로 반복되는 문자 시퀀스를 줄일 수 있습니다.

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

여러 일치가 있는 복잡한 예제

여러 일치 규칙을 보다 복잡한 이벤트 패턴으로 결합할 수 있습니다. 예를 들어 다음 이벤트 패턴은 anything-but과 numeric을 결합합니다.

```
{
  "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

이벤트 패턴을 작성할 때 키를 두 번 이상 포함하면 마지막 참조가 이벤트를 평가하는 데 사용됩니다. 예를 들어, 다음 패턴의 경우:

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

location을 평가할 때는 { "anything-but": "us-east" }만 고려됩니다.

$or 일치가 있는 복잡한 예제

여러 필드에서 일치하는 필드 값이 있는지 확인하는 복잡한 이벤트 패턴을 생성할 수도 있습니다. 여러 필드의 값 중 하나라도 일치하는 경우 $or를 사용하여 일치하는 이벤트 패턴을 만듭니다.

$or 구성의 개별 필드에 대한 패턴 일치에 숫자 일치 및 배열 등의 다른 필터 유형을 포함할 수 있습니다.

다음과 같은 조건 중 하나라도 충족되면 다음 이벤트 패턴이 일치합니다.

- c-count 필드가 0보다 크거나 5 이하입니다.
- d-count 필드가 10보다 작습니다.
- x-limit 필드는 3.018e2와 같습니다.

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

> **Note**
>
> 이벤트 패턴(예: PutRule, CreateArchive, UpdateArchive 및 TestEventPattern)을 허용하는 API는 $or를 사용하여 1,000개가 넘는 규칙 조합이 생성되면 InvalidEventPatternException을 발생시킵니다.
> 이벤트 패턴에서 규칙 조합 수를 결정하려면 이벤트 패턴에 있는 각 $or 배열의 총 인수의 수를 곱합니다. 예를 들어 위 패턴에는 세 개의 인수가 있는 단일 $or 배열이 포함되므로 총 규칙 조합 수도 3입니다. 두 개의 인수가 있는 다른 $or 배열을 추가하면 총 규칙 조합은 6개가 됩니다.

샌드박스를 사용하여 이벤트 패턴 테스트 EventBridge

규칙은 이벤트 패턴을 사용하여 이벤트를 선택하고 대상으로 이를 전송합니다. 이벤트 패턴은 일치하는 이벤트와 동일한 구조를 갖습니다. 이벤트 패턴은 이벤트와 일치할 수도 있고 아닐 수도 있습니다.

이벤트 패턴 정의는 일반적으로 새 규칙을 생성하거나 기존 규칙을 편집하는 대규모 프로세스의 일부입니다. 그러나 에서 EventBridge 샌드박스를 사용하면 규칙을 만들거나 편집할 필요 없이 이벤트 패턴을 빠르게 정의하고 샘플 이벤트를 사용하여 패턴이 원하는 이벤트와 일치하는지 확인할 수 있습니다. 이벤트 패턴을 테스트한 후에는 샌드박스에서 직접 해당 이벤트 패턴을 사용하여 새 규칙을 만들 수 있습니다. EventBridge

이벤트 패턴에 대한 자세한 내용은 ??? 섹션을 참조하세요.

> **Important**
>
> EventBridge에서는 higher-than-expected 요금 청구 및 스로틀링으로 이어질 수 있는 규칙을 만들 수 있습니다. 예를 들어 규칙이 끝없이 반복적으로 실행되는 무한 루프로 이어지는 규칙을 본의 아니게 생성할 수 있습니다. Amazon S3 버킷에서 ACL이 바뀐 것을 감지하고 소프트웨어를 트리거하여 ACL을 원하는 상태로 변경하는 규칙을 생성했다고 가정합니다. 이때 규칙이 부주의하게 작성되면 ACL에 대한 변경이 이어져 규칙을 다시 실행하면서 무한 루프에 빠지게 됩니다.
>
> 이러한 예상치 못한 결과를 최소화하기 위해 정확한 규칙 및 이벤트 패턴을 작성하는 방법에 대한 지침은 ??? 및 ??? 섹션을 참조하세요.

샌드박스를 사용하여 이벤트 패턴을 테스트하려면 EventBridge

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 개발자 리소스를 선택한 다음, 샌드박스를 선택하고 샌드박스 페이지에서 이벤트 패턴 탭을 선택합니다.
3. 이벤트 소스에서AWS 이벤트 또는 EventBridge 파트너 이벤트를 선택합니다.
4. 샘플 이벤트 섹션에서 이벤트 패턴을 테스트할 샘플 이벤트 유형을 선택합니다.

   다음 샘플 이벤트 유형을 사용할 수 있습니다.

   - AWS 이벤트 — support에서 생성된 이벤트 중에서 선택합니다. AWS 서비스
   - EventBridge 파트너 이벤트 — Salesforce와 같이 지원하는 EventBridge 타사 서비스에서 생성된 이벤트 중에서 선택합니다.

- 내 이벤트 입력 - JSON 텍스트로 자체 이벤트를 입력합니다.

  AWS 또는 파트너 이벤트를 출발점으로 사용하여 사용자 지정 이벤트를 직접 만들 수도 있습니다.

  1. AWS 이벤트 또는 EventBridge 파트너 이벤트를 선택하세요.
  2. 샘플 이벤트 드롭다운을 이용해 사용자 지정 이벤트의 시작점으로 사용할 이벤트를 선택합니다.

     EventBridge 샘플 이벤트를 표시합니다.
  3. 복사를 선택합니다.
  4. 이벤트 유형에서 내 이벤트 입력을 선택합니다.
  5. JSON 편집 창에서 샘플 이벤트 구조를 삭제하고 그 자리에 AWS 또는 파트너 이벤트를 붙여 넣습니다.
  6. 이벤트 JSON을 편집하여 자체 샘플 이벤트를 생성합니다.

5. 생성 방법을 선택합니다. EventBridge 스키마 또는 템플릿에서 이벤트 패턴을 만들거나 사용자 지정 이벤트 패턴을 만들 수 있습니다.

   Existing schema

   기존 EventBridge 스키마를 사용하여 이벤트 패턴을 만들려면 다음과 같이 하십시오.

   1. 생성 방법 섹션의 메서드에서 스키마 사용을 선택합니다.
   2. 이벤트 패턴 섹션의 스키마 유형에서 스키마 레지스트리에서 스키마 선택을 선택합니다.
   3. 스키마 레지스트리의 경우 드롭다운 상자를 선택하고 스키마 레지스트리 이름(예: `aws.events`)을 입력합니다. 표시되는 드롭다운 목록에서 옵션을 선택할 수도 있습니다.
   4. 스키마의 경우 드롭다운 상자를 선택하고 사용할 스키마 이름을 입력합니다. 예를 들어 `aws.s3@0bjectDeleted`입니다. 표시되는 드롭다운 목록에서 옵션을 선택할 수도 있습니다.
   5. 모델 섹션에서 속성 옆에 있는 편집 버튼을 선택하여 해당 속성을 엽니다. 필요에 따라 관계 및 값 필드를 설정한 다음, 설정을 선택하여 속성을 저장합니다.

      > **Note**
      >
      > 속성 정의에 대한 자세한 내용을 보려면 속성 이름 옆에 있는 정보 아이콘을 선택하세요. 이벤트에서 속성 특성을 설정하는 방법에 대한 참조를 보려면 속성 특성 대화 상자의 참고 섹션을 엽니다.

속성 특성을 삭제하려면 해당 속성의 편집 버튼을 선택한 다음, 지우기를 선택합니다.

6. 이벤트 패턴을 JSON 텍스트로 생성하고 검증하려면 JSON으로 이벤트 패턴 생성을 선택합니다.
7. 테스트 패턴과 비교하여 샘플 이벤트를 테스트하려면 테스트 패턴을 선택합니다.

   EventBridge 샘플 이벤트가 이벤트 패턴과 일치하는지 여부를 나타내는 메시지 상자를 표시합니다.

   다음 옵션 중 하나를 선택할 수도 있습니다.

   - 복사 - 이벤트 패턴을 디바이스의 클립보드에 복사합니다.
   - 정리 - 줄 바꿈, 탭, 공백을 추가하여 JSON 텍스트를 더 쉽게 읽을 수 있습니다.

Custom schema

사용자 지정 스키마를 작성하고 이벤트 패턴으로 변환하려면 다음을 수행하세요.

1. 생성 방법 섹션의 메서드에서 스키마 사용을 선택합니다.
2. 이벤트 패턴 섹션의 스키마 유형에서 스키마 입력을 선택합니다.
3. 텍스트 상자에 스키마를 입력합니다. 스키마의 형식을 유효한 JSON 텍스트로 지정해야 합니다.
4. 모델 섹션에서 속성 옆에 있는 편집 버튼을 선택하여 해당 속성을 엽니다. 필요에 따라 관계 및 값 필드를 설정한 다음, 설정을 선택하여 속성을 저장합니다.

   > **Note**
   >
   > 속성 정의에 대한 자세한 내용을 보려면 속성 이름 옆에 있는 정보 아이콘을 선택하세요. 이벤트에서 속성 특성을 설정하는 방법에 대한 참조를 보려면 속성 특성 대화 상자의 참고 섹션을 엽니다.
   > 속성 특성을 삭제하려면 해당 속성의 편집 버튼을 선택한 다음, 지우기를 선택합니다.

5. 이벤트 패턴을 JSON 텍스트로 생성하고 검증하려면 JSON으로 이벤트 패턴 생성을 선택합니다.
6. 테스트 패턴과 비교하여 샘플 이벤트를 테스트하려면 테스트 패턴을 선택합니다.

EventBridge 샘플 이벤트가 이벤트 패턴과 일치하는지 여부를 나타내는 메시지 상자를 표시합니다.

다음 옵션 중 하나를 선택할 수도 있습니다.

- 복사 - 이벤트 패턴을 디바이스의 클립보드에 복사합니다.
- 정리 - 줄 바꿈, 탭, 공백을 추가하여 JSON 텍스트를 더 쉽게 읽을 수 있습니다.

Event pattern

JSON 형식의 사용자 지정 이벤트 패턴을 작성하려면 다음을 수행하세요.

1. 생성 방법 섹션의 메서드에서 사용자 지정 패턴(JSON 편집기)을 선택합니다.
2. 이벤트 패턴의 경우 JSON 형식 텍스트로 사용자 지정 이벤트 패턴을 입력합니다.
3. 테스트 패턴과 비교하여 샘플 이벤트를 테스트하려면 테스트 패턴을 선택합니다.

   EventBridge 샘플 이벤트가 이벤트 패턴과 일치하는지 여부를 나타내는 메시지 상자를 표시합니다.

   다음 옵션 중 하나를 선택할 수도 있습니다.

   - 복사 - 이벤트 패턴을 디바이스의 클립보드에 복사합니다.
   - 정리 - 줄 바꿈, 탭, 공백을 추가하여 JSON 텍스트를 더 쉽게 읽을 수 있습니다.
   - 이벤트 패턴 양식 - 패턴 빌더에서 이벤트 패턴을 엽니다. 패턴 빌더에서 패턴을 있는 그대로 렌더링할 수 없는 경우 패턴 빌더를 열기 전에 EventBridge 경고 메시지가 표시됩니다.

6. (선택 사항) 이 이벤트 패턴으로 규칙을 생성하고 특정 이벤트 버스에 규칙을 할당하려면 패턴이 있는 규칙 생성하기를 선택합니다.

   EventBridge 규칙 만들기의 1단계로 이동합니다. 이 단계를 사용하여 규칙을 만들고 선택한 이벤트 버스에 할당할 수 있습니다.

   2단계 - 이벤트 패턴 작성에는 이미 지정한 이벤트 패턴 정보가 포함되어 있으며 이를 수락하거나 업데이트할 수 있습니다.

   규칙을 생성하는 방법에 대한 자세한 내용은 ??? 섹션을 참조하세요.

Amazon EventBridge 이벤트 패턴 정의 모범 사례

다음은 이벤트 버스 규칙에서 이벤트 패턴을 정의할 때 고려해야 할 몇 가지 모범 사례입니다.

무한 루프를 작성하지 마세요.

EventBridge에서는 규칙이 반복적으로 실행되는 무한 루프로 이어지는 규칙을 생성할 수 있습니다. 예를 들어 규칙이 S3 버킷에서 ACL이 바뀐 것을 감지할 경우 소프트웨어를 트리거하여 ACL을 원하는 상태로 변경합니다. 이때 규칙이 부주의하게 작성되면 ACL에 대한 변경이 이어져 규칙을 다시 실행하면서 무한 루프에 빠지게 됩니다.

이러한 문제를 방지하려면 규칙에 대한 이벤트 패턴을 실제로 대상에 전송하려는 이벤트와만 일치하도록 최대한 정확하게 작성합니다. 위 예시에서는 트리거된 작업이 동일한 규칙을 다시 실행하지 않도록 이벤트와 일치하는 이벤트 패턴을 생성합니다. 예를 들어 ACL이 변경된 후가 아니라 잘못된 상태인 것으로 확인되는 경우에만 이벤트와 일치하는 이벤트 패턴을 규칙에 생성합니다. 자세한 내용은 ??? 및 ??? 섹션을 참조하세요.

무한 루프는 예상보다 높은 요금을 빠르게 야기할 수 있습니다. 또한 제한으로 이어질 수 있으며 이벤트 전송이 지연될 수 있습니다. 간접 호출 속도의 상한선을 모니터링하여 예상치 못한 볼륨 급증에 대한 경고를 받을 수 있습니다.

예산 책정을 통해 요금이 지정된 한도를 초과할 경우에 알림을 받을 수 있습니다. 자세한 내용은 예산을 통해 비용 관리 섹션을 참조하세요.

이벤트 패턴을 최대한 정확하게 구성하세요.

이벤트 패턴이 정확할수록 실제로 원하는 이벤트와만 일치할 가능성이 높아지고, 새 이벤트가 이벤트 소스에 추가되거나 기존 이벤트가 새 속성을 포함하도록 업데이트될 때 예기치 않은 일치를 피할 수 있습니다.

이벤트 패턴에는 다음과 일치하는 필터가 포함될 수 있습니다.

- 이벤트에 대한 이벤트 메타데이터(예: `source`, `detail-type`, `account` 또는 `region`)입니다.
- 이벤트 데이터, 즉 `detail` 객체 내부의 필드입니다.
- 이벤트 콘텐츠 또는 `detail` 객체 내 필드의 실제 값입니다.

대부분의 패턴은 `source` 및 `detail-type` 필터만 지정하는 것처럼 단순합니다. 그러나 EventBridge 패턴에는 이벤트의 모든 키 또는 값을 기준으로 필터링할 수 있는 유연성이 포함됩니다. 또한 `prefix`

및 suffix 필터와 같은 콘텐츠 필터를 적용하여 패턴의 정밀도를 높일 수 있습니다. 자세한 정보는 ???을 참조하세요.

이벤트 소스 및 세부 유형을 필터로 지정하세요.

source 및 detail-type 메타데이터 필드를 사용하여 이벤트 패턴을 더 정확하게 구성하면 무한 루프가 생성되고 원치 않는 이벤트가 일치되는 현상을 줄일 수 있습니다.

둘 이상의 필드 내에서 특정 값을 매칭해야 하는 경우 단일 값 배열 내에 가능한 모든 값을 나열하는 대신 $or 비교 연산자를 사용하세요.

를 통해 AWS CloudTrail전달되는 이벤트의 경우 eventName 필드를 필터로 사용하는 것이 좋습니다.

다음 이벤트 패턴 예제는 Amazon Simple Queue Service 서비스 CreateQueue 또는 SetQueueAttributes 해당 서비스의 이벤트와 CreateKey 일치하거나 해당 AWS Key Management Service 서비스에서 DisableKeyRotation 발생한 이벤트입니다.

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

```
}
```

계정 및 리전을 필터로 지정하세요.

이벤트 패턴에 `account` 및 `region` 필드를 포함하면 교차 계정 또는 교차 리전 이벤트 매칭을 제한할 수 있습니다.

콘텐츠 필터를 지정하세요.

콘텐츠 기반 필터링은 이벤트 패턴의 길이를 최소한으로 유지하면서 이벤트 패턴 정밀도를 개선하는 데 도움이 될 수 있습니다. 예를 들어 가능한 모든 숫자 값을 나열하는 대신 숫자 범위를 기반으로 매칭하는 것이 유용할 수 있습니다.

자세한 정보는 ???을 참조하세요.

이벤트 소스 업데이트를 고려하여 이벤트 패턴의 범위를 지정하세요.

이벤트 패턴을 생성할 때 이벤트 스키마와 이벤트 도메인이 시간이 지날수록 진화하고 확장할 수 있다는 점을 고려해야 합니다. 여기서도 이벤트 패턴을 최대한 정확하게 구성하면 이벤트 소스가 변경되거나 확장되는 경우 예기치 않은 일치를 줄일 수 있습니다.

예를 들어 결제 관련 이벤트를 게시하는 새 마이크로 서비스의 이벤트와 매칭한다고 가정해 보겠습니다. 처음에 서비스는 도메인 `acme.payments`를 사용하고 단일 이벤트인 `Payment accepted`를 게시합니다.

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

이때 결제 허용 이벤트와 일치하는 간단한 이벤트 패턴을 생성할 수 있습니다.

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

그러나 서비스가 나중에 거부된 결제에 대한 새 이벤트를 도입한다고 가정해 보겠습니다.

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

이 경우 생성한 단순 이벤트 패턴은 이제 `Payment accepted` 및 `Payment rejected` 이벤트와 모두 일치합니다. EventBridge 두 가지 유형의 이벤트를 모두 지정된 처리 대상으로 라우팅하므로 처리 실패와 추가 처리 비용이 발생할 수 있습니다.

이벤트 패턴의 범위를 `Payment accepted` 이벤트로만 지정하려면 최소한 `source`와 `detail-type`을 지정해야 합니다.

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

이벤트 패턴에 계정과 리전을 지정하여 교차 계정 또는 교차 리전 이벤트가 이 규칙과 일치하는 경우에 추가로 제한할 수도 있습니다.

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

이벤트 패턴을 검증하세요.

규칙이 원하는 이벤트와 일치하는지 확인하려면 이벤트 패턴을 검증하는 것이 좋습니다. EventBridge 콘솔 또는 API를 사용하여 이벤트 패턴을 검증할 수 있습니다.

- EventBridge 콘솔에서 규칙 생성 과정에서 이벤트 패턴을 만들고 테스트하거나 샌드박스를 사용하여 개별적으로 이벤트 패턴을 만들고 테스트할 수 있습니다.
- 액션을 사용하여 프로그래밍 방식으로 이벤트 패턴을 테스트할 수 있습니다. TestEventPattern

아마존 EventBridge 규칙

각 이벤트 버스에 전송되는 이벤트의 용도를 지정합니다. EventBridge 이렇게 하려면 규칙을 만들어야 합니다. 규칙은 어떤 이벤트를 어떤 대상으로 전송하여 처리할지 지정합니다. 단일 규칙으로 이벤트를 여러 대상으로 전송한 다음, 병렬로 실행합니다.

다음과 같은 두 가지 유형의 규칙을 생성할 수 있습니다.

- 이벤트 데이터와 일치하는 규칙

  이벤트 데이터 기준 (이벤트 패턴이라고 함) 을 기반으로 수신 이벤트와 일치하는 규칙을 만들 수 있습니다. 이벤트 패턴은 규칙이 일치시키는 이벤트 구조와 필드를 정의합니다. 이벤트가 이벤트 패턴에 정의된 기준과 일치하는 경우 지정한 대상으로 이벤트를 EventBridge 보냅니다.

  자세한 설명은 ??? 섹션을 참조하세요.
- 일정에 따라 실행되는 규칙

  지정된 간격으로 지정된 대상에 이벤트를 보내는 규칙을 만들 수도 있습니다. 예를 들어 Lambda 함수를 정기적으로 실행하려면 일정에 따라 실행할 규칙을 만들 수 있습니다.

> **Note**
>
> EventBridge 는 하나의 중앙 관리형 서비스에서 작업을 생성, 실행 및 관리할 수 있는 서버리스 스케줄러인 Amazon EventBridge Scheduler를 제공합니다. EventBridge Scheduler는 고도로 사용자 지정이 가능하며, 대상 API 작업 및 서비스의 범위가 더 넓어 EventBridge 예정된 규칙보다 향상된 확장성을 제공합니다. AWS
> EventBridge 스케줄러를 사용하여 일정에 따라 대상을 호출하는 것이 좋습니다. 자세한 설명은 ??? 섹션을 참조하세요.

다음 동영상에서는 규칙의 기본 사항에 대해 설명합니다. What are rules

아마존 EventBridge 관리형 규칙

서비스는 사용자가 생성하는 규칙 외에도 해당 AWS 서비스의 특정 기능에 필요한 EventBridge 규칙을 AWS 계정에서 만들고 관리할 수 있습니다. 이를 관리형 정책이라고 합니다.

서비스에서 관리형 규칙을 생성할 때 해당 서비스에 규칙을 생성할 권한을 부여하는 IAM 정책을 만들 수도 있습니다. 이러한 방식으로 생성된 IAM 정책은 필수 규칙만 생성하도록 허용하는 리소스 수준 권한을 통해 범위가 좁아집니다.

강제 삭제 옵션을 사용하여 관리형 규칙을 삭제할 수 있지만 다른 서비스에 더 이상 해당 규칙이 필요하지 않다고 확신하는 경우에만 삭제해야 합니다. 그렇지 않은 경우 관리형 규칙을 삭제하면 해당 규칙을 사용하는 기능이 작동하지 않게 됩니다.

이벤트에 반응하는 Amazon EventBridge 규칙 생성

Amazon EventBridge에서 수신한 이벤트에 대해 조치를 취하기 위해 규칙을 생성할 수 있습니다. 이벤트가 규칙에 정의된 이벤트 패턴과 일치하면 EventBridge는 이벤트를 지정된 대상으로 보내고 규칙에 정의된 작업을 트리거합니다.

다음 비디오에서는 다양한 종류의 규칙을 만들고 테스트하는 방법을 살펴봅니다. Learning about rules

이벤트에 응답하는 Amazon EventBridge 규칙을 생성하려면 다음 절차를 따릅니다.

이벤트에 반응하는 규칙 만들기

다음 단계는 이벤트가 지정된 이벤트 버스로 전송될 때 EventBridge에서 이벤트를 일치시키는 데 사용하는 규칙을 만드는 방법을 안내합니다.

단계

- 규칙 정의
- 이벤트 패턴 작성
- 대상 선택
- 태그 구성 및 규칙 검토

규칙 정의

먼저 규칙을 식별할 수 있도록 규칙의 이름과 설명을 입력합니다. 또한 규칙이 이벤트 패턴에 맞는 이벤트를 찾는 이벤트 버스를 정의해야 합니다.

규칙 세부 정보를 정의하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 규칙을 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙의 이름을 입력하고 선택적으로 설명을 입력합니다.

   규칙은 동일한 AWS 리전과 동일한 이벤트 버스의 다른 규칙과 동일한 이름을 가질 수 없습니다.

5. 이벤트 버스에서 이 규칙과 연결할 이벤트 버스를 선택합니다. 이 규칙이 자신의 계정에서 발생하는 이벤트와 일치하도록 하려면 AWS 기본 이벤트 버스를 선택합니다. 사용자 계정의 AWS 서비스가 이벤트를 내보내면 그 이벤트는 계정의 기본 이벤트 버스로 이동합니다.
6. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다.
7. 다음을 선택합니다.

이벤트 패턴 작성

다음으로 이벤트 패턴을 작성합니다. 이렇게 하려면 이벤트 소스를 지정하고, 이벤트 패턴의 기준을 선택하고, 일치시킬 속성과 값을 정의합니다. 또한 JSON으로 이벤트 패턴을 생성하고 샘플 이벤트와 비교하여 테스트할 수 있습니다.

이벤트 패턴을 작성하려면

1. 이벤트 소스에서 AWS 이벤트 또는 EventBridge 파트너 이벤트를 선택합니다.
2. (선택 사항) 샘플 이벤트 섹션에서 이벤트 패턴을 테스트할 샘플 이벤트 유형을 선택합니다.

   다음 샘플 이벤트 유형을 사용할 수 있습니다.

   - AWS 이벤트 - 지원되는 AWS 서비스에서 발생한 이벤트 중에서 선택합니다.
   - EventBridge 파트너 이벤트 - Salesforce와 같이 EventBridge를 지원하는 타사 서비스에서 발생한 이벤트 중에서 선택합니다.
   - 내 이벤트 입력 - JSON 텍스트로 자체 이벤트를 입력합니다.

     AWS 또는 파트너 이벤트를 시작점으로 삼아 사용자 지정 이벤트를 자체적으로 만들 수도 있습니다.

     1. AWS 이벤트 또는 EventBridge 파트너 이벤트를 선택합니다.
     2. 샘플 이벤트 드롭다운을 이용해 사용자 지정 이벤트의 시작점으로 사용할 이벤트를 선택합니다.

        EventBridge는 샘플 이벤트를 표시합니다.
     3. 복사를 선택합니다.
     4. 이벤트 유형에서 내 이벤트 입력을 선택합니다.
     5. JSON 편집 창에서 샘플 이벤트 구조를 삭제하고 AWS 또는 파트너 이벤트를 대신 붙여넣습니다.
     6. 이벤트 JSON을 편집하여 자체 샘플 이벤트를 생성합니다.

3. 생성 방법을 선택합니다. EventBridge 스키마 또는 템플릿에서 이벤트 패턴을 생성하거나 사용자 지정 이벤트 패턴을 생성할 수 있습니다.

   Existing schema

   기존 EventBridge 스키마를 사용하여 이벤트 패턴을 생성하려면 다음을 수행하세요.

   1. 생성 방법 섹션의 메서드에서 스키마 사용을 선택합니다.
   2. 이벤트 패턴 섹션의 스키마 유형에서 스키마 레지스트리에서 스키마 선택을 선택합니다.
   3. 스키마 레지스트리의 경우 드롭다운 상자를 선택하고 스키마 레지스트리 이름(예: `aws.events`)을 입력합니다. 표시되는 드롭다운 목록에서 옵션을 선택할 수도 있습니다.
   4. 스키마의 경우 드롭다운 상자를 선택하고 사용할 스키마 이름을 입력합니다. 예: `aws.s3@ObjectDeleted`. 표시되는 드롭다운 목록에서 옵션을 선택할 수도 있습니다.
   5. 모델 섹션에서 속성 옆에 있는 편집 버튼을 선택하여 해당 속성을 엽니다. 필요에 따라 관계 및 값 필드를 설정한 다음, 설정을 선택하여 속성을 저장합니다.

      > **Note**
      >
      > 속성 정의에 대한 자세한 내용을 보려면 속성 이름 옆에 있는 정보 아이콘을 선택하세요. 이벤트에서 속성 특성을 설정하는 방법에 대한 참조를 보려면 속성 특성 대화 상자의 참고 섹션을 엽니다.
      > 속성 특성을 삭제하려면 해당 속성의 편집 버튼을 선택한 다음, 지우기를 선택합니다.

   6. 이벤트 패턴을 JSON 텍스트로 생성하고 검증하려면 JSON으로 이벤트 패턴 생성을 선택합니다.
   7. (선택 사항) 테스트 패턴과 비교하여 샘플 이벤트를 테스트하려면 테스트 패턴을 선택합니다.

      EventBridge는 샘플 이벤트가 이벤트 패턴과 일치하는지 여부를 나타내는 메시지 상자를 표시합니다.

      다음 옵션 중 하나를 선택할 수도 있습니다.

      - 복사 - 이벤트 패턴을 디바이스의 클립보드에 복사합니다.
      - 정리 - 줄 바꿈, 탭, 공백을 추가하여 JSON 텍스트를 더 쉽게 읽을 수 있습니다.

Custom schema

사용자 지정 스키마를 작성하고 이벤트 패턴으로 변환하려면 다음을 수행하세요.

1. 생성 방법 섹션의 메서드에서 스키마 사용을 선택합니다.
2. 이벤트 패턴 섹션의 스키마 유형에서 스키마 입력을 선택합니다.
3. 텍스트 상자에 스키마를 입력합니다. 스키마의 형식을 유효한 JSON 텍스트로 지정해야 합니다.
4. 모델 섹션에서 속성 옆에 있는 편집 버튼을 선택하여 해당 속성을 엽니다. 필요에 따라 관계 및 값 필드를 설정한 다음, 설정을 선택하여 속성을 저장합니다.

   > **Note**
   >
   > 속성 정의에 대한 자세한 내용을 보려면 속성 이름 옆에 있는 정보 아이콘을 선택하세요. 이벤트에서 속성 특성을 설정하는 방법에 대한 참조를 보려면 속성 특성 대화 상자의 참고 섹션을 엽니다.
   > 속성 특성을 삭제하려면 해당 속성의 편집 버튼을 선택한 다음, 지우기를 선택합니다.

5. 이벤트 패턴을 JSON 텍스트로 생성하고 검증하려면 JSON으로 이벤트 패턴 생성을 선택합니다.
6. (선택 사항) 테스트 패턴과 비교하여 샘플 이벤트를 테스트하려면 테스트 패턴을 선택합니다.

   EventBridge는 샘플 이벤트가 이벤트 패턴과 일치하는지 여부를 나타내는 메시지 상자를 표시합니다.

   다음 옵션 중 하나를 선택할 수도 있습니다.

   - 복사 - 이벤트 패턴을 디바이스의 클립보드에 복사합니다.
   - 정리 - 줄 바꿈, 탭, 공백을 추가하여 JSON 텍스트를 더 쉽게 읽을 수 있습니다.

Event pattern

JSON 형식의 사용자 지정 이벤트 패턴을 작성하려면 다음을 수행하세요.

1. 생성 방법 섹션의 메서드에서 사용자 지정 패턴(JSON 편집기)을 선택합니다.

2. 이벤트 패턴의 경우 JSON 형식 텍스트로 사용자 지정 이벤트 패턴을 입력합니다.
3. (선택 사항) 테스트 패턴과 비교하여 샘플 이벤트를 테스트하려면 테스트 패턴을 선택합니다.

   EventBridge는 샘플 이벤트가 이벤트 패턴과 일치하는지 여부를 나타내는 메시지 상자를 표시합니다.

   다음 옵션 중 하나를 선택할 수도 있습니다.

   - 복사 - 이벤트 패턴을 디바이스의 클립보드에 복사합니다.
   - 정리 - 줄 바꿈, 탭, 공백을 추가하여 JSON 텍스트를 더 쉽게 읽을 수 있습니다.
   - 이벤트 패턴 양식 - 패턴 빌더에서 이벤트 패턴을 엽니다. 패턴 빌더에서 패턴을 있는 그대로 렌더링할 수 없는 경우 EventBridge는 패턴 빌더를 열기 전에 경고를 표시합니다.

4. 다음을 선택합니다.

대상 선택

지정된 패턴과 일치하는 이벤트를 수신할 대상을 하나 이상 선택합니다. 대상은 EventBridge 이벤트 버스, Salesforce 등의 SaaS 파트너를 포함한 EventBridge API 대상 또는 다른 AWS 서비스를 포함할 수 있습니다.

대상을 선택하려면

1. 대상 유형에서 다음 대상 유형 중 하나를 선택합니다.

   Event bus

   EventBridge 이벤트 버스를 선택하려면 EventBridge 이벤트 버스를 선택하고 다음을 수행합니다.

   - 이 규칙과 동일하게 AWS 리전에서 이벤트 버스를 사용하려면:
     1. 동일한 계정 및 리전의 이벤트 버스를 선택합니다.
     2. 대상에 대한 이벤트 버스의 경우 드롭다운 상자를 선택하고 이벤트 버스의 이름을 입력합니다. 드롭다운 목록에서 이벤트 버스를 선택할 수도 있습니다.

        자세한 내용은 ??? 섹션을 참조하세요.
   - 다음 규칙에 따라 다른 AWS 리전 또는 계정의 이벤트 버스를 사용하려면:
     1. 다른 계정 또는 리전의 이벤트 버스를 선택합니다.

2. 대상 이벤트 버스에는 사용하려는 이벤트 버스의 ARN을 입력합니다.

   자세한 내용은 다음을 참조하세요.

   - ???
   - ???

API destination

EventBridge API 대상을 사용하려면 EventBridge API 대상을 선택한 후 다음 중 하나를 수행합니다.

- 기존 API 대상을 사용하려면 기존 API 대상 사용을 선택합니다. 그런 다음 드롭다운 목록에서 API 대상을 선택합니다.
- 새 API 대상을 생성하려면 새 API 대상 생성을 선택합니다. 그런 다음 대상에 대해 다음 세부 정보를 제공합니다.
  - 이름 - 대상의 이름을 입력합니다.

    이름은 AWS 계정 내에서 고유해야 합니다. 이름은 최대 64자까지 가능합니다. 유효한 문자는 A-Z, a-z, 0-9 및 . _ -(하이픈)입니다.
  - (선택 사항) 설명 – 대상에 대한 설명을 입력합니다.

    설명은 최대 512자까지 가능합니다.
  - API 대상 엔드포인트 — 대상의 URL 엔드포인트입니다.

    엔드포인트 URL은 **`https`**로 시작해야 합니다. **`*`**를 경로 파라미터 와일드카드로 포함할 수 있습니다. 대상 `HttpParameters` 속성에서 경로 파라미터를 설정할 수 있습니다.
  - HTTP 메서드 - 엔드포인트를 간접 호출할 때 사용할 HTTP 메서드를 선택합니다.
  - (선택 사항) 초당 간접 호출 속도 제한 — 이 대상에 대해 초당 허용되는 최대 간접 호출 수를 입력합니다.

    이 값은 0보다 커야 합니다. 기본적으로 이 값은 300으로 설정됩니다.
  - 연결 — 새 연결을 사용할지 기존 연결을 사용할지 선택합니다.
    - 기존 연결을 사용하려면 기존 연결 사용을 선택하고 드롭다운 목록에서 연결을 선택합니다.

- 이 대상에 대한 새 연결을 만들려면 새 연결 만들기를 선택한 다음 연결의 이름, 대상 유형 및 권한 부여 유형을 정의합니다. 이 연결에 대한 설명(선택 사항)을 추가할 수도 있습니다.

자세한 내용은 ??? 섹션을 참조하세요.

AWS 서비스

AWS 서비스를 사용하려면 AWS 서비스를 선택하고 다음을 수행합니다.

1. 대상 선택에서 대상으로 사용할 AWS 서비스를 선택합니다. 선택한 서비스에 대해 요청된 정보를 제공합니다.

> **Note**
>
> 표시되는 필드는 선택한 서비스에 따라 달라집니다. 사용 가능한 대상에 대한 자세한 내용은 EventBridge 콘솔에서 대상을 사용할 수 있습니다. 섹션을 참조하세요.

2. 여러 대상 유형에 대해 EventBridge에서는 대상에 이벤트를 보낼 권한이 필요합니다. 이 경우 EventBridge는 규칙 실행에 필요한 IAM 역할을 생성할 수 있습니다.

   실행 역할에서는 다음 중 하나를 수행합니다.

   - 이 규칙의 새 실행 역할을 만들려면:
     a. 이 특정 리소스에 대해 새 역할 생성을 선택합니다.
     b. 이 실행 역할의 이름을 입력하거나 EventBridge에서 생성된 이름을 사용합니다.
   - 이 규칙에 기존 실행 역할을 사용하려면:
     a. 기존 역할 사용을 선택합니다.
     b. 드롭다운 목록에서 사용할 실행 역할의 이름을 입력하거나 선택합니다.

3. (선택 사항) 추가 설정의 경우 대상 유형에 사용할 수 있는 선택적 설정을 지정합니다.

   Event bus

   (선택 사항) DLQ(Dead Letter Queue)에서 표준 Amazon SQS 대기열을 배달 못한 편지 대기열로 사용할지를 선택합니다. 이벤트가 대상에 성공적으로 전달되지 않은 경우 EventBridge는 이 규칙과 일치하는 이벤트를 배달 못한 편지 대기열로 보냅니다. 다음 중 하나를 수행합니다.

   - 배달 못한 편지 대기열을 사용하지 않으려면 없음(None)을 선택합니다.

- 현재 AWS 계정에서 배달 못한 편지 대기열로 사용할 대기열 선택(Select an Amazon SQS queue in the current account to use as the dead-letter queue)을 선택하고 드롭다운 목록에서 사용할 대기열을 선택합니다.
- 다른 AWS 계정에서 배달 못한 편지 대기열로 사용할 Amazon SQS 대기열 선택(Select an Amazon SQS queue in an other account as a dead-letter queue)을 선택하고 사용할 대기열의 ARN을 입력합니다. 메시지를 보낼 수 있는 EventBridge 권한을 부여하는 리소스 기반 정책을 대기열에 연결해야 합니다.

  자세한 내용은 DLQ(Dead Letter Queue)에 권한 부여 섹션을 참조하세요.

API destination

1. (선택 사항) 대상 입력 구성의 경우 일치하는 이벤트를 위해 대상으로 전송되는 텍스트를 사용자 지정하는 방법을 선택합니다. 다음 중 하나를 선택합니다.
   - 일치하는 이벤트 — EventBridge는 원본 소스 이벤트 전체를 대상으로 보냅니다. 이 값이 기본값입니다.
   - 일치하는 이벤트의 일부 — EventBridge는 원본 소스 이벤트의 지정된 부분만 대상에 보냅니다.

     일치하는 이벤트의 일부를 지정합니다.에서 EventBridge가 대상으로 전송할 이벤트 부분을 정의하는 JSON 경로를 지정합니다.
   - 상수(JSON 텍스트) - EventBridge는 지정된 JSON 텍스트만 대상으로 전송합니다. 원본 소스 이벤트의 어떤 부분도 전송되지 않습니다.

     JSON으로 상수를 지정에서 EventBridge가 이벤트 대신 대상으로 전송할 JSON 텍스트를 지정합니다.
   - 입력 변환기 - EventBridge가 대상으로 보낼 텍스트를 사용자 지정하도록 입력 변환기를 구성합니다. 자세한 내용은 ??? 섹션을 참조하세요.

     a. 입력 변환기 구성을 선택합니다.

     b. ???의 단계에 따라 입력 변환기를 구성합니다.
2. (선택 사항) 재시도 정책에서 오류 발생 후 EventBridge가 대상으로의 이벤트 전송을 재시도하는 방법을 지정합니다.
   - 최대 이벤트 기간 - EventBridge에서 처리되지 않은 이벤트를 유지할 수 있는 최대 시간(시간, 분, 초)을 입력합니다. 기본값은 18시간입니다.

- 재시도 - 오류 발생 후 EventBridge가 대상으로 이벤트 전송을 재시도해야 하는 최대 횟수를 입력합니다. 기본값은 185회입니다.

3. (선택 사항) DLQ(Dead Letter Queue)에서 표준 Amazon SQS 대기열을 배달 못한 편지 대기열로 사용할지를 선택합니다. 이벤트가 대상에 성공적으로 전달되지 않은 경우 EventBridge는 이 규칙과 일치하는 이벤트를 배달 못한 편지 대기열로 보냅니다. 다음 중 하나를 수행합니다.
   - 배달 못한 편지 대기열을 사용하지 않으려면 없음(None)을 선택합니다.
   - 현재 AWS 계정에서 배달 못한 편지 대기열로 사용할 대기열 선택(Select an Amazon SQS queue in the current account to use as the dead-letter queue)을 선택하고 드롭다운 목록에서 사용할 대기열을 선택합니다.
   - 다른 AWS 계정에서 배달 못한 편지 대기열로 사용할 Amazon SQS 대기열 선택(Select an Amazon SQS queue in an other account as a dead-letter queue)을 선택하고 사용할 대기열의 ARN을 입력합니다. 메시지를 보낼 수 있는 EventBridge 권한을 부여하는 리소스 기반 정책을 대기열에 연결해야 합니다.

     자세한 내용은 DLQ(Dead Letter Queue)에 권한 부여 섹션을 참조하세요.

AWS service

EventBridge는 특정 AWS 서비스에 대해 다음 필드를 모두 표시하지 않을 수 있다는 점에 유의합니다.

1. (선택 사항) 대상 입력 구성의 경우 일치하는 이벤트를 위해 대상으로 전송되는 텍스트를 사용자 지정하는 방법을 선택합니다. 다음 중 하나를 선택합니다.
   - 일치하는 이벤트 — EventBridge는 원본 소스 이벤트 전체를 대상으로 보냅니다. 이 값이 기본값입니다.
   - 일치하는 이벤트의 일부 — EventBridge는 원본 소스 이벤트의 지정된 부분만 대상에 보냅니다.

     일치하는 이벤트의 일부를 지정합니다.에서 EventBridge가 대상으로 전송할 이벤트 부분을 정의하는 JSON 경로를 지정합니다.
   - 상수(JSON 텍스트) - EventBridge는 지정된 JSON 텍스트만 대상으로 전송합니다. 원본 소스 이벤트의 어떤 부분도 전송되지 않습니다.

     JSON으로 상수를 지정에서 EventBridge가 이벤트 대신 대상으로 전송할 JSON 텍스트를 지정합니다.

- 입력 변환기 - EventBridge가 대상으로 보낼 텍스트를 사용자 지정하도록 입력 변환기를 구성합니다. 자세한 내용은 ??? 섹션을 참조하세요.

  a. 입력 변환기 구성을 선택합니다.

  b. ???의 단계에 따라 입력 변환기를 구성합니다.

2. (선택 사항) 재시도 정책에서 오류 발생 후 EventBridge가 대상으로의 이벤트 전송을 재시도하는 방법을 지정합니다.
   - 최대 이벤트 기간 - EventBridge에서 처리되지 않은 이벤트를 유지할 수 있는 최대 시간(시간, 분, 초)을 입력합니다. 기본값은 18시간입니다.
   - 재시도 - 오류 발생 후 EventBridge가 대상으로 이벤트 전송을 재시도해야 하는 최대 횟수를 입력합니다. 기본값은 185회입니다.
3. (선택 사항) DLQ(Dead Letter Queue)에서 표준 Amazon SQS 대기열을 배달 못한 편지 대기열로 사용할지를 선택합니다. 이벤트가 대상에 성공적으로 전달되지 않은 경우 EventBridge는 이 규칙과 일치하는 이벤트를 배달 못한 편지 대기열로 보냅니다. 다음 중 하나를 수행합니다.
   - 배달 못한 편지 대기열을 사용하지 않으려면 없음(None)을 선택합니다.
   - 현재 AWS 계정에서 배달 못한 편지 대기열로 사용할 대기열 선택(Select an Amazon SQS queue in the current account to use as the dead-letter queue)을 선택하고 드롭다운 목록에서 사용할 대기열을 선택합니다.
   - 다른 AWS 계정에서 배달 못한 편지 대기열로 사용할 Amazon SQS 대기열 선택(Select an Amazon SQS queue in an other account as a dead-letter queue)을 선택하고 사용할 대기열의 ARN을 입력합니다. 메시지를 보낼 수 있는 EventBridge 권한을 부여하는 리소스 기반 정책을 대기열에 연결해야 합니다.

     자세한 내용은 DLQ(Dead Letter Queue)에 권한 부여 섹션을 참조하세요.
4. (선택 사항) 이 규칙에 다른 대상을 추가하려면 다른 대상 추가를 선택합니다.
5. 다음을 선택합니다.

EventBridge는 특정 AWS 서비스에 대해 다음 필드를 모두 표시하지 않을 수 있다는 점에 유의합니다.

태그 구성 및 규칙 검토

마지막으로 규칙에 원하는 태그를 입력한 다음 규칙을 검토 및 생성합니다.

태그를 구성하고 규칙을 검토 및 생성하려면

1. (선택 사항) 규칙에 대해 하나 이상의 태그를 입력하세요. 자세한 내용은 아마존 EventBridge 태그 섹션을 참조하세요.
2. 다음을 선택합니다.
3. 새 규칙의 세부 정보를 검토합니다. 섹션을 변경하려면 해당 섹션 옆에 있는 편집 버튼을 선택합니다.

   규칙 세부 정보에 만족하면 규칙 생성을 선택합니다.

Amazon EventBridge와 함께 Amazon EventBridge 스케줄러 사용

Amazon EventBridge 스케줄러는 하나의 중앙 관리형 서비스에서 작업을 생성, 실행 및 관리할 수 있는 서버리스 스케줄러입니다. EventBridge 스케줄러를 사용하면 반복 패턴에 대해 cron 및 rate 표현식을 사용하여 일정을 만들거나 일회성 간접 호출을 구성할 수 있습니다. 전송을 위한 유연한 기간을 설정하고, 재시도 제한을 정의하고, 실패한 API 간접 호출의 최대 보존 시간을 설정할 수 있습니다.

EventBridge 스케줄러는 고도로 사용자 지정이 가능하며, 광범위한 대상 API 작업 및 AWS 서비스를 통해 EventBridge 예약 규칙보다 향상된 확장성을 제공합니다. EventBridge 스케줄러를 사용하여 일정에 따라 대상을 호출하는 것이 좋습니다.

주제

실행 역할 설정

새 일정을 생성할 때 EventBridge 스케줄러에 사용자를 대신하여 대상 API 작업을 간접적으로 호출할 수 있는 권한이 있어야 합니다. 실행 역할을 사용하여 EventBridge 스케줄러에 이러한 권한을 부여합니다. 일정의 실행 역할에 연결하는 권한 정책은 필요한 권한을 정의합니다. 이러한 권한은 EventBridge 스케줄러가 간접적으로 호출하려는 대상 API에 따라 달라집니다.

다음 절차와 같이 EventBridge 스케줄러 콘솔을 사용하여 일정을 생성하면 EventBridge 스케줄러가 선택한 대상을 기준으로 실행 역할을 자동으로 설정합니다. EventBridge 스케줄러 SDK, AWS CLI 또는 AWS CloudFormation 중 하나를 사용하여 일정을 생성하려면 EventBridge 스케줄러가 대상을 간접적으로 호출하는 데 필요한 권한을 부여하는 기존 실행 역할이 있어야 합니다. 일정에 대한 실행 역할을 수동으로 설정하는 방법에 대한 자세한 내용은 EventBridge 스케줄러 사용 설명서의 Setting up an execution role을 참조하세요.

일정 생성

콘솔을 사용하여 일정 생성

1. https://console.aws.amazon.com/scheduler/home에서 Amazon EventBridge 스케줄러 콘솔을 엽니다.
2. 일정 페이지에서 일정 생성을 선택합니다.

3. 일정 세부 정보 지정 페이지의 일정 이름 및 설명 섹션에서 다음을 수행합니다.

   a. 일정 이름에 일정의 이름을 입력합니다. 예: **`MyTestSchedule`**.

   b. (선택 사항) 설명에 일정에 대한 설명을 입력합니다. 예: **`My first schedule`**.

   c. 일정 그룹 드롭다운 목록에서 일정 그룹을 선택합니다. 그룹이 없는 경우 기본값을 선택합니다. 일정 그룹을 생성하려면 자체 일정 생성을 선택합니다.

      일정 그룹을 사용하여 일정 그룹에 태그를 추가합니다.

4. • 일정 옵션을 선택합니다.

발생	수행할 작업	
일회성 일정 일회성 일정은 사용자가 지정하는 날짜와 시간에 한 번만 대상을 간접적으로 호출합니다.	날짜 및 시간에 대해 다음을 수행합니다. • YYYY/MM/DD 형식으로 유효한 날짜를 입력합니다. • 24시간 hh:mm 형식으로 타임스탬프를 입력합니다. • 시간대에서 시간대를 선택하세요.	
반복되는 일정 반복 일정은 cron 표현식 또는 rate 표현식을 사용하여 지정한 속도로 대상을 간접적으로 호출합니다.	a. 일정 유형에서 다음 중 하나를 수행합니다. • cron 표현식을 사용하여 일정을 정의하려면 Cron 기반 일정을 선택하고 cron 표현식을 입력합니다. • rate 표현식을 사용하여 일정을 정의하려면 Rate 기반 일정을 선택하고 rate 표현식을 입력합니다.	

발생	수행할 작업	
	cron 및 rate 표현식에 대한 자세한 내용은 EventBridge Scheduler 사용 설명서의 EventBridge 스케줄러의 스케줄 유형을 참조하세요. b. 유연한 기간에서 끄기를 선택하여 옵션을 끄거나 미리 정의된 기간 중 하나를 선택합니다. 예를 들어, 15분을 선택하고 1시간에 한 번씩 대상을 간접적으로 호출하도록 반복 일정을 설정하면 일정은 매시간 시작 후 15분 이내에 실행됩니다.	

5. (선택 사항) 이전 단계에서 반복 일정을 선택한 경우 기간 섹션에서 다음을 수행합니다.
   a. 시간대에서 시간대를 선택합니다.
   b. 시작 날짜 및 시간에 YYYY/MM/DD 형식으로 유효한 날짜를 입력한 다음 24시간 hh:mm 형식으로 타임스탬프를 지정합니다.
   c. 종료 날짜 및 시간에 YYYY/MM/DD 형식으로 유효한 날짜를 입력한 다음 24시간 hh:mm 형식으로 타임스탬프를 지정합니다.
6. 다음을 선택합니다.
7. 대상 선택 페이지에서 EventBridge 스케줄러가 간접적으로 호출하는 AWS API 작업을 선택합니다.
   a. 대상 API의 경우 템플릿 형식의 대상을 선택합니다.
   b. Amazon EventBridge PutEvents를 선택합니다.
   c. PutEvents에서 다음을 지정합니다.

- EventBridge 이벤트 버스의 경우 드롭다운 메뉴에서 이벤트 버스를 선택합니다. 예: **`default`**.

  EventBridge 콘솔에서 새 이벤트 버스 생성을 선택하여 새 이벤트 버스를 생성할 수도 있습니다.
- 세부 유형에는 매칭하려는 이벤트의 세부 유형을 입력합니다. 예: **`Object Created`**.
- 소스에는 이벤트의 소스인 서비스 이름을 입력합니다.

  AWS 서비스 이벤트의 경우 서비스 접두사를 소스로 지정합니다. `aws.` 접두사는 포함하지 마십시오. 예를 들어, Amazon S3 이벤트의 경우 **`s3`**을 입력합니다.

  서비스의 접두사를 확인하려면 서비스 권한 부여 참조의 조건 키 테이블을 참조하십시오. 소스 및 세부 정보 유형 이벤트 값에 대한 자세한 내용은 ??? 섹션을 참조하십시오.
- (선택 사항): 세부 정보에 이벤트 패턴을 입력하여 EventBridge 스케줄러가 EventBridge로 보내는 이벤트를 추가로 필터링합니다.

  자세한 내용은 ??? 섹션을 참조하세요.

8. 다음을 선택합니다.
9. 설정 페이지에서 다음 작업을 수행합니다.

   a. 일정을 켜려면 일정 상태에서 일정 활성화를 토글합니다.

   b. 일정에 대한 재시도 정책을 구성하려면 재시도 정책 및 데드-레터 큐(DLQ)에서 다음을 수행합니다.

   - 재시도를 토글합니다.
   - 최대 이벤트 수명에 EventBridge 스케줄러가 처리되지 않은 이벤트를 보관해야 하는 최대 시간과 분을 입력합니다.
   - 최대 시간은 24시간입니다.
   - 최대 재시도 횟수에는 대상이 오류를 반환할 경우 EventBridge 스케줄러가 일정을 재시도하는 최대 횟수를 입력합니다.

     최댓값은 185회입니다.

   재시도 정책을 사용하면 일정이 대상을 간접적으로 호출하지 못할 경우 EventBridge 스케줄러가 일정을 다시 실행합니다. 구성된 경우 일정에 대한 최대 보존 기간과 재시도 횟수를 설정해야 합니다.

c. EventBridge 스케줄러가 전송되지 않은 이벤트를 저장하는 위치를 선택합니다.

DLQ(Dead Letter Queue) 옵션	수행할 작업	
저장 안 함	None을 선택합니다.	
일정을 생성하는 위치와 같은 AWS 계정에 이벤트 저장	a. 내 AWS 계정의 Amazon SQS 대기열을 DLQ로 선택을 선택합니다. b. Amazon SQS 대기열의 Amazon 리소스 이름(ARN)을 선택합니다.	
일정을 생성하는 위치와 다른 AWS 계정에 이벤트 저장	a. 다른 AWS 계정의 Amazon SQS 대기열을 DLQ로 지정을 선택합니다. b. Amazon SQS 대기열의 Amazon 리소스 이름(ARN)을 입력합니다.	

d. 고객 관리형 키를 사용하여 대상 입력을 암호화하려면 암호화에서 암호화 설정 사용자 지정(고급)을 선택합니다.

이 옵션을 선택하는 경우 기존 KMS 키 ARN을 입력하거나 AWS KMS key 생성을 선택하여 AWS KMS 콘솔로 이동합니다. EventBridge 스케줄러가 저장 데이터를 암호화하는 방법에 대한 자세한 내용은 Amazon EventBridge 스케줄러 사용 설명서의 Encryption at rest를 참조하세요.

e. EventBridge 스케줄러가 새 실행 역할을 생성하도록 하려면 이 일정에 대한 새 역할 생성을 선택합니다. 그런 다음 역할 이름을 입력합니다. 이 옵션을 선택하면 EventBridge 스케줄러가 템플릿 대상에 필요한 필수 권한을 역할에 연결합니다.

10. 다음을 선택합니다.
11. 일정 검토 및 생성 페이지에서 일정의 세부 정보를 검토합니다. 각 섹션에서 편집을 선택하여 해당 단계로 돌아가서 세부 정보를 편집합니다.
12. 일정 생성을 선택합니다.

일정 페이지에서 새 일정과 기존 일정 목록을 볼 수 있습니다. 상태 열에서 새 일정이 활성화됨 상태인지 확인합니다.

관련 리소스

EventBridge 스케줄러에 대한 자세한 내용은 다음을 참조하세요.

- EventBridge 스케줄러 사용 설명서
- EventBridge 스케줄러 API 참조
- EventBridge 스케줄러 요금

일정에 따라 실행되는 Amazon EventBridge 규칙 생성

규칙은 이벤트에 대한 응답으로 또는 특정 시간 간격으로 실행될 수 있습니다. 예를 들어 AWS Lambda 함수를 정기적으로 실행하려면 일정에 따라 실행되는 규칙을 생성하면 됩니다.

> **Note**
>
> EventBridge는 하나의 중앙 관리형 서비스에서 작업을 생성, 실행 및 관리할 수 있는 서버리스 스케줄러인 Amazon EventBridge 스케줄러를 제공합니다. EventBridge 스케줄러는 고도로 사용자 지정이 가능하며, 광범위한 대상 API 작업 및 AWS 서비스를 통해 EventBridge 예약 규칙보다 향상된 확장성을 제공합니다.
> EventBridge Scheduler를 사용하여 일정에 따라 대상을 간접 호출하는 것이 좋습니다. 자세한 내용은 ??? 섹션을 참조하세요.

EventBridge에서는 다음과 같은 2가지 유형의 예약된 규칙을 생성할 수 있습니다.

- 일정한 간격으로 실행되는 규칙

  EventBridge는 일정한 간격으로(예: 20분마다) 이러한 규칙을 실행합니다.

  예약된 규칙의 간격을 지정하려면 rate 표현식을 정의합니다.
- 특정 시간에 실행되는 규칙

  EventBridge는 특정 시간과 날짜에 이러한 규칙을 실행합니다(예: 매월 첫째 월요일 오전 8시(PST)).

예약된 규칙이 실행되는 시간과 날짜를 지정하려면 cron 표현식을 정의합니다.

rate 표현식은 정의하기가 더 간단하고 cron 표현식은 세부적인 일정 제어를 제공합니다. 예를 들어, cron 표현식을 사용하여 매주 또는 매월 특정 요일의 지정된 시간에 트리거되는 규칙을 정의할 수 있습니다. 반대로 rate 표현식은 매 시간 한 번 또는 매일 한 번과 같이 일정한 간격으로 규칙을 실행합니다.

예약된 모든 이벤트는 UTC+0 시간대를 사용하며 예약의 최소 단위는 1분입니다.

> **Note**
>
> EventBridge는 일정 표현식에 초 단위의 정밀성을 제공하지 않습니다. cron 표현식을 사용해 가장 정밀하게 설정할 수 있는 단위가 1분입니다. EventBridge와 대상 서비스가 분산되어 있기 때문에 예약된 규칙이 트리거되는 시간과 대상 서비스가 대상 리소스 실행하는 시간 사이에 몇 초의 지연이 있을 수 있습니다.

다음 비디오는 작업 예약에 대한 개요를 제공합니다. Creating scheduled tasks with EventBridge

주제

- 일정에 따라 실행되는 규칙 생성
- cron 표현식 참조
- rate 표현식 참조

일정에 따라 실행되는 규칙 생성

다음 단계는 정기적인 일정에 따라 실행되는 EventBridge 규칙을 만드는 방법을 안내합니다.

> **Note**
>
> 기본 이벤트 버스를 사용해야만 예약된 규칙을 생성할 수 있습니다.

단계

- 규칙 정의

- 일정 정의
- 대상 선택
- 태그 구성 및 규칙 검토

규칙 정의

먼저 규칙을 식별할 수 있도록 규칙의 이름과 설명을 입력합니다.

규칙 세부 정보를 정의하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 [Rules]를 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙의 이름을 입력하고 선택적으로 설명을 입력합니다.

   규칙은 동일한 AWS 리전과 동일한 이벤트 버스의 다른 규칙과 동일한 이름을 가질 수 없습니다.
5. 이벤트 버스에서 기본 이벤트 버스를 선택합니다. 기본 이벤트 버스를 사용해야만 예약된 규칙을 생성할 수 있습니다.
6. 규칙을 생성하는 즉시 적용되도록 하려면 선택한 이벤트 버스에 대해 규칙 활성화 옵션이 활성화되어 있어야 합니다.
7. 규칙 유형에서 스케줄을 선택합니다.

   이제 일정에 따라 실행되는 규칙을 계속 생성하거나 Amazon EventBridge 스케줄러를 사용할 수 있습니다.
8. 계속할 방법을 선택합니다.

   - EventBridge Scheduler를 사용하여 일정 생성

   > **Note**
   >
   > EventBridge Scheduler는 하나의 중앙 관리형 서비스에서 작업을 생성, 실행 및 관리할 수 있는 서버리스 스케줄러입니다. 이벤트 버스 및 규칙과 관계없이 일회성 및 반복 예약 기능을 제공합니다. EventBridge 스케줄러는 고도로 사용자 지정이 가능하며, 광범위한 대상 API 작업 및 AWS 서비스를 통해 EventBridge 예약 규칙보다 향상된 확장성을 제공합니다.

EventBridge Scheduler를 사용하여 일정에 따라 대상을 간접 호출하는 것이 좋습니다. 자세한 내용은 Amazon EventBridge 스케줄러 사용자 가이드의 Amazon EventBridge 스케줄러란?을 참조합니다.

1. EventBridge Scheduler에서 계속을 선택합니다.

   EventBridge는 EventBridge Scheduler 콘솔에서 일정 생성 페이지를 엽니다.
2. EventBridge Scheduler 콘솔에서 일정을 생성합니다.

- EventBridge를 계속 사용하여 기본 이벤트 버스에 대한 예약 규칙 생성
  1. 규칙 생성으로 이동을 선택합니다.

일정 정의

다음으로 일정 패턴을 정의합니다.

일정 패턴을 정의하려면

1. 일정 패턴에서 일정을 특정 시간에 실행할지 아니면 일정한 간격으로 실행할지를 선택합니다.

   Specific time

   1. 특정 시간(예: 매월 첫 번째 월요일 오전 8시 PST)에 실행되는 세분화된 일정을 선택합니다.
   2. Cron 표현식의 경우 EventBridge가 이 예약된 규칙을 실행할 시기를 결정하는 데 사용해야 하는 cron 표현식을 정의하는 필드를 지정합니다.

      모든 필드를 지정하고 나면 EventBridge는 EventBridge가 이 예약된 규칙을 실행할 다음 10개 날짜를 표시합니다. 해당 날짜를 UTC 또는 현지 시간대로 표시할지 선택할 수 있습니다.

      cron 표현식 구성에 대한 자세한 내용은 ??? 섹션을 참조하세요.

   Regular rate

   1. 일정한 간격(예: 10분마다)으로 실행되는 일정을 선택합니다.
   2. rate 표현식의 경우 값 및 단위 필드를 지정하여 EventBridge가 이 예약된 규칙을 실행해야 하는 간격을 정의합니다.

      rate 표현식 구성에 대한 자세한 내용은 ??? 섹션을 참조하세요.

2. 다음을 선택합니다.

대상 선택

지정된 패턴과 일치하는 이벤트를 수신할 대상을 하나 이상 선택합니다. 대상은 EventBridge 이벤트 버스, Salesforce 등의 SaaS 파트너를 포함한 EventBridge API 대상 또는 다른 AWS 서비스를 포함할 수 있습니다.

대상을 선택하려면

1. 대상 유형에서 다음 대상 유형 중 하나를 선택합니다.

   Event bus

   EventBridge 이벤트 버스를 선택하려면 EventBridge 이벤트 버스를 선택하고 다음을 수행합니다.

   - 이 규칙과 동일하게 AWS 리전에서 이벤트 버스를 사용하려면:
     1. 동일한 계정 및 리전의 이벤트 버스를 선택합니다.
     2. 대상에 대한 이벤트 버스의 경우 드롭다운 상자를 선택하고 이벤트 버스의 이름을 입력합니다. 드롭다운 목록에서 이벤트 버스를 선택할 수도 있습니다.

        자세한 내용은 ??? 섹션을 참조하세요.
   - 다음 규칙에 따라 다른 AWS 리전 또는 계정의 이벤트 버스를 사용하려면:
     1. 다른 계정 또는 리전의 이벤트 버스를 선택합니다.
     2. 대상 이벤트 버스에는 사용하려는 이벤트 버스의 ARN을 입력합니다.

        자세한 내용은 다음을 참조하세요.

        - ???
        - ???

   API destination

   EventBridge API 대상을 사용하려면 EventBridge API 대상을 선택한 후 다음 중 하나를 수행합니다.

   - 기존 API 대상을 사용하려면 기존 API 대상 사용을 선택합니다. 그런 다음 드롭다운 목록에서 API 대상을 선택합니다.

- 새 API 대상을 생성하려면 새 API 대상 생성을 선택합니다. 그런 다음 대상에 대해 다음 세부 정보를 제공합니다.
  - 이름 - 대상의 이름을 입력합니다.

    이름은 AWS 계정 내에서 고유해야 합니다. 이름은 최대 64자까지 가능합니다. 유효한 문자는 A-Z, a-z, 0-9 및 . _ -(하이픈)입니다.
  - (선택 사항) 설명 – 대상에 대한 설명을 입력합니다.

    설명은 최대 512자까지 가능합니다.
  - API 대상 엔드포인트 — 대상의 URL 엔드포인트입니다.

    엔드포인트 URL은 **`https`**로 시작해야 합니다. **`*`**를 경로 파라미터 와일드카드로 포함할 수 있습니다. 대상 `HttpParameters` 속성에서 경로 파라미터를 설정할 수 있습니다.
  - HTTP 메서드 - 엔드포인트를 간접 호출할 때 사용할 HTTP 메서드를 선택합니다.
  - (선택 사항) 초당 간접 호출 속도 제한 — 이 대상에 대해 초당 허용되는 최대 간접 호출 수를 입력합니다.

    이 값은 0보다 커야 합니다. 기본적으로 이 값은 300으로 설정됩니다.
  - 연결 — 새 연결을 사용할지 기존 연결을 사용할지 선택합니다.
    - 기존 연결을 사용하려면 기존 연결 사용을 선택하고 드롭다운 목록에서 연결을 선택합니다.
    - 이 대상에 대한 새 연결을 만들려면 새 연결 만들기를 선택한 다음 연결의 이름, 대상 유형 및 권한 부여 유형을 정의합니다. 이 연결에 대한 설명(선택 사항)을 추가할 수도 있습니다.

자세한 내용은 ??? 섹션을 참조하세요.

AWS 서비스

AWS 서비스를 사용하려면 AWS 서비스를 선택하고 다음을 수행합니다.

1. 대상 선택에서 대상으로 사용할 AWS 서비스를 선택합니다. 선택한 서비스에 대해 요청된 정보를 제공합니다.

> **Note**
>
> 표시되는 필드는 선택한 서비스에 따라 달라집니다. 사용 가능한 대상에 대한 자세한 내용은 EventBridge 콘솔에서 대상을 사용할 수 있습니다. 섹션을 참조하세요.

2. 여러 대상 유형에 대해 EventBridge에서는 대상에 이벤트를 보낼 권한이 필요합니다. 이 경우 EventBridge는 규칙 실행에 필요한 IAM 역할을 생성할 수 있습니다.

   실행 역할에서는 다음 중 하나를 수행합니다.

   - 이 규칙의 새 실행 역할을 만들려면:
     a. 이 특정 리소스에 대해 새 역할 생성을 선택합니다.
     b. 이 실행 역할의 이름을 입력하거나 EventBridge에서 생성된 이름을 사용합니다.
   - 이 규칙에 기존 실행 역할을 사용하려면:
     a. 기존 역할 사용을 선택합니다.
     b. 드롭다운 목록에서 사용할 실행 역할의 이름을 입력하거나 선택합니다.

3. (선택 사항) 추가 설정의 경우 대상 유형에 사용할 수 있는 선택적 설정을 지정합니다.

   Event bus

   (선택 사항) DLQ(Dead Letter Queue)에서 표준 Amazon SQS 대기열을 배달 못한 편지 대기열로 사용할지를 선택합니다. 이벤트가 대상에 성공적으로 전달되지 않은 경우 EventBridge는 이 규칙과 일치하는 이벤트를 DLQ(Dead Letter Queue)로 보냅니다. 다음 중 하나를 수행합니다.

   - 배달 못한 편지 대기열을 사용하지 않으려면 없음(None)을 선택합니다.
   - 현재 AWS 계정에서 배달 못한 편지 대기열로 사용할 대기열 선택(Select an Amazon SQS queue in the current account to use as the dead-letter queue)을 선택하고 드롭다운 목록에서 사용할 대기열을 선택합니다.
   - 다른 AWS 계정에서 배달 못한 편지 대기열로 사용할 Amazon SQS 대기열 선택(Select an Amazon SQS queue in an other account as a dead-letter queue)을 선택하고 사용할 대기열의 ARN을 입력합니다. 메시지를 보낼 수 있는 EventBridge 권한을 부여하는 리소스 기반 정책을 대기열에 연결해야 합니다.

     자세한 내용은 DLQ(Dead Letter Queue)에 권한 부여 섹션을 참조하세요.

API destination

1. (선택 사항) 대상 입력 구성의 경우 일치하는 이벤트를 위해 대상으로 전송되는 텍스트를 사용자 지정하는 방법을 선택합니다. 다음 중 하나를 선택합니다.
   - 일치하는 이벤트 — EventBridge는 원본 소스 이벤트 전체를 대상으로 보냅니다. 이 값이 기본값입니다.
   - 일치하는 이벤트의 일부 — EventBridge는 원본 소스 이벤트의 지정된 부분만 대상에 보냅니다.

     일치하는 이벤트의 일부를 지정합니다.에서 EventBridge가 대상으로 전송할 이벤트 부분을 정의하는 JSON 경로를 지정합니다.
   - 상수(JSON 텍스트) - EventBridge는 지정된 JSON 텍스트만 대상으로 전송합니다. 원본 소스 이벤트의 어떤 부분도 전송되지 않습니다.

     JSON으로 상수를 지정에서 EventBridge가 이벤트 대신 대상으로 전송할 JSON 텍스트를 지정합니다.
   - 입력 변환기 - EventBridge가 대상으로 보낼 텍스트를 사용자 지정하도록 입력 변환기를 구성합니다. 자세한 내용은 ??? 섹션을 참조하세요.

     a. 입력 변환기 구성을 선택합니다.

     b. ???의 단계에 따라 입력 변환기를 구성합니다.
2. (선택 사항) 재시도 정책에서 오류 발생 후 EventBridge가 대상으로의 이벤트 전송을 재시도하는 방법을 지정합니다.
   - 최대 이벤트 기간 - EventBridge에서 처리되지 않은 이벤트를 유지할 수 있는 최대 시간(시간, 분, 초)을 입력합니다. 기본값은 24시간입니다.
   - 재시도 - 오류 발생 후 EventBridge가 대상으로 이벤트 전송을 재시도해야 하는 최대 횟수를 입력합니다. 기본값은 185회입니다.
3. (선택 사항) DLQ(Dead Letter Queue)에서 표준 Amazon SQS 대기열을 배달 못한 편지 대기열로 사용할지를 선택합니다. 이벤트가 대상에 성공적으로 전달되지 않은 경우 EventBridge는 이 규칙과 일치하는 이벤트를 DLQ(Dead Letter Queue)로 보냅니다. 다음 중 하나를 수행합니다.
   - 배달 못한 편지 대기열을 사용하지 않으려면 없음(None)을 선택합니다.
   - 현재 AWS 계정에서 배달 못한 편지 대기열로 사용할 대기열 선택(Select an Amazon SQS queue in the current account to use as the dead-letter queue)을 선택하고 드롭다운 목록에서 사용할 대기열을 선택합니다.

- 다른 AWS 계정에서 배달 못한 편지 대기열로 사용할 Amazon SQS 대기열 선택(Select an Amazon SQS queue in an other account as a dead-letter queue)을 선택하고 사용할 대기열의 ARN을 입력합니다. 메시지를 보낼 수 있는 EventBridge 권한을 부여하는 리소스 기반 정책을 대기열에 연결해야 합니다.

  자세한 내용은 DLQ(Dead Letter Queue)에 권한 부여 섹션을 참조하세요.

AWS service

EventBridge는 특정 AWS 서비스에 대해 다음 필드를 모두 표시하지 않을 수 있다는 점에 유의합니다.

1. (선택 사항) 대상 입력 구성의 경우 일치하는 이벤트를 위해 대상으로 전송되는 텍스트를 사용자 지정하는 방법을 선택합니다. 다음 중 하나를 선택합니다.
   - 일치하는 이벤트 — EventBridge는 원본 소스 이벤트 전체를 대상으로 보냅니다. 이 값이 기본값입니다.
   - 일치하는 이벤트의 일부 — EventBridge는 원본 소스 이벤트의 지정된 부분만 대상에 보냅니다.

     일치하는 이벤트의 일부를 지정합니다.에서 EventBridge가 대상으로 전송할 이벤트 부분을 정의하는 JSON 경로를 지정합니다.
   - 상수(JSON 텍스트) - EventBridge는 지정된 JSON 텍스트만 대상으로 전송합니다. 원본 소스 이벤트의 어떤 부분도 전송되지 않습니다.

     JSON으로 상수를 지정에서 EventBridge가 이벤트 대신 대상으로 전송할 JSON 텍스트를 지정합니다.
   - 입력 변환기 - EventBridge가 대상으로 보낼 텍스트를 사용자 지정하도록 입력 변환기를 구성합니다. 자세한 내용은 ??? 섹션을 참조하세요.

     a. 입력 변환기 구성을 선택합니다.

     b. ???의 단계에 따라 입력 변환기를 구성합니다.
2. (선택 사항) 재시도 정책에서 오류 발생 후 EventBridge가 대상으로의 이벤트 전송을 재시도하는 방법을 지정합니다.
   - 최대 이벤트 기간 - EventBridge에서 처리되지 않은 이벤트를 유지할 수 있는 최대 시간(시간, 분, 초)을 입력합니다. 기본값은 24시간입니다.
   - 재시도 - 오류 발생 후 EventBridge가 대상으로 이벤트 전송을 재시도해야 하는 최대 횟수를 입력합니다. 기본값은 185회입니다.

3. (선택 사항) DLQ(Dead Letter Queue)에서 표준 Amazon SQS 대기열을 배달 못한 편지 대기열로 사용할지를 선택합니다. 이벤트가 대상에 성공적으로 전달되지 않은 경우 EventBridge는 이 규칙과 일치하는 이벤트를 DLQ(Dead Letter Queue)로 보냅니다. 다음 중 하나를 수행합니다.
   - 배달 못한 편지 대기열을 사용하지 않으려면 없음(None)을 선택합니다.
   - 현재 AWS 계정에서 배달 못한 편지 대기열로 사용할 대기열 선택(Select an Amazon SQS queue in the current account to use as the dead-letter queue)을 선택하고 드롭다운 목록에서 사용할 대기열을 선택합니다.
   - 다른 AWS 계정에서 배달 못한 편지 대기열로 사용할 Amazon SQS 대기열 선택(Select an Amazon SQS queue in an other account as a dead-letter queue)을 선택하고 사용할 대기열의 ARN을 입력합니다. 메시지를 보낼 수 있는 EventBridge 권한을 부여하는 리소스 기반 정책을 대기열에 연결해야 합니다.

     자세한 내용은 DLQ(Dead Letter Queue)에 권한 부여 섹션을 참조하세요.
4. (선택 사항) 이 규칙에 다른 대상을 추가하려면 다른 대상 추가를 선택합니다.
5. 다음을 선택합니다.

태그 구성 및 규칙 검토

마지막으로 규칙에 원하는 태그를 입력한 다음 규칙을 검토 및 생성합니다.

태그를 구성하고 규칙을 검토 및 생성하려면

1. (선택 사항) 규칙에 대해 하나 이상의 태그를 입력하세요. 자세한 내용은 아마존 EventBridge 태그 섹션을 참조하세요.
2. 다음을 선택합니다.
3. 새 규칙의 세부 정보를 검토합니다. 섹션을 변경하려면 해당 섹션 옆에 있는 편집 버튼을 선택합니다.

   규칙 세부 정보에 만족하면 규칙 생성을 선택합니다.

cron 표현식 참조

cron 표현식에는 각각 공백으로 구분되는 필수 필드 6개가 있습니다.

구문

```
cron(fields)
```

필드	값	와일드카드
분	0~59	, - * /
시간	0~23	, - * /
날짜	1~31	, - * ? / L W
월	1-12 또는 JAN-DEC	, - * /
요일	1-7 또는 SUN-SAT	, - * ? L #
연도	1970~2199	, - * /

와일드카드

- ,(쉼표) 와일드카드는 추가 값을 포함합니다. ‘월’ 필드에서 JAN, FEB, MAR은 1월, 2월, 3월을 포함한다는 의미입니다.
- -(대시) 와일드카드는 범위를 지정합니다. ‘일’ 필드에서 1-15는 지정된 달의 1일에서 15일까지 포함한다는 의미입니다.
- *(별표) 와일드카드는 필드의 모든 값을 포함합니다. ‘시간’ 필드에서 *는 모든 시간을 포함한다는 의미입니다. '날짜' 및 '요일' 필드 모두에서 *를 사용할 수 없습니다. 필드 중 하나에 사용할 경우 다른 하나에는 반드시 ?를 사용해야 합니다.
- /(슬래시) 와일드카드로 증분을 지정합니다. 예를 들어, '분' 필드에 1/10을 입력하면 지정한 시간의 1분부터 시작해서 매 10분 간격을 지정할 수 있습니다(즉, 11분, 21분, 31분 등).
- ?(물음표) 와일드카드는 any를 지정합니다. '날짜' 필드에 7을 입력하고 일주일 중 어느 날이라도 괜찮다면 '요일' 필드에는 ?을 입력합니다.
- '날짜' 또는 '요일' 필드에서 L 와일드카드로 해당 월 또는 주의 마지막 날을 지정할 수 있습니다.
- '날짜' 필드에서는 **W** 와일드카드로 어떤 한 평일을 지정할 수 있습니다. 예를 들어 '날짜' 필드에 **3W**를 입력하면 해당 월의 세 번째 평일에 가장 가까운 날을 지정할 수 있습니다.
- '요일' 필드의 # 와일드카드는 그 달에 속한 정해진 요일의 특정 인스턴스를 지정합니다. 예를 들어, 3#2는 그 달의 두 번째 화요일입니다. 3은 각 주의 셋째 날이므로 화요일을 나타내고 2는 그 달의 두 번째 해당 요일입니다.

> **Note**
>
> '#' 문자를 사용하는 경우 요일(day-of-week) 필드에 하나의 표현식만 정의할 수 있습니다. 예를 들어 "3#1,6#3"은(는) 두 개의 표현식으로 해석되기 때문에 유효하지 않습니다.

제한 사항

- 같은 cron 표현식에서 '날짜' 및 '요일' 필드를 지정할 수 없습니다. 필드 중 하나에 값 또는 *(별표)를 지정하는 경우 다른 필드에는 ?(물음표)를 사용해야 합니다.
- 1분보다 빠른 속도로 이어지는 cron 표현식은 지원되지 않습니다.

예시

예약에 따라 규칙을 생성할 때는 다음과 같이 동일한 cron 문자열을 사용할 수 있습니다.

분	시간	일	월	요일	연도	의미
0	10	*	*	?	*	매일 오전 10시(UTC+0)에 실행
15	12	*	*	?	*	매일 오후 12시 15분(UTC+0)에 실행
0	18	?	*	월-금	*	매주 월요일부터 금요일까지 오후 6시(UTC+0)에 실행
0	8	1	*	?	*	매월 1일 오전 8시

분	시간	일	월	요일	연도	의미
						(UTC+0)에 실행
0/15	*	*	*	?	*	15분마다 실행
0/10	*	?	*	월-금	*	월요일부터 금요일까지 10분마다 실행
0/5	8~17	?	*	월-금	*	월요일부터 금요일까지 오전 8시부터 오후 5시 55분(UTC +0) 사이에 5분마다 실행
0/30	20-2	?	*	월-금	*	시작일 오후 10시부터 다음 날 오전 2시까지 월요일부터 금요일까지 30분마다 실행(UTC) 월요일 오전 12시부터 오전 2시까지 실행합니다(UTC).

다음 예에서는 매일 오후 12시(UTC+0)에 실행되는 규칙을 생성합니다.

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

다음 예에서는 매일 오후 2시 5분과 오후 2시 35분(UTC+0)에 실행되는 규칙을 생성합니다.

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

다음 예에서는 2019년부터 2022년까지 매월 마지막 금요일 오전 10시 15분(UTC+0)에 실행되는 규칙을 생성합니다.

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

rate 표현식 참조

rate 표현식은 예약된 이벤트 규칙을 생성할 때 시작되며, 정의된 일정에 따라 실행됩니다.

rate 표현식에는 각각 공백으로 구분되는 필수 필드 2개가 있습니다.

구문

```
rate(value unit)
```

value

양수.

unit

시간 단위. 1의 값(예: `minute`)과 1을 초과하는 값(예: `minutes`)은 서로 다른 단위가 필요합니다.

유효값: 분 | 분 | 시간 | 시간 | 일 | 일

제한 사항

값이 1과 같을 경우에는 단위가 단수여야 합니다. 값이 1보다 크면 단위는 복수여야 합니다. 예를 들어, rate(1 hours)와 rate(5 hour)는 잘못된 식이며, rate(1 hour)와 rate(5 hours)가 유효한 식입니다.

예시

다음 예제는 AWS CLI `put-rule` 명령에서 rate 표현식을 사용하는 방법을 보여줍니다. 첫 번째 예제는 1분마다 규칙을 트리거하고, 두 번째 예제는 5분마다 규칙을 트리거하며, 다음은 한 시간에 한 번 트리거하고, 마지막 예제는 하루에 한 번 트리거합니다.

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

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

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

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

Amazon EventBridge 규칙 비활성화 또는 삭제

규칙이 이벤트를 처리하거나 일정에 따라 실행되는 것을 중지하려면 규칙을 삭제하거나 비활성화하면 됩니다. 다음 단계는 EventBridge 규칙을 삭제하거나 비활성화하는 방법을 안내합니다.

규칙을 삭제하거나 비활성화하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 규칙을 선택합니다.

   이벤트 버스에서 규칙과 연결된 이벤트 버스를 선택합니다.
3. 다음 중 하나를 수행합니다.

   a. 규칙을 삭제하려면 규칙 옆의 버튼을 선택하고 작업, 삭제, 삭제를 선택합니다.

      규칙이 관리형 규칙인 경우 해당 규칙이 관리형 규칙이고, 규칙을 삭제하면 규칙을 생성한 서비스에서 기능이 멈출 수 있음을 확인하도록 규칙 이름을 입력합니다. 계속하려면 규칙 이름을 입력하고 강제 삭제를 선택합니다.

   b. 규칙을 일시적으로 비활성화하려면 규칙 옆의 버튼을 선택하고 비활성화, 비활성화를 선택합니다.

      관리형 규칙은 비활성화할 수 없습니다.

Amazon EventBridge 규칙 정의 시 모범 사례

다음은 이벤트 버스에 대한 규칙을 생성할 때 고려해야 할 몇 가지 모범 사례입니다.

각 규칙에 단일 대상 설정

특정 규칙에 최대 5개의 대상을 지정할 수 있지만 각 규칙에 대해 단일 대상을 지정하면 규칙을 더 쉽게 관리할 수 있습니다. 둘 이상의 대상이 동일한 이벤트 세트를 수신해야 하는 경우 규칙을 복제하여 동일한 이벤트를 다른 대상에 전달하는 것이 좋습니다. 이러한 캡슐화는 규칙 유지 관리를 간소화합니다. 시간이 지남에 따라 이벤트 대상의 요구 사항이 달라지면 각 규칙과 해당 이벤트 패턴을 다른 규칙과 독립적으로 업데이트할 수 있습니다.

규칙 권한 설정

이벤트를 소비하는 애플리케이션 구성 요소 또는 서비스가 자체 규칙 관리를 제어할 수 있도록 할 수 있습니다. 고객이 채택하는 일반적인 아키텍처 접근 방식은 별도의 AWS 계정을 사용하여 이러한 애플

리케이션 구성 요소 또는 서비스를 격리하는 것입니다. 한 계정에서 다른 계정으로 이벤트가 흐르도록 하려면 이벤트를 다른 계정의 이벤트 버스로 라우팅하는 규칙을 한 이벤트 버스에 생성해야 합니다. 이벤트를 소비하는 팀 또는 서비스가 자체 규칙 관리를 제어할 수 있도록 할 수 있습니다. 리소스 정책을 통해 해당 계정에 적절한 권한을 지정하면 됩니다. 이는 모든 계정과 지역에 적용됩니다.

자세한 내용은 ??? 섹션을 참조하세요.

리소스 정책의 예는 GitHub에서 Multi-account design patterns with Amazon EventBridge를 참조하십시오.

규칙 성능 모니터링

규칙이 예상대로 작동하는지 확인하기 위해 규칙을 모니터링합니다.

- `TriggeredRules` 지표에 누락된 데이터 포인트 또는 이상 항목이 있는지 모니터링하면 주요 변경 사항을 적용한 게시자의 불일치를 감지하는 데 도움이 될 수 있습니다. 자세한 내용은 ??? 섹션을 참조하세요.
- 이상 항목 또는 최대 예상 개수에 대한 경보를 통해 규칙이 새 이벤트와 일치하는지 감지하는 데도 도움이 될 수 있습니다. AWS 서비스 및 SaaS 파트너를 포함한 이벤트 게시자가 새로운 사용 사례 및 기능을 활성화할 때 새로운 이벤트를 도입할 때 발생할 수 있습니다. 이러한 새 이벤트가 예상치 못한 상태에서 다운스트림 대상의 처리 속도보다 높은 볼륨으로 이어지면 이벤트 백로그가 발생할 수 있습니다.

  이런 예기치 않은 이벤트 처리로 인해 원치 않는 청구 요금이 발생할 수도 있습니다.

  또한 계정이 초당 총 대상 간접 호출 서비스 할당량을 초과할 경우 규칙 제한이 트리거될 수 있습니다. EventBridge는 여전히 제한된 규칙과 일치하는 이벤트를 전달하려고 시도하고 최대 24시간 동안 또는 대상의 사용자 지정 재시도 정책에 설명된 대로 재시도합니다. `ThrottledRules` 지표를 사용하여 제한된 규칙을 감지하고 알릴 수 있습니다.
- 지연 시간이 짧은 사용 사례의 경우 이벤트 버스의 상태를 나타내는 `IngestionToInvocationStartLatency`를 사용하여 지연 시간을 모니터링할 수도 있습니다. 긴 지연 시간이 30초 이상 지속되면 서비스 중단이나 규칙 제한이 발생할 수 있습니다.

Amazon EventBridge 및 AWS Serverless Application Model 템플릿 사용

EventBridge 콘솔에서 수동으로 규칙을 작성하고 테스트할 수 있으며, 이는 이벤트 패턴을 구체화할 때 개발 프로세스에 도움이 될 수 있습니다. 하지만 애플리케이션을 배포할 준비가 되면 AWS SAM과 같은 프레임워크를 사용하여 모든 서버리스 리소스를 일관되게 시작하는 것이 더 쉽습니다.

이 예제 애플리케이션을 사용하여 AWS SAM 템플릿을 사용하여 EventBridge 리소스를 구축하는 방법을 살펴보겠습니다. 이 예제의 template.yaml 파일은 네 개의 AWS Lambda 함수를 정의하는 AWS SAM 템플릿으로, Lambda 함수를 EventBridge와 통합하는 두 가지 방법을 보여줍니다.

이 예제 애플리케이션에 대한 자세한 내용은 ??? 섹션을 참조하십시오.

EventBridge 및 AWS SAM 템플릿을 사용하는 접근 방식으로 다음 두 가지가 있습니다. 하나의 규칙에 의해 하나의 Lambda 함수가 간접 호출되는 간단한 통합의 경우 결합된 템플릿 접근 방식을 사용하는 것이 좋습니다. 라우팅 로직이 복잡하거나 AWS SAM 템플릿 외부의 리소스에 연결하는 경우에는 분리된 템플릿 접근 방식이 더 좋습니다.

접근 방식:

- 결합된 템플릿
- 분리된 템플릿

결합된 템플릿

첫 번째 접근 방식은 `Events` 속성을 사용하여 EventBridge 규칙을 구성하는 것입니다. 다음 예제 코드는 Lambda 함수를 간접적으로 간접 호출하는 이벤트를 정의합니다.

> **Note**
>
> 이 예제는 모든 AWS 계정에 있는 기본 이벤트 버스에 규칙을 자동으로 생성합니다. 규칙을 사용자 지정 이벤트 버스에 연결하려면 템플릿에 `EventBusName`을 추가하면 됩니다.

```
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"
```

이 YAML 코드는 EventBridge 콘솔의 이벤트 패턴과 동일합니다. YAML에서는 이벤트 패턴만 정의하면 되며, 그러면 필요한 권한이 있는 IAM 역할이 AWS SAM에서 자동으로 생성됩니다.

분리된 템플릿

AWS SAM에서 EventBridge 구성을 정의하는 두 번째 접근 방식에서는 템플릿에서 리소스가 더 명확하게 구분됩니다.

1. 먼저 Lambda 함수를 정의합니다.

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

2. 다음으로 `AWS::Events::Rule` 리소스를 사용하여 규칙을 정의합니다. 속성은 이벤트 패턴을 정의하고 대상을 지정할 수도 있습니다. 여러 대상을 명시적으로 정의할 수 있습니다.

   ```
   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. 마지막으로, 대상을 간접적으로 간접 호출할 수 있는 권한을 EventBridge에 부여하는 `AWS::Lambda::Permission` 리소스를 정의합니다.

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

Amazon EventBridge 규칙에서 AWS CloudFormation 템플릿 생성

AWS CloudFormation을 사용하면 인프라를 코드로 취급하여 중앙 집중화되고 반복 가능한 방식으로 계정 및 리전 전반의 AWS 리소스를 구성하고 관리할 수 있습니다. CloudFormation은 프로비저닝 및 관리하려는 리소스를 정의하는 템플릿을 생성하게 하여 이 작업을 수행합니다.

EventBridge를 사용하면 계정의 기존 규칙에서 템플릿을 생성할 수 있으므로 CloudFormation 템플릿 개발을 바로 시작할 수 있습니다. 템플릿에 포함할 규칙을 하나 또는 여러 개 선택할 수 있습니다. 따라서 이러한 템플릿을 기반으로 사용해 CloudFormation 관리에 따른 리소스 스택을 생성할 수 있습니다.

CloudFormation에 대한 자세한 내용은 AWS CloudFormation 사용 설명서를 참조하십시오.

> **Note**
>
> EventBridge는 생성된 템플릿에 관리형 규칙을 포함하지 않습니다.

이벤트 버스에 포함된 규칙을 포함하여 기존 이벤트 버스에서 템플릿을 생성할 수도 있습니다.

하나 이상의 규칙에서 AWS CloudFormation 템플릿 생성

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 규칙을 선택합니다.
3. 이벤트 버스 선택에서 템플릿에 포함할 규칙이 포함된 이벤트 버스를 선택합니다.
4. 규칙에서 생성된 AWS CloudFormation 템플릿에 포함할 규칙을 선택합니다.

   단일 규칙의 경우 규칙 이름을 선택하여 규칙의 세부 정보 페이지를 표시할 수도 있습니다.
5. CloudFormation 템플릿을 선택한 다음, EventBridge에서 템플릿을 생성할 때 사용할 형식(JSON 또는 YAML)을 선택합니다.

   EventBridge는 선택한 형식으로 생성된 템플릿을 표시합니다.
6. EventBridge는 템플릿 파일을 다운로드하거나 템플릿을 클립보드에 복사하는 옵션을 제공합니다.
   - 템플릿 파일을 다운로드하려면 다운로드를 선택합니다.
   - 템플릿을 클립보드에 복사하려면 복사를 선택합니다.
7. 템플릿을 종료하려면 취소를 선택합니다.

사용 사례에 맞게 AWS CloudFormation 템플릿을 사용자 지정한 후에는 이를 사용하여 AWS CloudFormation에 스택을 생성할 수 있습니다.

Amazon EventBridge에서 생성된 CloudFormation 템플릿 사용 시 고려할 사항

EventBridge에서 생성한 CloudFormation 템플릿을 사용할 때는 다음 요소를 고려합니다.

- EventBridge는 생성 템플릿에 암호를 포함하지 않습니다.

  템플릿을 사용하여 CloudFormation 스택을 생성하거나 업데이트할 때 사용자가 암호 또는 기타 중요한 정보를 지정할 수 있는 템플릿 파라미터를 포함하도록 템플릿을 편집할 수 있습니다.

또한 사용자는 Secrets Manager를 이용해 원하는 리전에 보안 암호를 생성한 다음, 생성된 템플릿을 편집하여 동적 파라미터를 적용할 수 있습니다.

- 생성된 템플릿의 대상은 원래 이벤트 버스에 지정된 그대로 유지됩니다. 따라서 템플릿을 사용하여 다른 리전에 스택을 생성하기 전에 템플릿을 적절하게 편집하지 않으면 교차 리전 문제가 발생할 수 있습니다.

  또한 생성된 템플릿은 다운스트림 대상을 자동으로 생성하지 않습니다.

아마존 EventBridge 타겟

대상은 이벤트가 규칙에 정의된 이벤트 패턴과 일치할 때 이벤트를 EventBridge 보내는 리소스 또는 엔드포인트입니다. 규칙은 이벤트 데이터를 처리하고 관련 정보를 대상으로 보냅니다. 이벤트 데이터를 대상에 전달하려면 대상 리소스에 액세스할 수 있는 권한이 EventBridge 필요합니다. 각 규칙에 대해 최대 5개의 대상을 정의할 수 있습니다.

규칙에 대상을 추가하고 해당 규칙이 바로 실행되는 경우 새 대상이나 업데이트된 대상이 즉시 간접 호출되지 않을 수 있습니다. 변경 사항이 적용될 때까지 잠시 기다리십시오.

다음 동영상에서는 대상의 기본 사항을 다룹니다. 대상 정의

EventBridge 콘솔에서 대상을 사용할 수 있습니다.

EventBridge 콘솔에서 다음과 같은 이벤트 대상을 구성할 수 있습니다.

- API 대상
- API Gateway
- AWS AppSync;
- 배치 작업 대기열
- CloudWatch 로그 그룹
- CodeBuild 프로젝트
- CodePipeline
- 아마존 EBS `CreateSnapshot` API 직접 호출
- EC2 Image Builder
- EC2 `RebootInstances` API 직접 호출
- EC2 `StopInstances` API 직접 호출
- EC2 `TerminateInstances` API 직접 호출
- ECS 태스크
- 다른 계정 또는 리전의 이벤트 버스
- 동일한 계정 및 리전의 이벤트 버스
- Firehose 전송 스트림

- Glue 워크플로
- Incident Manager 대응 계획
- Inspector 평가 템플릿
- Kinesis 스트림
- Lambda 함수(ASYNC)
- Amazon Redshift 클러스터 데이터 API 쿼리
- Amazon Redshift Serverless 작업 그룹 데이터 API 쿼리
- SageMaker 파이프라인
- Amazon SNS 주제

  EventBridge Amazon SNS FIFO (선입 선출) 주제를 지원하지 않습니다.
- Amazon SQS 대기열
- Step Functions 상태 시스템(ASYNC)
- Systems Manager Automation
- Systems Manager OpsItem
- Systems Manager Run Command

대상 파라미터

일부 대상은 이벤트 페이로드의 정보를 대상으로 보내지 않고 대신 이벤트를 특정 API를 호출하는 트리거로 취급합니다. EventBridge Target 매개변수를 사용하여 해당 대상에 어떤 일이 발생하는지 결정합니다. 여기에는 다음이 포함됩니다.

- API 대상(API 대상으로 전송되는 데이터는 API의 구조와 일치해야 합니다. `InputTransformer` 객체를 사용하여 데이터가 올바르게 구조화되었는지 확인해야 합니다. 원본 이벤트 페이로드를 포함하려면 `InputTransformer`에서 해당 페이로드를 참조하세요.)
- API Gateway(API Gateway로 전송되는 데이터는 API의 구조와 일치해야 합니다. `InputTransformer` 객체를 사용하여 데이터가 올바르게 구조화되었는지 확인해야 합니다. 원본 이벤트 페이로드를 포함하려면 `InputTransformer`에서 해당 페이로드를 참조하세요.)
- Amazon EC2 Image Builder
- `RedshiftDataParameters`(Amazon Redshift Data API 클러스터)
- `SageMakerPipelineParameters`(Amazon SageMaker 런타임 모델 구축 파이프라인)

> **Note**
>
> EventBridge 모든 JSON Path 구문을 지원하지는 않으며 런타임에 이를 평가합니다. 지원되는 구문은 다음과 같습니다.
>
> - 점 표기법(예: `$.detail`)
> - 대시
> - 밑줄
> - 영숫자
> - 배열 인덱스
> - 와일드카드(*)

동적 경로 파라미터

일부 대상 파라미터는 선택적인 동적 JSON 경로 구문을 지원합니다. 이 구문을 사용하면 정적 값 대신 JSON 경로를 지정할 수 있습니다(예: `$.detail.state`). 전체 값은 일부가 아닌 JSON 경로여야 합니다. 예를 들어 `RedshiftParameters.Sql`은 `$.detail.state`일 수 있지만 `"SELECT * FROM $.detail.state"`일 수는 없습니다. 이러한 경로는 런타임 시 지정된 경로에 있는 이벤트 페이로드 자체의 데이터로 동적으로 대체됩니다. 동적 경로 파라미터는 입력 변환으로 생성된 새 값이나 변환된 값을 참조할 수 없습니다. 동적 파라미터 JSON 경로에 지원되는 구문은 입력을 변환할 때와 동일합니다. 자세한 내용은 ??? 섹션을 참조하세요.

다음 파라미터의 열거형이 아닌 모든 문자열 필드에 동적 구문을 사용할 수 있습니다.

- `EcsParameters`
- `HttpParameters`(`HeaderParameters` 키 제외)
- `RedshiftDataParameters`
- `SageMakerPipelineParameters`

권한

소유한 리소스에서 API를 호출하려면 적절한 권한이 EventBridge 필요합니다. AWS Lambda 및 Amazon SNS 리소스의 경우, 리소스 기반 정책을 EventBridge 사용합니다. EC2 인스턴스, Kinesis 데이터 스트림 및 Step Functions 상태 머신의 EventBridge 경우 파라미터에 지정한 IAM 역할을 사용합

니다. RoleARN PutTargets 구성된 IAM 권한 부여로 API Gateway 엔드포인트를 간접 호출할 수 있지만 권한 부여를 구성하지 않은 경우 역할은 선택 사항입니다. 자세한 정보는 아마존 EventBridge 및 AWS Identity and Access Management을 참조하세요.

다른 계정이 동일한 리전에 있고 권한을 부여한 경우 해당 계정으로 이벤트를 보낼 수 있습니다. 자세한 정보는 AWS 계정 간 Amazon EventBridge 이벤트 전송 및 수신을 참조하세요.

대상이 암호화된 경우 KMS 키 정책에 다음 섹션을 포함해야 합니다.

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

EventBridge 대상 세부 사항

AWS Batch 작업 대기열

를 통해 특정 매개변수를 구성할 AWS Batch submitJob 수 있습니다. BatchParameters

이벤트 페이로드에서 기타 항목을 지정할 수 있습니다. 이벤트 페이로드 (통과 또는 통과 InputTransformers) 에 다음 키가 포함된 경우 해당 키는 submitJob 요청 파라미터에 매핑됩니다.

- ContainerOverrides: containerOverrides

> **Note**
> 여기에는 명령, 환경, 메모리 및 vcpu만 포함됩니다.

- DependsOn: dependsOn

> **Note**
>
> 여기에는 jobId만 포함됩니다.

- `Parameters: parameters`

CloudWatch 로그 그룹

CloudWatch Logs InputTransformer대상과 함께 사용하지 않는 경우 이벤트 페이로드가 로그 메시지로 사용되고 이벤트 소스가 타임스탬프로 사용됩니다. 를 사용하는 경우 템플릿은 InputTransformer 다음과 같아야 합니다.

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

EventBridge 로그 스트림으로 전송된 항목을 일괄 처리하므로 트래픽에 따라 로그 스트림에 단일 또는 여러 이벤트를 전달할 EventBridge 수 있습니다.

CodeBuild 프로젝트

를 사용하여 InputTransformers입력 이벤트를 Target에 맞게 CodeBuild StartBuildRequest구성하면 매개변수가 일대일로 매핑되어 전달됩니다. `codeBuild.StartBuild`

Amazon ECS 태스크

를 사용하여 InputTransformersAmazon ECS RunTask TaskOverride구조와 일치하도록 Target에 대한 입력 이벤트를 구성하면 파라미터가 일대일로 매핑되어 전달됩니다. `ecs.RunTask`

Incident Manager 대응 계획

일치하는 이벤트가 경보에서 CloudWatch 발생한 경우 경보 상태 변경 세부 정보가 인시던트 관리자 호출의 트리거 세부 정보에 채워집니다. StartIncidentRequest

대상 구성

대상에 대한 EventBridge 설정을 구성하는 방법을 알아보십시오.

대상:

- API 대상
- 아마존 API Gateway용 아마존 EventBridge 타겟
- AWS AppSync 아마존 대상 EventBridge
- HTTP 엔드포인트 타겟을 위한 연결
- AWS 계정 간 Amazon EventBridge 이벤트 전송 및 수신
- AWS 지역 간 Amazon EventBridge 이벤트 전송 및 수신
- 동일한 계정 및 지역의 이벤트 버스 간에 Amazon EventBridge 이벤트 전송 및 수신

API 대상

Amazon EventBridge API 대상은 AWS 서비스 또는 리소스를 대상으로 호출하는 방법과 마찬가지로 규칙의 대상으로 호출할 수 있는 HTTP 엔드포인트입니다. API 대상을 사용하면 API 호출을 사용하여 AWS 서비스, 통합 서비스형 소프트웨어 (SaaS) 애플리케이션 및 외부 애플리케이션 간에 이벤트를 라우팅할 수 있습니다. AWS API 대상을 규칙의 대상으로 지정하면 규칙에 지정된 이벤트 패턴과 일치하는 모든 이벤트에 대해 HTTP 엔드포인트를 EventBridge 호출한 다음 요청과 함께 이벤트 정보를 전달합니다. EventBridge에서는 CONNECT 및 TRACE를 제외한 모든 HTTP 메서드를 요청에 사용할 수 있습니다. 가장 일반적으로 사용되는 HTTP 메서드는 PUT 및 POST입니다. 입력 변환기를 사용하여 이벤트를 특정 HTTP 엔드포인트 파라미터의 파라미터로 사용자 지정할 수도 있습니다. 자세한 정보는 아마존 EventBridge 인풋 트랜스포메이션을 참조하세요.

> **Note**
>
> API 대상은 인터페이스 VPC 엔드포인트와 같은 프라이빗 대상을 지원하지 않습니다. 여기에는 프라이빗 네트워크 및 Application Load Balancer 및 인터페이스 VPC 엔드포인트를 사용하는 가상 사설 클라우드 (VPC) 의 프라이빗 HTTPS API가 포함됩니다.
> 자세한 정보는 ???을 참조하세요.

> **Important**
>
> EventBridge API 대상 엔드포인트에 대한 요청의 최대 클라이언트 실행 제한 시간은 5초여야 합니다. 대상 엔드포인트가 응답하는 데 5초 이상 걸리는 경우 요청 제한 EventBridge 시간이 초과됩니다. EventBridge 제한 시간이 초과된 요청을 재시도 정책에 구성된 최대값까지 재시도합니다. 기본적으로 최댓값은 24시간 185회입니다. 최대 재시도 횟수를 초과하면 DLQ(Dead Letter Queue)가 있는 경우 이벤트가 DLQ(Dead Letter Queue)로 전송됩니다. 그렇지 않으면 이벤트가 삭제됩니다.

다음 동영상은 API 대상 사용을 보여줍니다. API 대상 사용

이 주제에서 수행할 작업

- API 대상 생성
- 이벤트를 API 대상으로 보내는 규칙 생성

- API 대상을 위한 서비스 연결 역할
- API 대상에 대한 요청 내 헤더
- API 대상 오류 코드
- 간접 호출 속도가 이벤트 전송에 미치는 영향
- API CloudEvents 대상으로 이벤트 전송
- API 대상 파트너

API 대상 생성

각 API 대상에는 연결이 필요합니다. 연결은 API 대상 엔드포인트에 권한을 부여하는 데 사용할 권한 부여 유형 및 보안 인증 정보를 지정합니다. 기존 연결을 선택하거나 API 대상을 생성할 때 동시에 연결을 생성할 수 있습니다. 자세한 내용은 ??? 섹션을 참조하세요.

콘솔을 사용하여 API 대상을 만들려면 EventBridge

1. 관리 권한이 있는 계정을 AWS 사용하여 EventBridge 로그인하고 EventBridge콘솔을 엽니다.
2. 왼쪽 탐색 창에서 API 대상을 선택합니다.
3. API 대상 테이블까지 아래로 스크롤한 다음, API 대상 생성을 선택합니다.
4. API 대상 생성 페이지에서 API 대상의 이름을 입력합니다. 최대 64개의 대문자 또는 소문자, 숫자, 점(.), 대시(-) 또는 밑줄(_) 문자를 사용할 수 있습니다.

   이 이름은 현재 리전의 계정에 대해 고유해야 합니다.
5. API 대상에 대한 설명을 입력합니다.
6. API 대상에 대한 API 대상 엔드포인트를 입력합니다. API 대상 엔드포인트는 이벤트의 HTTP 간접 호출 엔드포인트 대상입니다. 이 API 대상에 사용되는 연결에 포함한 권한 부여 정보는 이 엔드포인트에 대해 권한을 부여하는 데 사용됩니다. URL은 HTTPS를 사용해야 합니다.
7. API 대상 엔드포인트에 연결하는 데 사용할 HTTP 메서드를 입력합니다.
8. (선택 사항) 초당 간접 호출 속도 제한 필드에 API 대상 엔드포인트로 전송할 초당 최대 간접 호출 수를 입력합니다.

   설정한 속도 제한은 이벤트 전송 방식에 EventBridge 영향을 미칠 수 있습니다. 자세한 정보는 간접 호출 속도가 이벤트 전송에 미치는 영향을 참조하세요.
9. 연결에서 다음 중 하나를 수행합니다.
   - 기존 연결 사용을 선택한 다음, 이 API 대상에 사용할 연결을 선택합니다.

- 새 연결 생성을 선택하고 생성할 연결의 세부 정보를 입력합니다. 자세한 내용은 연결을 참조하세요.

10. Create를 선택합니다.

이벤트를 API 대상으로 보내는 규칙 생성

API 대상을 생성한 후 규칙의 대상으로 선택할 수 있습니다. API 대상을 특정 대상으로 사용하려면 올바른 권한이 있는 IAM 역할을 제공해야 합니다. 자세한 내용은 ??? 단원을 참조하십시오.

API 대상을 특정 대상으로 선택하는 것은 규칙 생성의 일부입니다.

콘솔을 사용하여 API 대상으로 이벤트를 보내는 규칙을 생성하려면

1. ??? 절차에서 해당 단계를 따릅니다.
2. ???단계에서 대상 유형으로 API 대상을 선택하라는 메시지가 표시되면:

   a. EventBridge API 대상을 선택합니다.
   b. 다음 중 하나를 수행하십시오.

      - 기존 API 대상 사용을 선택하고 기존 API 대상을 선택합니다.
      - 새 API 대상 생성을 선택하고 새 API 대상을 정의하는 데 필요한 설정을 지정합니다.

        필수 설정 지정에 대한 자세한 내용은 을 참조하십시오???.

   c. (선택 사항): 이벤트의 헤더 파라미터를 지정하려면 헤더 파라미터에서 헤더 파라미터 추가를 선택합니다.

      그런 다음 헤더 파라미터의 키와 값을 지정합니다.

   d. (선택 사항): 이벤트의 쿼리 문자열 파라미터를 지정하려면 쿼리 문자열 파라미터에서 쿼리 문자열 파라미터 추가를 선택합니다.

      그런 다음 쿼리 문자열 파라미터의 키와 값을 지정합니다.

3. 절차 단계에 따라 규칙 생성을 완료합니다.

API 대상을 위한 서비스 연결 역할

API 대상에 대한 연결을 생성하면 이름이 지정된 서비스 연결 역할이 AWS ServiceRoleForAmazonEventBridgeApiDestinations계정에 추가됩니다. EventBridge 서비스 연결 역할을 사용하여 Secrets Manager에서 시크릿을 생성하고 저장합니다. 서

비스 연결 역할에 필요한 권한을 부여하려면 정책을 역할에 EventBridge 연결합니다. AmazonEventBridgeApiDestinationsServiceRolePolicy 이 정책은 역할이 연결 보안 암호와 상호 작용하는 데 필요한 권한으로만 부여되는 권한을 제한합니다. 다른 권한은 포함되지 않으며, 역할은 보안 암호를 관리하도록 계정의 연결과만 상호 작용할 수 있습니다.

다음 정책은 AmazonEventBridgeApiDestinationsServiceRolePolicy입니다.

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

서비스 연결 역할에 대한 자세한 내용은 IAM 설명서의 서비스 연결 역할 사용을 참조하세요.

AmazonEventBridgeApiDestinationsServiceRolePolicy서비스 연결 역할은 다음 지역에서 지원됩니다. AWS

- 미국 동부(버지니아 북부)
- 미국 동부(오하이오)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오레곤)
- 아프리카(케이프타운)
- 아시아 태평양(홍콩)
- 아시아 태평양(뭄바이)
- 아시아 태평양(오사카)
- 아시아 태평양(서울)

- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 캐나다(중부)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(밀라노)
- 유럽(파리)
- 유럽(스톡홀름)
- 남아메리카(상파울루)
- 중국(닝샤)
- 중국(베이징)

API 대상에 대한 요청 내 헤더

다음 섹션에서는 API 대상에 대한 요청의 HTTP 헤더를 EventBridge 처리하는 방법을 자세히 설명합니다.

API 대상에 대한 요청에 포함된 헤더

API 대상에 사용되는 연결에 정의된 인증 헤더 외에도 각 EventBridge 요청에는 다음 헤더가 포함됩니다.

헤더 키	헤더 값
사용자 에이전트	아마존//EventBridgeApiDestinations
Content-Type	사용자 지정 콘텐츠 유형 값이 지정되지 않은 경우 다음 기본값을 콘텐츠 유형으로 EventBridge 포함합니다. application/json; charset=utf-8
Range	bytes=0-1048575

헤더 키	헤더 값
Accept-Encoding	gzip,deflate
연결	close
Content-Length	수신자에게 보낸 개체 본문의 크기(바이트)를 나타내는 개체 헤더입니다.
Host	요청을 보내는 서버의 호스트 및 포트 번호를 지정하는 요청 헤더입니다.

API 대상에 대한 요청에서 재정의할 수 없는 헤더

EventBridge 다음 헤더를 재정의할 수 없습니다.

- 사용자 에이전트
- Range

헤더는 API 대상에 대한 요청에서 EventBridge 제거됩니다.

EventBridge 모든 API 대상 요청에서 다음 헤더를 제거합니다.

- A-IM
- Accept-Charset
- Accept-Datetime
- Accept-Encoding
- Cache-Control
- 연결
- Content-Encoding
- Content-Length
- Content-MD5
- 날짜
- Expect
- 전달됨

- From
- Host
- HTTP2-Settings
- If-Match
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Max-Forwards
- 오리진(Origin)
- Pragma
- Proxy-Authorization
- Range
- 참조자
- TE
- 트레일러
- Transfer-Encoding
- 사용자 에이전트
- 업그레이드
- Via
- 경고

API 대상 오류 코드

이벤트를 API 대상으로 EventBridge 전달하려고 시도할 때 오류가 발생하면 EventBridge 다음을 수행합니다.

- 오류 코드 409, 429 및 5xx와 관련된 이벤트가 재시도됩니다.
- 오류 코드 1xx, 2xx, 3xx, 4xx(429 제외)와 관련된 이벤트는 재시도되지 않습니다.

EventBridge API 대상은 표준 HTTP 응답 헤더를 `Retry-After` 읽고 후속 요청을 하기 전에 기다려야 하는 시간을 확인합니다. EventBridge 정의된 재시도 정책과 헤더 중에서 보다 보수적인 값을 선택합

니다. `Retry-After` `Retry-After`값이 음수인 경우 해당 이벤트에 대한 전송 재시도를 EventBridge 중지합니다.

간접 호출 속도가 이벤트 전송에 미치는 영향

초당 간접 호출 속도를 생성된 간접 호출 수보다 훨씬 낮은 값으로 설정하면 이벤트에 대한 24시간 재시도 기간 내에 이벤트가 전달되지 않을 수 있습니다. 예를 들어 간접 호출 속도를 초당 10회 간접 호출로 설정했지만 초당 수천 개의 이벤트가 생성되는 경우, 24시간을 초과하는 전달 이벤트의 백로그가 빠르게 발생할 수 있습니다. 이벤트가 손실되지 않도록 하려면 나중에 이벤트를 처리할 수 있도록 간접 호출이 실패한 이벤트를 보낼 DLQ(Dead Letter Queue)를 설정하세요. 자세한 정보는 데드레터 대기열을 사용하여 전달되지 않은 이벤트를 처리합니다.을 참조하세요.

API CloudEvents 대상으로 이벤트 전송

CloudEvents 는 벤더 중립적인 이벤트 형식 지정 사양으로, 서비스, 플랫폼, 시스템 전반에 걸쳐 상호 운용성을 제공하는 것을 목표로 합니다. 를 EventBridge 사용하여 AWS 서비스 이벤트를 API 대상과 같은 대상으로 CloudEvents 전송되기 전으로 변환할 수 있습니다.

> **Note**
>
> 다음 절차는 소스 이벤트를 CloudEvents구조화 모드로 변환하는 방법을 설명합니다. CloudEvents 사양에서 구조적 모드 메시지는 전체 이벤트 (속성 및 데이터) 가 이벤트의 페이로드에 인코딩되는 메시지입니다.

사양에 대한 자세한 내용은 cloudevents.io를 CloudEvents 참조하십시오.

콘솔을 사용하여 AWS 이벤트를 해당 형식으로 변환하려면 CloudEvents

이벤트를 대상에 전달하기 전에 CloudEvents 형식으로 변환하려면 먼저 이벤트 버스 규칙을 만들어야 합니다. 규칙 정의의 일환으로 입력 변환기를 사용하여 지정한 대상으로 보내기 전에 EventBridge 변환 이벤트를 생성하십시오.

1. ??? 절차에서 해당 단계를 따릅니다.
2. ???단계에서 API 대상을 대상으로 선택하라는 메시지가 표시되면 다음을 입력합니다.
   a. EventBridge API 대상을 선택합니다.
   b. 다음 중 하나를 수행하십시오.
      - 기존 API 대상 사용을 선택하고 기존 API 대상을 선택합니다.

- 새 API 대상 생성을 선택하고 새 API 대상을 정의하는 데 필요한 설정을 지정합니다.

  필수 설정 지정에 대한 자세한 내용은 을 참조하십시오???.

c. 이벤트에 필요한 Content-Type 헤더 파라미터를 지정하십시오. CloudEvents

- 헤더 파라미터에서 헤더 파라미터 추가를 선택합니다.
- 키에 대해 지정합니다`Content-Type`.

  값으로 지정하십시오`application/cloudevents+json; charset=UTF-8`.

3. 대상의 실행 역할을 지정하십시오.
4. 소스 이벤트 데이터를 다음과 같은 CloudEvents 형식으로 변환하는 입력 변환기를 정의합니다.

a. 추가 설정에서 대상 입력 구성에서 입력 변환기를 선택합니다.

그런 다음 입력 변압기 구성을 선택합니다.

b. 대상 입력 트랜스포머에서 입력 경로를 지정합니다.

아래 입력 경로에서 region 속성은 해당 CloudEvents 형식의 사용자 지정 확장 속성입니다. 따라서 CloudEvents 사양을 준수하는 데 반드시 필요한 것은 아닙니다.

CloudEvents 핵심 사양에 정의되지 않은 확장 속성을 사용하고 생성할 수 있습니다. 알려진 확장 속성 목록을 비롯한 자세한 내용은 에 대한 CloudEvents 사양 설명서의 CloudEvents 확장 속성을 참조하십시오 GitHub.

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

c. 템플릿의 경우 템플릿을 입력하여 소스 이벤트 데이터를 해당 CloudEvents 형식으로 변환합니다.

아래 템플릿에서는 입력 경로의 `region` 속성이 CloudEvents 사양의 확장 속성이므로 반드시 필요한 `region` 것은 아닙니다.

```
{
```

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

5. 절차 단계에 따라 규칙 생성을 완료하십시오.

API 대상 파트너

다음 AWS 파트너가 제공한 정보를 사용하여 해당 서비스 또는 애플리케이션에 대한 API 대상 및 연결을 구성하십시오.

Cisco 클라우드 옵저버빌리티

API 대상 간접 호출 엔드포인트 URL:

`https://tenantName.observe.appdynamics.com/rest/awsevents/aws-eventbridge-integration/endpoint`

지원되는 권한 부여 유형:

OAuth 클라이언트 자격 증명

401 또는 407 응답이 반환되면 OAuth 토큰이 새로 고쳐집니다.

필요한 추가 권한 부여 파라미터:

Cisco 클라이언트 ID AppDynamics 및 클라이언트 시크릿

OAuth 엔드포인트:

`https://tenantName.observe.appdynamics.com/auth/tenantId/default/oauth2/token`

다음 OAuth 키/값 쌍 매개변수:

유형	키	값
바디 필드	grant_type	client_credentials

유형	키	값
헤더	Content-Type	애플리케이션/x-www-form-urlencoded; 문자셋=utf-8

AppDynamics 시스코 설명서:

AWS 이벤트 수집

일반적으로 사용되는 API 작업:

해당 사항 없음

추가 정보:

파트너 대상 드롭다운 AppDynamics 메뉴에서 Cisco를 선택하면 API 호출에 필요한 헤더 및 본문 키/값 쌍을 포함하여 필요한 OAuth 정보가 미리 채워집니다.

자세한 내용은 Cisco 설명서의 AWS 이벤트 통합을 참조하십시오. AppDynamics

컨플루언트

API 대상 간접 호출 엔드포인트 URL:

일반적으로 다음과 같은 형식입니다.

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

자세한 내용은 Confluent 설명서에서 REST 엔드포인트 주소 및 클러스터 ID 찾기를 참조하십시오.

지원되는 권한 부여 유형:

기본

필요한 추가 권한 부여 파라미터:

해당 사항 없음

컨플루언트 설명서:

프로듀스 레코드

아파치 카프카용 컨플루언트 REST 프록시

일반적으로 사용되는 API 작업:

POST

추가 정보:

이벤트 데이터를 엔드포인트가 처리할 수 있는 메시지로 변환하려면 대상 입력 변환기를 만드십시오.

- Kafka 파티션 키를 지정하지 않고 레코드를 생성하려면 입력 변환기에 다음 템플릿을 사용하세요. 입력 경로는 필요하지 않습니다.

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

- 이벤트 데이터 필드를 Kafka 파티션 키로 사용하여 레코드를 생성하려면 아래 입력 경로 및 템플릿 예제를 따르십시오. 이 예제에서는 `orderId` 필드의 입력 경로를 정의한 다음 해당 필드를 파티션 키로 지정합니다.

  먼저 이벤트 데이터 필드의 입력 경로를 정의합니다.

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

  그런 다음 입력 변환기 템플릿을 사용하여 데이터 필드를 파티션 키로 지정합니다.

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

Coralogix

API 대상 간접 호출 엔드포인트 URL

전체 엔드포인트 목록은 Coralogix API 참조를 참조하세요.

지원되는 권한 부여 유형

API 키

필요한 추가 권한 부여 파라미터

헤더 "x-amz-event-bridge-access-key", 값은 Coralogix API 키입니다.

Coralogix 설명서

아마존 EventBridge 인증

일반적으로 사용되는 API 작업

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

싱가포르: https://ingress.coralogixsg.com/aws/event-bridge

아일랜드: https://ingress.coralogix.com/aws/event-bridge

스톡홀름: https://ingress.eu2.coralogix.com/aws/event-bridge

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

추가 정보

이벤트는 `applicationName=[AWS Account]` 및 `subsystemName=[event.source]`를 포함한 로그 항목으로 저장됩니다.

Datadog

API 대상 간접 호출 엔드포인트 URL

전체 엔드포인트 목록은 Datadog API 참조를 참조하세요.

지원되는 권한 부여 유형

API 키

필요한 추가 권한 부여 파라미터

None

Datadog 설명서

인증

일반적으로 사용되는 API 작업

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

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

추가 정보

엔드포인트 URL은 Datadog 조직의 위치에 따라 다릅니다. 조직의 올바른 URL은 설명서를 참조하세요.

Freshworks

API 대상 간접 호출 엔드포인트 URL

엔드포인트 목록은 https://developers.freshworks.com/documentation/ 페이지를 참조하세요.

지원되는 권한 부여 유형

기본, API 키

필요한 추가 권한 부여 파라미터

해당 사항 없음

Freshworks 설명서

인증

일반적으로 사용되는 API 작업

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

추가 정보

없음

MongoDB

API 대상 간접 호출 엔드포인트 URL

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

지원되는 권한 부여 유형

API 키

이메일/암호

사용자 지정 JWT 인증

필요한 추가 권한 부여 파라미터

None

MongoDB 설명서

Atlas 데이터 API

엔드포인트

사용자 지정 HTTPS 엔드포인트

인증

일반적으로 사용되는 API 작업

None

추가 정보

없음

New Relic

API 대상 간접 호출 엔드포인트 URL

자세한 내용은 Our EU and US region data centers를 참조하세요.

이벤트

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

EU – https://insights-collector.eu01.nr-data.net/v1/accounts/*YOUR_NEW_RELIC_ACCOUNT_ID*/events

지표

US – https://metric-api.newrelic.com/metric/v1

EU – https://metric-api.eu.newrelic.com/metric/v1

로그

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

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

트레이스

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

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

지원되는 권한 부여 유형

API 키

New Relic 설명서

지표 API

이벤트 API

로그 API

트레이스 API

일반적으로 사용되는 API 작업

지표 API

이벤트 API

로그 API

트레이스 API

추가 정보

지표 API 제한

이벤트 API 제한

로그 API 제한

트레이스 API 제한

Operata

API 대상 간접 호출 엔드포인트 URL:

https://api.operata.io/v2/aws/events/contact-record

지원되는 권한 부여 유형:

기본

필요한 추가 권한 부여 파라미터:

None

Operata 설명서:

How do I create, view, change and revoke API Tokens?

Amazon EventBridge 스케줄러 AWS 파이프를 사용한 Operata 통합

일반적으로 사용되는 API 작업:

`POST https://api.operata.io/v2/aws/events/contact-record`

추가 정보:

`username`은 Operata 그룹 ID이고 암호는 API 토큰입니다.

Salesforce

API 대상 간접 호출 엔드포인트 URL

제목 — myDomainName https://.my.salesforce.com/services/data/ 버전번호 /sobjects*//* *SobjectEndpoint*

커스텀 플랫폼 이벤트 — myDomainName *https://.my.salesforce.com/services/data/#### /sobjects/ /* customPlatformEndpoint*

전체 엔드포인트 목록은 Salesforce API 참조를 참조하세요.

지원되는 권한 부여 유형

OAuth 클라이언트 자격 증명

401 또는 407 응답이 반환되면 OAUTH 토큰이 새로 고쳐집니다.

필요한 추가 권한 부여 파라미터

Salesforce 연결된 앱 클라이언트 ID 및 클라이언트 암호입니다.

다음 권한 부여 엔드포인트 중 하나입니다.

- 프로덕션 — *MyDomainName*https://.my.salesforce.com. /services/oauth2/token
- 확장 도메인이 없는 샌드박스— https://—.my. salesforce.com/services /oauth2/token *MyDomainName SandboxName*
- 향상된 도메인이 포함된 샌드박스 — ***MyDomainName SandboxName***https:// —.sandbox.my.salesforce.com/services/oauth2/token

다음 키/값 쌍은 다음과 같습니다.

Key(키)	값
grant_type	client_credentials

Salesforce 설명서

REST API 개발자 안내서

일반적으로 사용되는 API 작업

객체 메타데이터 작업

레코드 작업

추가 정보

EventBridge 콘솔을 사용하여 연결Salesforce, API 대상 및 정보를 라우팅하는 규칙을 생성하는 방법을 설명하는 자습서는 을 참조하십시오. Salesforce ???

Slack

API 대상 간접 호출 엔드포인트 URL

엔드포인트 및 기타 리소스 목록은 Using the Slack Web API를 참조하세요.

지원되는 권한 부여 유형

OAuth 2.0

401 또는 407 응답이 반환되면 OAUTH 토큰이 새로 고쳐집니다.

Slack 애플리케이션을 생성하여 워크스페이스에 설치하면 API 대상 연결을 통해 호출을 인증하는 데 사용할 OAuth 보유자 토큰이 자동으로 생성됩니다.

필요한 추가 권한 부여 파라미터

해당 사항 없음

Slack 설명서

기본 앱 설정

OAuth를 사용하여 설치

메시지 검색

메시지 전송

수신 웹후크를 사용하여 메시지 전송

일반적으로 사용되는 API 작업

https://slack.com/api/chat.postMessage

추가 정보

EventBridge 규칙을 구성할 때 강조해야 할 두 가지 구성이 있습니다.

- 콘텐츠 유형을 'application/json;charset=utf-8'로 정의하는 헤더 파라미터를 포함합니다.
- 입력 변환기를 사용하여 입력 이벤트를 Slack API의 예상 출력에 매핑합니다. 즉, Slack API로 전송되는 페이로드에 "채널" 및 "텍스트" 키/값 쌍이 있는지 확인합니다.

Shopify

API 대상 간접 호출 엔드포인트 URL

엔드포인트와 기타 리소스 및 메서드 목록은 Endpoints and requests를 참조하세요.

지원되는 권한 부여 유형

OAuth, API 키

> **Note**
>
> 401 또는 407 응답이 반환되면 OAUTH 토큰이 새로 고쳐집니다.

필요한 추가 권한 부여 파라미터

해당 사항 없음

Shopify 설명서

인증 및 권한 부여 개요

일반적으로 사용되는 API 작업

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

추가 정보

앱 생성

아마존 EventBridge 웹훅 배송

Shopify 관리자의 사용자 지정 앱에 대한 액세스 토큰

제품

Shopify 관리자 API

Splunk

API 대상 간접 호출 엔드포인트 URL

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

지원되는 권한 부여 유형

기본, API 키

필요한 추가 권한 부여 파라미터

None

Splunk 설명서

두 가지 권한 부여 유형에 모두 HEC 토큰 ID가 필요합니다. 자세한 내용은 Set up and use HTTP Event Collector in Splunk Web을 참조하세요.

일반적으로 사용되는 API 작업

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

추가 정보

API 키 — 엔드포인트를 구성할 때 API 키 이름은 “권한 부여”이고 값은 스플렁크 HEC 토큰 ID입니다. EventBridge

기본 (사용자 이름/암호) - 엔드포인트를 구성할 때 사용자 이름은 “Splunk”이고 암호는 Splunk HEC 토큰 ID입니다. EventBridge

Sumo Logic

API 대상 간접 호출 엔드포인트 URL

HTTP 로그 및 지표 소스 엔드포인트 URL은 사용자마다 다릅니다. 자세한 내용은 HTTP Logs and Metrics Source를 참조하세요.

지원되는 권한 부여 유형

Sumo Logic은 URL에 고유 키가 베이크되어 있으므로 HTTP 소스에 대한 인증이 필요하지 않습니다. 이러한 이유 때문에 해당 URL을 비밀로 취급해야 합니다.

EventBridge API 대상을 구성할 때는 인증 유형이 필요합니다. 이 요구 사항을 충족하려면 API 키를 선택하고 키 이름을 “dummy-key”로 지정하고 키 값을 “dummy-value”로 지정합니다.

필요한 추가 권한 부여 파라미터

해당 사항 없음

Sumo Logic 설명서

Sumo Logic이미 여러 AWS 서비스에서 로그 및 지표를 수집할 수 있는 호스팅 소스를 구축했으며 해당 웹 사이트의 정보를 사용하여 해당 소스로 작업할 수 있습니다. 자세한 내용은 Amazon Web Services를 참조하세요.

애플리케이션에서 사용자 지정 이벤트를 생성하고 이를 로그 또는 Sumo Logic 지표로 전송하려는 경우 EventBridge API 대상 및 Sumo Logic HTTP 로그 및 지표 소스 엔드포인트를 사용하십시오.

- 무료 Sumo Logic 인스턴스에 가입하고 생성하려면 Start your free trial today를 참조하세요.
- Sumo Logic 사용에 대한 자세한 내용은 HTTP Logs and Metrics Source를 참조하세요.

일반적으로 사용되는 API 작업

POST https://endpoint4.collection.us2.sumologic.com/receiver/v1/http/*UNIQUE_ID_PER_COLLECTOR*

추가 정보

없음

TriggerMesh

API 대상 간접 호출 엔드포인트 URL

HTTP용 이벤트 소스 주제의 정보를 사용하여 엔드포인트 URL을 공식화합니다. 엔드포인트 URL에는 다음 형식의 이벤트 소스 이름과 사용자 네임스페이스가 포함됩니다.

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

엔드포인트에 대한 요청에 기본 권한 부여 파라미터를 포함합니다.

지원되는 권한 부여 유형

기본

필요한 추가 권한 부여 파라미터

None

TriggerMesh 설명서

HTTP용 이벤트 소스

일반적으로 사용되는 API 작업

해당 사항 없음

추가 정보

없음

Zendesk

API 대상 간접 호출 엔드포인트 URL

https://developer.zendesk.com/rest_api/docs/support/tickets

지원되는 권한 부여 유형

기본, API 키

필요한 추가 권한 부여 파라미터

None

Zendesk 설명서

보안 및 인증

일반적으로 사용되는 API 작업

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

추가 정보

API EventBridge 요청은 Zendesk API 한도 계산에 포함됩니다. 사용자 플랜의 Zendesk 제한에 대한 자세한 내용은 Usage limits를 참조하세요.

계정과 데이터를 더 안전하게 보호하려면 기본 로그인 자격 증명 인증 대신 API 키를 사용하는 것이 좋습니다.

아마존 API Gateway용 아마존 EventBridge 타겟

Amazon API Gateway를 사용해 API를 생성, 게시, 유지 관리, 모니터링할 수 있습니다. EventBridgeAmazon은 API Gateway 엔드포인트로 이벤트 전송을 지원합니다. API Gateway 엔드포인트를 대상으로 지정하면 대상으로 전송되는 각 이벤트가 엔드포인트로 전송된 요청에 매핑됩니다.

> **Important**
>
> EventBridge API Gateway Edge에 최적화된 엔드포인트 및 리전 엔드포인트를 대상으로 사용할 수 있도록 지원합니다. 프라이빗 엔드포인트는 현재 지원되지 않습니다. 엔드포인트에 대한 자세한 내용은 https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-api-endpoint-types.html 섹션을 참조하세요.

다음과 같은 사용 사례에 API Gateway 대상을 사용할 수 있습니다.

- API Gateway에 호스팅된 고객 지정 API를 AWS 호출하거나 타사 이벤트를 기반으로 호출합니다.
- 일정에 따라 주기적으로 엔드포인트를 간접 호출합니다.

EventBridge JSON 이벤트 정보는 HTTP 요청 본문으로 엔드포인트에 전송됩니다. 다음과 같이 대상의 `HttpParameters` 필드에 다른 요청 속성을 지정할 수 있습니다.

- `PathParameterValues`는 엔드포인트 ARN의 모든 경로 변수에 순차적으로 해당하는 값을 나열합니다(예: `"arn:aws:execute-api:us-east-1:112233445566:myapi/dev/POST/pets/*/*"`).
- `QueryStringParameters`호출된 엔드포인트에 EventBridge 추가되는 쿼리 문자열 파라미터를 나타냅니다.
- `HeaderParameters`는 요청에 추가할 HTTP 헤더를 정의합니다.

> **Note**
>
> 보안상의 이유로 다음 HTTP 헤더 키는 허용되지 않습니다.
>
> - `X-Amz` 또는 `X-Amzn` 접두사가 붙은 모든 항목
> - `Authorization`
> - `Connection`
> - `Content-Encoding`
> - `Content-Length`
> - `Host`
> - `Max-Forwards`
> - `TE`
> - `Transfer-Encoding`
> - `Trailer`
> - `Upgrade`
> - `Via`
> - `WWW-Authenticate`
> - `X-Forwarded-For`

동적 파라미터

API Gateway 대상을 간접 호출할 때 대상으로 전송되는 이벤트에 데이터를 동적으로 추가할 수 있습니다. 자세한 내용은 the section called “대상 파라미터” 단원을 참조하십시오.

간접 호출 재시도

모든 대상과 마찬가지로 일부 실패한 호출을 EventBridge 재시도합니다. API Gateway의 경우 5xx 또는 429 HTTP 상태 코드와 함께 전송된 응답을 지수적 백오프 및 지터를 사용하여 최대 24시간 동안 EventBridge 재시도합니다. 그런 다음 CloudWatch Amazon에 `FailedInvocations` 지표를 EventBridge 게시합니다. EventBridge 다른 4xx HTTP 오류는 재시도하지 않습니다.

제한 시간

EventBridge rule API Gateway 요청의 최대 클라이언트 실행 제한 시간은 5초여야 합니다. API Gateway가 응답하는 데 5초 이상 걸리는 경우 요청 EventBridge 제한 시간을 초과한 다음 다시 시도합니다.

EventBridge 파이프 API Gateway 요청의 최대 제한 시간은 API Gateway 최대값인 29초입니다.

AWS AppSync 아마존 대상 EventBridge

AWS AppSync 개발자는 안전한 서버리스 고성능 GraphQL 및 Pub/Sub API를 사용하여 애플리케이션과 서비스를 데이터 및 이벤트에 연결할 수 있습니다. 를 사용하면 GraphQL 변이가 있는 응용 프로그램에 실시간 데이터 업데이트를 게시할 수 있습니다. AWS AppSync EventBridge 일치하는 이벤트에 대해 유효한 GraphQL 변형 연산 호출을 지원합니다. AWS AppSync API 변형을 대상으로 지정하면 은 변형 작업을 통해 이벤트를 AWS AppSync 처리한 다음 해당 돌연변이에 연결된 구독을 트리거할 수 있습니다.

> **Note**
>
> EventBridge AWS AppSync 공개 GraphQL API를 지원합니다. EventBridge 현재 프라이빗 API는 지원하지 AWS AppSync 않습니다.

다음과 같은 사용 사례에 AWS AppSync GraphQL API 타겟을 사용할 수 있습니다.

- 이벤트 데이터를 구성된 데이터 소스로 푸시, 변환 및 저장합니다.
- 연결된 애플리케이션 클라이언트에 실시간 알림을 전송합니다.

> **Note**
>
> AWS AppSync 대상은 권한 부여 유형을 사용한 AWS AppSync GraphQL API 호출만 지원합니다. `AWS_IAM`

AWS AppSync GraphQL API에 대한 자세한 내용은 개발자 안내서의 GraphQL 및 아키텍처를 참조하십시오. AWS AppSync AWS AppSync

콘솔을 사용하여 규칙의 AWS AppSync 대상을 지정하려면 EventBridge

1. 규칙을 만들거나 편집하십시오.
2. 대상에서 AWS 서비스 및 AWS AppSync를 선택하여 대상을 지정하십시오.
3. 선택 세트와 함께 파싱하여 실행할 변형 작업을 지정합니다.

   - AWS AppSync API를 선택한 다음 호출할 GraphQL API 뮤테이션을 선택합니다.
   - 파라미터 및 선택 세트 구성에서 키-값 매핑 또는 입력 변환기를 사용하여 선택 세트를 만들도록 선택합니다.

     Key-value mapping

     키-값 매핑을 사용하여 선택 세트를 만들려면 다음과 같이 하십시오.

     - API 파라미터의 변수를 지정합니다. 각 변수는 정적 값이거나 이벤트 페이로드의 동적 JSON 경로 표현식일 수 있습니다.
     - 선택 세트에서 응답에 포함할 변수를 선택합니다.

     Input transformer

     입력 변환기를 사용하여 선택 세트를 만들려면 다음과 같이 하십시오.

     - 사용할 변수를 정의하는 입력 경로를 지정합니다.
     - 입력 템플릿을 지정하여 대상에 전달하려는 정보를 정의하고 형식을 지정합니다.

     자세한 정보는 ???을 참조하세요.
4. 실행 역할에서 새 역할을 생성할지 또는 기존 역할을 사용할지 선택합니다.
5. 규칙 생성 또는 편집을 완료합니다.

예: 아마존 AWS AppSync 대상 EventBridge

다음 예제에서는 전달을 위한 이벤트 형식을 지정하는 입력 변환을 정의하는 것을 포함하여 EventBridge 규칙의 AWS AppSync 대상을 지정하는 방법을 살펴보겠습니다.

다음 스키마로 정의된 AWS AppSync GraphQL `Ec2EventAPI` API가 있다고 가정해 보겠습니다.

```
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"])
}
```

이 API를 사용하는 애플리케이션 클라이언트는 `subscribeToEvent` 구독을 구독할 수 있으며, 이는 `pushEvent` 변형에 의해 트리거됩니다.

뮤테이션을 통해 AppSync API로 이벤트를 보내는 타겟이 있는 EventBridge 규칙을 만들 수 있습니다. `pushEvent` 변형이 간접 호출되면 구독한 모든 클라이언트가 이벤트를 수신합니다.

이 API를 EventBridge 규칙의 대상으로 지정하려면 다음과 같이 하십시오.

1. 규칙 대상의 Amazon 리소스 이름(ARN)을 `Ec2EventAPI` API의GraphQL 엔드포인트로 설정합니다.
2. 변형 GraphQL 작업을 대상 파라미터로 지정합니다.

```
mutation CreatePushEvent($id: ID!, $statusCode: String, $instanceId: String) {
  pushEvent(id: $input, statusCode: $statusCode, instanceId: $instanceId) {
    id
```

```
    statusCode
    instanceId
  }
}
```

변형 선택 세트에는 GraphQL 구독에서 구독하려는 모든 필드가 포함되어야 합니다.

3. 입력 변환기를 구성하여 일치하는 이벤트의 데이터가 작업에 사용되는 방식을 지정합니다.

   “EC2 Instance Launch Successful” 샘플 이벤트를 선택했다고 가정해 보겠습니다.

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

대상 입력 변환기의 입력 경로를 사용하여 템플릿에서 사용할 다음 변수를 정의할 수 있습니다.

```
{
  "id": "$.id",
  "statusCode": "$.detail.StatusCode",
  "EC2InstanceId": "$.detail.EC2InstanceId"
}
```

입력 변환기 템플릿을 작성하여 변형 작업에 EventBridge 전달되는 변수를 정의합니다 AWS AppSync . 템플릿은 JSON으로 평가되어야 합니다. 입력 경로가 주어지면 다음 템플릿을 작성할 수 있습니다.

```
{
  "id": <id>,
  "statusCode": <statusCode>,
  "instanceId": <EC2InstanceId>
}
```

HTTP 엔드포인트 타겟을 위한 연결

연결은 지정된 HTTP 엔드포인트에 연결하는 EventBridge 데 사용할 권한 부여 방법과 자격 증명을 정의합니다. 권한 부여 설정을 구성하고 연결을 생성하면 인증 정보를 안전하게 AWS Secrets Manager 저장하기 위한 암호가 생성됩니다. HTTP 엔드포인트 대상에 맞게 연결에 포함할 추가 매개 변수를 추가할 수도 있습니다.

다음과 함께 연결을 사용하십시오.

- API 대상

  API 대상을 생성할 때 사용할 연결을 지정합니다. 계정에서 기존 연결을 선택하거나 API 대상을 생성할 때 연결을 생성할 수 있습니다.

연결을 위한 권한 부여 방법

EventBridge 연결은 다음과 같은 인증 방법을 지원합니다.

- 기본
- API 키

  기본 및 API 키 인증의 경우 필요한 인증 헤더를 EventBridge 자동으로 채웁니다.

- OAuth

  EventBridge 또한 OAuth 인증의 경우 클라이언트 ID와 암호를 액세스 토큰으로 교환한 다음 안전하게 관리합니다.

  401 또는 407 응답이 반환되면 OAUTH 토큰이 새로 고쳐집니다.

연결을 생성할 때 엔드포인트에 대한 권한 부여에 필요한 헤더, 본문 및 쿼리 파라미터를 포함할 수도 있습니다. 엔드포인트에 대한 권한 부여가 동일할 경우 둘 이상의 HTTP 엔드포인트에 동일한 연결을 사용할 수 있습니다.

연결을 생성하고 권한 부여 매개변수를 추가하면 에서 비밀이 EventBridge 생성됩니다 AWS Secrets Manager. Secrets Manager 암호 저장 및 액세스 비용은 API 대상 사용 요금에 포함됩니다. API 대상과 함께 암호를 사용하는 모범 사례에 대해 자세히 알아보려면 사용 CloudFormation 설명서를 참조하십시오 AWS::Events::ApiDestination.

> **Note**
>
> 연결을 성공적으로 생성하거나 업데이트하려면 Secrets Manager를 사용할 권한이 있는 계정을 사용해야 합니다. 필요한 권한은 AmazonEventBridgeFullAccess 정책에 포함되어 있습니다. 연결을 위해 계정에 생성된 서비스 연결 역할에도 동일한 권한이 부여됩니다.

HTTP 엔드포인트 대상에 대한 연결 생성

콘솔을 사용하여 HTTP 엔드포인트와 함께 사용할 연결을 만들려면 EventBridge

1. 관리 권한이 있는 계정을 AWS 사용하여 EventBridge 로그인하고 EventBridge 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 API 대상을 선택합니다.
3. API 대상 테이블까지 아래로 스크롤한 다음, 연결 탭을 선택합니다.
4. 연결 생성을 선택합니다.
5. 연결 생성 페이지에서 연결에 대한 연결 이름을 입력합니다.
6. 연결에 대한 설명을 입력합니다.
7. 권한 부여 유형에서는 이 연결을 사용하는 API 대상에 지정된 HTTP 엔드포인트에 대한 연결을 승인하는 데 사용할 권한 부여 유형을 선택합니다. 다음 중 하나를 수행하십시오.
   - 기본(사용자 이름/암호)을 선택한 다음, HTTP 엔드포인트에 권한을 부여하는 데 사용할 사용자 이름과 암호를 입력합니다.

- OAuth 클라이언트 자격 증명을 선택한 다음, 엔드포인트에 권한을 부여하는 데 사용할 권한 부여 엔드포인트, HTTP 메서드, 클라이언트 ID, 클라이언트 암호를 입력합니다.

  OAuth HTTP 파라미터에서 권한 부여 엔드포인트에 대한 권한 부여에 포함할 추가 파라미터를 추가합니다. 드롭다운 목록에서 파라미터를 선택한 다음, 키와 값을 입력합니다. 추가 파라미터를 포함하려면 파라미터 추가를 선택합니다.

  간접 호출 HTTP 파라미터에서 권한 부여 요청에 포함할 추가 파라미터를 추가합니다. 파라미터를 추가하려면 드롭다운 목록에서 파라미터를 선택한 다음, 키와 값을 입력합니다. 추가 파라미터를 포함하려면 파라미터 추가를 선택합니다.

- API 키를 선택하고 API 키 권한 부여에 사용할 API 키 이름 및 관련 값을 입력합니다.

  간접 호출 HTTP 파라미터에서 권한 부여 요청에 포함할 추가 파라미터를 추가합니다. 파라미터를 추가하려면 드롭다운 목록에서 파라미터를 선택한 다음, 키와 값을 입력합니다. 추가 파라미터를 포함하려면 파라미터 추가를 선택합니다.

8. Create를 선택합니다.

EventBridge 콘솔을 사용하여 연결 편집

기존 연결을 편집할 수 있습니다.

EventBridge 콘솔을 사용하여 연결을 편집하려면

1. 관리 권한이 있는 계정을 AWS 사용하여 EventBridge 로그인하고 EventBridge 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 API 대상을 선택합니다.
3. API 대상 테이블까지 아래로 스크롤한 다음, 연결 탭을 선택합니다.
4. 연결 테이블에서 편집할 연결을 선택합니다.
5. 연결 세부 정보 페이지에서 편집을 선택합니다.
6. 연결 값을 업데이트한 다음, 업데이트를 선택합니다.

콘솔을 사용하여 연결 인증 취소하기 EventBridge

연결 권한 부여를 취소하면 모든 권한 부여 파라미터가 제거됩니다. 권한 부여 파라미터를 제거하면 연결에서 보안 암호가 제거되므로 새 연결을 만들지 않고도 다시 사용할 수 있습니다.

> **Note**
>
> HTTP 엔드포인트에 요청을 성공적으로 보내려면 승인되지 않은 연결을 사용하는 모든 HTTP 엔드포인트를 업데이트하여 다른 연결을 사용해야 합니다.

연결 권한 부여를 취소하려면

1. 관리 권한이 있는 계정을 AWS 사용하여 EventBridge 로그인하고 콘솔을 엽니다. EventBridge
2. 왼쪽 탐색 창에서 API 대상을 선택합니다.
3. API 대상 테이블까지 아래로 스크롤한 다음, 연결 탭을 선택합니다.
4. 연결 테이블에서 연결을 선택합니다.
5. 연결 세부 정보 페이지에서 권한 부여 철회를 선택합니다.
6. 연결의 권한 부여를 취소하시겠어요? 대화 상자에서 연결 이름을 입력한 다음, 권한 부여 철회를 선택합니다.

프로세스가 완료될 때까지 연결 상태가 권한 부여 철회 중으로 변경됩니다. 그 후 상태가 권한 부여 철회됨으로 변경됩니다. 이제 연결을 편집하여 새 권한 부여 파라미터를 추가할 수 있습니다.

AWS 계정 간 Amazon EventBridge 이벤트 전송 및 수신

AWS 계정 내 이벤트 버스 간에 이벤트를 보내고 EventBridge 받도록 구성할 수 있습니다. 계정 간에 이벤트를 보내거나 EventBridge 받도록 구성한 경우 계정의 이벤트 버스로 이벤트를 보내거나 이벤트 버스에서 이벤트를 수신할 수 있는 계정을 지정할 수 있습니다. AWS 이벤트 버스와 관련된 특정 규칙의 이벤트 또는 특정 소스의 이벤트를 허용하거나 거부할 수도 있습니다. 자세한 내용은 Amazon EventBridge 리소스 정책을 통한 계정 간 액세스 단순화를 참조하십시오.

> **Note**
>
> 를 사용하는 AWS Organizations경우 조직을 지정하고 해당 조직의 모든 계정에 액세스 권한을 부여할 수 있습니다. 또한 다른 계정으로 이벤트를 전송할 경우 전송 이벤트 버스에는 IAM 역할이 연결되어 있어야 합니다. 자세한 내용은AWS Organizations 사용 설명서에서 AWS Organizations란 무엇입니까? 단원을 참조하세요.

> **Note**
>
> Incident Manager 대응 계획을 대상으로 사용하는 경우 계정과 공유되는 모든 대응 계획을 기본적으로 사용할 수 있습니다.

대상 지역이 지역 간 지원되는 대상 지역이면 모든 지역의 동일한 지역 내 AWS 계정에 있는 이벤트 버스 간에 이벤트를 보내고 받을 수 있고, 다른 지역의 계정 간에도 이벤트를 보내고 받을 수 있습니다.

다른 계정의 이벤트 버스로 이벤트를 보내거나 이벤트 버스에서 이벤트를 EventBridge 수신하도록 구성하는 단계는 다음과 같습니다.

- 수신자 계정에서 이벤트 버스의 권한을 편집하여 지정된 AWS 계정, 조직 또는 모든 AWS 계정이 수신자 계정으로 이벤트를 보낼 수 있도록 합니다.
- 발신자 계정에서 수신자 계정의 이벤트 버스를 대상으로 갖는 하나 이상의 규칙을 설정합니다.

  발신자 계정이 AWS 조직으로부터 이벤트를 전송할 수 있는 권한을 상속받는 경우, 발신자 계정에는 수신자 계정으로 이벤트를 전송할 수 있는 정책이 포함된 IAM 역할도 있어야 합니다. 를 사용하여 수신자 AWS Management Console 계정의 이벤트 버스를 대상으로 하는 규칙을 생성하면 역할이 자동으로 생성됩니다. 를 AWS CLI사용하는 경우 역할을 수동으로 만들어야 합니다.
- 수신자 계정에서 발신자 계정이 전송하는 이벤트를 일치시키는 하나 이상의 규칙을 설정합니다.

한 계정에서 다른 계정으로 전송되는 이벤트에 대해서는 전송 계정에서 사용자 지정 이벤트로 요금이 부과됩니다. 수신 계정에는 요금이 부과되지 않습니다. 자세한 내용은 Amazon EventBridge 요금을 참조하십시오.

수신자 계정은 발신자 계정에서 수신되는 이벤트를 제3의 계정으로 전송하는 규칙을 설정하는 경우 이러한 이벤트는 제3의 계정으로 전송되지 않습니다.

동일한 계정에 세 개의 이벤트 버스가 있고 첫 번째 이벤트 버스에서 두 번째 이벤트 버스의 이벤트를 세 번째 이벤트 버스로 전달하는 규칙을 설정한 경우 해당 이벤트는 세 번째 이벤트 버스로 전송되지 않습니다.

다음 동영상은 계정 간 라우팅 이벤트를 다룹니다. 이벤트를 다른 AWS 계정의 버스로 라우팅하는 방법을 다룹니다.

다른 AWS 계정의 이벤트를 허용할 수 있는 권한을 부여하십시오.

다른 계정 또는 조직으로부터 이벤트를 수신하려면 먼저 이벤트를 수신하려는 이벤트 버스에 대한 권한을 편집해야 합니다. 기본 이벤트 버스는 AWS 서비스, 승인된 다른 AWS 계정 및 PutEvents 통화의 이벤트를 수락합니다. 이벤트 버스에 대한 권한은 이벤트 버스에 연결된 리소스 기반 정책을 사용해 부여되거나 거부됩니다. 정책에서 AWS 계정 ID를 사용하여 다른 계정에 권한을 부여하거나 조직 ID를 사용하여 AWS 조직에 권한을 부여할 수 있습니다. 예제 정책을 비롯하여 이벤트 버스 권한에 대한 자세한 내용은 Amazon EventBridge 이벤트 버스에 대한 권한 섹션을 참조하세요.

> **Note**
>
> EventBridge 이제 모든 새로운 크로스 어카운트 이벤트 버스 타겟에서 IAM 역할을 추가해야 합니다. 이는 2023년 3월 2일 이후에 생성된 이벤트 버스 대상에만 적용됩니다. 해당 날짜 이전에 IAM 역할 없이 생성된 애플리케이션은 영향을 받지 않습니다. 그러나 사용자에게 다른 계정의 리소스에 대한 액세스 권한을 부여하는 IAM 역할을 추가하는 것이 좋습니다. 이렇게 하면 서비스 제어 정책(SCP)을 사용해 조직의 경계를 적용하여 조직에 속한 계정에서 이벤트를 보내고 받을 수 있는 사람을 결정할 수 있습니다.

> **Important**
>
> 모든 AWS 계정으로부터 이벤트를 수신하기로 선택한 경우, 다른 계정으로부터 수신할 이벤트만 일치시키는 규칙을 생성하도록 주의하십시오. 더욱 안전한 규칙을 생성하려면 이벤트를 수신할 계정 하나 이상의 계정 ID가 입력되는 Account 필드가 각 규칙의 모든 패턴에 포함되어야 합니다. 이벤트 패턴에 계정 필드가 포함되는 규칙은 Account 필드에 나열되어 있지 않은 계정에서 전송된 이벤트와 일치하지 않습니다. 자세한 정보는 아마존 EventBridge 이벤트을 참조하세요.

AWS 계정 간 이벤트 규칙

계정이 다른 AWS 계정의 이벤트 버스로부터 이벤트를 수신하도록 설정된 경우 해당 이벤트에 맞는 규칙을 작성할 수 있습니다. 다른 계정의 이벤트 버스에서 수신할 이벤트에 일치하도록 규칙의 이벤트 패턴을 설정합니다.

규칙의 이벤트 패턴에 account를 지정하지 않을 경우 다른 계정의 이벤트 버스로부터 수신하는 이벤트를 일치시키는 계정의 모든 규칙(신규 및 기존)이 해당 이벤트를 기준으로 트리거됩니다. 다른 계정

의 이벤트 버스에서 이벤트를 수신할 때 자체 계정에서 생성된 이벤트 패턴에서만 규칙이 트리거되도록 하려면 규칙의 이벤트 패턴에 account를 추가하고 자체 계정 ID를 지정해야 합니다.

모든 AWS 계정의 이벤트 버스 이벤트를 수락하도록 AWS 계정을 설정하는 경우 계정의 모든 EventBridge 규칙에 추가하는 account 것이 좋습니다. 이렇게 하면 계정 내 규칙이 알 AWS 수 없는 계정에서 발생한 이벤트에 대해 트리거되는 것을 방지할 수 있습니다. 규칙에 account 필드를 지정할 때 AWS 계정 2개 이상의 계정 ID를 필드에 지정할 수 있습니다.

권한을 부여한 AWS 계정의 모든 이벤트 버스에서 일치하는 이벤트에 대해 규칙이 트리거되도록 하려면 규칙 account 필드에 *를 지정하지 마십시오. 이벤트의 account 필드에는 *가 절대 나타나지 않으므로 그렇게 하면 모든 이벤트가 일치하지 않을 것입니다. 대신 규칙에서 account 필드만 생략하세요.

AWS 계정 간에 이벤트를 전송하는 규칙 생성

다른 계정의 이벤트 버스를 대상으로 지정하는 것은 규칙 생성의 일부입니다.

콘솔을 사용하여 다른 AWS 계정으로 이벤트를 보내는 규칙 생성하기

1. ??? 절차에서 해당 단계를 따릅니다.
2. ??? 단계에서 대상 유형을 선택하라는 메시지가 표시되면
   a. EventBridge 이벤트 버스를 선택합니다.
   b. 다른 계정 또는 리전의 이벤트 버스를 선택합니다.
   c. 대상 이벤트 버스에는 사용하려는 이벤트 버스의 ARN을 입력합니다.
3. 절차 단계에 따라 규칙 생성을 완료합니다.

AWS 지역 간 Amazon EventBridge 이벤트 전송 및 수신

AWS 지역 간에 이벤트를 보내고 EventBridge 받도록 구성할 수 있습니다. 특정 리전의 이벤트, 이벤트 버스와 관련된 특정 규칙의 이벤트 또는 특정 소스의 이벤트를 허용하거나 거부할 수도 있습니다. 자세한 내용은 Amazon을 통한 지역 간 이벤트 라우팅 소개를 참조하십시오. EventBridge

지원되는 대상 리전은 다음과 같습니다.

- 미국 동부(버지니아 북부)
- 미국 동부(오하이오)
- 미국 서부(캘리포니아 북부)

- 미국 서부(오레곤)
- 아프리카(케이프타운)
- 아시아 태평양(홍콩)
- 아시아 태평양(도쿄)
- 아시아 태평양(서울)
- 아시아 태평양(오사카)
- 아시아 태평양(뭄바이)
- 아시아 태평양(하이데라바드)
- 아시아 태평양(싱가포르)
- 아시아 태평양(자카르타)
- 아시아 태평양(시드니)
- 아시아 태평양(멜버른)
- 캐나다(중부)
- 캐나다 서부(캘거리)
- 유럽(프랑크푸르트)
- 유럽(스페인)
- 유럽(취리히)
- 유럽(스톡홀름)
- 유럽(밀라노)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(파리)
- 이스라엘(텔아비브)
- 중동(UAE)
- 중동(바레인)
- 남아메리카(상파울루)

다음 동영상은 https://console.aws.amazon.com/events/ AWS CloudFormation, 및 AWS Serverless Application Model: 지역 간 이벤트 라우팅을 사용하여 지역 간 이벤트를 라우팅하는 방법을 설명합니다. 지역 간 이벤트 라우팅

이벤트를 다른 AWS 지역으로 보내는 규칙 만들기

다른 AWS 지역의 이벤트 버스를 대상으로 지정하는 것은 규칙 생성의 일부입니다.

콘솔을 사용하여 다른 AWS 계정으로 이벤트를 보내는 규칙을 만들려면

1. ??? 절차에서 해당 단계를 따릅니다.
2. ??? 단계에서 대상 유형을 선택하라는 메시지가 표시되면

   a. EventBridge 이벤트 버스를 선택합니다.
   b. 다른 계정 또는 리전의 이벤트 버스를 선택합니다.
   c. 대상 이벤트 버스에는 사용하려는 이벤트 버스의 ARN을 입력합니다.
3. 절차 단계에 따라 규칙 생성을 완료합니다.

동일한 계정 및 지역의 이벤트 버스 간에 Amazon EventBridge 이벤트 전송 및 수신

동일한 AWS 계정 및 지역의 이벤트 버스 간에 이벤트를 보내고 EventBridge 받도록 구성할 수 있습니다.

이벤트 버스 간에 이벤트를 보내거나 EventBridge 받도록 구성하면 발신자 이벤트 버스의 IAM 역할을 사용하여 발신자 이벤트 버스에 수신자 이벤트 버스에 이벤트를 전송할 권한을 부여합니다. 수신자 이벤트 버스에서 리소스 기반 정책을 사용하여 발신자 이벤트 버스로부터 이벤트를 수신할 수 있는 권한을 수신자 이벤트 버스에 부여합니다. 특정 이벤트 버스의 이벤트, 이벤트 버스와 관련된 특정 규칙의 이벤트 또는 특정 소스의 이벤트를 허용하거나 거부할 수도 있습니다. 예제 정책을 비롯하여 이벤트 버스 권한에 대한 자세한 내용은 Amazon EventBridge 이벤트 버스에 대한 권한 섹션을 참조하세요.

계정의 이벤트 버스로 이벤트를 보내거나 이벤트 버스 간에 이벤트를 EventBridge 수신하도록 구성하는 단계는 다음과 같습니다.

- 기존 IAM 역할을 사용하려면 발신자 이벤트 버스 권한을 수신자 이벤트 버스에 부여하거나 수신자 이벤트 버스 권한을 반신자 이벤트 버스에 부여해야 합니다.
- 발신자 이벤트 버스에서 수신자 이벤트 버스를 대상으로 하는 하나 이상의 규칙을 설정하고 IAM 역할을 생성합니다. 역할에 연결되어야 하는 정책의 예는 ??? 섹션을 참조하세요.
- 수신자 이벤트 버스에서 이벤트가 다른 이벤트 버스로부터 전달될 수 있도록 권한을 편집합니다.
- 수신자 이벤트에서 발신자 이벤트 버스가 전송하는 이벤트를 일치시키는 하나 이상의 규칙을 설정합니다.

> **Note**
>
> EventBridge 발신자 이벤트 버스에서 수신한 이벤트를 세 번째 이벤트 버스로 라우팅할 수 없습니다.

한 이벤트 버스에서 다른 이벤트 버스로 전송된 이벤트는 사용자 지정 이벤트로 요금이 부과됩니다. 자세한 내용은 Amazon EventBridge 요금을 참조하십시오.

동일한 AWS 계정 및 지역의 다른 이벤트 버스로 이벤트를 보내는 규칙 생성

이벤트를 다른 이벤트 버스로 보내려면 이벤트 버스를 대상으로 하는 규칙을 생성합니다. 동일한 AWS 계정 및 지역의 이벤트 버스를 대상으로 지정하는 것은 규칙 생성의 일부입니다.

콘솔을 사용하여 동일한 AWS 계정 및 지역의 다른 이벤트 버스로 이벤트를 보내는 규칙을 만들려면

1. ??? 절차에서 해당 단계를 따릅니다.
2. ??? 단계에서 대상 유형을 선택하라는 메시지가 표시되면
    a. EventBridge 이벤트 버스를 선택합니다.
    b. 동일한 AWS 계정 및 지역에서 이벤트 버스를 선택합니다.
    c. 이벤트 버스를 대상으로 하려면 드롭다운 목록에서 이벤트 버스를 선택합니다.
3. 절차 단계에 따라 규칙 생성을 완료합니다.

아마존 EventBridge 인풋 트랜스포메이션

정보를 규칙 대상으로 EventBridge 전달하기 전에 이벤트의 텍스트를 사용자 정의할 수 있습니다. 콘솔 또는 API의 입력 변환기를 통해 JSON 경로를 사용하여 원래 이벤트 소스의 값을 참조하는 변수를 정의합니다. 변환된 이벤트는 원래 이벤트 대신 대상으로 전송됩니다. 그러나 동적 경로 파라미터는 변환된 이벤트가 아닌 원래 이벤트를 참조해야 합니다. 입력에서 각 값을 할당하여 최대 100개의 변수를 정의할 수 있습니다. 그런 다음, <*variable-name*>으로 입력 템플릿 내에서 이러한 변수를 사용할 수 있습니다.

입력 변환기 사용에 대한 자습서는 ??? 섹션을 참조하세요.

> **Note**
>
> EventBridge 모든 JSON Path 구문을 지원하지는 않으며 런타임에 이를 평가합니다. 지원되는 구문은 다음과 같습니다.
>
> - 점 표기법(예: `$.detail`)
> - 대시
> - 밑줄
> - 영숫자
> - 배열 인덱스
> - 와일드카드(*)

이 주제에서 수행할 작업

- 미리 정의된 변수
- 입력 변환 예제
- API를 사용하여 입력을 변환합니다. EventBridge
- 를 사용하여 입력을 변환합니다. AWS CloudFormation
- 입력 변환과 관련된 일반적인 문제
- 규칙 생성 과정으로 입력 변환기 구성
- EventBridge 샌드박스를 사용하여 타겟 입력 트랜스포머를 테스트합니다.

미리 정의된 변수

JSON 경로를 정의하지 않고 사용할 수 있는 미리 정의된 변수가 있습니다. 이러한 변수는 예약되어 있으며 이러한 이름으로 변수를 만들 수 없습니다.

- `aws.events.rule-arn`— 규칙의 Amazon 리소스 이름 (ARN). EventBridge
- `aws.events.rule-name`— EventBridge 규칙의 이름.
- `aws.events.event.ingestion-time`— 이벤트가 수신된 시간 EventBridge. 이는 ISO 8601 타임스탬프입니다. 이 변수는 에서 EventBridge 생성되며 덮어쓸 수 없습니다.
- `aws.events.event` — JSON 형식의 원본 이벤트 페이로드입니다(`detail` 필드 제외). 내용이 이스케이프되지 않으므로 JSON 필드의 값으로만 사용할 수 있습니다.
- `aws.events.event.json` — JSON 형식의 전체 원본 이벤트 페이로드입니다(`detail` 필드 포함). 내용이 이스케이프되지 않으므로 JSON 필드의 값으로만 사용할 수 있습니다.

입력 변환 예제

다음은 Amazon EC2 이벤트 예제입니다.

```
{
  "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"
  }
}
```

콘솔에서 규칙을 정의할 때 입력 구성에서 입력 변환기 옵션을 선택합니다. 이 옵션은 텍스트 상자 두 개를 표시합니다. 하나는 입력 경로, 하나는 입력 템플릿에 대한 텍스트 상자입니다.

입력 경로는 변수를 정의하는 데 사용됩니다. JSON 경로를 사용하여 이벤트의 항목을 참조하고 해당 값을 변수에 저장합니다. 예를 들어 첫 번째 텍스트 상자에 다음을 입력하여 이벤트 예제의 값을 참조하는 입력 경로를 만들 수 있습니다. 대괄호와 인덱스를 사용해 배열에서 항목을 가져올 수도 있습니다.

> **Note**
>
> EventBridge 유효한 JSON 출력을 보장하기 위해 런타임 시 입력 변환기를 대체합니다. 따라서 JSON 경로 파라미터를 참조하는 변수는 따옴표로 묶고 JSON 객체 또는 배열을 참조하는 변수는 따옴표로 묶지 마세요.

```
{
  "timestamp" : "$.time",
  "instance" : "$.detail.instance-id",
  "state" : "$.detail.state",
  "resource" : "$.resources[0]"
}
```

이는 <timestamp>, <instance>, <state> 및 <resource>라는 4가지 변수를 정의합니다. 입력 템플릿을 만들 때 이러한 변수를 참조할 수 있습니다.

입력 템플릿은 대상에 전달하려는 정보에 대한 템플릿입니다. 문자열이나 JSON을 대상에 전달하는 템플릿을 만들 수 있습니다. 다음 입력 템플릿 예는 이전 이벤트 및 입력 경로를 사용하여 이벤트를 대상으로 라우팅하기 전에 예제 출력으로 변환합니다.

설명	Template	출력
단순 문자열	`"instance <instance> is in <state>"`	`"instance i-0123456789 is in RUNNING"`
이스케이프된 따옴표가 있는 문자열	`"instance \"<instance>\" is in <state>"`	`"instance \"i-0123456789\" is in RUNNING"` 이 동작은 콘솔에서의 동작이라는 점에 유의하세요. EventBridge AWS CLI 는

설명	Template	출력
		슬래시 문자를 이스케이프하고 결과는 "instance "i-0123456789" is in RUNNING"입니다.

단순 JSON	<pre>{
	  "instance" :
	 <instance>,
	  "state": <state>
	}</pre>
		<pre>{
		  "instance" :
		 "i-0123456789",
		  "state": "RUNNING"
		}</pre>

문자열과 변수가 포함된 JSON	<pre>{
	 "instance" : <instance
	>,
	 "state": "<state>",
	 "instanceStatus":
	 "instance \"<instance>
	\" is in <state>"
	}</pre>
		<pre>{
		 "instance" : "i-012345
		6789",
		 "state": "RUNNING",
		 "instanceStatus":
		 "instance \"i-01234
		56789\" is in RUNNING"
		}</pre>

변수와 정적 정보가 혼합된 JSON	<pre>{
	  "instance" :
	 <instance>,
	  "state": [9, <state>,
	 true],
	  "Transformed" : "Yes"
	}</pre>
		<pre>{
		  "instance" :
		 "i-0123456789",
		  "state": [
		  9,
		  "RUNNING",
		  true
		  ],
		  "Transformed" : "Yes"
		}</pre>

설명	Template	출력
JSON에 예약된 변수 포함	`{ "instance" : <instance>, "state": <state>, "ruleArn" : <aws.events.rule-arn>, "ruleName" : <aws.events.rule-name>, "originalEvent" : <aws.events.event.json> }`	`{ "instance" : "i-0123456789", "state": "RUNNING", "ruleArn" : "arn:aws:events:us-east-2:123456789012:rule/example", "ruleName" : "example", "originalEvent" : { ... // commented for brevity } }`
문자열에 예약된 변수 포함	`"<aws.events.rule-name> triggered"`	`"example triggered"`
아마존 CloudWatch 로그 그룹	`{ "timestamp" : <timestamp>, "message": "instance \"<instance>\" is in <state>" }`	`{ "timestamp" : 2015-11-11T21:29:54Z, "message": "instance "i-0123456789" is in RUNNING }`

API를 사용하여 입력을 변환합니다. EventBridge

EventBridge API를 사용하여 입력을 변환하는 방법에 대한 자세한 내용은 입력 변환기를 사용하여 이벤트에서 데이터를 추출하고 해당 데이터를 대상에 입력하기 항목을 참조하십시오.

를 사용하여 입력을 변환합니다. AWS CloudFormation

입력을 변환하는 AWS CloudFormation 데 사용하는 방법에 대한 자세한 내용은 을 참조하십시오 AWS::Events::Rule InputTransformer.

입력 변환과 관련된 일반적인 문제

입력을 변환할 때 흔히 발생하는 몇 가지 문제는 다음과 같습니다. EventBridge

- 문자열의 경우 따옴표가 필요합니다.
- 템플릿에 대한 JSON 경로를 만들 때 검증이 수행되지 않습니다.
- 변수를 지정하여 이벤트에 존재하지 않는 JSON 경로를 일치시키는 경우 변수가 생성되지 않기 때문에 출력에 나타나지 않습니다.
- `aws.events.event.json`과 같은 JSON 속성은 JSON 필드의 값으로만 사용할 수 있으며 다른 문자열에서는 인라인으로 사용할 수 없습니다.
- EventBridge 대상의 입력 템플릿을 채울 때 입력 경로에서 추출한 값을 이스케이프하지 않습니다.
- JSON 경로가 JSON 개체 또는 배열을 참조하지만 변수가 문자열에서 참조되는 경우 유효한 문자열이 되도록 내부 따옴표를 EventBridge 제거합니다. 예를 들어`$.detail`, “Detail is”를 `<detail>` 가리키는 변수의 경우 <detail>객체에서 따옴표가 EventBridge 제거됩니다.

  따라서 단일 JSON 경로 변수를 기반으로 JSON 객체를 출력하려면 해당 변수를 키로 배치해야 합니다. 이 예에서는 `{"detail": <detail>}`입니다.
- 문자열을 나타내는 변수에는 따옴표가 필요하지 않습니다. 허용되지만 변환 출력이 유효한 JSON인지 확인하기 위해 변환 중에 문자열 변수 값에 EventBridge 자동으로 따옴표를 추가합니다. EventBridge JSON 개체 또는 배열을 나타내는 변수에는 따옴표를 추가하지 않습니다. JSON 객체 또는 배열을 나타내는 변수에 따옴표를 추가하지 마세요.

  예를 들어 다음 입력 템플릿에는 문자열과 JSON 객체를 모두 나타내는 변수가 포함되어 있습니다.

```
{
  "ruleArn" : <aws.events.rule-arn>,
  "ruleName" : <aws.events.rule-name>,
  "originalEvent" : <aws.events.event.json>
}
```

  올바른 따옴표를 사용하여 유효한 JSON을 생성합니다.

```
{
  "ruleArn" : "arn:aws:events:us-east-2:123456789012:rule/example",
  "ruleName" : "example",
  "originalEvent" : {
    ... // commented for brevity
  }
```

```
}
```

- 여러 줄 문자열로 출력되는 (JSON이 아닌) 텍스트의 경우 입력 템플릿의 각 줄을 큰 따옴표로 묶으십시오.

  예를 들어, 다음 이벤트 패턴에 대해 Amazon Inspector Finding EVENTS를 일치시키는 경우를 예로 들어 보겠습니다.

```
{
  "detail": {
    "severity": ["HIGH"],
    "status": ["ACTIVE"]
  },
  "detail-type": ["Inspector2 Finding"],
  "source": ["inspector2"]
}
```

  그리고 다음 입력 경로를 사용하세요.

```
{
  "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"
}
```

  아래 입력 템플릿을 사용하여 여러 줄 문자열 출력을 생성할 수 있습니다.

```
"<severity> severity finding <title>"
"Description: <description>"
"ARN: \"<arn>\""
"Type: <type>"
"AWS Account: <account>"
"Region: <region>"
"EC2 Instance: <instance>"
```

```
"Platform: <platform>"
"AMI: <ami>"
```

규칙 생성 과정으로 입력 변환기 구성

규칙을 만드는 과정에서 지정된 대상으로 이벤트를 보내기 전에 일치하는 이벤트를 처리하는 EventBridge 데 사용할 입력 변환기를 지정할 수 있습니다. AWS 서비스 또는 API 대상인 대상에 대해 입력 변환기를 구성할 수 있습니다.

규칙의 일부로 대상 입력 변환기를 생성하려면

1. ???에 설명된 대로 규칙을 생성하는 단계를 따르세요.
2. 3단계 - 대상 선택에서 추가 설정을 확장합니다.
3. 대상 입력 구성의 경우 드롭다운에서 입력 변환기를 선택합니다.

   입력 변환기 구성을 클릭합니다.

   EventBridge 입력 변환기 구성 대화 상자를 표시합니다.
4. 샘플 이벤트 섹션에서 이벤트 패턴을 테스트할 샘플 이벤트 유형을 선택합니다. AWS 이벤트 또는 파트너 이벤트를 선택하거나 사용자 지정 이벤트를 입력할 수 있습니다.

   AWS events

   지원되는 AWS 서비스에서 발생한 이벤트 중에서 선택합니다.

   1. AWS 이벤트를 선택합니다.
   2. 샘플 이벤트에서 원하는 AWS 이벤트를 선택합니다. 이벤트는 AWS 서비스별로 구성됩니다.

      이벤트를 선택하면 샘플 이벤트가 EventBridge 채워집니다.

      예를 들어, S3 객체 생성을 선택하면 샘플 S3 객체 생성 이벤트가 EventBridge 표시됩니다.
   3. (선택 사항) 복사를 선택하여 샘플 이벤트를 디바이스의 클립보드에 복사할 수도 있습니다.

   Partner events

   지원하는 타사 서비스 (예: Salesforce) 에서 EventBridge 생성된 이벤트 중에서 선택하십시오.

1. 파트너 이벤트를 선택하세요EventBridge .
2. 샘플 이벤트에서 원하는 파트너 이벤트를 선택합니다. 이벤트는 파트너별로 구성됩니다.

   이벤트를 선택하면 샘플 이벤트가 EventBridge 채워집니다.
3. (선택 사항) 복사를 선택하여 샘플 이벤트를 디바이스의 클립보드에 복사할 수도 있습니다.

Enter your own

JSON 텍스트로 자체 이벤트를 입력합니다.

1. 직접 입력을 선택합니다.
2. EventBridge 필수 이벤트 속성의 템플릿으로 샘플 이벤트를 채웁니다.
3. 필요에 따라 샘플 이벤트를 편집하고 추가합니다. 샘플 이벤트는 유효한 JSON이어야 합니다.
4. (선택 사항) 다음 옵션 중 하나를 선택할 수도 있습니다.
   - 복사 - 샘플 이벤트를 디바이스의 클립보드에 복사합니다.
   - 정리 - 줄 바꿈, 탭, 공백을 추가하여 JSON 텍스트를 더 쉽게 읽을 수 있습니다.

5. (선택 사항) 예시 입력 경로, 템플릿 및 출력 섹션을 확장하여 다음 예를 확인합니다.
   - JSON 경로를 사용하여 이벤트 데이터를 나타내는 변수를 정의하는 방법
   - 입력 변환기 템플릿에서 해당 변수를 사용하는 방법
   - 타겟으로 EventBridge 전송되는 결과 출력입니다.

   입력 변환의 자세한 예는 ??? 섹션을 참조하세요.
6. 대상 입력 변환기 섹션에서 입력 템플릿에 사용할 변수를 정의합니다.

   변수는 JSON 경로를 사용하여 원래 이벤트 소스의 값을 참조합니다. 그런 다음 입력 템플릿에서 해당 변수를 참조하여 대상으로 EventBridge 전달되는 변환된 이벤트에 원본 소스 이벤트의 데이터를 포함시킬 수 있습니다. 최대 100개의 변수를 정의할 수 있습니다. 입력 변환기는 유효한 JSON이어야 합니다.

   예를 들어 이 입력 변환기의 샘플 AWS 이벤트로 S3 Object Created 이벤트를 선택했다고 가정해 보겠습니다. 그러면 템플릿에서 사용할 다음 변수를 정의할 수 있습니다.

```
{
```

```
  "requester": "$.detail.requester",
  "key": "$.detail.object.key",
  "bucket": "$.detail.bucket.name"
}
```

(선택 사항) 복사를 선택하여 입력 변환기를 디바이스의 클립보드에 복사할 수도 있습니다.

7. 템플릿 섹션에서 대상으로 EventBridge 전달되는 내용을 결정하는 데 사용할 템플릿을 작성합니다.

   JSON, 문자열, 정적 정보, 정의한 변수 및 예약된 변수를 사용할 수 있습니다. 입력 변환의 자세한 예는 ??? 섹션을 참조하세요.

   예를 들어 이전 예에서 변수를 정의했다고 가정하겠습니다. 그러면 해당 변수, 예약된 변수 및 정적 정보를 참조하는 다음 템플릿을 작성할 수 있습니다.

```
{
    "message": "<requester> has created the object \"<key>\" in the bucket
 \"<bucket>\"",
    "RuleName": <aws.events.rule-name>,
    "ruleArn" : <aws.events.rule-arn>,
    "Transformed": "Yes"
}
```

   (선택 사항) 복사를 선택하여 템플릿을 디바이스의 클립보드에 복사할 수도 있습니다.

8. 템플릿을 테스트하려면 출력 생성을 선택합니다.

   EventBridge 입력 템플릿을 기반으로 샘플 이벤트를 처리하고 출력에서 생성된 변환된 출력을 표시합니다. 이 정보는 원래 소스 이벤트 대신 대상에 EventBridge 전달될 정보입니다.

   위에서 설명한 예시 입력 템플릿에 대해 생성된 출력은 다음과 같습니다.

```
{
    "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"
}
```

   (선택 사항) 복사를 선택하여 생성된 출력을 디바이스의 클립보드에 복사할 수도 있습니다.

9. 확인을 선택합니다.
10. ???에 설명된 대로 규칙을 생성하는 나머지 단계를 따르세요.

EventBridge 샌드박스를 사용하여 타겟 입력 트랜스포머를 테스트합니다.

정보를 규칙의 대상으로 EventBridge 전달하기 전에 입력 변환기를 사용하여 이벤트의 텍스트를 사용자 지정할 수 있습니다.

일반적으로 입력 변환기 구성은 새 규칙을 생성하거나 기존 규칙을 편집할 때 대상을 지정하는 대규모 프로세스의 일부입니다. 그러나 에서 EventBridge 샌드박스를 사용하면 규칙을 만들거나 편집할 필요 없이 입력 트랜스포머를 빠르게 구성하고 샘플 이벤트를 사용하여 원하는 출력을 얻는지 확인할 수 있습니다.

입력 변환에 대한 자세한 내용은 ??? 섹션을 참조하세요.

대상 입력 변환기를 테스트하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 개발자 리소스에서 샌드박스를 선택하고 샌드박스 페이지에서 대상 입력 변환기 탭을 선택합니다.
3. 샘플 이벤트 섹션에서 이벤트 패턴을 테스트할 샘플 이벤트 유형을 선택합니다. AWS 이벤트 또는 파트너 이벤트를 선택하거나 사용자 지정 이벤트를 직접 입력할 수 있습니다.

   AWS events

   지원되는 AWS 서비스에서 발생한 이벤트 중에서 선택합니다.

   1. AWS 이벤트를 선택합니다.
   2. 샘플 이벤트에서 원하는 AWS 이벤트를 선택합니다. 이벤트는 AWS 서비스별로 구성됩니다.

      이벤트를 선택하면 샘플 이벤트가 EventBridge 채워집니다.

      예를 들어, S3 객체 생성을 선택하면 샘플 S3 객체 생성 이벤트가 EventBridge 표시됩니다.
   3. (선택 사항) 복사를 선택하여 샘플 이벤트를 디바이스의 클립보드에 복사할 수도 있습니다.

   Partner events

   지원하는 타사 서비스 (예: Salesforce) 에서 EventBridge 생성된 이벤트 중에서 선택하십시오.

1. 파트너 이벤트를 선택하세요EventBridge .
2. 샘플 이벤트에서 원하는 파트너 이벤트를 선택합니다. 이벤트는 파트너별로 구성됩니다.

   이벤트를 선택하면 샘플 이벤트가 EventBridge 채워집니다.
3. (선택 사항) 복사를 선택하여 샘플 이벤트를 디바이스의 클립보드에 복사할 수도 있습니다.

Enter your own

JSON 텍스트로 자체 이벤트를 입력합니다.

1. 직접 입력을 선택합니다.
2. EventBridge 필수 이벤트 속성의 템플릿으로 샘플 이벤트를 채웁니다.
3. 필요에 따라 샘플 이벤트를 편집하고 추가합니다. 샘플 이벤트는 유효한 JSON이어야 합니다.
4. (선택 사항) 다음 옵션 중 하나를 선택할 수도 있습니다.
   - 복사 - 샘플 이벤트를 디바이스의 클립보드에 복사합니다.
   - 정리 - 줄 바꿈, 탭, 공백을 추가하여 JSON 텍스트를 더 쉽게 읽을 수 있습니다.

4. (선택 사항) 예시 입력 경로, 템플릿 및 출력 섹션을 확장하여 다음 예를 확인합니다.
   - JSON 경로를 사용하여 이벤트 데이터를 나타내는 변수를 정의하는 방법
   - 입력 변환기 템플릿에서 해당 변수를 사용하는 방법
   - 타겟으로 EventBridge 전송되는 결과 출력입니다.

   입력 변환의 자세한 예는 ??? 섹션을 참조하세요.
5. 대상 입력 변환기 섹션에서 입력 템플릿에 사용할 변수를 정의합니다.

   변수는 JSON 경로를 사용하여 원래 이벤트 소스의 값을 참조합니다. 그런 다음 입력 템플릿에서 해당 변수를 참조하여 대상으로 EventBridge 전달되는 변환된 이벤트에 원본 소스 이벤트의 데이터를 포함시킬 수 있습니다. 최대 100개의 변수를 정의할 수 있습니다. 입력 변환기는 유효한 JSON이어야 합니다.

   예를 들어 이 입력 변환기의 샘플 AWS 이벤트로 S3 Object Created 이벤트를 선택했다고 가정해 보겠습니다. 그러면 템플릿에서 사용할 다음 변수를 정의할 수 있습니다.

```
{
```

```
  "requester": "$.detail.requester",
  "key": "$.detail.object.key",
  "bucket": "$.detail.bucket.name"
}
```

(선택 사항) 복사를 선택하여 입력 변환기를 디바이스의 클립보드에 복사할 수도 있습니다.

6. 템플릿 섹션에서 대상으로 EventBridge 전달되는 내용을 결정하는 데 사용할 템플릿을 작성합니다.

   JSON, 문자열, 정적 정보, 정의한 변수 및 예약된 변수를 사용할 수 있습니다. 입력 변환의 자세한 예는 ??? 섹션을 참조하세요.

   예를 들어 이전 예에서 변수를 정의했다고 가정하겠습니다. 그러면 해당 변수, 예약된 변수 및 정적 정보를 참조하는 다음 템플릿을 작성할 수 있습니다.

```
{
    "message": "<requester> has created the object \"<key>\" in the bucket
 \"<bucket>\"",
    "RuleName": <aws.events.rule-name>,
    "ruleArn" : <aws.events.rule-arn>,
    "Transformed": "Yes"
}
```

   (선택 사항) 복사를 선택하여 템플릿을 디바이스의 클립보드에 복사할 수도 있습니다.

7. 템플릿을 테스트하려면 출력 생성을 선택합니다.

   EventBridge 입력 템플릿을 기반으로 샘플 이벤트를 처리하고 출력에서 생성된 변환된 출력을 표시합니다. 이 정보는 원래 소스 이벤트 대신 대상에 EventBridge 전달될 정보입니다.

   위에서 설명한 예시 입력 템플릿에 대해 생성된 출력은 다음과 같습니다.

```
{
    "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"
}
```

   (선택 사항) 복사를 선택하여 생성된 출력을 디바이스의 클립보드에 복사할 수도 있습니다.

Amazon EventBridge 아카이브 및 재생

EventBridge에서는 이벤트 아카이브를 생성하여 나중에 쉽게 이벤트를 재생할 수 있습니다. 예를 들어 이벤트를 재생하여 오류를 복구하거나 애플리케이션의 새 기능을 검증할 수 있습니다.

> **Note**
>
> 이벤트 버스에 게시되는 이벤트와 아카이브에 도착하는 이벤트 사이에 지연이 발생할 수 있습니다. 모든 이벤트가 재생되도록 보관된 이벤트의 재생을 10분 동안 연기하는 것이 좋습니다.

다음 비디오에서는 아카이브 및 재생 사용에 대해 보여줍니다. 아카이브 및 재생 생성

주제

- 아마존 EventBridge 이벤트 보관
- 보관된 Amazon EventBridge 이벤트 재생

아마존 EventBridge 이벤트 보관

에서 EventBridge 아카이브를 생성할 때 이벤트 패턴을 지정하여 아카이브로 전송할 이벤트를 결정할 수 있습니다. EventBridge 이벤트 패턴과 일치하는 이벤트를 아카이브로 보냅니다. 또한 이벤트가 폐기되기 전에 아카이브에 이벤트를 저장하도록 보존 기간을 설정합니다.

기본적으로 AWS 자체 CMK에 따른 256비트 고급 암호화 표준 (AES-256) 을 사용하여 아카이브의 이벤트 데이터를 EventBridge 암호화하므로 무단 액세스로부터 데이터를 보호하는 데 도움이 됩니다.

> **Note**
>
> `DescribeArchive`작업의 `EventCount` 및 `SizeBytes` 값의 조정 기간은 24시간입니다. 따라서 최근에 만료되었거나 새로 보관된 이벤트는 이 값에 즉시 반영되지 않을 수 있습니다.

모든 이벤트에 대한 아카이브를 생성하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 아카이브를 선택합니다.
3. 아카이브 생성을 선택합니다.
4. 아카이브 세부 정보에서 아카이브의 이름을 입력합니다. 이 이름은 선택한 리전의 계정에 대해 고유해야 합니다.

   아카이브를 생성한 후에는 이름을 변경할 수 없습니다.
5. (선택 사항) 아카이브에 대한 설명을 입력합니다.
6. 소스의 경우 아카이브로 전송할 이벤트를 내보내는 이벤트 버스를 선택합니다.
7. 보존 기간에서 다음 중 하나를 수행합니다.
   - 이벤트를 아카이브에 보존하고 삭제하지 않으려면 무기한을 선택합니다.
   - 이벤트를 보존할 일수를 입력합니다. 지정된 일수가 지나면 아카이브에서 이벤트를 EventBridge 삭제합니다.
8. 다음을 선택합니다.
9. 이벤트 패턴에서 이벤트 필터링 없음을 선택합니다.
10. 아카이브 생성을 선택합니다.

이벤트 패턴이 있는 아카이브를 생성하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 아카이브를 선택합니다.
3. 아카이브 생성을 선택합니다.
4. 아카이브 세부 정보에서 아카이브의 이름을 입력합니다. 이 이름은 선택한 리전의 계정에 대해 고유해야 합니다.

   아카이브를 생성한 후에는 이름을 변경할 수 없습니다.
5. (선택 사항) 아카이브에 대한 설명을 입력합니다.
6. 소스의 경우 아카이브로 전송할 이벤트를 내보내는 이벤트 버스를 선택합니다.
7. 보존 기간에서 다음 중 하나를 수행합니다.
   - 이벤트를 아카이브에 보존하고 삭제하지 않으려면 무기한*을 선택합니다.
   - 이벤트를 보존할 일수를 입력합니다. 지정된 일수가 지나면 아카이브에서 이벤트를 EventBridge 삭제합니다.
8. 다음을 선택합니다.
9. 이벤트 패턴에서 이벤트 패턴 일치를 기준으로 이벤트 필터링을 선택합니다.
10. 다음 중 하나를 수행합니다.
    - 패턴 빌더를 선택한 다음, 서비스 공급자를 선택합니다. AWS를 선택하는 경우 패턴에 사용할 AWS 서비스 이름과 이벤트 유형도 선택합니다.
    - JSON 편집기를 선택하여 패턴을 수동으로 생성합니다. 규칙에서 패턴을 복사한 다음, JSON 편집기에 붙여넣을 수도 있습니다.
11. 아카이브 생성을 선택합니다.

이벤트가 아카이브로 성공적으로 전송되었는지 확인하려면 EventBridge API `DescribeArchive`작업을 사용하여 아카이브의 이벤트 수가 `EventCount` 반영되는지 확인할 수 있습니다. 값이 0이면 아카이브에 이벤트가 없는 것입니다.

보관된 Amazon EventBridge 이벤트 재생

아카이브를 생성한 후 아카이브에서 이벤트를 재생할 수 있습니다. 예를 들어 추가 기능으로 애플리케이션을 업데이트하는 경우, 이벤트가 재처리되도록 과거 이벤트를 재생하여 애플리케이션의 일관성을 유지할 수 있습니다. 아카이브를 이용해 이벤트를 재생하여 새 기능을 사용할 수도 있습니다. 이벤트를 재생할 때 이벤트를 재생할 아카이브, 이벤트를 재생할 시작 및 종료 시간, 이벤트 버스 또는 이벤트를 재생할 하나 이상의 규칙을 지정할 수 있습니다.

이벤트는 아카이브에 추가된 순서대로 재생되지 않을 수도 있습니다. 재생 시 이벤트 시간을 기준으로 재생할 이벤트를 처리하고 1분 간격으로 재생합니다. 이벤트 시작 시간과 이벤트 종료 시간을 20분 범위로 지정하는 경우 이벤트는 먼저 해당 20분 범위 중 처음 1분부터 재생됩니다. 그런 다음, 두 번째 1분의 이벤트가 재생됩니다. EventBridge API의 `DescribeReplay` 작업을 사용하여 재생 진행 상황을 확인할 수 있습니다. `EventLastReplayedTime`은(는) 재생된 마지막 이벤트의 타임스탬프를 반환합니다.

이벤트는 AWS 계정의 초당 `PutEvents` 트랜잭션 한도를 기준으로 하되, 이와는 별도로 재생됩니다. PutEvents의 한도 증가를 요청할 수 있습니다. 자세한 내용은 Amazon EventBridge 할당량을 참조하세요.

> **Note**
>
> AWS 리전별 계정당 활성 동시 재생을 최대 10개 보유할 수 있습니다.

이벤트 재생을 시작하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 재생을 선택합니다.
3. 다시 재생 새로 시작을 선택합니다.
4. 재생의 이름을 입력하고 선택적으로 설명을 입력합니다.
5. 소스에서 이벤트를 재생할 아카이브를 선택합니다.
6. 대상의 경우 이벤트가 발생된 동일한 이벤트 버스에만 이벤트를 재생할 수 있습니다.
7. 규칙 지정에 대해 다음 중 하나를 수행합니다.
   - 이벤트를 모든 규칙에 따라 재생하려면 모든 규칙을 선택합니다.
   - 규칙 지정을 선택한 다음, 이벤트를 재생할 규칙을 하나 또는 여러 개 선택합니다.

8. 다시 재생 기간에서 시작 시간과 종료 시간의 날짜, 시간 및 시간대를 지정합니다. 시작 시간과 종료 시간 사이에 발생한 이벤트만 재생됩니다.
9. 재생 시작을 선택합니다.

보관된 이벤트가 재생되면 재생 상태는 완료됨입니다.

재생을 시작한 다음, 중단하려는 경우 상태가 시작 중 또는 실행 중이면 취소할 수 있습니다.

재생을 취소하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 재생을 선택합니다.
3. 취소할 재생을 선택합니다.
4. 취소를 선택합니다.

아마존 EventBridge 파이프

Amazon EventBridge Pipes는 소스를 타겟에 연결합니다. 파이프는 고급 변환 및 강화를 지원하여 지원되는 소스와대상 간의 point-to-point 통합을 위한 것입니다. 이벤트 기반 아키텍처를 개발할 때 전문 지식 및 통합 코드의 필요성이 줄여 회사 애플리케이션 전반의 일관성을 높입니다. 파이프를 설정하려면 소스를 선택하고, 선택적 필터링을 추가하고, 선택적 보강을 정의하고, 이벤트 데이터의 대상을 선택합니다.

> **Note**
>
> 이벤트 버스를 사용하여 이벤트를 라우팅할 수도 있습니다. 이벤트 버스는 이벤트 기반 서비스 간에 이벤트를 many-to-many 라우팅하는 데 매우 적합합니다. 자세한 정보는 ???을 참조하세요.

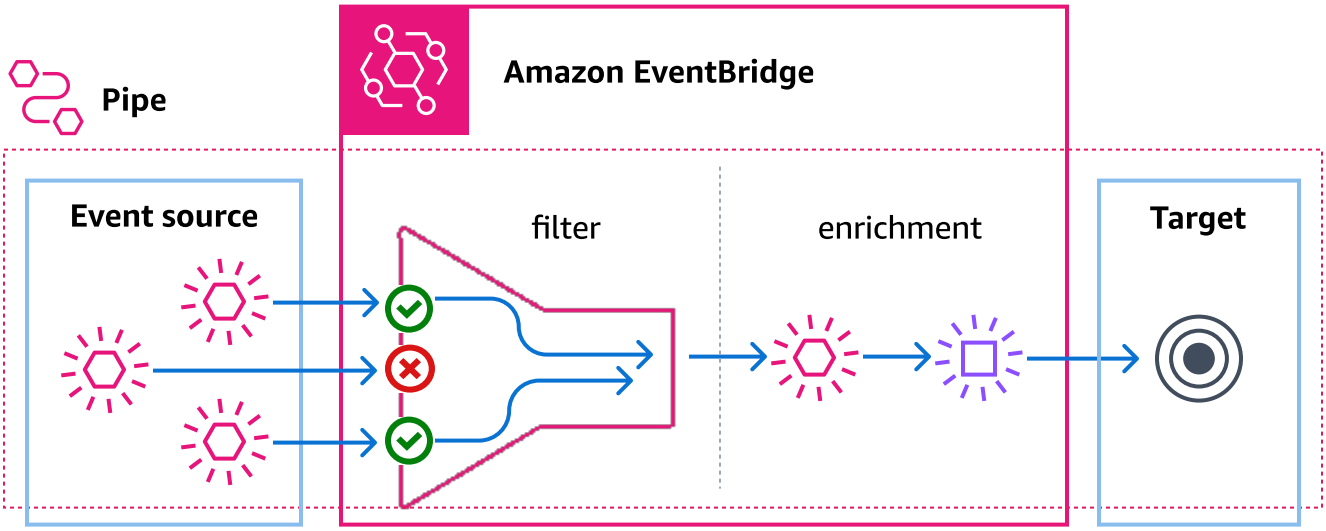
파이프 작동 방식 EventBridge

개괄적으로 EventBridge 파이프의 작동 원리는 다음과 같습니다.

1. 계정에서 파이프를 생성합니다. 여기에는 다음이 포함됩니다.
   - 파이프에서 이벤트를 수신할 지원되는 이벤트 소스 중 하나를 지정합니다.
   - 선택적으로 파이프가 소스로부터 수신한 이벤트의 하위 집합만 처리하도록 필터를 구성할 수 있습니다.
   - 선택적으로 이벤트 데이터를 대상으로 전송하기 전에 이벤트 데이터를 보강하는 보강 단계를 구성할 수 있습니다.
   - 파이프에서 이벤트를 전송할 지원되는 대상 중 하나를 지정합니다.
2. 이벤트 소스는 파이프로 이벤트를 보내기 시작하고 파이프는 이벤트를 처리한 다음 대상으로 전송합니다.
   - 필터를 구성한 경우 파이프는 이벤트를 평가하여 해당 필터와 일치하는 경우에만 대상으로 이벤트를 보냅니다.

     필터와 일치하는 이벤트에 대해서만 요금이 부과됩니다.
   - 보강을 구성한 경우 파이프는 이벤트를 대상으로 보내기 전에 이벤트에 대해 보강을 수행합니다.

     이벤트가 일괄 처리되는 경우 보강을 통해 일괄 처리에서 이벤트 순서가 유지됩니다.



예를 들어 파이프를 사용하여 전자 상거래 시스템을 만들 수 있습니다. 배송 주소와 같은 고객 정보가 포함된 API가 있다고 가정해 보겠습니다.

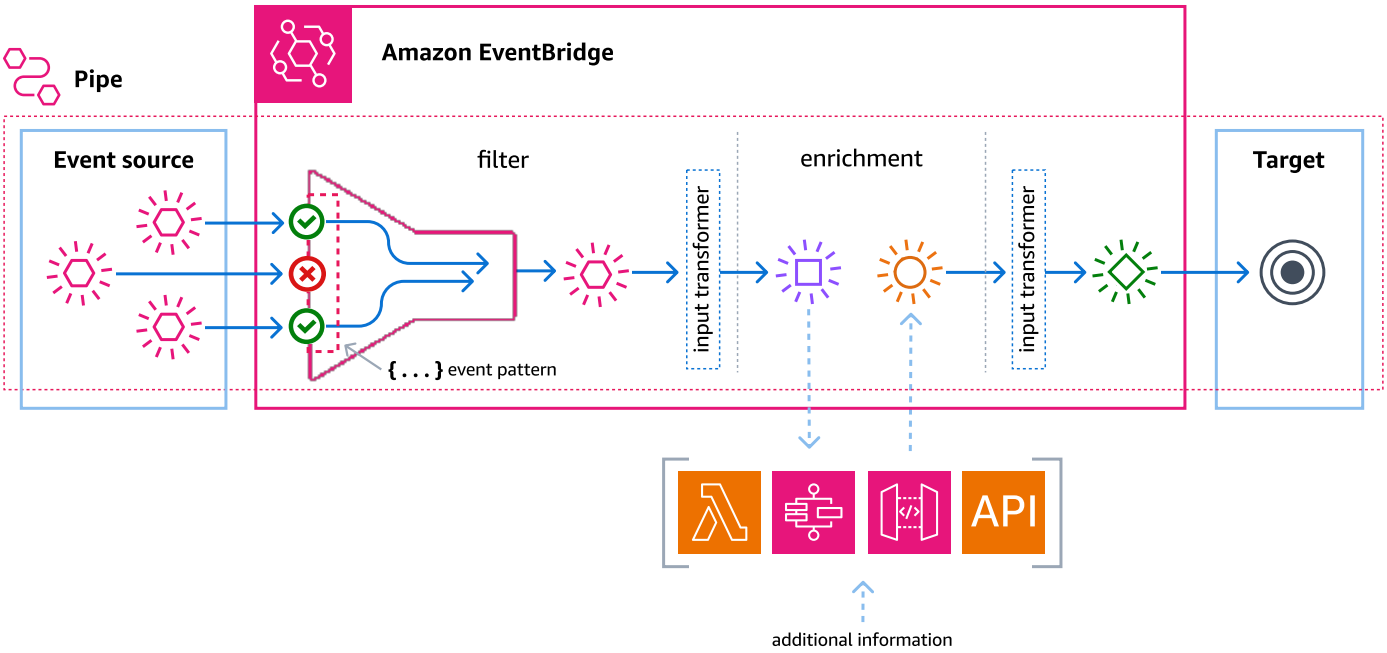
1. 그런 다음 다음을 사용하여 파이프를 생성합니다.
   - Amazon SQS 주문 수신 메시지 대기열을 이벤트 소스로 사용
   - 보강으로서의 EventBridge API 데스티네이션
   - 타겟이 되는 AWS Step Functions 스테이트 머신
2. 그런 다음 Amazon SQS 주문 수신 메시지가 대기열에 나타나면 해당 메시지가 파이프로 전송됩니다.
3. 그러면 파이프가 해당 데이터를 API Destination Enrichment로 전송하고, EventBridge API Destination Enrichment는 해당 주문에 대한 고객 정보를 반환합니다.
4. 마지막으로 파이프는 보강된 데이터를 AWS Step Functions 스테이트 머신으로 전송하고, 스테이트 머신은 주문을 처리합니다.

EventBridge 파이프 개념

EventBridge 파이프의 기본 구성 요소를 자세히 살펴보겠습니다.

파이프

파이프는 단일 소스에서 단일 대상으로 이벤트를 라우팅합니다. 파이프에는 특정 이벤트를 필터링하고 이벤트 데이터가 대상으로 전송되기 전에 이벤트 데이터를 보강하는 기능도 포함되어 있습니다.



소스

EventBridge Pipes는 다양한 소스로부터 이벤트 데이터를 수신하고 해당 데이터에 선택적 필터 및 보강을 적용한 다음 이를 대상으로 보냅니다. 소스가 파이프로 전송된 이벤트에 순서를 적용하는 경우 해당 순서는 대상에 도달하는 전체 프로세스에서 유지됩니다.

소스에 대한 자세한 내용은 ??? 단원을 참조하십시오.

필터

파이프는 특정 소스의 이벤트를 필터링한 다음 해당 이벤트의 일부만 처리할 수 있습니다. 파이프에 필터링을 구성하려면 파이프가 대상으로 전송할 이벤트를 결정하는 데 사용하는 이벤트 패턴을 정의합니다.

필터와 일치하는 이벤트에 대해서만 요금이 부과됩니다.

자세한 정보는 ???을 참조하세요.

보강

EventBridge Pipes의 보강 단계를 사용하면 타겟으로 데이터를 보내기 전에 소스의 데이터를 개선할 수 있습니다. 예를 들어 전체 티켓 데이터가 포함되지 않은 티켓 생성 이벤트를 수신할 수 있습니다. 보강을 사용하면 Lambda 함수로 `get-ticket` API를 호출하여 전체 티켓 세부 정보를 확인할 수 있습니다. 그 후 파이프가 해당 정보를 대상으로 전송할 수 있습니다.

이벤트 데이터 보강에 대한 자세한 내용은 ??? 단원을 참조하세요.

대상

이벤트 데이터를 필터링하고 보강한 후 파이프가 Amazon Kinesis 스트림 또는 Amazon 로그 그룹과 같은 특정 대상으로 전송하도록 지정할 수 있습니다. CloudWatch 사용 가능한 대상 목록은 ??? 단원을 참조하세요.

데이터가 개선된 후와 파이프를 통해 대상으로 전송되기 전에 데이터를 변환할 수 있습니다. 자세한 정보는 ???을 참조하세요.

각각 소스가 다른 여러 파이프는 동일한 대상으로 이벤트를 전송할 수 있습니다.

파이프와 이벤트 버스를 함께 사용하여 여러 대상에 이벤트를 보낼 수도 있습니다. 일반적인 사용 사례는 이벤트 버스를 대상으로 하는 파이프를 만드는 것입니다. 이 파이프는 이벤트를 이벤트 버스로 보낸 다음 이벤트 버스가 해당 이벤트를 여러 대상으로 전송합니다. 예를 들어, 소스용 DynamoDB 스트림과 대상 이벤트 버스를 사용하여 파이프를 생성할 수 있습니다. 파이프는 DynamoDB 스트림에서 이벤트를 수신하여 이벤트 버스로 전송합니다. 그런 다음 이벤트 버스에서 지정한 규칙에 따라 이벤트를 여러 대상으로 전송합니다.

Amazon EventBridge 파이프에 대한 권한

파이프를 설정할 때 기존 실행 역할을 사용하거나 EventBridge에서 필요한 권한이 있는 실행 역할을 생성하도록 할 수 있습니다. EventBridge 파이프에 필요한 권한은 소스 유형에 따라 다르며 아래에 나열되어 있습니다. 실행 역할을 직접 설정하는 경우 이러한 권한을 직접 추가해야 합니다.

> **Note**
>
> 소스에 액세스하는 데 필요한 권한이 정확히 어느 정도인지 확실하지 않은 경우 EventBridge 파이프 콘솔을 사용하여 새 역할을 만든 다음 정책에 나열된 작업을 검사합니다.

주제

- DynamoDB 실행 역할 권한
- Kinesis 실행 역할 권한
- Amazon MQ 실행 역할 권한
- Amazon MSK 실행 역할 권한
- 자체 관리형 Apache Kafka 실행 역할 권한

- Amazon SQS 실행 역할 권한
- 보강 및 대상 권한

DynamoDB 실행 역할 권한

DynamoDB 스트림의 경우 DynamoDB 데이터 스트림과 관련된 리소스를 관리하려면 EventBridge 파이프에 다음 권한이 필요합니다.

- `dynamodb:DescribeStream`
- `dynamodb:GetRecords`
- `dynamodb:GetShardIterator`
- `dynamodb:ListStreams`

실패한 배치의 레코드를 파이프 DLQ(Dead Letter Queue)로 보내려면 파이프 실행 역할에 다음 권한이 필요합니다.

- `sqs:SendMessage`

Kinesis 실행 역할 권한

Kinesis의 경우 Kinesis 데이터 스트림과 관련된 리소스를 관리하려면 EventBridge 파이프에 다음 권한이 필요합니다.

- `kinesis:DescribeStream`
- `kinesis:DescribeStreamSummary`
- `kinesis:GetRecords`
- `kinesis:GetShardIterator`
- `kinesis:ListShards`
- `kinesis:ListStreams`
- `kinesis:SubscribeToShard`

실패한 배치의 레코드를 파이프 DLQ(Dead Letter Queue)로 보내려면 파이프 실행 역할에 다음 권한이 필요합니다.

- `sqs:SendMessage`

Amazon MQ 실행 역할 권한

Amazon MQ의 경우, Amazon MQ 메시지 브로커와 관련된 리소스를 관리하기 위해 EventBridge 파이프에 다음 권한이 필요합니다.

- mq:DescribeBroker
- secretsmanager:GetSecretValue
- ec2:CreateNetworkInterface
- ec2:DeleteNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:DescribeSecurityGroups
- ec2:DescribeSubnets
- ec2:DescribeVpcs
- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents

Amazon MSK 실행 역할 권한

Amazon MSK의 경우 Amazon MSK 주제와 관련된 리소스를 관리하려면 EventBridge에 다음 권한이 필요합니다.

> **Note**
>
> IAM 역할 기반 인증을 사용하는 경우 실행 역할에는 아래 나열된 권한 외에 ???에 나열된 권한이 필요합니다.

- kafka:DescribeClusterV2
- kafka:GetBootstrapBrokers
- ec2:CreateNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:DescribeVpcs
- ec2:DeleteNetworkInterface

- ec2:DescribeSubnets
- ec2:DescribeSecurityGroups
- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents

자체 관리형 Apache Kafka 실행 역할 권한

자체 관리형 Apache Kafka의 경우 자체 관리형 Apache Kafka 스트림과 관련된 리소스를 관리하기 위해 EventBridge에 다음과 같은 권한이 필요합니다.

필요한 권한

Amazon CloudWatch Logs의 로그 그룹에 로그를 생성하고 저장하려면 파이프의 실행 역할에 다음 권한이 있어야 합니다.

- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents

선택적 권한

파이프에는 다음 권한이 필요할 수도 있습니다.

- Secrets Manager 비밀 정보를 설명합니다.
- AWS Key Management Service(AWS KMS) 고객 관리형 키에 액세스합니다.
- Amazon VPC에 액세스합니다.

Secrets Manager 및 AWS KMS 권한

Apache Kafka 브로커에 대해 구성하는 액세스 제어 유형에 따라 파이프에는 Secrets Manager 비밀 정보에 액세스하거나 AWS KMS 고객 관리형 키를 복호화할 수 있는 권한이 필요할 수 있습니다. 리소스에 액세스하려면 함수의 실행 역할에 다음 권한이 주어져야 합니다.

- secretsmanager:GetSecretValue
- kms:Decrypt

VPC 권한

VPC 내의 사용자만 자체 관리형 Apache Kafka 클러스터에 액세스할 수 있는 경우 파이프에 Amazon VPC 리소스에 액세스할 수 있는 권한이 있어야 합니다. 이러한 리소스에는 VPC, 서브넷, 보안 그룹 및 네트워크 인터페이스가 있습니다. 리소스에 액세스하려면 파이프의 실행 역할에 다음 권한이 주어져야 합니다.

- `ec2:CreateNetworkInterface`
- `ec2:DescribeNetworkInterfaces`
- `ec2:DescribeVpcs`
- `ec2:DeleteNetworkInterface`
- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`

Amazon SQS 실행 역할 권한

Amazon SQS의 경우 Amazon SQS 대기열과 관련된 리소스를 관리하려면 EventBridge에 다음 권한이 필요합니다.

- `sqs:ReceiveMessage`
- `sqs:DeleteMessage`
- `sqs:GetQueueAttributes`

보강 및 대상 권한

소유하고 있는 리소스에 대해 API 직접 호출을 수행할 수 있으려면 EventBridge 파이프에 적절한 권한이 필요합니다. EventBridge 파이프는 IAM 보안 주체 `pipes.amazonaws.com`을 사용하여 보강 및 대상 호출을 위해 파이프에 지정하는 IAM 역할을 사용합니다.

아마존 EventBridge 파이프 생성

EventBridge Pipes를 사용하면 고급 이벤트 변환 및 강화를 포함하여 소스와 대상 간의 point-to-point 통합을 생성할 수 있습니다. EventBridge 파이프를 생성하려면 다음 단계를 수행합니다.

1. ???
2. ???

3. ???
4. ???
5. ???

CLI를 사용하여 파이프를 생성하는 방법에 대한 자세한 내용은 AWS CLI 명령 참조의 create-pipe 항목을 참조하십시오.AWS

소스 지정

시작하려면 파이프에서 이벤트를 수신할 소스를 지정합니다.

콘솔을 사용하여 파이프 소스를 지정하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 파이프를 선택합니다.
3. 파이프 생성을 선택합니다.
4. 파이프 이름을 입력합니다.
5. (선택 사항) 파이프에 대한 설명을 추가합니다.
6. 파이프 만들기 탭의 소스에서 이 파이프에 지정할 소스 유형을 선택하고 소스를 구성합니다.

   구성 속성은 선택한 소스 유형에 따라 다릅니다.

   Confluent

   콘솔을 사용하여 Confluent Cloud 스트림을 소스로 구성하려면

   1. 소스에서 컨플루언트 클라우드를 선택합니다.
   2. 부트스트랩 서버의 경우 브로커의 `host:port` 쌍 주소를 입력합니다.
   3. 주제 이름에 파이프가 읽을 주제의 이름을 입력합니다.
   4. (선택 사항) VPC에서 원하는 VPC를 선택합니다. 그런 다음, VPC 서브넷에서 원하는 서브넷을 선택합니다. VPC 보안 그룹에서 보안 그룹을 선택합니다.
   5. 인증 (선택 사항) 의 경우 인증 사용을 켜고 다음을 수행하십시오.
      a. 인증 방법에서 인증 유형을 선택합니다.
      b. 비밀 키에서 비밀 키를 선택합니다.

      자세한 내용은 Confluent 설명서에서 Confluent Cloud 리소스에 대한 인증을 참조하십시오.

6. (선택 사항) 추가 설정 - 선택 사항에서 다음을 수행합니다.
   a. 시작 위치에는 다음 중 하나를 선택합니다.
      - 최신 - 샤드에 있는 최근 레코드로 스트림 읽기를 시작합니다.
      - 트림 호라이즌 - 샤드에서 트리밍되지 않은 마지막 레코드로 스트림 읽기를 시작합니다. 샤드에서 가장 오래된 레코드입니다.
   b. 배치 크기 - 선택 사항에는 각 배치의 최대 레코드 수를 입력합니다. 기본 값은 100입니다.
   c. 배치 기간 - 선택 사항의 경우 진행하기 전에 레코드를 수집할 최대 시간(초)을 입력합니다.

DynamoDB

1. 소스에서 DynamoDB를 선택합니다.
2. DynamoDB 스트림의 경우 소스로 사용할 스트림을 선택합니다.
3. 시작 위치에는 다음 중 하나를 선택합니다.
   - 최신 - 샤드에 있는 최근 레코드로 스트림 읽기를 시작합니다.
   - 트림 호라이즌 - 샤드에서 트리밍되지 않은 마지막 레코드로 스트림 읽기를 시작합니다. 샤드에서 가장 오래된 레코드입니다.
4. (선택 사항) 추가 설정 - 선택 사항에서 다음을 수행합니다.
   a. 배치 크기 - 선택 사항에는 각 배치의 최대 레코드 수를 입력합니다. 기본 값은 100입니다.
   b. 배치 기간 - 선택 사항의 경우 진행하기 전에 레코드를 수집할 최대 시간(초)을 입력합니다.
   c. 샤드당 동시 배치 - 선택 사항에는 동시에 읽을 수 있는 동일한 샤드의 배치 수를 입력합니다.
   d. 부분 배치 항목 실패 시의 경우 다음을 선택합니다.
      - AUTOMATIC_BISECT - 각 배치를 반으로 나누고 모든 레코드가 처리되거나 배치에 하나의 실패한 메시지가 남을 때까지 절반마다 다시 시도합니다.

> **Note**
>
> AUTOMATIC_BISECT를 선택하지 않으면 실패한 특정 레코드를 반환할 수 있으며 해당 레코드만 다시 시도됩니다.

Kinesis

콘솔을 사용하여 Kinesis 소스를 구성하려면

1. 소스에서 Kinesis를 선택합니다.
2. Kinesis 스트림의 경우 소스로 사용할 스트림을 선택합니다.
3. 시작 위치에는 다음 중 하나를 선택합니다.
   - 최신 - 샤드에 있는 최근 레코드로 스트림 읽기를 시작합니다.
   - 트림 호라이즌 - 샤드에서 트리밍되지 않은 마지막 레코드로 스트림 읽기를 시작합니다. 샤드에서 가장 오래된 레코드입니다.
   - 타임 스탬프에서 - 지정된 시간부터 스트림 읽기를 시작합니다. 타임스탬프에서 YYYY/MM/DD 및 hh:mm:ss 형식으로 데이터와 시간을 입력합니다.
4. (선택 사항) 추가 설정 - 선택 사항에서 다음을 수행합니다.
   a. 배치 크기 - 선택 사항에는 각 배치의 최대 레코드 수를 입력합니다. 기본 값은 100입니다.
   b. (선택 사항) 배치 기간 - 선택 사항의 경우 진행하기 전에 레코드를 수집할 최대 시간(초)을 입력합니다.
   c. 샤드당 동시 배치 - 선택 사항에는 동시에 읽을 수 있는 동일한 샤드의 배치 수를 입력합니다.
   d. 부분 배치 항목 실패 시의 경우 다음을 선택합니다.
      - AUTOMATIC_BISECT - 각 배치를 반으로 나누고 모든 레코드가 처리되거나 배치에 하나의 실패한 메시지가 남을 때까지 절반마다 다시 시도합니다.

> **Note**
>
> AUTOMATIC_BISECT를 선택하지 않으면 실패한 특정 레코드를 반환할 수 있으며 해당 레코드만 다시 시도됩니다.

Amazon MQ

콘솔을 사용하여 Amazon MQ 소스를 구성하려면

1. 소스에서 Amazon MQ를 선택합니다.

2. Amazon MQ 브로커의 경우 소스로 사용할 스트림을 선택합니다.
3. 대기열 이름에 파이프가 읽을 대기열의 이름을 입력합니다.
4. 인증 방법으로 BASIC_AUTH를 선택합니다.
5. 비밀 키에서 비밀 키를 선택합니다.
6. (선택 사항) 추가 설정 - 선택 사항에서 다음을 수행합니다.
   a. 배치 크기 - 선택 사항에는 각 배치의 최대 메시지 수를 입력합니다. 기본 값은 100입니다.
   b. 배치 기간 - 선택 사항의 경우 진행하기 전에 레코드를 수집할 최대 시간(초)을 입력합니다.

Amazon MSK

콘솔을 사용하여 Amazon MSK 소스를 구성하려면

1. 소스에서 Amazon MSK를 선택합니다.
2. Amazon MSK 클러스터에서 사용하려는 클러스터를 선택합니다.
3. 주제 이름에 파이프가 읽을 주제의 이름을 입력합니다.
4. (선택 사항) 소비자 그룹 ID - 선택 사항에서 파이프를 조인할 소비자 그룹의 ID를 입력합니다.
5. (선택 사항) 인증 - 선택 사항의 경우 인증 사용을 설정하고 다음을 수행합니다.
   a. 인증 방법에서 원하는 유형을 선택합니다.
   b. 비밀 키에서 비밀 키를 선택합니다.
6. (선택 사항) 추가 설정 - 선택 사항에서 다음을 수행합니다.
   a. 배치 크기 - 선택 사항에는 각 배치의 최대 레코드 수를 입력합니다. 기본 값은 100입니다.
   b. 배치 기간 - 선택 사항의 경우 진행하기 전에 레코드를 수집할 최대 시간(초)을 입력합니다.
   c. 시작 위치에는 다음 중 하나를 선택합니다.
      - 최신 - 샤드에 있는 최근 레코드로 주제 읽기를 시작합니다.
      - 트림 호라이즌 - 샤드에서 트리밍되지 않은 마지막 레코드로 주제 읽기를 시작합니다. 샤드에서 가장 오래된 레코드입니다.

> **Note**
>
> 트림 호라이즌은 Apache Kafka의 가장 빠른 항목과 동일합니다.

Self managed Apache Kafka

콘솔을 사용하여 자체 관리형 Apache Kafka 소스를 구성하려면

1. 소스에서 자체 관리형 Apache Kafka를 선택합니다.
2. 부트스트랩 서버의 경우 브로커의 `host:port` 쌍 주소를 입력합니다.
3. 주제 이름에 파이프가 읽을 주제의 이름을 입력합니다.
4. (선택 사항) VPC에서 원하는 VPC를 선택합니다. 그런 다음, VPC 서브넷에서 원하는 서브넷을 선택합니다. VPC 보안 그룹에서 보안 그룹을 선택합니다.
5. (선택 사항) 인증 - 선택 사항의 경우 인증 사용을 설정하고 다음을 수행합니다.
   a. 인증 방법에서 인증 유형을 선택합니다.
   b. 비밀 키에서 비밀 키를 선택합니다.
6. (선택 사항) 추가 설정 - 선택 사항에서 다음을 수행합니다.
   a. 시작 위치에는 다음 중 하나를 선택합니다.
      - 최신 - 샤드에 있는 최근 레코드로 스트림 읽기를 시작합니다.
      - 트림 호라이즌 - 샤드에서 트리밍되지 않은 마지막 레코드로 스트림 읽기를 시작합니다. 샤드에서 가장 오래된 레코드입니다.
   b. 배치 크기 - 선택 사항에는 각 배치의 최대 레코드 수를 입력합니다. 기본 값은 100입니다.
   c. 배치 기간 - 선택 사항의 경우 진행하기 전에 레코드를 수집할 최대 시간(초)을 입력합니다.

Amazon SQS

콘솔을 사용하여 Amazon SQS 소스를 구성하려면

1. 소스에서 SQS를 선택합니다.
2. SQS 대기열의 경우 사용하려는 대기열을 선택합니다.
3. (선택 사항) 추가 설정 - 선택 사항에서 다음을 수행합니다.

a. 배치 크기 - 선택 사항에는 각 배치의 최대 레코드 수를 입력합니다. 기본 값은 100입니다.

b. 배치 기간 - 선택 사항의 경우 진행하기 전에 레코드를 수집할 최대 시간(초)을 입력합니다.

이벤트 필터링 구성(선택 사항)

소스에서 대상으로 이벤트의 하위 집합만 전송하도록 파이프에 필터링을 추가할 수 있습니다.

콘솔을 사용하여 필터링을 구성하려면

1. 필터링을 선택합니다.
2. 샘플 이벤트 - 선택 사항에는 이벤트 패턴을 작성하는 데 사용할 수 있는 샘플 이벤트가 표시되거나 직접 입력을 선택하여 자체 이벤트를 입력할 수 있습니다.
3. 이벤트 패턴에서 이벤트를 필터링하는 데 사용할 이벤트 패턴을 입력합니다. 필터 구성에 대한 자세한 내용은 을 참조하십시오. ???

   다음은 City 필드에 Seattle 값이 있는 이벤트만 보내는 이벤트 패턴의 예입니다.

```
{
  "data": {
    "City": ["Seattle"]
  }
}
```

이제 이벤트가 필터링되었으므로 파이프에 선택적 보강과 대상을 추가할 수 있습니다.

이벤트 보강 정의(선택 사항)

보강을 위해 Lambda 함수, 상태 머신, AWS Step Functions Amazon API Gateway 또는 API 대상으로 이벤트 데이터를 전송할 수 있습니다.

보강을 선택하려면

1. 보강을 선택합니다.
2. 세부 정보의 서비스에서 보강에 사용할 서비스 및 관련 설정을 선택합니다.

데이터를 개선하도록 전송하기 전에 데이터를 변환할 수도 있습니다.

(선택 사항) 입력 변환기를 정의하려면

1. 보강 입력 변환기 - 선택 사항을 선택합니다.
2. 샘플 이벤트/이벤트 페이로드에서 샘플 이벤트 유형을 선택합니다.
3. 변환기에는 "Event happened at <$.detail.field>."와 같은 변환기 구문을 입력합니다. 여기서 <$.detail.field>는 샘플 이벤트의 필드에 대한 참조입니다. 샘플 이벤트의 필드를 두 번 클릭하여 변환기에 추가할 수도 있습니다.
4. 출력의 경우 출력이 원하는 대로 표시되는지 확인합니다.

이제 데이터가 필터링되고 개선되었으므로 이벤트 데이터를 전송할 대상을 정의해야 합니다.

대상 구성

대상을 구성하려면

1. [Target]을 선택합니다.
2. 세부 정보의 대상 서비스에서 대상을 선택합니다. 표시되는 필드는 선택한 대상에 따라 달라집니다. 필요에 따라 이 대상 유형과 관련된 정보를 입력합니다.

데이터를 대상으로 전송하기 전에 데이터를 변환할 수도 있습니다.

(선택 사항) 입력 변환기를 정의하려면

1. 대상 입력 변환기 - 선택 사항을 선택합니다.
2. 샘플 이벤트/이벤트 페이로드에서 샘플 이벤트 유형을 선택합니다.
3. 변환기에는 "Event happened at <$.detail.field>."와 같은 변환기 구문을 입력합니다. 여기서 <$.detail.field>는 샘플 이벤트의 필드에 대한 참조입니다. 샘플 이벤트의 필드를 두 번 클릭하여 변환기에 추가할 수도 있습니다.
4. 출력의 경우 출력이 원하는 대로 표시되는지 확인합니다.

이제 파이프가 구성되었으므로 해당 설정이 올바르게 구성되었는지 확인하세요.

파이프 설정 구성

파이프는 기본적으로 활성화되지만 비활성화할 수 있습니다. 파이프에 대한 권한을 지정하고, 파이프 로깅을 설정하고, 태그를 추가할 수도 있습니다.

파이프 설정을 구성하려면

1. 파이프 설정 탭을 선택합니다.
2. 기본적으로 새로 생성된 파이프는 생성되는 즉시 활성화됩니다. 비활성 파이프를 생성하려면 활성화의 파이프 활성화에서 활성을 끕니다.
3. 권한의 실행 역할에서 다음 중 하나를 수행합니다.

   a. 이 파이프에 대한 새 실행 역할을 EventBridge 생성하려면 이 특정 리소스에 대한 새 역할 생성을 선택하십시오. 역할 이름에서 선택적으로 역할 이름을 편집할 수 있습니다.

   b. 기존 실행 역할을 사용하려면 기존 역할 사용을 선택합니다. 역할 이름에서 역할을 선택합니다.
4. (선택 사항) a Kinesis 또는 DynamoDB 스트림을 파이프 소스로 지정한 경우 재시도 정책과 데드 레터 큐 (DLQ) 를 구성할 수 있습니다.

   재시도 정책 및 DLQ(Dead Letter Queue) - 선택 사항에서 다음을 수행합니다.

   재시도 정책에서 다음을 수행합니다.

   a. 재시도 정책을 설정하려면 재시도를 켭니다. 기본적으로 새로 생성된 파이프에는 재시도 정책이 설정되어 있지 않습니다.

   b. 최대 이벤트 기간(Maximum age of event)에 1분(00:01)에서 24시간(24:00) 사이의 값을 입력합니다.

   c. 재시도(Retry attempts)에 0에서 185 사이의 숫자를 입력합니다.

   d. DLQ(Dead Letter Queue)를 사용하려면 DLQ(Dead Letter Queue)를 설정하고 원하는 방법을 선택한 다음, 사용할 대기열이나 주제를 선택합니다. 기본적으로 새로 생성된 파이프는 DLQ를 사용하지 않습니다.
5. (선택 사항) 로그 - 선택 사항에서 해당 로그를 구성하는 방법을 비롯해 EventBridge 파이프가 지원되는 서비스에 로깅 정보를 보내는 방법을 설정할 수 있습니다.

   파이프 레코드 로깅에 대한 자세한 내용은 ??? 섹션을 참조하세요.

CloudWatch 로그는 로그 수준과 마찬가지로 기본적으로 로그 대상으로 선택됩니다. ERROR 따라서 EventBridge Pipes는 기본적으로 세부 ERROR 수준이 포함된 로그 레코드를 보내는 새 CloudWatch 로그 그룹을 생성합니다.

EventBridge Pipes가 지원되는 로그 대상 중 하나로 로그 레코드를 보내도록 하려면 다음과 같이 하십시오.

a. 로그 - 선택 사항에서 로그 레코드를 전송할 대상을 선택합니다.

b. 로그 레벨의 경우 로그 EventBridge 레코드에 포함할 정보 수준을 선택합니다. ERROR 로그 수준이 기본적으로 선택됩니다.

   자세한 정보는 ???을 참조하세요.

c. 이벤트 페이로드 정보와 서비스 요청 및 응답 정보를 로그 레코드에 EventBridge 포함하려면 실행 데이터 포함을 선택합니다.

   자세한 정보는 ???을 참조하세요.

d. 선택한 각 로그 대상을 구성합니다.

   CloudWatch Logs 로그의 경우 CloudWatch 로그에서 다음을 수행하십시오.

   - CloudWatch 로그 그룹의 경우 새 로그 그룹을 EventBridge 만들지 여부를 선택하거나 기존 로그 그룹을 선택하거나 기존 로그 그룹의 ARN을 지정할 수 있습니다.
   - 새 로그 그룹의 경우 원하는 대로 로그 그룹 이름을 편집합니다.

   CloudWatch 로그는 기본적으로 선택됩니다.

   Firehose 스트림 로그의 경우 Firehose 스트림 로그에서 Firehose 스트림을 선택합니다.

   Amazon S3 로그의 경우 S3 로그에서 다음을 수행하십시오.

   - 로그 대상으로 사용할 버킷 이름을 입력합니다.
   - 버킷 소유자의 AWS 계정 ID를 입력합니다.
   - EventBridge 가 S3 객체를 생성할 때 사용할 접두사 텍스트를 입력합니다.

     자세한 내용은 Amazon Simple Storage Service 사용 설명서에서 접두어를 사용한 객체 구성을 참조하세요.

   - S3 로그 레코드를 EventBridge 포맷할 방법을 선택합니다.

- `json`: JSON
- `plain`: 일반 텍스트
- `w3c`: W3C 확장 로깅 파일 형식

6. (선택 사항) 태그 - 선택 사항에서 새 태그 추가를 선택하고 규칙에 대해 하나 이상의 태그를 입력합니다. 자세한 정보는 ???을 참조하세요.
7. 파이프 생성을 선택합니다.

구성 파라미터 검증

파이프가 생성된 후 다음 구성 파라미터를 EventBridge 검증합니다.

- IAM 역할 - 파이프가 생성된 후에는 파이프의 소스를 변경할 수 없으므로 제공된 IAM 역할이 소스에 액세스할 수 EventBridge 있는지 확인합니다.

> **Note**
>
> EventBridge 인리치먼트나 타겟은 파이프가 생성된 후에 업데이트될 수 있으므로 동일한 검증을 수행하지 않습니다.

- 일괄 처리 — 소스의 배치 크기가 대상의 최대 배치 크기를 초과하지 않는지 EventBridge 검증합니다. 그럴 경우 배치 크기를 EventBridge 줄여야 합니다. 또한 대상이 일괄 처리를 지원하지 않는 경우 소스에 EventBridge 대해 일괄 처리를 구성할 수 없습니다.
- 보강 — 배치 크기 1만 지원되므로 API Gateway 및 API 대상 강화의 배치 크기가 1인지 EventBridge 확인합니다.

파이프 시작 또는 중지

기본적으로 파이프는 `Running`이며 생성 시 이벤트를 처리합니다.

Amazon SQS, Kinesis 또는 DynamoDB 소스로 파이프를 생성하는 경우 파이프를 생성하는 데 보통 1~2분 정도 걸릴 수 있습니다.

Amazon MSK, 자체 관리형 Apache Kafka 또는 Amazon MQ 소스를 사용하여 파이프를 생성하는 경우 파이프를 생성하는 데 최대 10분이 걸릴 수 있습니다.

콘솔을 사용하여 이벤트를 처리하지 않고 파이프를 생성하려면

- 파이프 활성화 설정을 끕니다.

이벤트를 프로그래밍 방식으로 처리하지 않고 파이프를 만들려면

- API 직접 호출에서 `DesiredState`를 `Stopped`로 설정합니다.

콘솔을 사용하여 기존 파이프를 시작 또는 중지하려면

- 파이프 설정 탭의 활성화에서 파이프 활성화에 대해 활성을 켜거나 끕니다.

프로그래밍 방식으로 기존 파이프를 시작 또는 중지하려면

- API 직접 호출에서 `DesiredState` 파라미터를 `RUNNING` 또는 `STOPPED` 중 하나로 설정합니다.

파이프가 `STOPPED`되는 시점과 더 이상 이벤트를 처리하지 않는 시점 사이에는 지연이 있을 수 있습니다.

- Amazon SQS 및 스트림 소스의 경우 이 지연은 일반적으로 2분 미만입니다.
- Amazon MQ 및 Apache Kafka 소스의 경우 이 지연은 최대 15분까지 걸릴 수 있습니다.

아마존 EventBridge 파이프 소스

EventBridge Pipes는 다양한 소스로부터 이벤트 데이터를 수신하고 해당 데이터에 선택적 필터 및 보강을 적용한 다음 목적지로 전송합니다.

소스가 Pipes로 전송된 이벤트에 순서를 적용하는 EventBridge 경우 해당 순서는 대상에 도달하는 전체 프로세스에서 유지됩니다.

다음 AWS 서비스를 EventBridge Pipes의 소스로 지정할 수 있습니다.

- Amazon DynamoDB 스트림
- Amazon Kinesis 스트림
- Amazon MQ 브로커
- Amazon MSK 스트림
- Amazon SQS 대기열

- 아파치 카프카 스트림

  Apache Kafka 스트림을 파이프 소스로 지정하는 경우 직접 관리하는 Apache Kafka 스트림 또는 다음과 같은 타사 공급자가 관리하는 스트림을 지정할 수 있습니다.

  - Confluent Cloud
  - CloudKarafka
  - Redpanda

소스로서의 Amazon DynamoDB 스트림

EventBridge 파이프를 사용하여 DynamoDB 스트림에서 레코드를 수신할 수 있습니다. 그런 다음, 해당 레코드를 처리를 위해 대상으로 보내기 전에 선택적으로 필터링하거나 개선할 수 있습니다. 파이프를 설정할 때 선택할 수 있는 Amazon DynamoDB Streams 관련 설정이 있습니다. EventBridge 파이프는 데이터 스트림의 레코드 순서를 유지하여 데이터를 대상으로 보냅니다.

> **Important**
>
> 파이프의 소스인 DynamoDB 스트림을 비활성화하면 스트림을 다시 활성화하더라도 해당 파이프를 사용할 수 없습니다. 이는 다음과 같은 이유로 발생합니다.
>
> - 소스가 비활성화된 파이프는 중지, 시작 또는 업데이트할 수 없습니다.
> - 파이프를 생성한 후에는 새 소스로 업데이트할 수 없습니다. DynamoDB 스트림을 다시 활성화하면 해당 스트림에 새 Amazon 리소스 이름(ARN)이 할당되고 더 이상 파이프와 연결되지 않습니다.
>
> DynamoDB 스트림을 다시 활성화하는 경우 스트림의 새 ARN을 사용하여 새 파이프를 생성해야 합니다.

예제 이벤트

다음 샘플 이벤트는 파이프가 수신한 정보를 보여줍니다. 이 이벤트를 사용하여 이벤트 패턴을 생성 및 필터링하거나 입력 변환을 정의할 수 있습니다. 모든 필드를 필터링할 수 있는 것은 아닙니다. 필터링할 수 있는 필드에 대한 자세한 내용은 ??? 섹션을 참조하세요.

```
[
  {
```

```
    "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"
  }
]
```

폴링 및 배치 처리 스트림

EventBridge는 초당 4회의 기본 속도로 레코드에 대해 DynamoDB 스트림의 샤드를 폴링합니다. 레코드를 사용할 수 있으면 EventBridge가 이벤트를 처리하고 결과를 기다립니다. 처리가 성공하면 EventBridge가 레코드를 더 받을 때까지 폴링을 재개합니다.

기본적으로, EventBridge는 레코드가 사용 가능하게 되는 즉시 파이프를 간접 호출합니다. EventBridge가 소스에서 읽는 배치에 레코드가 하나만 있는 경우 이벤트 하나만 처리됩니다. 적은 수의 레코드를 처리하는 것을 피하려면 배치 기간을 구성하여 파이프가 최대 5분 동안 레코드를 버퍼링하도록 지정하면 됩니다. 이벤트를 처리하기 전에 EventBridge는 전체 배치가 수집되거나 배치 작업 기간이 만료되거나 배치가 페이로드 한도인 6MB에 도달할 때까지 소스에서 레코드를 계속 읽습니다.

또한 각 샤드의 여러 배치를 병렬로 처리하여 동시성을 높일 수 있습니다. EventBridge는 각 샤드에서 최대 10개의 배치를 동시에 처리할 수 있습니다. 샤드당 동시 배치 수를 증가하는 경우 EventBridge에서는 파티션 키 수준에서의 순차적 처리를 여전히 보장합니다.

Kinesis 또는 DynamoDB 데이터 스트림의 한 샤드와 하나 이상의 파이프 실행을 동시에 처리하도록 `ParallelizationFactor` 설정을 구성합니다. EventBridge가 병렬화 계수를 통해 샤드에서 폴링하는 동시 배치의 수는 1(기본값)부터 10까지 지정할 수 있습니다. 예를 들어 `ParallelizationFactor`를 2로 설정하는 경우 최대 100개의 Kinesis 데이터 샤드를 처리하기 위한 200개의 동시 EventBridge 파이프 실행을 보유할 수 있습니다. 이는 데이터 볼륨이 일시적이고 `IteratorAge`가 높을 때 처리량을 확장하는 데 도움을 줍니다. Kinesis 집계를 사용하는 경우에는 병렬화 계수가 작동하지 않습니다.

폴링 및 스트리밍 시작 위치

파이프 생성 및 업데이트 중 스트림 소스 폴링은 최종적으로 일관됩니다.

- 파이프 생성 중 스트림에서 이벤트 폴링을 시작하는 데 몇 분 정도 걸릴 수 있습니다.
- 소스 폴링 구성에 대한 파이프 업데이트 중 스트림에서 이벤트 폴링을 중지했다가 다시 시작하는 데 몇 분 정도 걸릴 수 있습니다.

즉, 스트림의 시작 위치로 LATEST를 지정하면 파이프 생성 또는 업데이트 중에 전송된 이벤트가 파이프에서 누락될 수 있습니다. 누락된 이벤트가 없도록 하기 위해서는 스트림 시작 위치를 TRIM_HORIZON으로 지정하세요.

배치 항목 실패 보고

EventBridge가 소스의 스트리밍 데이터를 소비하고 처리할 때 기본적으로 배치가 완전히 성공한 경우에만 배치의 가장 높은 시퀀스 번호로 체크포인트를 수행합니다. 실패한 배치에서 정상 처리된 메시지를 재처리하지 않으려면 성공한 메시지와 실패한 메시지를 나타내는 객체를 반환하도록 보강 또는 대상을 구성하면 됩니다. 이를 부분 일괄 응답이라고 합니다.

자세한 내용은 ??? 섹션을 참조하세요.

성공 및 실패 조건

다음 중 하나를 반환할 경우 EventBridge는 배치를 완전한 성공으로 처리합니다.

- 비어 있는 `batchItemFailure` 목록
- null `batchItemFailure` 목록
- 비어 있는 `EventResponse`
- null `EventResponse`

다음 중 하나를 반환할 경우 EventBridge는 배치를 완전한 실패로 처리합니다.

- 빈 문자열 `itemIdentifier`
- null `itemIdentifier`
- 키 이름이 잘못된 `itemIdentifier`

EventBridge는 재시도 전략에 따라 실패를 재시도합니다.

Amazon Kinesis 스트림을 소스로 사용

EventBridge 파이프를 사용하여 Kinesis 데이터 스트림에서 레코드를 수신할 수 있습니다. 그런 다음, 해당 레코드를 처리를 위해 사용 가능한 대상 중 하나로 보내기 전에 선택적으로 필터링하거나 개선할 수 있습니다. 파이프를 설정할 때 선택할 수 있는 Kinesis 관련 설정이 있습니다. EventBridge 파이프는 데이터 스트림의 레코드 순서를 유지하여 데이터를 대상으로 보냅니다.

Kinesis 데이터 스트림은 샤드의 집합입니다. 샤드마다 데이터 레코드 시퀀스가 포함되어 있습니다. 소비자는 Kinesis 데이터 스트림의 데이터를 처리하는 애플리케이션입니다. EventBridge 파이프를 공유 처리량 소비자(표준 반복기)에 매핑하거나 향상된 팬 아웃 기능이 있는 전용 처리량 소비자에 매핑할 수 있습니다.

표준 반복기의 경우 EventBridge는 HTTP 프로토콜을 사용하여 레코드에 대해 Kinesis 스트림의 각 샤드를 폴링합니다. 파이프는 샤드의 다른 소비자와 읽기 처리량을 공유합니다.

지연 시간을 최소화하고 읽기 처리량을 최대화하기 위해 향상된 팬아웃으로 데이터 스트림 소비자를 생성할 수 있습니다. 스트림 소비자는 각 샤드에 대해 전용 연결을 설정하므로 스트림에서 읽는 다른 애플리케이션에 영향을 주지 않습니다. 전용 처리량은 많은 애플리케이션에서 동일한 데이터를 읽거나, 큰 레코드를 갖는 스트림을 재처리하는 경우에 유용합니다. Kinesis는 HTTP/2를 통해 레코드를 EventBridge로 푸시합니다. Kinesis 데이터 스트림에 대한 자세한 내용은 Amazon Kinesis Data Streams에서 데이터 읽기를 참조하세요.

예제 이벤트

다음 샘플 이벤트는 파이프가 수신한 정보를 보여줍니다. 이 이벤트를 사용하여 이벤트 패턴을 생성 및 필터링하거나 입력 변환을 정의할 수 있습니다. 모든 필드를 필터링할 수 있는 것은 아닙니다. 필터링할 수 있는 필드에 대한 자세한 내용은 ??? 섹션을 참조하세요.


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

폴링 및 배치 처리 스트림

EventBridge는 초당 4회의 기본 속도로 레코드에 대해 Kinesis 스트림의 샤드를 폴링합니다. 레코드를 사용할 수 있으면 EventBridge가 이벤트를 처리하고 결과를 기다립니다. 처리가 성공하면 EventBridge가 레코드를 더 받을 때까지 폴링을 재개합니다.

기본적으로, EventBridge는 레코드가 사용 가능하게 되는 즉시 파이프를 간접 호출합니다. EventBridge가 소스에서 읽는 배치에 레코드가 하나만 있는 경우 이벤트 하나만 처리됩니다. 적은 수의 레코드를 처리하는 것을 피하려면 배치 기간을 구성하여 파이프가 최대 5분 동안 레코드를 버퍼링하도록 지정하면 됩니다. 이벤트를 처리하기 전에 EventBridge는 전체 배치가 수집되거나 배치 작업 기간이 만료되거나 배치가 페이로드 한도인 6MB에 도달할 때까지 소스에서 레코드를 계속 읽습니다.

또한 각 샤드의 여러 배치를 병렬로 처리하여 동시성을 높일 수 있습니다. EventBridge는 각 샤드에서 최대 10개의 배치를 동시에 처리할 수 있습니다. 샤드당 동시 배치 수를 증가하는 경우 EventBridge에서는 파티션 키 수준에서의 순차적 처리를 여전히 보장합니다.

Kinesis 또는 DynamoDB 데이터 스트림의 한 샤드와 하나 이상의 파이프 실행을 동시에 처리하도록 `ParallelizationFactor` 설정을 구성합니다. EventBridge가 병렬화 계수를 통해 샤드에서 폴링하는 동시 배치의 수는 1(기본값)부터 10까지 지정할 수 있습니다. 예를 들어 `ParallelizationFactor`를 2로 설정하는 경우 최대 100개의 Kinesis 데이터 샤드를 처리하기 위한 200개의 동시 EventBridge 파이프 실행을 보유할 수 있습니다. 이는 데이터 볼륨이 일시적이고

IteratorAge가 높을 때 처리량을 확장하는 데 도움을 줍니다. Kinesis 집계를 사용하는 경우에는 병렬화 계수가 작동하지 않습니다.

폴링 및 스트리밍 시작 위치

파이프 생성 및 업데이트 중 스트림 소스 폴링은 최종적으로 일관됩니다.

- 파이프 생성 중 스트림에서 이벤트 폴링을 시작하는 데 몇 분 정도 걸릴 수 있습니다.
- 소스 폴링 구성에 대한 파이프 업데이트 중 스트림에서 이벤트 폴링을 중지했다가 다시 시작하는 데 몇 분 정도 걸릴 수 있습니다.

즉, 스트림의 시작 위치로 LATEST를 지정하면 파이프 생성 또는 업데이트 중에 전송된 이벤트가 파이프에서 누락될 수 있습니다. 누락된 이벤트가 없도록 하기 위해서는 스트림 시작 위치를 TRIM_HORIZON 또는 AT_TIMESTAMP로 지정하세요.

배치 항목 실패 보고

EventBridge가 소스의 스트리밍 데이터를 소비하고 처리할 때 기본적으로 배치가 완전히 성공한 경우에만 배치의 가장 높은 시퀀스 번호로 체크포인트를 수행합니다. 실패한 배치에서 정상 처리된 메시지를 재처리하지 않으려면 성공한 메시지와 실패한 메시지를 나타내는 객체를 반환하도록 보강 또는 대상을 구성하면 됩니다. 이를 부분 배치 응답이라고 합니다.

자세한 내용은 ??? 섹션을 참조하세요.

성공 및 실패 조건

다음 중 하나를 반환할 경우 EventBridge는 배치를 완전한 성공으로 처리합니다.

- 비어 있는 batchItemFailure 목록
- null batchItemFailure 목록
- 비어 있는 EventResponse
- null EventResponse

다음 중 하나를 반환할 경우 EventBridge는 배치를 완전한 실패로 처리합니다.

- 빈 문자열 itemIdentifier
- null itemIdentifier
- 키 이름이 잘못된 itemIdentifier

EventBridge는 재시도 전략에 따라 실패를 재시도합니다.

Amazon MQ 메시지 브로커를 소스로 사용

EventBridge Pipes를 사용하여 Amazon MQ 메시지 브로커로부터 레코드를 수신할 수 있습니다. 그런 다음, 해당 레코드를 처리를 위해 사용 가능한 대상 중 하나로 보내기 전에 선택적으로 필터링하거나 개선할 수 있습니다. 파이프를 설정할 때 선택할 수 있는 Amazon MQ 전용 설정이 있습니다. EventBridge Pipes는 데이터를 목적지로 전송할 때 메시지 브로커의 레코드 순서를 유지합니다.

Amazon MQ는 Apache ActiveMQ 및 RabbitMQ를 위한 관리형 메시지 브로커 서비스입니다. 메시지 브로커를 사용하면 소프트웨어 애플리케이션 및 구성 요소가 다양한 프로그래밍 언어, 운영 체제 및 공식 메시징 프로토콜을 사용하여 이벤트 대상인 토픽 또는 대기열과 통신할 수 있습니다.

Amazon MQ는 ActiveMQ 또는 RabbitMQ 브로커를 설치하여 사용자 대신 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 관리할 수도 있습니다. 브로커가 설치되면 다양한 네트워크 토폴로지와 기타 인프라 요구 사항을 인스턴스에 제공합니다.

Amazon MQ 소스에는 다음과 같은 구성 제한 사항이 있습니다.

- 크로스 어카운트 — 크로스 어카운트 처리는 EventBridge 지원하지 않습니다. 다른 AWS 계정에 있는 Amazon MQ 메시지 브로커의 레코드를 처리하는 EventBridge 데는 사용할 수 없습니다.
- 인증 - ActiveMQ의 경우 ActiveMQ만 지원됩니다. SimpleAuthenticationPlugin RabbitMQ의 경우 PLAIN 인증 메커니즘만 지원됩니다. 보안 인증 정보를 관리하려면 AWS Secrets Manager를 사용하세요. ActiveMQ 인증에 대한 자세한 내용은 Amazon MQ 개발자 안내서의 ActiveMQ 브로커와 LDAP 통합을 참조하세요.
- 연결 할당량 - 브로커에는 와이어 레벨 프로토콜당 허용되는 최대 연결 수가 있습니다. 이 할당량은 브로커 인스턴스 유형에 따라 결정됩니다. 자세한 내용은 Amazon MQ 개발자 안내서에서 *Amazon MQ의 할당량*의 브로커 섹션을 참조하세요.
- 연결성 - 퍼블릭 또는 프라이빗 Virtual Private Cloud(VPC)에서 브로커를 생성할 수 있습니다. 프라이빗 VPC의 경우 메시지를 수신하려면 파이프가 VPC에 대한 액세스 권한이 있어야 합니다.
- 이벤트 대상 - 대기열 대상만 지원됩니다. 그러나 파이프와 상호 작용하는 동안 내부적으로는 토픽으로 작동하고 외부에서는 대기열로 작동하는 가상 토픽을 사용할 수 있습니다. 자세한 내용은 Apache ActiveMQ 웹사이트의 가상 대상과 RabbitMQ 웹사이트의 가상 호스트를 참조하세요.
- 네트워크 토폴로지 - ActiveMQ의 경우 파이프당 하나의 단일 인스턴스 또는 대기 브로커만 지원됩니다. RabbitMQ의 경우 파이프당 하나의 단일 인스턴스 브로커 또는 클러스터 배포만 지원됩니다. 단일 인스턴스 브로커에는 장애 조치 엔드포인트가 필요합니다. 이러한 브로커 배포 모드에 대한 자세한 내용은 Amazon MQ 개발자 안내서에서 Active MQ 브로커 아키텍처 및 Rabbit MQ 브로커 아키텍처를 참조하세요.

- 프로토콜 - 지원되는 프로토콜은 사용하는 Amazon MQ 통합에 따라 다릅니다.
  - ActiveMQ EventBridge 통합의 경우 /Java 메시지 서비스 (JMS) 프로토콜을 사용하여 메시지를 사용합니다 OpenWire. 다른 프로토콜에서는 메시지 소비가 지원되지 않습니다. EventBridge JMS 프로토콜 내에서의 TextMessage및 BytesMessage작업만 지원합니다. OpenWire 프로토콜에 대한 자세한 내용은 Apache ActiveMQ 웹 사이트를 참조하십시오 OpenWire.
  - RabbitMQ 통합의 경우 AMQP 0-9-1 프로토콜을 EventBridge 사용하여 메시지를 사용합니다. 메시지 소비를 위한 다른 프로토콜은 지원되지 않습니다. RabbitMQ의 AMQP 0-9-1 프로토콜 구현에 대한 자세한 내용은 RabbitMQ 웹사이트의 AMQP 0-9-1 전체 참조 가이드를 참조하세요.

EventBridge Amazon MQ에서 지원하는 최신 버전의 ActiveMQ 및 RabbitMQ를 자동으로 지원합니다. 지원되는 최신 버전은 Amazon MQ 개발자 가이드의 Amazon MQ 릴리스 노트를 참조하세요

> **Note**
>
> 기본적으로 Amazon MQ에는 브로커에 대한 주별 유지 관리 기간이 있습니다. 해당 기간 동안에는 브로커를 사용할 수 없습니다. 대기 모드가 없는 브로커의 경우 기간이 종료될 때까지 메시지를 처리하지 EventBridge 않습니다.

이벤트 예제

다음 샘플 이벤트는 파이프가 수신한 정보를 보여줍니다. 이 이벤트를 사용하여 이벤트 패턴을 생성 및 필터링하거나 입력 변환을 정의할 수 있습니다. 모든 필드를 필터링할 수 있는 것은 아닙니다. 필터링할 수 있는 필드에 대한 자세한 내용은 ??? 섹션을 참조하세요.

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=="
  }
]
```


소비자 그룹

Amazon MQ와 상호 작용하려면 Amazon MQ 브로커로부터 읽을 수 있는 소비자 그룹을 EventBridge 생성하십시오. 소비자 그룹은 파이프 UUID와 동일한 ID를 사용하여 생성됩니다.

Amazon MQ 소스의 경우 레코드를 EventBridge 일괄 처리하여 단일 페이로드로 함수에 전송합니다. 동작을 제어하려면 일괄 처리 기간 및 배치 크기를 구성할 수 있습니다. EventBridge 다음 중 하나가 발생할 때까지 메시지를 가져옵니다.

- 처리된 레코드의 페이로드 크기는 최대 6MB에 이릅니다.
- 일괄 처리 기간이 만료됩니다.
- 레코드 수가 전체 배치 크기에 도달했습니다.

EventBridge 배치를 단일 페이로드로 변환한 다음 함수를 호출합니다. 메시지는 지속되거나 역직렬화되지 않습니다. 대신 소비자 그룹은 메시지를 바이트의 BLOB로 검색합니다. 그런 다음 JSON 페이로드에 base64로 인코딩합니다. 파이프가 일괄 처리된 메시지에 대해 오류를 반환하는 경우, 처리가 성공하거나 메시지가 만료될 때까지 전체 메시지 배치를 EventBridge 재시도합니다.

네트워크 구성

기본적으로 Amazon MQ 브로커는 `PubliclyAccessible` 플래그가 false로 설정되어 생성됩니다. 브로커가 퍼블릭 IP 주소를 수신하는 경우에만 `PubliclyAccessible`이 true로 설정됩니다. 파이프의 전체 액세스를 위해 브로커는 퍼블릭 엔드포인트를 사용하거나 VPC에 대한 액세스를 제공해야 합니다.

Amazon MQ 브로커에 공개적으로 액세스할 수 없는 경우 브로커와 연결된 Amazon VPC (가상 사설 클라우드) 리소스에 액세스할 수 EventBridge 있어야 합니다.

- Amazon MQ 브로커의 EventBridge VPC에 액세스하려면 소스의 서브넷에 대한 아웃바운드 인터넷 액세스를 사용할 수 있습니다. 퍼블릭 서브넷의 경우 이는 관리형 NAT 게이트웨이여야 합니다. 프라이빗 서브넷의 경우 이는 NAT 게이트웨이 또는 자체 NAT일 수 있습니다. NAT에 퍼블릭 IP 주소가 있고 인터넷에 연결할 수 있는지 확인합니다.
- EventBridge 또한 Pipes는 이벤트 전송을 지원하므로 퍼블릭 인터넷을 AWS PrivateLink거치지 않고도 Amazon Virtual Private Cloud (Amazon VPC) 에 있는 이벤트 소스에서 Pipes 대상으로 이벤트를 보낼 수 있습니다. 인터넷 게이트웨이를 배포하거나, 방화벽 규칙을 구성하거나, 프록시 서버를 설정할 필요 없이 Pipes를 사용하여 Amazon Managed Streaming for Apache Kafka (Amazon MSK), 자체 관리형 Apache Kafka 및 프라이빗 서브넷에 있는 Amazon MQ 소스에서 폴링할 수 있습니다.

  VPC 엔드포인트를 설정하려면 사용 설명서의 VPC 엔드포인트 생성을 참조하십시오.AWS PrivateLink 서비스 이름에서 을 선택합니다. `com.amazonaws.region.pipes-data`

다음 규칙(최소)으로 Amazon VPC 보안 그룹을 구성합니다.

- 인바운드 규칙 — Amazon MQ 브로커 포트에서 소스에 지정된 보안 그룹에 대한 모든 트래픽을 허용합니다.

- 아웃바운드 규칙 - 모든 대상에 대해 포트 443의 모든 트래픽을 허용합니다. Amazon MQ 브로커 포트에서 소스에 지정된 보안 그룹에 대한 모든 트래픽을 허용합니다.

  브로커 포트에는 다음이 포함됩니다.

  - 일반 텍스트의 경우 9092
  - TLS의 경우 9094
  - SASL의 경우 9096
  - IAM의 경우 9098

> **Note**
>
> Amazon VPC 구성은 Amazon MQ API를 통해 검색할 수 있습니다. 설정하는 동안 이 옵션을 구성할 필요가 없습니다.

Amazon Managed Streaming for Apache Kafka 주제를 소스로 사용

EventBridge 파이프를 사용하여 Apache Kafka용 아마존 매니지드 스트리밍 (Amazon MSK) 주제로부터 레코드를 수신할 수 있습니다. 그런 다음, 해당 레코드를 처리를 위해 사용 가능한 대상 중 하나로 보내기 전에 선택적으로 필터링하거나 개선할 수 있습니다. 파이프를 설정할 때 선택할 수 있는 Amazon MSK만의 설정이 있습니다. EventBridge Pipes는 데이터를 목적지로 전송할 때 메시지 브로커의 레코드 순서를 유지합니다.

Amazon MSK는 Apache Kafka를 사용하여 스트리밍 데이터를 처리하는 애플리케이션을 구축하고 실행하는 데 사용할 수 있는 완전관리형 서비스입니다. Amazon MSK는 Apache Kafka를 실행하는 클러스터의 설정, 크기 조정, 관리를 간소화합니다. Amazon MSK를 사용하면 여러 가용 영역과 AWS Identity and Access Management (IAM) 을 통한 보안을 위해 애플리케이션을 구성할 수 있습니다. Amazon MSK는 Kafka의 여러 오픈 소스 버전을 지원합니다.

Amazon MSK를 소스로 사용하는 것은 아마존 심플 큐 서비스 (Amazon SQS) 또는 아마존 키네시스를 사용하는 것과 비슷하게 작동합니다. EventBridge소스의 새 메시지를 내부적으로 폴링한 다음 대상을 동기적으로 호출합니다. EventBridge 메시지를 일괄적으로 읽고 이를 함수에 이벤트 페이로드로 제공합니다. 최대 배치 크기는 구성 가능합니다. (기본값은 100개의 메시지입니다.)

Apache Kafka 기반 소스의 경우 배치 창 및 EventBridge 배치 크기와 같은 처리 제어 매개 변수를 지원합니다.

EventBridge 각 파티션에 대해 순차적으로 메시지를 읽습니다. 각 배치를 EventBridge 처리한 후 해당 배치에 있는 메시지의 오프셋을 커밋합니다. 파이프의 대상이 일괄 처리된 메시지에 대해 오류를 반환하는 경우, 처리가 성공하거나 메시지가 만료될 때까지 전체 메시지 배치를 EventBridge 재시도합니다.

EventBridge 대상을 호출할 때 이벤트에서 메시지 배치를 전송합니다. 이벤트 페이로드에는 메시지 배열이 포함됩니다. 각 배열 항목에는 Amazon MSK 주제 및 파티션 식별자에 대한 세부 정보와 함께 타임스탬프 및 base64로 인코딩된 메시지가 포함됩니다.

이벤트 예제

다음 샘플 이벤트는 파이프가 수신한 정보를 보여줍니다. 이 이벤트를 사용하여 이벤트 패턴을 생성 및 필터링하거나 입력 변환을 정의할 수 있습니다. 모든 필드를 필터링할 수 있는 것은 아닙니다. 필터링할 수 있는 필드에 대한 자세한 내용은 ??? 섹션을 참조하세요.


```
[
  {
    "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
```

```
        ]
      }
    ]
  }
]
```

폴링 및 스트리밍 시작 위치

파이프 생성 및 업데이트 중 스트림 소스 폴링은 최종적으로 일관됩니다.

- 파이프 생성 중 스트림에서 이벤트 폴링을 시작하는 데 몇 분 정도 걸릴 수 있습니다.
- 소스 폴링 구성에 대한 파이프 업데이트 중 스트림에서 이벤트 폴링을 중지했다가 다시 시작하는 데 몇 분 정도 걸릴 수 있습니다.

즉, 스트림의 시작 위치로 `LATEST`를 지정하면 파이프 생성 또는 업데이트 중에 전송된 이벤트가 파이프에서 누락될 수 있습니다. 누락된 이벤트가 없도록 하기 위해서는 스트림 시작 위치를 `TRIM_HORIZON`으로 지정하세요.

MSK 클러스터 인증

EventBridge Amazon MSK 클러스터에 액세스하고, 레코드를 검색하고, 기타 작업을 수행할 권한이 필요합니다. Amazon MSK는 MSK 클러스터에 대한 클라이언트 액세스를 제어하는 몇 가지 옵션을 지원합니다. 어떤 인증 방법이 사용되는지에 대한 자세한 내용은 ??? 단원을 참조하세요.

클러스터 액세스 옵션

- 인증되지 않은 액세스
- SASL/SCRAM 인증
- IAM 역할 기반 인증
- 상호 TLS 인증
- mTLS 암호 구성
- 부트스트랩 EventBridge 브로커 선택 방법

인증되지 않은 액세스

개발에는 인증되지 않은 액세스만 사용하는 것이 좋습니다. 클러스터에 대해 IAM 역할 기반 인증이 비활성화된 경우에만 인증되지 않은 액세스가 가능합니다.

SASL/SCRAM 인증

Amazon MSK는 전송 계층 보안(TLS) 암호화를 통해 Simple Authentication and Security Layer/Salted Challenge Response Authentication Mechanism(SASL/SCRAM) 인증을 지원합니다. 클러스터에 EventBridge 연결하려면 인증 자격 증명 (로그인 자격 증명) 을 비밀에 저장합니다. AWS Secrets Manager

Secrets Manager 사용에 대한 자세한 내용은 Amazon Managed Streaming for Apache Kafka 개발자 안내서의 AWS Secrets Manager를 사용한 사용자 이름 및 암호 인증을 참조하세요.

Amazon MSK는 SASL/PLAIN 인증을 지원하지 않습니다.

IAM 역할 기반 인증

IAM을 사용하여 MSK 클러스터에 연결하는 클라이언트의 자격 증명을 인증할 수 있습니다. MSK 클러스터에서 IAM 인증이 활성화되어 있고 인증을 위한 암호를 제공하지 않으면 EventBridge 자동으로 IAM 인증을 기본적으로 사용합니다. IAM 사용자 또는 역할 기반 정책을 생성하고 배포하려면 IAM 콘솔 또는 API를 사용합니다. 자세한 내용은 Amazon Managed Streaming for Apache Kafka 개발자 안내서의 IAM 액세스 제어를 참조하세요.

MSK 클러스터에 연결하고, 레코드를 읽고, 기타 필요한 작업을 수행할 수 있도록 EventBridge 하려면 파이프의 실행 역할에 다음 권한을 추가하십시오.

```
{
    "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"
            ]
```

```
        }
    ]
}
```

이러한 권한의 범위를 특정 클러스터, 주제 및 그룹으로 설정할 수 있습니다. 자세한 내용은 Amazon Managed Streaming for Apache Kafka 개발자 안내서의 Amazon MSK Kafka 작업을 참조하세요.

상호 TLS 인증

상호 TLS(mTLS)는 클라이언트와 서버 간의 양방향 인증을 제공합니다. 클라이언트는 서버가 클라이언트를 확인할 수 있도록 서버에 인증서를 보내고, 서버는 클라이언트가 서버를 확인할 수 있도록 클라이언트에 인증서를 보냅니다.

Amazon MSK의 경우 클라이언트 EventBridge 역할을 합니다. 클라이언트 인증서 (Secrets Manager에서는 보안 인증서) 를 구성하여 MSK EventBridge 클러스터의 브로커에 인증합니다. 클라이언트 인증서는 서버의 신뢰 저장소에 있는 인증 기관(CA)에서 서명해야 합니다. MSK 클러스터는 브로커를 EventBridge 인증하기 위해 서버 인증서를 전송합니다. EventBridge 서버 인증서는 신뢰 저장소에 있는 CA에서 서명해야 합니다. AWS

Amazon MSK의 모든 브로커는 기본적으로 신뢰하는 EventBridge Amazon 신뢰 서비스 CA에서 서명한 공개 인증서를 사용하기 때문에 Amazon MSK는 자체 서명 서버 인증서를 지원하지 않습니다.

Amazon MSK용 mTLS에 대한 자세한 내용은 Amazon Managed Streaming for Apache Kafka 개발자 안내서의 상호 TLS 인증을 참조하세요.

mTLS 암호 구성

CLIENT_CERTIFICATE_TLS_AUTH 비밀 정보에 인증서 필드와 프라이빗 키 필드가 필요합니다. 암호화된 프라이빗 키의 경우 비밀 정보에 프라이빗 키 암호가 필요합니다. 인증서와 프라이빗 키는 모두 PEM 형식이어야 합니다.

> **Note**
>
> EventBridge PBES1 (PBES2 제외) 개인 키 암호화 알고리즘을 지원합니다.

인증서 필드에는 클라이언트 인증서부터 시작하여 중간 인증서가 이어지고 루트 인증서로 끝나는 인증서 목록이 포함되어야 합니다. 각 인증서는 다음 구조의 새 줄에서 시작해야 합니다.

```
-----BEGIN CERTIFICATE-----
```

```
        <certificate contents>
-----END CERTIFICATE-----
```

Secrets Manager는 최대 65,536바이트의 보안 정보를 지원하므로 긴 인증서 체인을 위한 충분한 공간입니다.

프라이빗 키는 다음 구조의 PKCS #8 형식이어야 합니다.

```
-----BEGIN PRIVATE KEY-----
         <private key contents>
-----END PRIVATE KEY-----
```

암호화된 프라이빗 키의 경우 다음 구조를 사용합니다.

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
          <private key contents>
-----END ENCRYPTED PRIVATE KEY-----
```

다음 예제에서는 암호화된 프라이빗 키를 사용한 mTLS 인증용 비밀 정보 콘텐츠를 표시합니다. 암호화된 개인 키의 경우 비밀 정보에 프라이빗 키 암호를 포함합니다.

```
{
 "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-----"
```

```
}
```

부트스트랩 EventBridge 브로커 선택 방법

EventBridge 클러스터에서 사용할 수 있는 인증 방법과 인증을 위한 암호 제공 여부에 따라 부트스트랩 브로커를 선택합니다. mTLS 또는 SASL/SCRAM에 대한 암호를 제공하면 해당 인증 방법이 자동으로 선택됩니다. EventBridge 암호를 제공하지 않는 경우 클러스터에서 활성화된 가장 강력한 인증 방법을 EventBridge 선택합니다. 다음은 가장 강력한 인증부터 가장 약한 인증까지 브로커를 EventBridge 선택하는 우선 순위입니다.

- mTLS(MTL에 제공되는 암호)
- SASL/SCRAM(SASL/SCRAM에 대해 제공되는 암호)
- SASL IAM(암호가 제공되지 않았으며 IAM 인증이 활성화됨)
- 인증되지 않은 TLS(암호가 제공되지 않고 IAM 인증이 활성화되지 않음)
- 일반 텍스트(암호가 제공되지 않고 IAM 인증 및 인증되지 않은 TLS가 모두 활성화되지 않음)

> **Note**
>
> 가장 안전한 브로커 유형에 연결할 EventBridge 수 없는 경우 다른 (약한) 브로커 유형에 연결을 시도하지 않습니다. 더 약한 브로커 유형을 EventBridge 선택하려면 클러스터에서 더 강력한 인증 방법을 모두 비활성화하세요.

네트워크 구성

EventBridge Amazon MSK 클러스터와 연결된 Amazon VPC (가상 사설 클라우드) 리소스에 액세스할 수 있어야 합니다.

- Amazon MSK 클러스터의 VPC에 액세스하려면 소스의 서브넷에 대한 아웃바운드 인터넷 액세스를 사용할 EventBridge 수 있습니다. 퍼블릭 서브넷의 경우 이는 관리형 NAT 게이트웨이여야 합니다. 프라이빗 서브넷의 경우 이는 NAT 게이트웨이 또는 자체 NAT일 수 있습니다. NAT에 퍼블릭 IP 주소가 있고 인터넷에 연결할 수 있는지 확인합니다.
- EventBridge 또한 Pipes는 이벤트 전송을 지원하므로 퍼블릭 인터넷을 AWS PrivateLink거치지 않고도 Amazon Virtual Private Cloud (Amazon VPC) 에 있는 이벤트 소스에서 Pipes 대상으로 이벤트를 보낼 수 있습니다. 인터넷 게이트웨이를 배포하거나, 방화벽 규칙을 구성하거나, 프록시 서버를 설정할 필요 없이 Pipes를 사용하여 Amazon Managed Streaming for Apache Kafka (Amazon MSK), 자체 관리형 Apache Kafka 및 프라이빗 서브넷에 있는 Amazon MQ 소스에서 폴링할 수 있습니다.

VPC 엔드포인트를 설정하려면 사용 설명서의 VPC 엔드포인트 만들기를 참조하십시오.AWS PrivateLink 서비스 이름에서 을 선택합니다. `com.amazonaws.region.pipes-data`

다음 규칙(최소)으로 Amazon VPC 보안 그룹을 구성합니다.

- 인바운드 규칙 — Amazon MSK 브로커 포트에서 소스에 지정된 보안 그룹에 대한 모든 트래픽을 허용합니다.
- 아웃바운드 규칙 - 모든 대상에 대해 포트 443의 모든 트래픽을 허용합니다. Amazon MSK 브로커 포트에서 소스에 지정된 보안 그룹에 대한 모든 트래픽을 허용합니다.

  브로커 포트에는 다음이 포함됩니다.
  - 일반 텍스트의 경우 9092
  - TLS의 경우 9094
  - SASL의 경우 9096
  - IAM의 경우 9098

> **Note**
>
> Amazon VPC 구성은 Amazon MSK API를 통해 검색할 수 있습니다. 설정하는 동안 이 옵션을 구성할 필요가 없습니다.

사용자 지정이 가능한 소비자 그룹 ID

Apache Kafka를 소스로 설정할 때 소비자 그룹 ID를 지정할 수 있습니다. 이 소비자 그룹 ID는 파이프를 조인하려는 Apache Kafka 소비자 그룹의 기존 식별자입니다. 이 기능을 사용하여 진행 중인 Apache Kafka 레코드 처리 설정을 다른 소비자의 Apache Kafka 레코드 처리 설정으로 마이그레이션 할 수 있습니다. EventBridge

소비자 그룹 ID를 지정하고 해당 소비자 그룹 내에 다른 활성 폴러가 있는 경우 Apache Kafka는 모든 소비자에게 메시지를 배포합니다. 즉, Apache Kafka 주제에 대한 모든 메시지를 EventBridge 수신하지는 않습니다. 해당 주제의 모든 메시지를 EventBridge 처리하려면 해당 소비자 그룹의 다른 설문조사를 모두 끄십시오.

또한 소비자 그룹 ID를 지정하고 Apache Kafka가 동일한 ID를 가진 유효한 기존 소비자 그룹을 찾는 경우 파이프의 매개 변수를 EventBridge 무시합니다. `StartingPosition` 대신 소비자 그룹의 커

밋된 오프셋에 따라 레코드 처리를 EventBridge 시작합니다. 소비자 그룹 ID를 지정했는데 Apache Kafka가 기존 소비자 그룹을 찾을 수 없는 경우 지정된 것으로 EventBridge 소스를 구성합니다. `StartingPosition`

지정하는 소비자 그룹 ID는 모든 Apache Kafka 이벤트 소스 중에서 고유해야 합니다. 지정된 소비자 그룹 ID로 파이프를 생성한 후에는 이 값을 업데이트할 수 없습니다.

Amazon MSK 소스의 Auto Scaling

Amazon MSK 소스를 처음 생성할 때 Apache Kafka 주제의 모든 파티션을 처리하도록 한 명의 소비자를 EventBridge 할당합니다. 각 소비자는 증가한 워크로드를 처리하기 위해 여러 프로세서를 병렬로 실행합니다. 또한 EventBridge 워크로드에 따라 소비자 수를 자동으로 늘리거나 줄입니다. 각 파티션에서 메시지 순서를 유지하기 위해 최대 소비자 수는 주제의 파티션당 하나의 소비자입니다.

1분 간격으로 주제에 있는 모든 파티션의 소비자 오프셋 지연을 EventBridge 평가합니다. 지연이 너무 높으면 파티션이 처리할 수 있는 속도보다 EventBridge 더 빨리 메시지를 수신하는 것입니다. 필요한 경우 주제에 소비자를 EventBridge 추가하거나 제거합니다. 소비자를 추가 또는 제거하는 크기 조정 프로세스는 평가 후 3분 이내에 진행됩니다.

대상에 과부하가 걸리면 소비자 수를 EventBridge 줄이십시오. 이 동작은 소비자가 검색하고 파이프에 보낼 수 있는 메시지 수를 줄임으로써 파이프의 워크로드를 줄입니다.

소스로서의 아파치 카프카 스트림

Apache Kafka는 데이터 파이프라인 및 스트리밍 분석과 같은 워크로드를 지원하는 오픈 소스 이벤트 스트리밍 플랫폼입니다. Apache Kafka용 아마존 매니지드 스트리밍 (Amazon MSK) 또는 자체 관리형 아파치 Kafka 클러스터를 사용할 수 있습니다. 자체 관리형 클러스터는 AWS 용어로 호스팅되지 않은 모든 Apache Kafka 클러스터를 말합니다. AWS여기에는 직접 관리하는 클러스터와 타사 공급자가 호스팅하는 클러스터 (예:, 또는) 가 모두 포함됩니다. Confluent CloudCloudKarafkaRedpanda

클러스터의 다른 AWS 호스팅 옵션에 대한 자세한 내용은 AWS 빅데이터 블로그의 Apache Kafka 실행 모범 사례를 참조하십시오. AWS

소스로서의 아파치 카프카는 아마존 심플 큐 서비스 (Amazon SQS) 또는 아마존 키네시스를 사용하는 것과 비슷하게 작동합니다. EventBridge소스의 새 메시지를 내부적으로 폴링한 다음 대상을 동기적으로 호출합니다. EventBridge 메시지를 일괄적으로 읽고 이를 함수에 이벤트 페이로드로 제공합니다. 최대 배치 크기는 구성 가능합니다. (기본값은 100개의 메시지입니다.)

Apache Kafka 기반 소스의 경우 배치 창 및 EventBridge 배치 크기와 같은 처리 제어 매개 변수를 지원합니다.

EventBridge 파이프를 호출할 때 이벤트 매개 변수로 메시지 배치를 전송합니다. 이벤트 페이로드에는 메시지 배열이 포함됩니다. 각 배열 항목에는 Apache Kafka 주제 및 Apache Kafka 파티션 식별자에 대한 세부 정보와 함께 타임스탬프 및 base64로 인코딩된 메시지가 포함됩니다.

이벤트 예제

다음 샘플 이벤트는 파이프가 수신한 정보를 보여줍니다. 이 이벤트를 사용하여 이벤트 패턴을 생성 및 필터링하거나 입력 변환을 정의할 수 있습니다. 모든 필드를 필터링할 수 있는 것은 아닙니다. 필터링할 수 있는 필드에 대한 자세한 내용은 ??? 섹션을 참조하세요.

```
[
  {
    "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 클러스터 인증

EventBridge Pipes는 자체 관리형 Apache Kafka 클러스터를 사용하여 인증하는 몇 가지 방법을 지원합니다. 지원되는 인증 방법 중 하나를 사용하도록 Apache Kafka 클러스터를 구성해야 합니다. Apache Kafka 보안에 대한 자세한 내용은 Apache Kafka 설명서의 보안 섹션을 참조하세요.

VPC 액세스

VPC 내의 Apache Kafka 사용자만 Apache Kafka 브로커에 액세스할 수 있는 자체 관리형 Apache Kafka 환경을 사용하는 경우 Apache Kafka 소스에서 Amazon VPC (가상 사설 클라우드) 를 구성해야 합니다.

SASL/SCRAM 인증

EventBridge Pipes는 전송 계층 보안 (TLS) 암호화를 통한 단순 인증 및 보안 계층/솔티드 챌린지 응답 인증 메커니즘 (SASL/SCRAM) 인증을 지원합니다. EventBridge Pipes는 암호화된 자격 증명을 전송하여 클러스터에서 인증합니다. SASL/SCRAM 인증에 관한 자세한 내용은 RFC 5802를 참조하세요.

EventBridge 파이프는 TLS 암호화를 통한 SASL/PLAIN 인증을 지원합니다. SASL/PLAIN 인증을 사용하면 Pipes는 자격 증명을 EventBridge 암호화되지 않은 일반 텍스트 (암호화되지 않은) 로 서버에 보냅니다.

SASL 인증의 경우 로그인 자격 증명을 AWS Secrets Manager에 보안 암호로 저장합니다.

상호 TLS 인증

상호 TLS(mTLS)는 클라이언트와 서버 간의 양방향 인증을 제공합니다. 클라이언트는 서버가 클라이언트를 확인할 수 있도록 서버에 인증서를 보내고, 서버는 클라이언트가 서버를 확인할 수 있도록 클라이언트에 인증서를 보냅니다.

자체 관리형 Apache Kafka에서는 파이프가 클라이언트 역할을 합니다. EventBridge Apache Kafka 브로커를 사용하여 EventBridge 파이프를 인증하려면 클라이언트 인증서 (Secrets Manager에서 암호로) 를 구성합니다. 클라이언트 인증서는 서버의 신뢰 저장소에 있는 인증 기관(CA)에서 서명해야 합니다.

Apache Kafka 클러스터는 Pipes로 Apache Kafka 브로커를 인증하기 위해 서버 인증서를 EventBridge Pipes로 전송합니다. EventBridge 서버 인증서는 퍼블릭 CA 인증서 또는 프라이빗 CA/자체 서명 인증서일 수 있습니다. 공개 CA 인증서는 Pipes 신뢰 저장소에 있는 CA에서 서명해야 합니다. EventBridge 사설 CA/자체 서명 인증서의 경우 서버 루트 CA 인증서를 Secrets Manager에서 암호로 구성합니다. EventBridge 파이프는 루트 인증서를 사용하여 Apache Kafka 브로커를 확인합니다.

mTLS에 대한 자세한 내용은 소스로 Amazon MSK에 대한 상호 TLS 인증 도입을 참조하세요.

클라이언트 인증서 비밀 정보 구성

CLIENT_CERTIFICATE_TLS_AUTH 비밀 정보에 인증서 필드와 프라이빗 키 필드가 필요합니다. 암호화된 프라이빗 키의 경우 비밀 정보에 프라이빗 키 암호가 필요합니다. 인증서와 프라이빗 키는 모두 PEM 형식이어야 합니다.

> **Note**
>
> EventBridge Pipes는 PBES1 (PBES2 제외) 개인 키 암호화 알고리즘을 지원합니다.

인증서 필드에는 클라이언트 인증서부터 시작하여 중간 인증서가 이어지고 루트 인증서로 끝나는 인증서 목록이 포함되어야 합니다. 각 인증서는 다음 구조의 새 줄에서 시작해야 합니다.

```
-----BEGIN CERTIFICATE-----
        <certificate contents>
-----END CERTIFICATE-----
```

Secrets Manager는 최대 65,536바이트의 보안 정보를 지원하므로 긴 인증서 체인을 위한 충분한 공간입니다.

프라이빗 키는 다음 구조의 PKCS #8 형식이어야 합니다.

```
-----BEGIN PRIVATE KEY-----
         <private key contents>
-----END PRIVATE KEY-----
```

암호화된 프라이빗 키의 경우 다음 구조를 사용합니다.

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
          <private key contents>
-----END ENCRYPTED PRIVATE KEY-----
```

다음 예제에서는 암호화된 프라이빗 키를 사용한 mTLS 인증용 비밀 정보 콘텐츠를 표시합니다. 암호화된 프라이빗 키의 경우 비밀 정보에 프라이빗 키 암호를 포함합니다.

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

서버 루트 CA 인증서 비밀 정보 구성

Apache Kafka 브로커가 프라이빗 CA에서 서명한 인증서로 TLS 암호화를 사용하는 경우 이 비밀 정보를 생성합니다. VPC, SASL/SCRAM, SASL/PLAIN 또는 mTLS 인증에 TLS 암호화를 사용할 수 있습니다.

서버 루트 CA 인증서 비밀 정보에는 PEM 형식의 Apache Kafka 브로커의 루트 CA 인증서가 포함된 필드가 필요합니다. 다음 예제는 비밀 정보의 구조를 보여줍니다.

```
{
    "certificate": "-----BEGIN CERTIFICATE-----
  MIID7zCCAtegAwIBAgIBADANBgkqhkiG9w0BAQsFADCBmDELMAkGA1UEBhMCVVMx
  EDAOBgNVBAgTB0FyaXpvbmExEzARBgNVBAcTClNjb3R0c2RhbGUxJTAjBgNVBAoT
  HFN0YXJmaWVsZCBUZWNobm9sb2dpZXMsIEluYy4xOzA5BgNVBAMTMlN0YXJmaWVs
  ZCBTZXJ2aWNlcyBSb290IENlcnRpZmljYXRlIEF1dG...
  -----END CERTIFICATE-----"
```

네트워크 구성

프라이빗 VPC 연결을 사용하는 자체 관리형 Apache Kafka 환경을 사용하는 경우 Apache Kafka 브로커와 연결된 Amazon VPC (가상 사설 클라우드) 리소스에 액세스할 수 EventBridge 있어야 합니다.

- Apache Kafka 클러스터의 VPC에 액세스하려면 소스의 서브넷에 대한 아웃바운드 인터넷 액세스를 사용할 EventBridge 수 있습니다. 퍼블릭 서브넷의 경우 이는 관리형 NAT 게이트웨이여야 합니다.

프라이빗 서브넷의 경우 이는 NAT 게이트웨이 또는 자체 NAT일 수 있습니다. NAT에 퍼블릭 IP 주소가 있고 인터넷에 연결할 수 있는지 확인합니다.

- EventBridge 또한 Pipes는 를 통한 AWS PrivateLink이벤트 전달을 지원하므로 공용 인터넷을 통과하지 않고도 Amazon Virtual Private Cloud (Amazon VPC) 에 있는 이벤트 소스에서 Pipes 대상으로 이벤트를 보낼 수 있습니다. 인터넷 게이트웨이를 배포하거나, 방화벽 규칙을 구성하거나, 프록시 서버를 설정할 필요 없이 Pipes를 사용하여 Amazon Managed Streaming for Apache Kafka (Amazon MSK), 자체 관리형 Apache Kafka 및 프라이빗 서브넷에 있는 Amazon MQ 소스에서 폴링할 수 있습니다.

  VPC 엔드포인트를 설정하려면 사용 설명서의 VPC 엔드포인트 만들기를 참조하십시오.AWS PrivateLink 서비스 이름에서 을 선택합니다. `com.amazonaws.region.pipes-data`

다음 규칙(최소)으로 Amazon VPC 보안 그룹을 구성합니다.

- 인바운드 규칙 - 소스에 지정된 보안 그룹에 대한 Apache Kafka 브로커 포트의 모든 트래픽을 허용합니다.
- 아웃바운드 규칙 - 모든 대상에 대해 포트 443의 모든 트래픽을 허용합니다. 소스에 지정된 보안 그룹에 대한 Apache Kafka 브로커 포트의 모든 트래픽을 허용합니다.

  브로커 포트에는 다음이 포함됩니다.

  - 일반 텍스트의 경우 9092
  - TLS의 경우 9094
  - SASL의 경우 9096
  - IAM의 경우 9098

Apache Kafka 소스를 사용한 소비자 자동 스케일링

Apache Kafka 소스를 처음 생성할 때 Kafka 주제의 모든 파티션을 처리하도록 한 명의 EventBridge 소비자를 할당합니다. 각 소비자는 증가한 워크로드를 처리하기 위해 여러 프로세서를 병렬로 실행합니다. 또한 EventBridge 워크로드에 따라 소비자 수를 자동으로 늘리거나 줄입니다. 각 파티션에서 메시지 순서를 유지하기 위해 최대 소비자 수는 주제의 파티션당 하나의 소비자입니다.

1분 간격으로 주제에 있는 모든 파티션의 소비자 오프셋 지연을 EventBridge 평가합니다. 지연이 너무 높으면 파티션이 처리할 수 있는 속도보다 EventBridge 더 빨리 메시지를 수신하는 것입니다. 필요한 경우 주제에 소비자를 EventBridge 추가하거나 제거합니다. 소비자를 추가 또는 제거하는 크기 조정 프로세스는 평가 후 3분 이내에 진행됩니다.

대상에 과부하가 걸리면 소비자 수를 EventBridge 줄이십시오. 이 동작은 소비자가 검색하고 함수에 보낼 수 있는 메시지 수를 줄임으로써 함수의 워크로드를 줄입니다.

Amazon Simple Queue Service를 소스로 사용

EventBridge Pipes를 사용하여 Amazon SQS 대기열에서 레코드를 수신할 수 있습니다. 그런 다음, 해당 레코드를 처리를 위해 사용 가능한 대상으로 보내기 전에 선택적으로 필터링하거나 개선할 수 있습니다.

파이프를 사용하여 Amazon Simple Queue 서비스 (Amazon SQS) 대기열에 있는 메시지를 처리할 수 있습니다. EventBridge 파이프는 표준 대기열과 선입선출 (FIFO) 대기열을 지원합니다. Amazon SQS를 사용하면 작업을 대기열로 전송하고 비동기식으로 처리하여 애플리케이션의 구성 요소 중 하나에서 작업을 오프로드할 수 있습니다.

EventBridge 대기열을 폴링하고 대기열 메시지가 포함된 이벤트와 동시에 파이프를 호출합니다. EventBridge 메시지를 일괄적으로 읽고 일괄 처리할 때마다 파이프를 한 번씩 호출합니다. 파이프가 배치를 성공적으로 처리하면 대기열에서 해당 메시지가 EventBridge 삭제됩니다.

기본적으로 대기열에 있는 최대 10개의 메시지를 동시에 EventBridge 폴링하여 해당 배치를 파이프로 보냅니다. 적은 수의 레코드로 파이프를 간접 호출하는 것을 피하려면 배치 기간을 구성하여 이벤트 소스가 최대 5분 동안 레코드를 버퍼링하도록 지정할 수 있습니다. 파이프를 호출하기 전에 다음 중 하나가 발생할 때까지 Amazon SQS 표준 대기열에서 메시지를 EventBridge 계속 폴링합니다.

- 일괄 처리 기간이 만료됩니다.
- 간접 호출 페이로드 크기 할당량에 도달했습니다.
- 구성된 최대 배치 크기에 도달했습니다.

> **Note**
>
> 배치 창을 사용하고 Amazon SQS 대기열에 트래픽이 적은 경우 파이프를 호출하기 전에 최대 20초까지 기다릴 EventBridge 수 있습니다. 이는 배치 기간을 20초 미만으로 설정한 경우에도 마찬가지입니다. FIFO 대기열의 경우 레코드에는 중복 제거 및 시퀀싱과 관련된 추가 속성이 포함되어 있습니다.

일괄 EventBridge 읽기 시 메시지는 대기열에 남아 있지만 대기열의 가시성 제한 시간 동안 숨겨집니다. 파이프가 배치를 성공적으로 처리하면 대기열에서 메시지가 EventBridge 삭제됩니다. 기본적으로

배치를 처리하는 동안 파이프에 오류가 발생하면 해당 배치의 모든 메시지가 대기열에 다시 표시됩니다. 이러한 이유로 파이프 코드는 의도하지 않은 부작용 없이 동일한 메시지를 여러 번 처리할 수 있어야 합니다. 파이프 응답에 배치 항목 실패를 포함시켜 이 재처리 동작을 수정할 수 있습니다. 다음 예제에서는 메시지 두 개의 배치에 대한 이벤트를 보여 줍니다.

이벤트 예제

다음 샘플 이벤트는 파이프가 수신한 정보를 보여줍니다. 이 이벤트를 사용하여 이벤트 패턴을 생성 및 필터링하거나 입력 변환을 정의할 수 있습니다. 모든 필드를 필터링할 수 있는 것은 아닙니다. 필터링할 수 있는 필드에 대한 자세한 내용은 ??? 섹션을 참조하세요.

표준 대기열


```
[
  {
    "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 대기열

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

조정 및 처리

표준 대기열의 경우 긴 폴링을 EventBridge 사용하여 대기열이 활성화될 때까지 대기열을 폴링합니다. 메시지가 준비되면 최대 5개의 배치를 EventBridge 읽고 파이프로 보냅니다. 메시지를 계속 사용할 수 있는 경우 배치를 읽는 프로세스 수를 분당 최대 300개 더 EventBridge 늘리십시오. 파이프가 동시에 처리할 수 있는 최대 배치 수는 1,000개입니다.

FIFO 대기열의 경우 수신한 순서대로 파이프로 메시지를 EventBridge 보냅니다. FIFO 대기열에 메시지를 전송할 때 메시지 그룹 ID를 지정합니다. Amazon SQS를 사용하면 동일한 그룹의 메시지를 순서대로 쉽게 전송할 수 EventBridge 있습니다. EventBridge 수신된 메시지를 그룹으로 정렬하고 한 그룹에 대해 한 번에 한 배치만 전송합니다. 파이프에서 오류가 반환되면 파이프는 동일한 그룹으로부터 추가 메시지를 EventBridge 수신하기 전에 영향을 받은 메시지에 대해 모든 재시도를 시도합니다.

파이프와 함께 EventBridge 사용할 대기열 구성

파이프의 소스로 제공할 Amazon SQS 대기열을 생성합니다. 그런 다음 파이프가 각 이벤트 배치를 처리할 시간을 허용하고 규모가 커지면 제한 오류에 EventBridge 대응하여 재시도할 수 있도록 대기열을 구성합니다.

각 레코드 배치를 처리하는 파이프 시간을 허용하려면 소스 대기열의 가시성 제한 시간을 파이프 강화 및 대상 구성 요소의 총 런타임의 6배 이상으로 설정합니다. 이전 배치를 처리하는 동안 파이프가 병목 EventBridge 현상이 발생하는 경우 추가 시간을 통해 재시도할 수 있습니다.

파이프가 메시지를 여러 번 처리하지 못하면 Amazon SQS에서 메시지를 DLQ(Dead Letter Queue)로 보낼 수 있습니다. 파이프에서 오류가 반환되면 해당 오류를 대기열에 EventBridge 보관하세요. 가시성 시간 초과가 발생하면 EventBridge은(는) 메시지를 다시 수신합니다. 여러 번 수신한 후 메시지를 두 번째 대기열로 보내려면 소스 대기열에 DLQ(Dead Letter Queue)를 구성합니다.

> **Note**
>
> 파이프가 아닌 소스 대기열에서 DLQ(Dead Letter Queue)를 구성해야 합니다. 파이프에 구성한 DLQ(Dead Letter Queue)는 소스 대기열이 아닌 파이프의 비동기 간접 호출 대기열에 사용됩니다.

파이프가 오류를 반환하거나 최대 동시성 상태이기 때문에 간접 호출할 수 없는 경우에는 추가 시도로 처리에 성공할 수 있습니다. 메시지를 DLQ(Dead Letter Queue)로 보내기 전에 메시지를 처리할 수 있는 기회를 더 많이 제공하려면 원본 대기열의 리드라이브 정책에서 `maxReceiveCount`를 5이상으로 설정합니다.

배치 항목 실패 보고

소스에서 스트리밍 데이터를 EventBridge 사용하고 처리할 때는 기본적으로 일괄 처리의 가장 높은 시퀀스 번호로 체크포인트를 지정하지만 일괄 처리가 완료될 때만 검사합니다. 실패한 배치에서 정상 처리된 메시지를 재처리하지 않으려면 성공한 메시지와 실패한 메시지를 나타내는 객체를 반환하도록 보강 또는 대상을 구성하면 됩니다. 이를 부분 배치 응답이라고 합니다.

자세한 설명은 ??? 섹션을 참조하세요.

성공 및 실패 조건

다음 중 하나를 반환하면 배치를 완전히 성공한 것으로 EventBridge 간주합니다.

- 비어 있는 batchItemFailure 목록
- null batchItemFailure 목록
- 비어 있는 EventResponse
- null EventResponse

다음 중 하나를 반환하면 배치를 완전한 실패로 EventBridge 간주합니다.

- 빈 문자열 itemIdentifier
- null itemIdentifier
- 키 이름이 잘못된 itemIdentifier

EventBridge 재시도 전략에 따라 실패를 재시도합니다.

아마존 EventBridge 파이프 필터링

EventBridge Pipes를 사용하면 특정 소스의 이벤트를 필터링하고 그 중 일부만 처리할 수 있습니다. 이 필터링은 이벤트 패턴을 사용하여 EventBridge 이벤트 버스 또는 Lambda 이벤트 소스 매핑에서 필터링하는 것과 동일한 방식으로 작동합니다. 이벤트 패턴에 대한 자세한 내용은 ??? 섹션을 참조하세요.

필터 기준 FilterCriteria 객체는 필터(Filters) 목록으로 구성된 구조입니다. 각 필터는 필터링 패턴(Pattern)을 정의하는 구조입니다. Pattern은 JSON 필터 규칙의 문자열 표현입니다. FilterCriteria 객체는 다음 예제와 같은 형태를 가집니다.

```
{
  "Filters": [
    {"Pattern": "{ \"Metadata1\": [ rule1 ], \"data\": { \"Data1\": [ rule2 ] }}"
    }
  ]
}
```

명확성을 더하기 위해 일반 JSON으로 확장된 Pattern 필터의 값은 다음과 같습니다.

```
{
  "Metadata1": [ pattern1 ],
  "data": {"Data1": [ pattern2 ]}
}
```

FilterCriteria 객체의 주요 부분은 메타데이터 속성 및 데이터 속성입니다.

- 메타데이터 속성은 이벤트 객체의 필드입니다. 예제에서 FilterCriteria.Metadata1은 메타데이터 속성을 가리킵니다.
- 데이터 속성은 이벤트 본문의 필드입니다. 예제에서 FilterCriteria.Data1은 데이터 속성을 가리킵니다.

예를 들어 Kinesis 스트림에 다음과 같은 이벤트가 포함되어 있다고 가정해 보겠습니다.

```
{
  "kinesisSchemaVersion": "1.0",
  "partitionKey": "1",
  "sequenceNumber": "49590338271490256608559692538361571095921575989136588898",
  "data": {"City": "Seattle",
    "State": "WA",
    "Temperature": "46",
    "Month": "December"
  },
  "approximateArrivalTimestamp": 1545084650.987
}
```

이벤트가 파이프를 통과하여 흐르면 data 필드가 base64로 인코딩되어 다음과 같이 표시됩니다.

```
{
  "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 이벤트의 메타데이터 속성은 data 객체 외부의 모든 필드(예: partitionKey 또는 sequenceNumber)입니다.

Kinesis 이벤트의 데이터 속성은 data 객체 내부의 필드(예: City또는 Temperature)입니다.

이 이벤트와 일치하도록 필터링하면 디코딩된 필드에 필터를 사용하면 됩니다. 예를 들어 partitionKey 및 City를 필터링하려면 다음 필터를 사용합니다.

```
{
  "partitionKey": [
    "1"
  ],
  "data": {
    "City": [
      "Seattle"
    ]
  }
}
```

이벤트 필터를 생성할 때 Pipes는 이벤트 EventBridge 콘텐츠에 액세스할 수 있습니다. 이 콘텐츠는 Amazon SQS body 필드와 같이 JSON으로 이스케이프 처리되거나 Kinesis data 필드와 같이 base64로 인코딩됩니다. 데이터가 유효한 JSON인 경우 대상 매개 변수의 입력 템플릿 또는 JSON 경로가 콘텐츠를 직접 참조할 수 있습니다. 예를 들어 Kinesis 이벤트 소스가 유효한 JSON인 경우 <$.data.someKey>를 사용하여 변수를 참조할 수 있습니다.

이벤트 패턴을 생성할 때 폴링 작업으로 추가된 필드가 아니라 소스 API에서 보낸 필드를 기준으로 필터링할 수 있습니다. 다음 필드는 이벤트 패턴에 사용할 수 없습니다.

- awsRegion
- eventSource
- eventSourceARN
- eventVersion
- eventID
- eventName
- invokeIdentityArn
- eventSourceKey

메시지 및 데이터 필드

모든 EventBridge Pipe 소스에는 핵심 메시지 또는 데이터가 포함된 필드가 있습니다. 이를 메시지 필드 또는 데이터 필드라고 합니다. 이러한 필드는 JSON으로 이스케이프 처리되거나 base64로 인코딩

될 수 있기 때문에 특별하지만, 유효한 JSON인 경우 본문이 이스케이프되지 않은 것처럼 JSON 패턴으로 필터링할 수 있습니다. 이러한 필드의 내용은 입력 변환기에서도 원활하게 사용할 수 있습니다.

Amazon SQS 메시지를 올바르게 필터링

Amazon SQS 메시지가 필터 기준을 충족하지 않는 경우 대기열에서 메시지를 EventBridge 자동으로 제거합니다. 이러한 메시지를 Amazon SQS에서 수동으로 삭제할 필요가 없습니다.

Amazon SQS 경우, 메시지 body는 임의의 문자열이 될 수 있습니다. 그러나 `FilterCriteria`에서 유효한 JSON 형식의 body를 기대하는 경우 문제가 될 수 있습니다. 반대 시나리오도 마찬가지입니다. 수신 메시지 body가 유효한 JSON 형식이지만 필터 기준이 body를 일반 문자열로 예상하는 경우 의도하지 않은 동작이 발생할 수 있습니다.

이 문제를 방지하려면 `FilterCriteria`에서 body의 형식이 대기열에서 받은 메시지의 body의 예상 형식과 일치하는지 확인합니다. 메시지를 필터링하기 전에 수신 메시지의 body 형식과 필터 패턴을 EventBridge 자동으로 평가합니다. body 일치하지 않는 부분이 있는 경우 메시지를 EventBridge 삭제합니다. 다음 표에는 이 평가가 요약되어 있습니다.

수신 메시지 **body** 형식	필터 패턴 **body** 형식	결과적 작업
일반 문자열	일반 문자열	EventBridge 필터 기준에 따라 필터링합니다.
일반 문자열	데이터 속성에 대한 필터 패턴 없음	EventBridge 필터 기준에 따라 필터링합니다 (다른 메타데이터 속성에만 해당).
일반 문자열	유효한 JSON	EventBridge 메시지를 삭제합니다.
유효한 JSON	일반 문자열	EventBridge 메시지를 삭제합니다.
유효한 JSON	데이터 속성에 대한 필터 패턴 없음	EventBridge 필터 기준에 따라 필터링합니다 (다른 메타데이터 속성에만 해당).
유효한 JSON	유효한 JSON	EventBridge 필터 기준에 따라 필터링합니다.

body일부로 포함하지 않는 경우 이 검사를 EventBridge 건너뛰십시오`FilterCriteria`.

Kinesis 및 DynamoDB 메시지를 올바르게 필터링

필터 기준에서 Kinesis 또는 DynamoDB 레코드를 처리한 후 스트림 반복기가 이 레코드를 너머 진행됩니다. 레코드가 필터 조건을 충족하지 않으면 이벤트 소스에서 레코드를 수동으로 삭제할 필요가 없습니다. 보존 기간이 지나면 Kinesis 및 DynamoDB는 이러한 이전 레코드를 자동으로 삭제합니다. 레코드를 더 빨리 삭제하려면 데이터 보존 기간 변경을 참조하세요.

스트림 이벤트 소스에서 이벤트를 올바르게 필터링하려면 데이터 필드와 데이터 필드의 필터 기준이 모두 유효한 JSON 형식이어야 합니다. (Kinesis의 경우 데이터 필드는 `data`이고 DynamoDB의 경우 데이터 필드는 `dynamodb`입니다.) 필드 중 하나가 유효한 JSON 형식이 아닌 경우 메시지를 EventBridge 삭제하거나 예외가 발생합니다. 다음 표에는 특정 동작이 요약되어 있습니다.

수신 데이터 형식(**`data`** 또는 **`dynamodb`**)	데이터 속성에 대한 필터 패턴 형식	결과적 작업
유효한 JSON	유효한 JSON	EventBridge 필터 기준에 따라 필터링합니다.
유효한 JSON	데이터 속성에 대한 필터 패턴 없음	EventBridge 필터 기준에 따라 필터링합니다 (다른 메타데이터 속성에만 해당).
유효한 JSON	JSON 외	EventBridge 파이프 또는 업데이트 시 예외가 발생합니다. 데이터 속성에 대한 필터 패턴은 유효한 JSON 형식이어야 합니다.
JSON 외	유효한 JSON	EventBridge 레코드를 삭제합니다.
JSON 외	데이터 속성에 대한 필터 패턴 없음	EventBridge 필터 기준에 따라 필터링합니다 (다른 메타데이터 속성에만 해당).
JSON 외	JSON 외	EventBridge 파이프 생성 또는 업데이트 시 예외가 발생합니

수신 데이터 형식(**data** 또는 **dynamodb**)	데이터 속성에 대한 필터 패턴 형식	결과적 작업
		다. 데이터 속성에 대한 필터 패턴은 유효한 JSON 형식이어야 합니다.

Amazon Managed Streaming for Apache Kafka, 자체 관리형 Apache Kafka, Amazon MQ 메시지를 올바르게 필터링

Amazon MQ 소스의 경우 메시지 필드는 `data`입니다. Apache Kafka 소스(Amazon MSK 및 자체 관리형 Apache Kafka)의 경우 `key` 및 `value`라는 두 개의 메시지 필드가 있습니다.

EventBridge 필터에 포함된 모든 필드와 일치하지 않는 메시지를 삭제합니다. Apache Kafka의 경우 함수를 성공적으로 호출한 후 일치하는 메시지와 일치하지 않는 메시지에 대해 오프셋을 EventBridge 커밋합니다. Amazon MQ의 경우 함수를 성공적으로 호출한 후 일치하는 메시지를 확인하고, EventBridge 일치하지 않는 메시지를 필터링할 때 이를 확인합니다.

Apache Kafka와 Amazon MQ 메시지는 UTF-8 인코딩 문자열이어야 하며, 일반 문자열이거나 JSON 형식이어야 합니다. 필터 기준을 적용하기 전에 Apache Kafka와 Amazon MQ 바이트 배열을 UTF-8 바이트로 EventBridge 디코딩하기 때문입니다. 메시지가 UTF-16 또는 ASCII 같은 다른 인코딩을 사용하거나 메시지 형식이 형식과 일치하지 않는 경우 메타데이터 필터만 처리합니다. `FilterCriteria` EventBridge 다음 표에는 특정 동작이 요약되어 있습니다.

수신 메시지 형식 (**data** 또는 **key** 및**value**)	메시지 속성에 대한 필터 패턴 형식	결과적 작업
일반 문자열	일반 문자열	EventBridge 필터 기준에 따라 필터링합니다.
일반 문자열	데이터 속성에 대한 필터 패턴 없음	EventBridge 필터 기준에 따라 필터링합니다 (다른 메타데이터 속성에만 해당).
일반 문자열	유효한 JSON	EventBridge 필터 기준에 따라 필터링합니다 (다른 메타데이터 속성에만 해당).

수신 메시지 형식 (**data** 또는 **key** 및**value**)	메시지 속성에 대한 필터 패턴 형식	결과적 작업
유효한 JSON	일반 문자열	EventBridge 필터 기준에 따라 필터링합니다 (다른 메타데이터 속성에만 해당).
유효한 JSON	데이터 속성에 대한 필터 패턴 없음	EventBridge 필터 기준에 따라 필터링합니다 (다른 메타데이터 속성에만 해당).
유효한 JSON	유효한 JSON	EventBridge 필터 기준에 따라 필터링합니다.
UTF-8이 아닌 인코딩 문자열	JSON, 일반 문자열 또는 패턴 없음	EventBridge 필터 기준에 따라 필터링합니다 (다른 메타데이터 속성에만 해당).

Lambda ESM과 파이프의 차이점 EventBridge

이벤트를 필터링할 때 Lambda EventBridge ESM과 Pipes는 일반적으로 같은 방식으로 작동합니다. 주요 차이점은 ESM 페이로드에 `eventSourceKey` 필드가 없다는 것입니다.

Amazon EventBridge 파이프 이벤트 보강

EventBridge 파이프의 보강 단계에 따라 소스의 데이터를 대상으로 전송하기 전에 개선할 수 있습니다. 예를 들어 전체 티켓 데이터가 포함되지 않은 티켓 생성 이벤트를 수신할 수 있습니다. 보강을 사용하면 Lambda 함수로 `get-ticket` API를 호출하여 전체 티켓 세부 정보를 확인할 수 있습니다. 그 후 파이프가 해당 정보를 대상으로 전송할 수 있습니다.

EventBridge에서 파이프를 설정할 때 다음 보강을 구성할 수 있습니다.

- API 대상
- Amazon API Gateway
- Lambda 함수
- Step Functions 상태 시스템

> **Note**
>
> EventBridge 파이프는 Express 워크플로만 보강으로 지원합니다.

EventBridge는 대상을 간접 호출하기 전에 보강의 응답을 기다려야 하기 때문에 보강을 동기식으로 간접 호출합니다.

보강 응답은 최대 6MB 크기로 제한됩니다.

개선을 위해 데이터를 전송하기 전에 소스에서 수신한 데이터를 변환할 수도 있습니다. 자세한 내용은 ??? 섹션을 참조하세요.

보강을 사용하여 이벤트 필터링

EventBridge 파이프는 보강 응답을 구성된 대상에 직접 전달합니다. 여기에는 배치를 지원하는 대상에 대한 배열 응답이 포함됩니다. 일괄 처리 동작에 대한 자세한 내용은 ??? 섹션을 참조하세요. 또한 보강을 필터로 사용하고 소스에서 수신한 것보다 적은 수의 이벤트를 전달할 수 있습니다. 대상을 간접 호출하지 않으려면 빈 응답(예: "", {} 또는 [])을 반환합니다.

> **Note**
>
> 빈 페이로드로 대상을 간접 호출하려면 빈 JSON `[{}]`이 포함된 배열을 반환합니다.

보강 간접 호출

EventBridge는 대상을 간접 호출하기 전에 보강의 응답을 기다려야 하기 때문에 보강을 동기식으로 간접 호출합니다(간접 호출 유형은 `REQUEST_RESPONSE`로 설정됨).

> **Note**
>
> Step Functions 상태 시스템의 경우 EventBridge는 동기식으로 간접 호출될 수 있는 Express 워크플로만 보강으로 지원합니다.

아마존 EventBridge 파이프 타겟

파이프의 데이터를 특정 대상으로 보낼 수 있습니다. 에서 파이프를 설정할 때 다음 대상을 구성할 수 있습니다 EventBridge.

- API 대상
- API Gateway
- 배치 작업 대기열
- CloudWatch 로그 그룹
- ECS 태스크
- 동일한 계정 및 리전의 이벤트 버스
- Firehose 전송 스트림
- Inspector 평가 템플릿
- Kinesis 스트림
- Lambda 함수(SYNC 또는 ASYNC)
- Redshift 클러스터 데이터 API 쿼리
- SageMaker 파이프라인
- Amazon SNS 주제(SNS FIFO 주제는 지원되지 않음)
- Amazon SQS 대기열
- Step Functions 상태 시스템
  - Express 워크플로(SYNC 또는 ASYNC)
  - 표준 워크플로(ASYNC)
- Timestream LiveAnalytics 테이블용

대상 파라미터

일부 대상 서비스는 이벤트 페이로드를 대상에 보내지 않고 대신 이벤트를 특정 API를 호출하는 트리거로 취급합니다. EventBridge `PipeTargetParameters`를 사용하여 해당 API로 전송할 정보를 지정합니다. 여기에는 다음이 포함됩니다.

- API 대상(API 대상으로 전송되는 데이터는 API의 구조와 일치해야 합니다. `InputTemplate` 객체를 사용하여 데이터가 올바르게 구조화되었는지 확인해야 합니다. 원본 이벤트 페이로드를 포함하려면 `InputTemplate`에서 해당 페이로드를 참조하세요.)

- API Gateway(API Gateway로 전송되는 데이터는 API의 구조와 일치해야 합니다. `InputTemplate` 객체를 사용하여 데이터가 올바르게 구조화되었는지 확인해야 합니다. 원본 이벤트 페이로드를 포함하려면 `InputTemplate`에서 해당 페이로드를 참조하세요.)
- `PipeTargetRedshiftDataParameters`(Amazon Redshift Data API 클러스터)
- `PipeTargetSageMakerPipelineParameters`(Amazon SageMaker 런타임 모델 구축 파이프라인)
- `PipeTargetBatchJobParameters` (AWS Batch)

> **Note**
>
> EventBridge 모든 JSON Path 구문을 지원하지는 않으며 런타임에 이를 평가합니다. 지원되는 구문은 다음과 같습니다.
>
> - 점 표기법(예: `$.detail`)
> - 대시
> - 밑줄
> - 영숫자
> - 배열 인덱스
> - 와일드카드(*)

동적 경로 파라미터

EventBridge 파이프 대상 매개변수는 선택적 동적 JSON 경로 구문을 지원합니다. 이 구문을 사용하여 정적 값 대신 JSON 경로를 지정할 수 있습니다(예: `$.detail.state`). 전체 값은 일부가 아닌 JSON 경로여야 합니다. 예를 들어 `RedshiftParameters.Sql`은 `$.detail.state`일 수 있지만 `"SELECT * FROM $.detail.state"`일 수는 없습니다. 이러한 경로는 런타임 시 지정된 경로에 있는 이벤트 페이로드 자체의 데이터로 동적으로 대체됩니다. 동적 경로 파라미터는 입력 변환으로 생성된 새 값이나 변환된 값을 참조할 수 없습니다. 동적 파라미터 JSON 경로에 지원되는 구문은 입력을 변환할 때와 동일합니다. 자세한 정보는 ???을 참조하세요.

동적 구문은 다음을 제외하고 모든 EventBridge Pipes 보강 및 대상 매개변수의 모든 문자열, 비열거형 필드에 사용할 수 있습니다.

- `PipeTargetCloudWatchLogsParameters.LogStreamName`
- `PipeTargetEventBridgeEventBusParameters.EndpointId`

- PipeEnrichmentHttpParameters.HeaderParameters
- PipeTargetHttpParameters.HeaderParameters

예를 들어 파이프 Kinesis 대상을 소스 이벤트의 사용자 지정 키로 설정하려면 를 설정하십시오. PartitionKey KinesisTargetParameter PartitionKey다음으로:

- Kinesis 소스의 경우 "$.data.*someKey*"
- Amazon SQS 소스의 경우 "$.body.*someKey*"

그런 다음 이벤트 페이로드가 유효한 JSON 문자열 (예:) 이면 JSON 경로에서 값을 EventBridge 추출하여 대상 파라미터로 사용합니다. {"*someKey*":"*someValue*"} 이 예제에서는 EventBridge Kinesis를 "*SomeValue* “PartitionKey로 설정합니다.

권한

소유한 리소스에서 API를 호출하려면 Pipes에 적절한 권한이 EventBridge 있어야 합니다. EventBridge PIPES는 파이프에서 지정한 IAM 역할을 사용하여 강화에 사용하고 IAM 보안 주체를 사용하여 호출을 타겟팅합니다. pipes.amazonaws.com

간접 호출 대상

EventBridge 다음과 같은 방법으로 대상을 호출할 수 있습니다.

- 동기적으로 (호출 유형이 로 REQUEST_RESPONSE 설정됨) - 계속하기 전에 대상의 응답을 EventBridge 기다립니다.
- 비동기적으로 (호출 유형을 로 설정FIRE_AND_FORGET) - 응답을 EventBridge 기다리지 않고 계속 진행합니다.

기본적으로 순서가 지정된 소스가 있는 파이프의 경우 다음 이벤트로 진행하기 전에 대상의 응답이 필요하므로 대상을 동기적으로 EventBridge 호출합니다.

표준 Amazon SQS 대기열과 같은 소스가 주문을 적용하지 않는 경우 지원되는 대상을 동기 또는 EventBridge 비동기 방식으로 호출할 수 있습니다.

Lambda 함수 및 Step Functions 상태 머신을 사용하여 간접 호출 유형을 구성할 수 있습니다.

> **Note**
>
> Step Functions 상태 머신의 경우 표준 워크플로를 비동기식으로 간접 호출해야 합니다.

EventBridge 파이프, 대상 세부 사항

AWS Batch 작업 대기열

모든 AWS Batch `submitJob` 매개변수는 모든 Pipe 매개변수로 명시적으로 구성되며`BatchParameters`, 모든 Pipe 매개변수와 마찬가지로 수신 이벤트 페이로드에 대한 JSON 경로를 사용하여 동적으로 구성할 수 있습니다.

CloudWatch 로그 그룹

입력 변환기를 사용하든 사용하지 않든, 이벤트 페이로드는 로그 메시지로 사용됩니다. `PipeTarget`의 `CloudWatchLogsParameters`를 통해 `Timestamp`(또는 대상의 명시적인 `LogStreamName`)를 설정할 수 있습니다. 모든 파이프 파라미터와 마찬가지로 수신 이벤트 페이로드에 대한 JSON 경로를 사용할 때 이러한 파라미터는 동적일 수 있습니다.

Amazon ECS 태스크

모든 Amazon ECS `runTask` 파라미터는 `EcsParameters`를 통해 명시적으로 구성됩니다. 모든 파이프 파라미터와 마찬가지로 수신 이벤트 페이로드에 대한 JSON 경로를 사용할 때 이러한 파라미터는 동적일 수 있습니다.

Lambda 함수 및 Step Functions 워크플로

Lambda 및 Step Functions에는 배치 API가 없습니다. 파이프 소스의 이벤트 배치를 처리하기 위해 배치는 JSON 배열로 변환되어 Lambda 또는 Step Functions 대상에 대한 입력으로 전달됩니다. 자세한 정보는 ???을 참조하세요.

Timestream LiveAnalytics 테이블용

Timestream for LiveAnalytics 테이블을 파이프 타겟으로 지정할 때 고려할 사항은 다음과 같습니다.

- Apache Kafka 스트림 (제공 업체Amazon MSK 또는 타사 공급자 포함) 은 현재 파이프 소스로 지원되지 않습니다.
- a Kinesis 또는 DynamoDB 스트림을 파이프 소스로 지정한 경우 재시도 횟수를 지정해야 합니다.

자세한 내용은 ???을(를) 참조하세요.

Amazon EventBridge Pipes 일괄 처리 및 동시성

일괄 처리 동작

EventBridge Pipes는 소스에서 이를 지원하는 대상으로의 일괄 처리를 지원합니다. 또한 AWS Lambda 및 AWS Step Functions에 대해 보강을 위한 일괄 처리가 지원됩니다. 서비스마다 지원하는 일괄 처리 수준이 다르기 때문에 대상에서 지원하는 것보다 더 큰 배치 크기로 파이프를 구성할 수 없습니다. 예를 들어 Amazon Kinesis 스트림 소스는 최대 10,000개의 레코드 배치 크기를 지원하지만 Amazon Simple Queue Service는 배치당 메시지를 최대 10개까지 대상으로 지원합니다. 따라서 Kinesis 스트림에서 Amazon SQS 대기열로의 파이프는 소스에서 구성된 최대 배치 크기가 10개일 수 있습니다.

일괄 처리를 지원하지 않는 보강 또는 대상으로 파이프를 구성하면 소스에서 일괄 처리를 활성화할 수 없습니다.

소스에서 일괄 처리가 활성화되면 JSON 레코드 배열이 파이프를 통해 전달된 후 지원되는 보강 또는 대상의 배치 API에 매핑됩니다. 입력 변환기는 배열 전체가 아닌 배열의 각 개별 JSON 레코드에 별도로 적용됩니다. 이러한 배열의 예를 보려면 ??? 섹션을 참조하여 특정 소스를 선택하세요. 파이프는 배치 크기가 1인 경우에도 지원되는 보강 또는 대상에 배치 API를 사용합니다. 보강 또는 대상에 배치 API가 없지만 Lambda 및 Step Functions와 같은 전체 JSON 페이로드를 수신하는 경우, 전체 JSON 배열이 요청 한 번으로 전송됩니다. 배치 크기가 1인 경우에도 요청은 JSON 배열로 전송됩니다.

파이프가 소스에서 일괄 처리되도록 구성되어 있고 대상이 일괄 처리를 지원하는 경우, 보강에서 JSON 항목의 배열을 반환할 수 있습니다. 이 배열에는 원본 소스보다 짧거나 긴 배열이 포함될 수 있습니다. 그러나 배열이 대상에서 지원하는 배치 크기보다 크면 파이프가 대상을 간접 호출하지 않습니다.

지원되는 배치 가능 대상

대상	최대 배치 크기
CloudWatch 로그	10,000개
EventBridge 이벤트 버스	10
Firehose 스트림	500

대상	최대 배치 크기
Kinesis 스트림	500
Lambda 함수	고객 정의
Step Functions 상태 시스템	고객 정의
Amazon SNS 주제	10
Amazon SQS 대기열	10

다음 보강 및 대상은 처리를 위해 전체 배치 이벤트 페이로드를 수신하며 배치 크기가 아닌 이벤트의 총 페이로드 크기로 제한됩니다.

- Step Functions 상태 시스템(262,144자)
- Lambda 함수(6MB)

부분적 배치 실패

Amazon SQS 및 스트림 소스 (예: Kinesis 및 DynamoDB) 의 경우 Pipes는 대상 장애에 대한 부분 배치 EventBridge 장애 처리를 지원합니다. 대상이 일괄 처리를 지원하는데 일괄 처리 중 일부만 성공하면 나머지 페이로드에 대한 일괄 처리를 EventBridge 자동으로 재시도합니다. 가장 up-to-date 풍부한 콘텐츠의 경우 구성된 보강을 다시 호출하는 것을 포함하여 전체 파이프를 통해 이러한 재시도가 이루어집니다.

보강에 대한 부분적 배치 장애 처리는 지원되지 않습니다.

Lambda 및 Step Functions 대상의 경우 대상에서 구조가 정의된 페이로드를 반환하여 부분 장애를 지정할 수도 있습니다. 이는 재시도해야 하는 이벤트를 나타냅니다.

부분 장애 페이로드 구조 예시

```
{
  "batchItemFailures": [
    {
      "itemIdentifier": "id2"
    },
    {
```

```
      "itemIdentifier": "id4"
    }
]
```

이 예시에서 `itemIdentifier`는 대상이 원본 소스에서 처리한 이벤트의 ID와 일치합니다. Amazon SQS의 경우 `messageId`입니다. Kinesis와 DynamoDB의 경우 `eventID`입니다. EventBridge Pipes가 대상의 부분 배치 장애를 적절하게 처리하려면 보강에서 반환되는 모든 배열 페이로드에 이러한 필드를 포함해야 합니다.

처리량 및 동시성 동작

파이프가 수신하여 보강 또는 대상으로 이동하는 모든 이벤트 또는 이벤트 배치는 파이프 실행으로 간주됩니다. `STARTED` 상태에 있는 파이프는 소스의 이벤트를 지속적으로 폴링하여 사용 가능한 백로그와 구성된 일괄 처리 설정에 따라 규모를 확장하거나 축소합니다.

동시 파이프 실행에 대한 할당량, 계정 및 리전별 파이프 수에 대한 내용은 ??? 섹션을 참조하세요.

기본적으로 단일 파이프는 소스에 따라 다음과 같은 최대 동시 실행 수로 확장됩니다.

- DynamoDB - 동시 실행은 파이프에 구성된 `ParallelizationFactor`에 스트림의 샤드 수를 곱한 만큼 증가할 수 있습니다.
- Apache Kafka - 동시 실행은 주제에 대한 파티션 수를 최대 1,000개까지 늘릴 수 있습니다.
- Kinesis - 동시 실행은 파이프에 구성된 `ParallelizationFactor`에 스트림의 샤드 수를 곱한 만큼 증가할 수 있습니다.
- Amazon MQ – 5
- Amazon SQS – 1,250

최대 폴링 처리량 또는 동시성 한도를 높이기 위한 요구 사항이 있는 경우 지원팀에 문의하세요.

> **Note**
>
> 실행 한도는 최선의 안전 제한으로 간주됩니다. 폴링이 이러한 값 아래로 제한되지 않지만 파이프나 계정이 해당 권장 값보다 높게 버스트될 수 있습니다.

파이프 실행은 보강 및 대상 처리를 포함하여 최대 5분으로 제한됩니다. 현재 이 한도는 늘릴 수 없습니다.

Amazon SQS FIFO 대기열, Kinesis 및 DynamoDB Streams, Apache Kafka 주제 등 소스가 엄격하게 정렬된 파이프는 FIFO 대기열의 메시지 그룹 ID 수 또는 Kinesis 대기열의 샤드 수와 같은 소스 구성에 따라 동시성이 추가로 제한됩니다. 이러한 제약 조건 내에서 순서가 엄격하게 보장되므로 정렬된 소스가 있는 파이프는 이러한 동시성 한도를 초과할 수 없습니다.

Amazon EventBridge Pipes 입력 변환

Amazon EventBridge 파이프는 데이터를 보강과 대상으로 전달할 때 선택적 입력 변환기를 지원합니다. 입력 변환기를 사용하여 보강 또는 대상 서비스의 요구 사항에 맞게 JSON 이벤트 입력 페이로드를 재구성할 수 있습니다. Amazon API Gateway 및 API 대상의 경우 다음과 같이 입력 이벤트를 API의 RESTful 모델에 맞출 수 있습니다. 입력 변환기는 `InputTemplate` 파라미터로 모델링됩니다. 자유 텍스트, 이벤트 페이로드의 JSON 경로 또는 이벤트 페이로드에 대한 인라인 JSON 경로를 포함하는 JSON 객체일 수 있습니다. 보강을 위해 이벤트 페이로드는 소스에서 가져옵니다. 대상의 경우 파이프에 구성된 경우 보강에서 반환되는 것이 이벤트 페이로드입니다. 이벤트 페이로드의 서비스별 데이터 외에도 `InputTemplate`의 예약 변수를 사용하여 파이프의 데이터를 참조할 수 있습니다.

배열의 항목에 액세스하려면 대괄호 표기법을 사용하십시오.

> **Note**
>
> EventBridge는 일부 JSON 경로 구문을 지원하며 런타임 시 이를 평가합니다. 지원되는 구문은 다음과 같습니다.
>
> - 점 표기법(예: `$.detail`)
> - 대시
> - 밑줄
> - 영숫자
> - 배열 인덱스
> - 와일드카드(*)

다음은 Amazon SQS 이벤트 페이로드를 참조하는 샘플 `InputTemplate` 파라미터입니다.

정적 문자열

```
InputTemplate: "Hello, sender"
```

JSON 경로

```
InputTemplate: <$.attributes.SenderId>
```

동적 문자열

```
InputTemplate: "Hello, <$.attributes.SenderId>"
```

정적 JSON

```
InputTemplate: >
{
  "key1": "value1",
  "key2": "value2",
  "key3": "value3",
}
```

다이나믹 JSON

```
InputTemplate: >
{
  "key1": "value1"
  "key2": <$.body.key>,
  "d": <aws.pipes.event.ingestion-time>
}
```

대괄호 표기법을 사용하여 배열의 항목에 액세스합니다.

```
InputTemplate: >
{
  "key1": "value1"
  "key2": <$.body.Records[3]>,
  "d": <aws.pipes.event.ingestion-time>
}
```

> **Note**
>
> EventBridge는 유효한 JSON 출력을 보장하기 위해 런타임 시 입력 변환기를 대체합니다. 따라서 JSON 경로 파라미터를 참조하는 변수는 따옴표로 묶고 JSON 객체 또는 배열을 참조하는 변수는 따옴표로 묶지 마세요.

예약된 변수

입력 템플릿은 다음 예약된 변수를 사용할 수 있습니다.

- <aws.pipes.pipe-arn> - 파이프의 Amazon 리소스 이름(ARN)입니다.
- <aws.pipes.pipe-name> - 파이프의 이름입니다.
- <aws.pipes.source-arn> – 파이프의 이벤트 소스의 ARN입니다.
- <aws.pipes.enrichment-arn> – 파이프의 보강에 대한 ARN입니다.
- <aws.pipes.target-arn> – 파이프의 대상 ARN입니다.
- <aws.pipes.event.ingestion-time> – 입력 변환기가 이벤트를 수신한 시간입니다. 이는 ISO 8601 타임스탬프입니다. 이 시간은 보강이 이벤트 처리를 완료한 시기에 따라 보강 입력 변환기와 대상 입력 변환기의 시간이 다릅니다.
- <aws.pipes.event> – 입력 변환기가 수신한 이벤트입니다.

  보강 입력 변환기의 경우 이는 소스에서 발생한 이벤트입니다. 여기에는 소스의 원본 페이로드와 추가적인 서비스별 메타데이터가 포함됩니다. 서비스 관련 예제는 ??? 섹션을 참조하세요.

  대상 입력 변환기의 경우 추가 메타데이터 없이 보강(구성된 경우)에서 반환되는 이벤트입니다. 따라서 보강 반환 페이로드는 JSON이 아닐 수 있습니다. 파이프에 보강이 구성되어 있지 않은 경우 메타데이터가 있는 소스에서 발생한 이벤트입니다.
- <aws.pipes.event.json> - aws.pipes.event와 같지만 원본 페이로드(소스에서 또는 보강을 통해 반환됨)가 JSON인 경우에만 변수에 값이 있습니다. 파이프에 인코딩된 필드(예: Amazon SQS body 필드 또는 Kinesis data)가 있는 경우 해당 필드는 디코딩되어 유효한 JSON으로 바뀝니다. 이스케이프되지 않기 때문에 변수는 JSON 필드의 값으로만 사용할 수 있습니다. 자세한 내용은 ??? 섹션을 참조하세요.

입력 변환 예제

다음은 샘플 이벤트로 사용할 수 있는 Amazon EC2 이벤트의 예입니다.

```
{
  "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"
  }
}
```

다음 JSON을 변환기로 사용하겠습니다.

```
{
  "instance" : <$.detail.instance-id>,
  "state": <$.detail.state>,
  "pipeArn" : <aws.pipes.pipe-arn>,
  "pipeName" : <aws.pipes.pipe-name>,
  "originalEvent" : <aws.pipes.event.json>
}
```

결과 출력은 다음과 같습니다.

```
{
  "instance" : "i-0123456789",
  "state": "RUNNING",
  "pipeArn" : "arn:aws:pipe:us-east-1:123456789012:pipe/example",
  "pipeName" : "example",
  "originalEvent" : {
    ... // commented for brevity
  }
}
```

암시적 본문 데이터 구문 분석

수신 페이로드의 다음 필드는 JSON으로 이스케이프 처리되거나(예: Amazon SQS body 객체) base64로 인코딩될 수 있습니다(예: Kinesis `data` 객체). 필터링과 입력 변환 모두에서 EventBridge는 이러한 필드를 유효한 JSON으로 변환하므로 하위 값을 직접 참조할 수 있습니다. 예를 들어, Kinesis의 경우 `<$.data.someKey>`입니다.

추가 메타데이터 없이 대상이 원본 페이로드를 수신하도록 하려면 소스별 본문 데이터와 함께 입력 변환기를 사용하십시오. Amazon SQS의 경우 <$.body>, Kinesis의 경우 <$.data>를 예로 들 수 있습니다. 원본 페이로드가 유효한 JSON 문자열(예: {"key": "value"})인 경우 소스별 본문 데이터와 함께 입력 변환기를 사용하면 원본 소스 페이로드 내의 따옴표가 제거됩니다. 예를 들어, {"key": "value"}는 대상에 전달되면 "{key: value}"가 됩니다. 대상에 유효한 JSON 페이로드(예: EventBridge Lambda 또는 Step Functions)가 필요한 경우 전송이 실패합니다. 잘못된 JSON을 생성하지 않고 대상이 원본 소스 데이터를 수신하도록 하려면 소스 본문 데이터 입력 변환기를 JSON으로 래핑하십시오. 예: {"data": <$.data>}.

암시적 본문 구문 분석을 사용하여 대부분의 파이프 대상 또는 보강 파라미터의 값을 동적으로 채울 수도 있습니다. 자세한 정보는 ??? 섹션을 참조하세요.

> **Note**
>
> 원본 페이로드가 유효한 JSON인 경우 이 필드에는 이스케이프 처리되지 않고 base64로 인코딩되지 않은 JSON이 포함됩니다. 하지만 페이로드가 유효한 JSON이 아닌 경우, EventBridge는 Amazon SQS를 제외하고 아래 나열된 필드를 base64로 인코딩합니다.

- Active MQ – `data`
- Kinesis – `data`
- Amazon MSK – `key` 및 `value`
- Rabbit MQ – `data`
- 자체 관리형 Apache Kafka; – `key` 및 `value`
- Amazon SQS – body

입력 변환과 관련된 일반적인 문제

다음은 EventBridge 파이프에서 입력을 변환할 때 발생하는 몇 가지 일반적인 문제입니다.

- 문자열의 경우 따옴표가 필요합니다.
- 템플릿에 대한 JSON 경로를 만들 때 검증이 수행되지 않습니다.
- 변수를 지정하여 이벤트에 존재하지 않는 JSON 경로를 일치시키는 경우 변수가 생성되지 않기 때문에 출력에 나타나지 않습니다.
- `aws.pipes.event.json`과 같은 JSON 속성은 JSON 필드의 값으로만 사용할 수 있으며 다른 문자열에서는 인라인으로 사용할 수 없습니다.

- EventBridge는 대상에 대한 입력 템플릿을 채울 때 입력 경로에서 추출된 값을 이스케이프하지 않습니다.
- JSON 경로가 JSON 객체 또는 배열을 참조하지만 변수가 문자열에서 참조되는 경우, EventBridge는 문자열이 유효하도록 모든 내부 따옴표를 제거합니다. 예를 들어 "Body is <$.body>"를 입력하면 EventBridge가 객체에서 따옴표를 제거하게 됩니다.

  따라서 단일 JSON 경로 변수를 기반으로 JSON 객체를 출력하려면 해당 변수를 키로 배치해야 합니다. 이 예에서는 `{"body": <$.body>}`입니다.
- 문자열을 나타내는 변수에는 따옴표가 필요하지 않습니다. 따옴표가 허용되지만 EventBridge 파이프는 변환 중에 문자열 변수 값에 자동으로 따옴표를 추가하여 변환 출력이 유효한 JSON인지 확인합니다. EventBridge 파이프는 JSON 객체 또는 배열을 나타내는 변수에 따옴표를 추가하지 않습니다. JSON 객체 또는 배열을 나타내는 변수에 따옴표를 추가하지 마세요.

  예를 들어 다음 입력 템플릿에는 문자열과 JSON 객체를 모두 나타내는 변수가 포함되어 있습니다.

  ```
  {
    "pipeArn" : <aws.pipes.pipe-arn>,
    "pipeName" : <aws.pipes.pipe-name>,
    "originalEvent" : <aws.pipes.event.json>
  }
  ```

  올바른 따옴표를 사용하여 유효한 JSON을 생성합니다.

  ```
  {
    "pipeArn" : "arn:aws:events:us-east-2:123456789012:pipe/example",
    "pipeName" : "example",
    "originalEvent" : {
      ... // commented for brevity
    }
  }
  ```
- Lambda 또는 Step Functions 보강 또는 대상의 경우 배치 크기가 1이더라도 배치는 JSON 배열로 대상에 전달됩니다. 하지만 입력 변환기는 여전히 배열 전체가 아닌 JSON 배열의 개별 레코드에 적용됩니다. 자세한 내용은 ??? 섹션을 참조하세요.

로그 아마존 EventBridge 파이프

EventBridge 파이프 로깅을 사용하면 Pipes가 EventBridge 파이프 성능을 자세히 설명하는 레코드를 지원되는 AWS 서비스에 전송하도록 할 수 있습니다. 로그를 사용하면 파이프의 실행 성능을 파악하고 문제 해결 및 디버깅에 유용합니다.

Pipes가 레코드를 전달하는 로그 대상으로 다음 AWS 서비스를 선택할 수 있습니다. EventBridge

- CloudWatch 로그

  EventBridge 지정된 로그 로그 그룹에 CloudWatch 로그 레코드를 전달합니다.

  CloudWatch 로그를 사용하면 사용하는 모든 시스템, 애플리케이션 및 AWS 서비스의 로그를 확장성이 뛰어난 단일 서비스로 중앙 집중화할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 로그 그룹 및 로그 스트림 작업을 참조하십시오.
- Firehose 스트림 로그

  EventBridge Firehose 전송 스트림에 로그 레코드를 전달합니다.

  Amazon Data Firehose는 지원되는 타사 서비스 공급자가 소유한 사용자 지정 HTTP 엔드포인트 또는 HTTP 엔드포인트는 물론 특정 AWS 서비스와 같은 대상에 실시간 스트리밍 데이터를 전송하는 완전 관리형 서비스입니다. 자세한 내용은 Amazon Data Firehose 사용 설명서의 Amazon Data Firehose 전송 스트림 생성을 참조하십시오.
- Amazon S3 로그

  EventBridge 로그 레코드를 Amazon S3 객체로 지정된 버킷에 전달합니다.

  Amazon S3는 업계 최고의 확장성, 데이터 가용성, 보안 및 성능을 제공하는 객체 스토리지 서비스입니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 Amazon S3에서 객체 업로드, 다운로드 및 작업을 참조하세요.

Amazon EventBridge 파이프 로깅 작동 방식

파이프 실행은 파이프가 수신하여 보강 및/또는 대상으로 이동하는 이벤트 또는 이벤트 배치입니다. 활성화된 경우, 이벤트 배치가 처리될 때 수행하는 각 실행 단계에 대한 로그 레코드를 EventBridge 생성합니다. 레코드에 포함된 정보는 이벤트 배치(단일 이벤트 또는 최대 10,000개 이벤트)에 적용됩니다.

파이프 소스 및 대상에서 이벤트 배치 크기를 구성할 수 있습니다. 자세한 정보는 ???을 참조하세요.

각 로그 대상으로 전송되는 레코드 데이터는 동일합니다.

Amazon CloudWatch Logs 대상이 구성된 경우 모든 대상으로 전송되는 로그 레코드의 한도는 256kb입니다. 필요에 따라 필드가 잘립니다.

다음과 같은 방법으로 선택한 로그 대상으로 레코드 EventBridge 전송을 사용자 지정할 수 있습니다.

- 선택한 로그 대상으로 레코드를 EventBridge 전송하는 실행 단계를 결정하는 로그 수준을 지정할 수 있습니다. 자세한 정보는 ???을 참조하세요.
- EventBridge 파이프가 관련 있는 실행 단계의 레코드에 실행 데이터를 포함할지 여부를 지정할 수 있습니다. 이 데이터에는 다음이 포함됩니다.
  - 이벤트 배치의 페이로드
  - AWS 인리치먼트 또는 대상 서비스에 전송된 요청
  - AWS 강화 또는 대상 서비스에서 반환한 응답

  자세한 정보는 ???을 참조하세요.

EventBridge 파이프 로그 수준 지정

선택한 로그 대상으로 레코드를 EventBridge 전송하는 실행 단계 유형을 지정할 수 있습니다.

로그 레코드에 포함할 세부 정보 수준을 다음 중에서 선택합니다. 로그 수준은 파이프에 지정된 모든 로그 대상에 적용됩니다. 각 로그 수준에는 이전 로그 수준의 실행 단계가 포함됩니다.

- 끄기 - 지정된 로그 대상으로 레코드를 보내지 EventBridge 않습니다. 이것이 기본 설정입니다.
- 오류 — 파이프 실행 중에 생성된 오류와 관련된 모든 레코드를 지정된 로그 대상으로 EventBridge 보냅니다.
- 정보 — 오류와 관련된 모든 레코드와 파이프 실행 중에 수행된 다른 단계를 선택하여 지정된 로그 대상으로 EventBridge 전송합니다.
- TRACE — 파이프 실행의 모든 단계에서 생성된 모든 레코드를 지정된 로그 대상으로 EventBridge 보냅니다.

EventBridge 콘솔에서는 기본적으로 CloudWatch 로그 수준과 마찬가지로 로그가 ERROR 로그 대상으로 선택됩니다. 따라서 EventBridge Pipes는 기본적으로 세부 ERROR 수준이 포함된 로그 레코드를 보내는 새 CloudWatch 로그 그룹을 생성합니다. 프로그래밍 방식으로 로그를 구성할 때는 기본값이 선택되지 않습니다.

아래는 각 로그 수준에 포함된 실행 단계를 나타낸 표입니다.

단계	TRACE	INFO	ERROR	OFF
실행 실패	x	x	x	
실행 부분적으로 실패	x	x	x	
실행 시작됨	x	x		
실행 성공	x	x		
실행 제한됨	x	x	x	
실행 제한 시간	x	x	x	
보강 간접 호출 실패	x	x	x	
보강 간접 호출 건너뜀	x	x		
보강 간접 호출 시작됨	x			
보강 간접 호출 성공	x			
보강 단계 진입	x	x		
보강 단계 실패	x	x	x	
보강 단계 성공	x	x		
보강 변환 실패	x	x	x	
보강 변환 시작	x			
보강 변환 성공	x			
대상 간접 호출 실패	x	x	x	
대상 간접 호출 부분 실패	x	x	x	
대상 간접 호출 건너뜀	x			
대상 간접 호출 시작됨	x			

단계	TRACE	INFO	ERROR	OFF
대상 간접 호출 성공	x			
대상 단계 진입	x	x		
대상 단계 실패	x	x	x	
대상 단계 부분 실패	x	x	x	
대상 단계 건너뜀	x			
대상 단계 성공	x	x		
대상 변환 실패	x	x	x	
대상 변환 시작됨	x			
대상 변환 성공	x			

EventBridge Pipes 로그에 실행 데이터 포함

생성되는 레코드에 실행 데이터를 EventBridge 포함하도록 형식을 지정할 수 있습니다. 실행 데이터에는 이벤트 배치 페이로드를 나타내는 필드는 물론 보강 및 대상으로 전송된 요청 및 응답이 포함됩니다.

실행 데이터는 문제 해결 및 디버깅에 유용합니다. 이 `payload` 필드에는 배치에 포함된 각 이벤트의 실제 내용이 포함되므로 개별 이벤트를 특정 파이프 실행과 연관시킬 수 있습니다.

실행 데이터를 포함하도록 선택하면 파이프에 지정된 모든 로그 대상에 해당 데이터가 포함됩니다.

> **Important**
>
> 이러한 필드에는 민감한 정보가 포함될 수 있습니다. EventBridge 로깅 중에는 이러한 필드의 내용을 삭제하려고 시도하지 않습니다.

실행 데이터를 포함할 때 관련 레코드에 다음 필드를 EventBridge 추가합니다.

- **`payload`**

파이프에서 처리 중인 이벤트 배치의 내용을 나타냅니다.

EventBridge 이벤트 배치 콘텐츠가 업데이트되었을 수 있는 단계에서 생성된 레코드에 payload 필드를 포함합니다. 여기에는 다음 단계가 포함됩니다.

- EXECUTION_STARTED
- ENRICHMENT_TRANSFORMATION_SUCCEEDED
- ENRICHMENT_STAGE_SUCCEEDED
- TARGET_TRANSFORMATION_SUCCEEDED
- TARGET_STAGE_SUCCEEDED

- **awsRequest**

  보강 또는 대상에 전송된 요청을 JSON 문자열로 나타냅니다. API 대상으로 전송된 요청의 경우 이는 해당 엔드포인트로 전송된 HTTP 요청을 나타냅니다.

  EventBridge 보강 및 타깃팅의 최종 단계, 즉 지정된 보강 또는 대상 서비스에 대해 요청을 실행하거나 실행을 시도한 후 EventBridge 생성된 레코드에 awsRequest 필드를 포함합니다. 여기에는 다음 단계가 포함됩니다.

  - ENRICHMENT_INVOCATION_FAILED
  - ENRICHMENT_INVOCATION_SUCCEEDED
  - TARGET_INVOCATION_FAILED
  - TARGET_INVOCATION_PARTIALLY_FAILED
  - TARGET_INVOCATION_SUCCEEDED

- **awsResponse**

  보강 또는 대상이 반환한 응답을 JSON 형식으로 나타냅니다. API 대상으로 전송된 요청의 경우 이는 해당 엔드포인트에서 반환된 HTTP 응답을 나타냅니다.

  와 마찬가지로awsRequest, 보강 및 대상 서비스의 최종 단계에서 생성된 레코드, 즉 지정된 보강 또는 대상 서비스에 대해 요청을 실행하거나 실행을 시도한 후 응답을 EventBridge 받은 후에 생성된 레코드에 awsResponse 필드를 EventBridge 포함합니다. 여기에는 다음 단계가 포함됩니다.

  - ENRICHMENT_INVOCATION_FAILED
  - ENRICHMENT_INVOCATION_SUCCEEDED
  - TARGET_INVOCATION_FAILED
  - TARGET_INVOCATION_PARTIALLY_FAILED

- TARGET_INVOCATION_SUCCEEDED

파이프 실행 단계에 대한 설명은 ??? 단원을 참조하세요.

Pipes 로그 레코드의 실행 데이터 잘라내기 EventBridge

파이프의 로그 레코드에 실행 데이터를 EventBridge 포함하도록 선택하면 레코드가 256KB 크기 제한을 초과할 수 있습니다. 이를 방지하기 위해 는 다음과 같은 순서로 실행 데이터 필드를 EventBridge 자동으로 잘라냅니다. EventBridge 다음 필드를 잘라내기 전에 각 필드를 완전히 잘라냅니다. EventBridge 데이터 문자열 끝에서 문자를 제거하기만 하면 필드 데이터를 자릅니다. 데이터 중요도에 따라 자르려고 시도하지 않으며 잘라내면 JSON 형식이 무효화됩니다.

- payload
- awsRequest
- awsResponse

이벤트에서 EventBridge 필드가 잘리는 경우 잘린 데이터 필드 목록이 필드에 포함됩니다. truncatedFields

파이프 로그 레코드의 오류 보고 EventBridge

EventBridge 또한 오류 상태를 나타내는 파이프 실행 단계에 가능한 경우 오류 데이터를 포함합니다. 이러한 단계는 다음과 같습니다.

- ExecutionThrottled
- ExecutionTimeout
- ExecutionFailed
- ExecutionPartiallyFailed
- EnrichmentTransformationFailed
- EnrichmentInvocationFailed
- EnrichmentStageFailed
- TargetTransformationFailed
- TargetInvocationFailed
- TargetInvocationPartiallyFailed
- TargetStageFailed

- `TargetStagePartiallyFailed`

EventBridge 파이프 실행 단계

파이프 실행 단계의 흐름을 이해하면 로그를 사용하여 파이프의 성능을 문제 해결하거나 디버깅하는 데 도움이 될 수 있습니다.

파이프 실행은 파이프가 수신하여 보강 또는 대상으로 이동하는 이벤트 또는 일련의 이벤트입니다. 활성화된 경우 이벤트 배치가 처리될 때 수행하는 각 실행 단계에 대한 로그 레코드를 EventBridge 생성합니다.

상위 수준에서 실행에는 보강과 대상이라는 두 단계 또는 일련의 단계가 포함됩니다. 각 단계는 변환 및 간접 호출 단계로 구성됩니다.

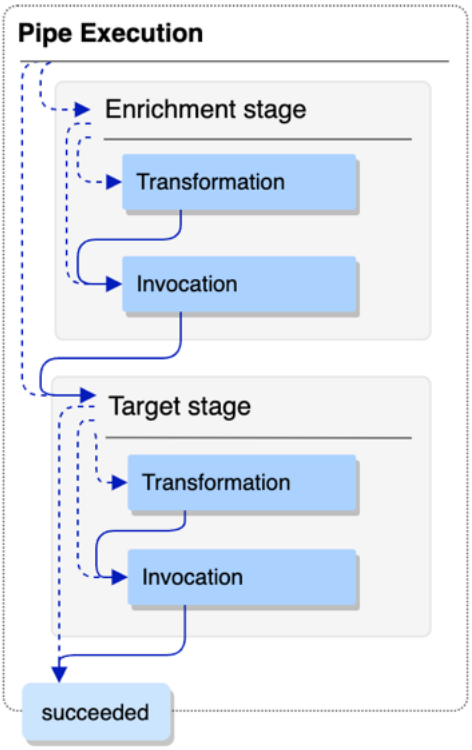
성공적인 파이프 실행의 주요 단계는 다음과 같습니다.

- 파이프 실행이 시작됩니다.
- 이벤트에 대한 보강을 지정한 경우 실행이 보강 단계로 들어갑니다. 보강을 지정하지 않은 경우 실행은 대상 단계로 진행됩니다.

  보강 단계에서 파이프는 사용자가 지정한 모든 변환을 수행한 다음 보강을 간접 호출합니다.
- 대상 단계에서 파이프는 사용자가 지정한 모든 변환을 수행한 다음 대상을 간접 호출합니다.

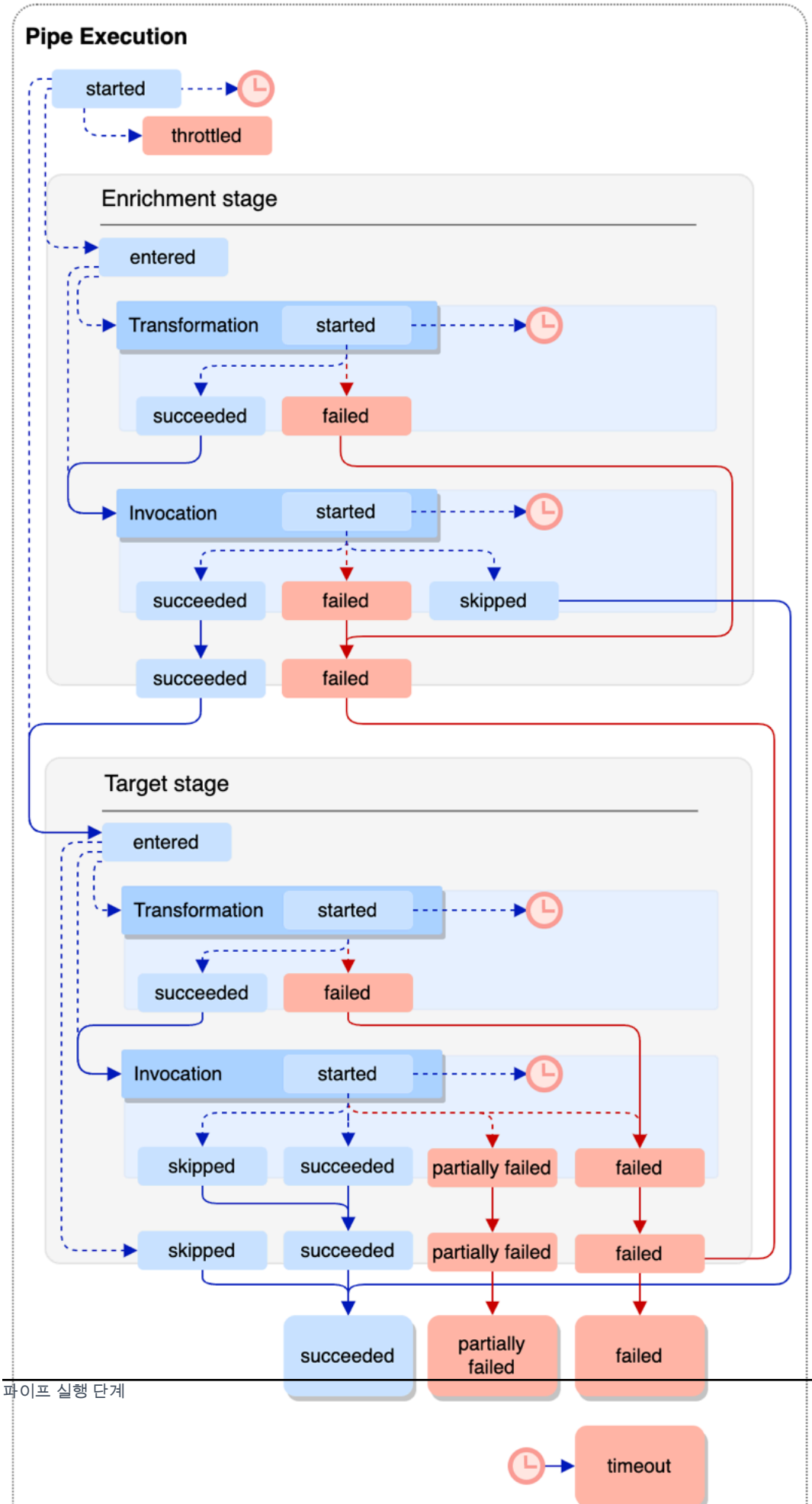
  변환이나 대상을 지정하지 않은 경우 실행은 대상 단계를 건너뛰게 됩니다.
- 파이프 실행이 성공적으로 완료됩니다.

아래 다이어그램은 이 흐름을 보여줍니다. 분기 경로는 점선 형식으로 표시됩니다.



아래 다이어그램은 가능한 모든 실행 단계를 나타낸 파이프 실행 흐름을 자세히 보여줍니다. 다시 말하지만, 분기 경로는 점선 형식으로 표시됩니다.

파이프 실행 단계의 전체 목록은 ??? 단원을 참조하세요.


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

대상 간접 호출로 인해 일괄 처리가 부분적으로 실패할 수 있다는 점에 유의하세요. 자세한 정보는 ???을 참조하세요.

EventBridge 파이프 로그 스키마 참조

다음 참조는 EventBridge 파이프 로그 레코드의 스키마를 자세히 설명합니다.

각 로그 레코드는 파이프 실행 단계를 나타내며, 파이프 소스 및 대상이 일괄 처리되도록 구성된 경우 최대 10,000개의 이벤트를 포함할 수 있습니다.

자세한 정보는 ???을 참조하세요.

```
{
    "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

파이프 실행 ID.

파이프 실행은 파이프가 수신하여 보강 또는 대상으로 이동하는 이벤트 또는 일련의 이벤트입니다. 자세한 정보는 ???을 참조하세요.

timestamp

로그 이벤트가 발생한 날짜 및 시간입니다.

단위: 밀리초

messageType

레코드가 생성된 파이프 실행 단계입니다.

파이프 실행 단계에 대한 자세한 내용은 ??? 단원을 참조하세요.

resourceArn

파이프의 Amazon 리소스 이름(ARN)입니다.

logLevel

파이프 로그에 지정된 세부 수준입니다.

유효한 값: `ERROR` | `INFO` | `TRACE`

자세한 정보는 ???을 참조하세요.

payload

파이프에서 처리 중인 이벤트 배치의 내용입니다.

EventBridge 이 파이프의 로그에 실행 데이터를 포함하도록 지정한 경우에만 이 필드를 포함합니다. 자세한 내용은 ??? 섹션을 참조하세요.

> **Important**
>
> 이러한 필드에는 민감한 정보가 포함될 수 있습니다. EventBridge 로깅 중에는 이러한 필드의 내용을 삭제하려고 시도하지 않습니다.

자세한 정보는 ???을 참조하세요.

awsRequest

JSON 형식으로 보강 또는 대상에 전송된 요청입니다. API 대상으로 전송된 요청의 경우 이는 해당 엔드포인트로 전송된 HTTP 요청을 나타냅니다.

EventBridge 이 파이프의 로그에 실행 데이터를 포함하도록 지정한 경우에만 이 필드를 포함합니다. 자세한 내용은 ??? 섹션을 참조하세요.

> **Important**
>
> 이러한 필드에는 민감한 정보가 포함될 수 있습니다. EventBridge 로깅 중에는 이러한 필드의 내용을 삭제하려고 시도하지 않습니다.

자세한 정보는 ???을 참조하세요.

awsResponse

보강 또는 대상이 JSON 형식으로 반환한 응답입니다. API 대상으로 전송된 요청의 경우 이는 API 대상 서비스 자체에서 반환된 응답이 아니라 해당 엔드포인트에서 반환된 HTTP 응답을 나타냅니다.

EventBridge 이 파이프의 로그에 실행 데이터를 포함하도록 지정한 경우에만 이 필드를 포함합니다. 자세한 내용은 ??? 섹션을 참조하세요.

> **Important**
>
> 이러한 필드에는 민감한 정보가 포함될 수 있습니다. EventBridge 로깅 중에는 이러한 필드의 내용을 삭제하려고 시도하지 않습니다.

자세한 정보는 ???을 참조하세요.

truncatedFields

레코드를 256KB 크기 제한 미만으로 유지하기 위해 실행 데이터 필드 EventBridge 목록이 잘렸습니다.

실행 데이터 필드를 잘라낼 필요가 없는 경우에는 EventBridge 이 필드가 표시되지만 `null`

자세한 정보는 ???을 참조하세요.

error

이 파이프 실행 단계에서 생성된 모든 오류에 대한 정보를 포함합니다.

이 파이프 실행 단계에서 오류가 생성되지 않은 경우 이 필드는 존재하지만 `null`입니다.

httpStatusCode

호출된 서비스에서 반환한 HTTP 상태 코드입니다.

message

호출된 서비스에서 반환된 오류 메시지입니다.

details

호출된 서비스에서 반환한 모든 세부 오류 정보입니다.

awsService

호출된 서비스의 이름입니다.

requestId

호출된 서비스에서 이 요청에 대한 요청 ID입니다.

Amazon CloudWatch Logs를 사용하여 Amazon EventBridge Pipes 로깅 AWS CloudTrail 및 모니터링

CloudWatch 지표를 사용하여 EventBridge 파이프 호출을 기록하고 파이프의 사용 CloudTrail 및 상태를 모니터링할 수 있습니다.

CloudWatch 지표

EventBridge Pipes는 파이프 실행이 제한되는 것부터 성공적으로 호출되는 대상에 이르기까지 모든 것에 대해 CloudWatch 1분마다 Amazon에 지표를 전송합니다.

지표	설명	차원	단위
`Concurrency`	파이프의 동시 실행 수입니다.	AwsAccountId	None
`Duration`	파이프 실행에 걸린 시간입니다.	PipeName	밀리초
`EventCount`	파이프가 처리한 이벤트 수입니다.	PipeName	None
`EventSize`	파이프를 간접 호출한 이벤트의 페이로드 크기입니다.	PipeName	바이트
`ExecutionThrottled`	병목 현상이 발생한 파이프 실행 횟수입니다. **Note** 실행이 제한되지 않은 경우 이 값은 `0`입니다.	AwsAccountId, PipeName	None

지표	설명	차원	단위
ExecutionTimeout	실행을 완료하기 전에 제한 시간이 초과된 파이프 실행 횟수입니다. **Note** 실행 시간이 초과되지 않은 경우 이 값은 0입니다.	PipeName	None
ExecutionFailed	파이프 실행이 실패한 횟수입니다. **Note** 실패한 실행이 없는 경우 이 값은 0입니다.	PipeName	None
ExecutionPartiallyFailed	파이프의 실행 중 부분적으로 실패한 횟수입니다. **Note** 부분적으로 실패한 실행이 없는 경우 이 값은 0입니다.	PipeName	None
EnrichmentStageDuration	보강 단계를 완료하는 데 걸린 시간입니다.	PipeName	밀리초
EnrichmentStageFailed	파이프의 보강 단계 실행에 실패한 횟수입니다. **Note** 실패한 실행이 없는 경우 이 값은 0입니다.	PipeName	None

지표	설명	차원	단위
`Invocations`	총 간접 호출 수입니다.	AwsAccountId, PipeName	None
`TargetStageDuration`	대상 단계를 완료하는 데 걸린 시간입니다.	PipeName	밀리초
`TargetStageFailed`	파이프의 대상 단계 실행에 실패한 횟수입니다. **Note** 실패한 실행이 없는 경우 이 값은 `0`입니다.	PipeName	None
`TargetStagePartiallyFailed`	파이프의 대상 단계 실행이 부분적으로 실패한 횟수입니다. **Note** 부분적으로 실패한 대상 단계 실행이 없는 경우 이 값은 `0`입니다.	PipeName	None
`TargetStageSkipped`	파이프의 대상 단계 실행을 건너뛴 횟수입니다 (예: 보강이 빈 페이로드를 반환하는 경우).	PipeName	개수

CloudWatch 지표의 치수

CloudWatch 지표에는 차원 또는 정렬 가능한 속성이 있으며, 이러한 속성은 다음과 같습니다.

측정기준	설명
`AwsAccountId`	계정 ID를 기준으로 사용할 수 있는 지표를 필터링합니다.

측정기준	설명
PipeName	파이프 이름을 기준으로 사용할 수 있는 지표를 필터링합니다.

Amazon EventBridge Pipes 오류 처리 및 문제 해결

재시도 동작 및 오류 처리

EventBridge Pipes는 소스 서비스, 강화 또는 대상 서비스 또는 재시도 가능한 모든 AWS 실패에 대해 자동으로 보강 및 대상 호출을 재시도합니다. EventBridge 그러나 보강 또는 대상 고객 구현에서 오류가 발생하는 경우 파이프 폴링 처리량은 점차 줄어듭니다. 거의 연속적인 4xx 오류(예: IAM 권한 부여 문제 또는 리소스 누락)의 경우 StateReason에 설명 메시지를 추가하여 파이프를 자동으로 비활성화할 수 있습니다.

파이프 간접 호출 오류 및 재시도 동작

파이프를 간접 호출하면 파이프 내부 오류와 고객 간접 호출 오류라는 두 가지 유형의 주요 오류가 발생할 수 있습니다.

파이프 내부 오류

파이프 내부 오류는 Pipes 서비스에서 관리하는 호출 측면에서 발생하는 오류입니다. EventBridge

이러한 유형의 오류에는 다음과 같은 문제가 포함될 수 있습니다.

- 고객 대상 서비스를 간접 호출하려고 할 때 HTTP 연결에 실패합니다.
- 파이프 서비스 자체의 가용성이 일시적으로 저하됩니다.

일반적으로 EventBridge Pipes는 내부 오류를 무한정 재시도하고 소스에서 레코드가 만료될 때만 중지합니다.

스트림 소스가 있는 파이프의 경우 EventBridge Pipes는 내부 오류에 대한 재시도 횟수를 스트림 원본의 재시도 정책에 지정된 최대 재시도 횟수로 계산하지 않습니다. Amazon SQS 소스가 있는 파이프의 경우, EventBridge Pipes는 내부 오류에 대한 재시도 횟수를 Amazon SQS 소스의 최대 수신 수에 계산하지 않습니다.

고객 간접 호출 오류

고객 간접 호출 오류는 사용자가 관리하는 구성 또는 코드로 인해 발생하는 오류입니다.

이러한 유형의 오류에는 다음과 같은 문제가 포함될 수 있습니다.

- 파이프에 대한 권한이 부족하여 대상을 간접 호출할 수 없습니다.
- 동기식 간접 호출 고객의 Lambda, Step Functions, API 대상 또는 API Gateway 엔드포인트에 로직 오류가 있습니다.

고객 호출 오류의 경우 Pipes는 다음을 수행합니다 EventBridge .

- 스트림 소스가 있는 파이프의 경우 Pipes는 EventBridge 파이프 재시도 정책에 구성된 최대 재시도 시간까지 또는 최대 레코드 보존 기간이 만료될 때까지 (둘 중 먼저 도래하는 날짜 기준) 까지 재시도합니다.
- Amazon SQS 소스가 있는 EventBridge 파이프의 경우 Pipes는 소스 대기열의 최대 수신 수까지 고객 오류를 재시도합니다.
- Apache Kafka 또는 Amazon MQ 소스가 있는 파이프의 경우 내부 오류를 EventBridge 재시도하는 것과 동일하게 고객 오류를 재시도합니다.

컴퓨팅 대상이 있는 파이프의 경우 파이프를 동기적으로 호출해야 Pipes가 고객 컴퓨팅 로직에서 발생하는 모든 런타임 오류를 인지하고 해당 오류를 재시도할 수 있습니다. EventBridge Step Functions 표준 워크플로의 로직에서 발생한 오류에 대해서는 해당 대상이 비동기식으로 간접 호출되어야 하므로 파이프를 재시도할 수 없습니다.

Amazon SQS 및 스트림 소스 (예: Kinesis 및 DynamoDB) 의 경우 Pipes는 대상 장애에 대한 부분 배치 EventBridge 장애 처리를 지원합니다. 자세한 내용은 부분적 배치 실패를 참조하세요.

파이프 DLQ 동작

파이프는 소스의 DLQ(Dead Letter Queue) 동작을 상속합니다.

- 소스 Amazon SQS 대기열에 구성된 DLQ가 있는 경우, 지정된 횟수만큼 시도한 후에 메시지가 DLQ로 자동 전송됩니다.
- DynamoDB 및 Kinesis 스트림과 같은 스트림 소스에서는 파이프 및 경로 이벤트에 대한 DLQ를 구성할 수 있습니다. DynamoDB 및 Kinesis 스트림 소스는 Amazon SQS 대기열과 Amazon SNS 주제를 DLQ 대상으로 지원합니다.

Kinesis 또는 DynamoDB 소스가 있는 파이프에 `DeadLetterConfig`를 지정하는 경우 파이프의 `MaximumRecordAgeInSeconds` 속성이 소스 이벤트의 `MaximumRecordAge`보다 작은지 확인하세

요. MaximumRecordAgeInSeconds는 파이프 폴러가 이벤트를 중단하고 DLQ에 전달하는 시기를 제어하고 MaximumRecordAge는 메시지가 삭제되기 전에 소스 스트림에 표시되는 기간을 제어합니다. 따라서 이벤트가 DLQ로 전송되는 시점과 소스에서 자동으로 삭제되는 시점 사이에 충분한 시간을 두도록 MaximumRecordAgeInSeconds를 소스 MaximumRecordAge보다 작은 값으로 설정하여 이벤트가 DLQ로 이동한 이유를 파악할 수 있습니다.

Amazon MQ 소스의 경우 DLQ를 메시지 브로커에서 직접 구성할 수 있습니다.

EventBridge Pipes는 스트림 소스에 대한 선입선출 (FIFO) DLQ를 지원하지 않습니다.

EventBridge Pipes는 Amazon MSK 스트림 및 자체 관리형 아파치 카프카 스트림 소스용 DLQ를 지원하지 않습니다.

파이프 실패 상태

파이프 생성, 삭제 및 업데이트는 실패 상태를 초래할 수 있는 비동기 작업입니다. 마찬가지로 오류로 인해 파이프가 자동으로 비활성화될 수 있습니다. 모든 경우에 파이프 StateReason은 오류 문제를 해결하는 데 도움이 되는 정보를 제공합니다.

다음은 가능한 StateReason 값의 샘플입니다.

- 스트림을 찾을 수 없음. 처리를 재개하려면 파이프를 삭제하고 새 파이프를 생성합니다.
- 파이프에는 큐 작업 (sqs:ReceiveMessage, sqs: 및 sqs:) 을 수행하는 데 필요한 권한이 없습니다. DeleteMessage GetQueueAttributes
- 연결 오류. VPC는 파이프에 연결할 수 있어야 합니다. NAT 게이트웨이 또는 VPC 엔드포인트를 파이프 데이터에 구성하여 액세스를 제공할 수 있습니다. NAT 게이트웨이 또는 VPC 엔드포인트를 파이프 데이터에 설정하는 방법은 설명서를 참조하십시오. AWS
- MSK 클러스터에는 연결된 보안 그룹이 없음

업데이트된 StateReason에 따라 파이프가 자동으로 중지될 수 있습니다. 가능한 이유는 다음과 같습니다.

- Step Functions 표준 워크플로가 보강으로 구성되었습니다.
- Step Functions 표준 워크플로가 동기식으로 간접 호출할 대상으로 구성되었습니다.

사용자 지정 암호화 오류

AWS-managed AWS KMS 키가 아닌 AWS KMS 사용자 지정 암호화 키 (CMK) 를 사용하도록 소스를 구성하는 경우 파이프의 실행 역할 복호화 권한을 명시적으로 부여해야 합니다. 이렇게 하려면 사용자 지정 CMK 정책에 다음과 같은 추가 권한을 포함하세요.

```
  {
      "Sid": "Allow Pipes access",
      "Effect": "Allow",
      "Principal": {
          "AWS": "arn:aws:iam::01234567890:role/service-role/
Amazon_EventBridge_Pipe_DDBStreamSourcePipe_12345678"
      },
      "Action": "kms:Decrypt",
      "Resource": "*"
  }
```

위 역할을 파이프의 실행 역할로 바꿉니다.

이는 다음을 포함하여 CMK를 사용하는 모든 파이프 소스에 해당됩니다. AWS KMS

- Amazon DynamoDB Streams
- Amazon Kinesis Data Streams
- Amazon MQ
- Amazon MSK
- Amazon SQS

자습서: 소스 이벤트를 필터링하는 EventBridge 파이프 만들기

이 자습서에서는 DynamoDB 스트림 소스를 Amazon SQS 대기열 대상에 연결하는 파이프를 생성합니다. 여기에는 대기열에 전달할 이벤트를 필터링할 때 파이프에서 사용할 이벤트 패턴을 지정하는 것이 포함됩니다. 그런 다음 파이프를 테스트하여 원하는 이벤트만 전달되는지 확인합니다.

사전 조건: 소스 및 대상 생성

파이프를 만들기 전에 파이프가 연결할 소스와 대상을 만들어야 합니다. 이 경우 Amazon DynamoDB 데이터 스트림은 파이프 소스 역할을 하고 Amazon SQS 대기열은 파이프 대상 역할을 합니다.

이 단계를 단순화하기 위해 소스 및 대상 리소스를 AWS CloudFormation을 프로비저닝하는 데 사용할 수 있습니다. 이를 위해 다음 리소스를 정의하는 CloudFormation 템플릿을 생성해야 합니다.

- 파이프 소스

  DynamoDB 테이블의 항목 변경 사항에 대한 정렬된 정보 흐름을 제공하기 위해 활성화된 스트림이 있는 Amazon DynamoDB 테이블(이름: `pipe-tutorial-source`)입니다.
- 파이프 대상

  파이프에서 DynamoDB 이벤트 스트림을 수신하기 위한 Amazon SQS 대기열(이름: `pipe-tutorial-target`).

파이프 리소스 프로비저닝을 위한 CloudFormation 템플릿을 만들려면

1. 아래 ??? 섹션의 JSON 템플릿 텍스트를 복사합니다.
2. 템플릿을 JSON 파일(예: `~/pipe-tutorial-resources.json`)로 저장합니다.

다음으로 방금 생성한 템플릿 파일을 사용하여 CloudFormation 스택을 프로비저닝합니다.

> **Note**
>
> CloudFormation 스택을 생성한 후에는 스택에서 프로비저닝한 AWS 리소스에 대한 요금이 부과됩니다.

AWS CLI를 사용하여 자습서 사전 조건 프로비저닝

- 다음 CLI 명령을 실행합니다. 여기서 `--template-body`는 템플릿 파일의 위치를 지정합니다.

```
aws cloudformation create-stack --stack-name pipe-tuturial-resources --template-
body file://~/pipe-tutorial-resources.json
```

CloudFormation 콘솔을 사용하여 자습서 사전 조건 프로비저닝

1. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.
2. 스택을 선택한 다음 스택 생성을 선택하고 새 리소스 사용(표준)을 선택합니다.

   CloudFormation에 스택 생성 마법사가 표시됩니다.

3. 사전 조건 - 템플릿 준비에서 템플릿 준비 완료를 선택합니다.
4. 템플릿 지정에서 템플릿 파일 업로드를 선택한 다음 파일을 선택하고 다음을 선택합니다.
5. 스택과 해당 스택이 프로비저닝할 리소스를 구성합니다.
   - 스택 이름에 `pipe-tuturial-resources`을 입력합니다.
   - 파라미터의 경우 DynamoDB 테이블 및 Amazon SQS 대기열의 기본 이름을 그대로 둡니다.
   - 다음을 선택합니다.
6. 다음을 선택한 후 제출을 선택합니다.

   CloudFormation에서는 스택을 생성하고 템플릿에 정의된 리소스를 프로비저닝합니다.

CloudFormation에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation이란 무엇입니까?를 참조하세요.

1단계: 파이프 생성

파이프 소스와 대상이 프로비저닝되었으므로 이제 파이프를 생성하여 두 서비스를 연결할 수 있습니다.

EventBridge 콘솔을 사용하여 파이프 생성

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 파이프를 선택합니다.
3. 파이프 생성을 선택합니다.
4. 이름에 파이프 이름을 `pipe-tutorial`로 지정합니다.
5. DynamoDB 데이터 스트림 소스를 지정합니다.
   a. 세부 정보에서 소스에 대해 DynamoDB 데이터 스트림을 선택합니다.

      EventBridge는 DynamoDB 관련 소스 구성 설정을 표시합니다.

   b. DynamoDB 스트림의 경우 `pipe-tutorial-source`를 선택합니다.

      시작 위치를 기본값인 `Latest`로 설정된 상태로 유지합니다.

   c. 다음을 선택합니다.
6. 이벤트 패턴을 지정하고 테스트하여 이벤트를 필터링합니다.

필터링을 사용하면 파이프가 보강 또는 대상으로 보내는 이벤트를 제어할 수 있습니다. 파이프는 이벤트 패턴과 일치하는 이벤트만 보강 또는 대상으로 보냅니다.

자세한 내용은 ??? 섹션을 참조하세요.

> **Note**
>
> 보강 또는 대상으로 전송된 이벤트에 대해서만 요금이 청구됩니다.

a. 샘플 이벤트 - 선택 사항에서 AWS 이벤트를 선택한 상태로 두고 DynamoDB 스트림 샘플 이벤트 1이 선택되었는지 확인합니다.

   이벤트 패턴을 테스트하는 데 사용할 샘플 이벤트입니다.

b. 이벤트 패턴에서 다음 이벤트 패턴을 입력합니다.

```
{
  "eventName": ["INSERT", "MODIFY"]
}
```

c. 패턴 테스트를 선택합니다.

   EventBridge는 샘플 이벤트가 이벤트 패턴과 일치한다는 메시지를 표시합니다. 샘플 이벤트에 `INSERT`라는 `eventName` 값이 있기 때문입니다.

d. 다음을 선택합니다.

7. 보강 지정을 건너뛰려면 다음을 선택합니다.

   이 예시에서는 보강을 선택하지 않습니다. 강화를 사용하면 소스의 데이터를 대상으로 보내기 전에 개선하는 서비스를 선택할 수 있습니다. 자세한 내용은 ??? 섹션을 참조하세요.

8. Amazon SQS 대기열을 파이프 대상으로 지정합니다.

   a. 세부 정보에서 대상 서비스에 대해 Amazon SQS 대기열을 선택합니다.

   b. 대기열에서 `pipe-tutorial-target`을 선택합니다.

   c. 대상 입력 변환기 섹션은 비워둡니다.

      자세한 내용은 ??? 섹션을 참조하세요.

9. 파이프 생성을 선택합니다.

EventBridge는 파이프를 생성하고 파이프 세부 정보 페이지를 표시합니다. 상태가 Running으로 업데이트되면 파이프가 준비된 것입니다.

2단계: 파이프 필터 이벤트 확인

파이프가 설정되었지만 아직 테이블에서 이벤트를 수신하지 않았습니다.

파이프를 테스트하려면 DynamoDB 테이블의 항목을 업데이트해야 합니다. 각 업데이트는 DynamoDB 스트림이 파이프로 보내는 이벤트를 생성합니다. 지정한 이벤트 패턴과 일치하는 것도 있고 그렇지 않은 것도 있습니다. 그런 다음 Amazon SQS 대기열을 검사하여 파이프가 이벤트 패턴과 일치하는 이벤트만 전송했는지 확인할 수 있습니다.

테이블 항목을 업데이트하여 이벤트를 생성합니다.

1. https://console.aws.amazon.com/dynamodb/에서 DynamoDB 콘솔을 엽니다.
2. 왼쪽 탐색 메뉴에서 테이블을 선택합니다. `pipe-tutorial-source` 테이블을 선택합니다.

   DynamoDB는 `pipe-tutorial-source`에 대한 테이블 세부 정보 페이지를 표시합니다.
3. 테이블 항목 탐색을 선택한 다음 항목 생성을 선택합니다.

   DynamoDB는 항목 생성 페이지를 표시합니다.
4. 속성에서 새 테이블 항목을 생성합니다.

   a. 앨범의 경우 `Album A`를 입력합니다.

   b. 아티스트에 `Artist A`를 입력합니다.

   c. 항목 생성을 선택합니다.
5. 테이블 항목 업데이트:

   a. 반환된 항목에서 앨범 A를 선택합니다.

   b. 새 속성 추가를 선택한 다음 문자열을 선택합니다.

   c. `Song A` 값으로 Song의 새 값을 입력합니다.

   d. 변경 사항 저장을 선택합니다.
6. 테이블 항목 삭제:

   a. 반환된 항목에서 앨범 A를 확인합니다.

   b. 작업 메뉴에서 항목 삭제를 선택합니다.

테이블 항목을 세 번 업데이트했습니다. 그러면 DynamoDB 데이터 스트림에 대한 세 가지 이벤트가 생성됩니다.

- 항목을 생성할 때의 `INSERT` 이벤트입니다.
- 항목에 속성을 추가했을 때의 `MODIFY` 이벤트입니다.
- 항목을 삭제했을 때 `REMOVE` 이벤트입니다.

그러나 파이프에 대해 지정한 이벤트 패턴은 `INSERT` 또는 `MODIFY` 이벤트가 아닌 모든 이벤트를 필터링해야 합니다. 다음으로 파이프가 예상 이벤트를 대기열에 전달했는지 확인합니다.

예상된 이벤트가 대기열에 전달되었는지 확인

1. https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. `pipe-tutorial-target` 대기열을 선택합니다.

   Amazon SQS는 대기열 세부 정보 페이지를 표시합니다.
3. 메시지 전송 및 수신을 선택한 다음 메시지 수신에서 메시지 폴링을 선택합니다.

   대기열은 파이프를 폴링한 다음 수신한 이벤트를 나열합니다.
4. 전달된 이벤트 JSON을 보려면 이벤트 이름을 선택합니다.

대기열에는 두 개의 이벤트가 있어야 합니다. 하나는 `eventName`이 `INSERT`고 다른 하나는 `eventName`이 `MODIFY`입니다. 하지만 파이프는 테이블 항목을 삭제하는 이벤트를 전달하지 않았습니다. 해당 이벤트에는 `REMOVE`라는 `eventName`이 있었고 이는 파이프에 지정한 이벤트 패턴과 일치하지 않았기 때문입니다.

3단계: 리소스 정리

먼저 파이프 자체를 삭제합니다.

EventBridge 콘솔을 사용하여 파이프 삭제

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 파이프를 선택합니다.
3. `pipe-tutorial` 파이프를 선택하고 삭제를 선택합니다.

그런 다음 CloudFormation 스택을 삭제하여 스택 내에 프로비저닝된 리소스의 지속적인 사용에 대한 요금이 청구되지 않도록 합니다.

CLI를 사용하여 자습서 사전 조건 삭제 AWS

- 다음 CLI 명령을 실행합니다. 여기서 `--stack-name`은 스택 이름을 지정합니다.

```
aws cloudformation delete-stack --stack-name pipe-tuturial-resources
```

AWS CloudFormation 콘솔을 사용하여 자습서 사전 조건 삭제

1. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.
2. 스택 페이지에서 스택을 선택한 다음 삭제를 선택합니다.
3. 삭제를 선택하여 작업을 확인합니다.

사전 조건 생성을 위한 AWS CloudFormation 템플릿

아래 JSON을 사용하여 이 자습서에 필요한 소스 및 대상 리소스를 프로비저닝하기 위한 CloudFormation 템플릿을 생성합니다.

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

EventBridge 파이프에서 AWS CloudFormation 템플릿 생성

AWS CloudFormation 인프라를 코드로 취급하여 중앙 집중적이고 반복 가능한 방식으로 계정 및 지역 전반의 AWS 리소스를 구성하고 관리할 수 있습니다. CloudFormation 프로비저닝 및 관리하려는 리소스를 정의하는 템플릿을 생성하여 이를 수행할 수 있습니다.

EventBridge 템플릿 개발을 CloudFormation 빠르게 시작할 수 있도록 계정의 기존 파이프에서 템플릿을 생성할 수 있습니다. 템플릿에 포함할 파이프를 하나 또는 여러 개 선택할 수 있습니다. 그런 다음 이러한 템플릿을 기반으로 사용하여 관리 중인 CloudFormation 리소스 스택을 만들 수 있습니다.

에 대한 CloudFormation 자세한 내용은 AWS CloudFormation 사용 설명서를 참조하십시오.

이벤트 버스의 경우 이벤트 버스 및 이벤트 버스 규칙에서 CloudFormation 템플릿을 생성할 수 있습니다.

EventBridge 파이프 템플릿에 포함된 리소스

CloudFormation 템플릿을 EventBridge 생성할 때 선택한 각 파이프에 대한 AWS::Pipes::Pipe리소스가 생성됩니다. 또한, 설명된 조건에 따른 다음 리소스도 EventBridge 포함됩니다.

- AWS::Events::ApiDestination

  파이프에 API 대상 (보강 또는 대상) 이 포함된 경우 이를 CloudFormation 템플릿에 리소스로 EventBridge AWS::Events::ApiDestination 포함하세요.
- AWS::Events::EventBus

  파이프에 이벤트 버스가 대상으로 포함된 경우 CloudFormation 템플릿에 이벤트 버스를 리소스로 EventBridge 포함하세요. AWS::Events::EventBus
- AWS::IAM::Role

  파이프를 구성할 때 새 실행 역할을 EventBridge 만든 경우 해당 역할을 템플릿에 AWS::IAM::Role 리소스로 EventBridge 포함하도록 선택할 수 있습니다. EventBridge 생성한 역할은 포함되지 않습니다. (어느 경우든 AWS::Pipes::Pipe 리소스 `RoleArn` 속성에는 역할의 ARN이 포함됩니다.)

Pipes에서 EventBridge 생성된 CloudFormation 템플릿 사용 시 고려 사항

에서 생성한 CloudFormation 템플릿을 사용할 때는 다음 요소를 고려하십시오 EventBridge.

- EventBridge 생성 템플릿에 비밀번호를 포함하지 않습니다.

템플릿을 사용하여 CloudFormation 스택을 생성하거나 업데이트할 때 사용자가 암호 또는 기타 민감한 정보를 지정할 수 있는 템플릿 매개 변수를 포함하도록 템플릿을 편집할 수 있습니다.

또한 사용자는 Secrets Manager를 이용해 원하는 리전에 보안 암호를 생성한 다음, 생성된 템플릿을 편집하여 동적 파라미터를 적용할 수 있습니다.

- 생성된 템플릿의 대상은 원래 파이프에 지정된 그대로 유지됩니다. 따라서 템플릿을 사용하여 다른 리전에 스택을 생성하기 전에 템플릿을 적절하게 편집하지 않으면 교차 리전 문제가 발생할 수 있습니다.

  또한 생성된 템플릿은 다운스트림 대상을 자동으로 생성하지 않습니다.

EventBridge 파이프에서 CloudFormation 템플릿 생성

EventBridge 콘솔을 사용하여 하나 이상의 파이프에서 CloudFormation 템플릿을 생성하려면 다음과 같이 하십시오.

하나 이상의 파이프에서 CloudFormation 템플릿을 생성하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 파이프를 선택합니다.
3. 파이프에서 생성된 CloudFormation 템플릿에 포함할 파이프를 하나 이상 선택합니다.

   단일 파이프의 경우 파이프 이름을 선택하여 파이프의 세부 정보 페이지를 표시할 수도 있습니다.
4. CloudFormation 템플릿을 선택한 다음 템플릿을 생성할 형식 (JSON 또는 YAML) EventBridge 을 선택합니다.

   EventBridge 선택한 형식으로 생성된 템플릿을 표시합니다.
5. 선택한 파이프에 대해 새 실행 역할을 생성한 후 해당 역할을 EventBridge 템플릿에 포함하려면 콘솔에서 생성한 IAM 역할 포함을 EventBridge 대신 선택합니다.
6. EventBridge 템플릿 파일을 다운로드하거나 템플릿을 클립보드에 복사할 수 있는 옵션을 제공합니다.

   - 템플릿 파일을 다운로드하려면 다운로드를 선택합니다.
   - 템플릿을 클립보드에 복사하려면 복사를 선택합니다.
7. 템플릿을 종료하려면 취소를 선택합니다.

글로벌 엔드포인트 및 이벤트 복제를 통해 애플리케이션의 리전별 내결함성 설정

Amazon EventBridge 글로벌 엔드포인트로 애플리케이션의 가용성을 개선할 수 있습니다. 글로벌 엔드포인트는 추가 비용 없이 애플리케이션에 리전별 내결함성을 구성하는 데 도움이 됩니다. 시작하려면 Amazon Route 53 상태 확인을 엔드포인트에 할당합니다. 장애 조치가 시작되면 상태 확인에서 '비정상' 상태를 보고합니다. 장애 조치가 시작된 후 몇 분 내에 모든 사용자 지정 이벤트가 보조 리전의 이벤트 버스로 라우팅되고 해당 이벤트 버스에서 처리됩니다. 상태 확인에서 '정상' 상태가 보고되면 기본 리전의 이벤트 버스에서 이벤트를 처리합니다.

글로벌 엔드포인트를 사용하는 경우 이벤트 복제를 활성화할 수 있습니다. 이벤트 복제는 관리형 규칙을 사용하여 모든 사용자 지정 이벤트를 기본 및 보조 리전의 이벤트 버스로 전송합니다.

> **Note**
>
> 사용자 지정 버스를 사용하는 경우 장애 조치가 제대로 작동하려면 각 리전에 이름과 계정이 동일한 사용자 지정 버스가 필요합니다.

주제

복구 시간 목표 및 복구 시점 목표

Recovery Time Objective(RTO)는 장애 발생 후 보조 리전에서 이벤트 수신을 시작하는 데 걸리는 시간입니다. RTO의 경우 시간에는 CloudWatch 경보를 트리거하고 Route 53 상태 확인의 상태를 업데이트하는 기간이 포함됩니다. Recovery Point Objective(RPO)는 장애 발생 시 처리되지 않은 상태로 남

아 있는 데이터의 측정값입니다. RPO의 경우 시간에는 보조 리전에 복제되지 않고 서비스 또는 리전이 복구될 때까지 기본 리전에서 멈춘 이벤트가 포함됩니다. 글로벌 엔드포인트를 사용할 때 경보 구성에 대한 권장 가이드를 따르면 RTO 및 RPO가 360초(최대 420초)일 것으로 예상할 수 있습니다.

이벤트 복제

이벤트는 보조 리전에서 비동기적으로 처리됩니다. 즉, 두 리전에서 이벤트가 동시에 처리된다는 보장은 없습니다. 장애 조치가 트리거되면 이벤트는 보조 리전에서 처리되며 사용 가능한 경우 기본 리전에서 처리됩니다. 이벤트 복제를 활성화하면 월별 비용이 증가합니다. 자세한 내용은 Amazon EventBridge 요금을 참조하십시오.

다음과 같은 이유로 글로벌 엔드포인트 설정 시 이벤트 복제를 활성화하는 것이 좋습니다.

- 이벤트 복제를 통해 글로벌 엔드포인트가 올바르게 구성되었는지 확인할 수 있습니다. 이에 따라 장애 조치 발생 시에도 문제를 해결할 수 있습니다.
- 장애 조치 이벤트를 자동으로 복구하려면 이벤트 복제가 필요합니다. 이벤트 복제를 활성화하지 않은 경우, 이벤트가 기본 리전으로 돌아가기 전에 수동으로 Route 53 상태 확인을 '정상'으로 재설정해야 합니다.

복제된 이벤트 페이로드

다음은 복제된 이벤트 페이로드의 예입니다.

> **Note**
>
> `region`에는 이벤트가 복제된 리전이 나열됩니다.

```
{
    "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"
    }
}
```

글로벌 엔드포인트 생성

글로벌 엔드포인트를 설정하려면 다음 단계를 완료합니다.

1. 기본 및 보조 리전에 모두 일치하는 이벤트 버스와 규칙이 있는지 확인합니다.
2. Route 53 상태 확인을 생성하여 이벤트 버스를 모니터링합니다. 상태 확인 생성 시 도움이 필요하면 글로벌 엔드포인트를 생성할 때 새 상태 확인을 선택하세요.
3. 글로벌 엔드포인트를 생성합니다.

Route 53 상태 확인을 설정한 후에는 글로벌 엔드포인트를 생성할 수 있습니다.

콘솔을 사용하여 글로벌 엔드포인트 생성하기

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 글로벌 엔드포인트를 선택합니다.
3. 엔드포인트 생성을 선택합니다.
4. 엔드포인트 이름 및 설명을 입력합니다.
5. 기본 리전의 이벤트 버스에서 엔드포인트와 연결하려는 이벤트 버스를 선택합니다.
6. 보조 리전에는 장애 조치 시 이벤트를 전달하려는 리전을 선택합니다.

> **Note**
>
> 보조 리전의 이벤트 버스는 자동으로 채워지며 편집할 수 없습니다.

7. 장애 조치 및 복구를 트리거하기 위한 Route 53 상태 확인의 경우 엔드포인트에서 모니터링할 상태 확인을 선택합니다. 아직 상태 확인이 없는 경우 New Health check를 선택하여 AWS CloudFormation 콘솔을 열고 CloudFormation 템플릿을 사용하여 상태 확인을 생성합니다.

> **Note**
>
> 데이터가 누락되면 상태 확인에 실패합니다. 이벤트를 간헐적으로만 전송해야 하는 경우에는 더 긴 MinimumEvaluationPeriod이벤트를 사용하거나 누락된 데이터를 '위반' 대신 '누락'으로 처리하는 것이 좋습니다.

8. (선택 사항) 이벤트 복제의 경우 다음을 수행합니다.

   a. 이벤트 복제 활성화됨을 선택합니다.

   b. 실행 역할에서 새 AWS Identity and Access Management 역할을 생성할지 기존 역할을 사용할지 선택합니다. 다음을 따릅니다.

      - 이 특정 리소스에 대해 새 역할 생성을 선택합니다. 선택적으로 역할 이름을 업데이트하여 새 역할을 생성할 수 있습니다.
      - 기존 역할 사용을 선택합니다. 그런 다음, 실행 역할에서 사용할 역할을 선택합니다.

9. 생성을 선택합니다.

API를 사용하여 글로벌 엔드포인트 생성하기

EventBridge API를 사용하여 글로벌 엔드포인트를 생성하려면 Amazon EventBridge API 참조를 참조하십시오 CreateEndpoint.

AWS CloudFormation을 사용하여 글로벌 엔드포인트 생성하기

AWS CloudFormation API를 사용하여 글로벌 엔드포인트를 생성하려면 사용 AWS CloudFormation 설명서를 참조하십시오 AWS::Events::Endpoints.

SDK를 사용하여 글로벌 엔드포인트로 작업하기 AWS

> **Note**
>
> 조만간 C++에 대한 지원이 제공될 예정입니다.

AWS SDK를 사용하여 글로벌 엔드포인트를 사용할 때는 다음 사항에 유의하세요.

- 특정 SDK에 AWS 공용 런타임 (CRT) 라이브러리가 설치되어 있어야 합니다. CRT가 설치되어 있지 않은 경우 설치해야 할 항목을 나타내는 예외 메시지가 표시됩니다. 자세한 내용은 다음 자료를 참조하세요.
  - AWS 공통 런타임(CRT) 라이브러리
  - awslabs/ aws-crt-java
  - 아슬라브/ aws-crt-nodejs
  - 아슬라브/ aws-crt-python
- 글로벌 엔드포인트를 생성한 후에는 사용하는 모든 `PutEvents` 호출에 `endpointId` 및 `EventBusName`을 추가해야 합니다.
- 글로벌 엔드포인트는 서명 버전 4A를 지원합니다. 이 버전의 SIGv4를 사용하면 여러 AWS 리전에 대한 요청에 서명할 수 있습니다. 이는 여러 리전 중 하나에서 데이터 액세스가 발생할 수 있는 API 작업에 유용합니다. AWS SDK를 사용할 때 자격 증명을 제공하면 추가 구성 없이 글로벌 엔드포인트에 대한 요청에 서명 버전 4A가 사용됩니다. SigV4A에 대한 자세한 내용은 AWS 일반 참조의 AWS API 요청에 서명을 참조하세요.

  글로벌 AWS STS 엔드포인트 (sts.amazonaws.com) 에서 임시 자격 증명을 요청하는 경우 기본적으로 AWS STS SigV4A를 지원하지 않는 자격 증명을 제공합니다. 자세한 내용은 사용 설명서의 지역별 관리를 참조하십시오 AWS STS . AWSAWS Identity and Access Management

사용 가능한 리전

다음 리전은 글로벌 엔드포인트를 지원합니다.

- 미국 동부(버지니아 북부)
- 미국 동부(오하이오)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오레곤)
- 캐나다(중부)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(밀라노)
- 유럽(파리)

- 유럽(스톡홀름)
- 아시아 태평양(뭄바이)
- 아시아 태평양(오사카)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 남아메리카(상파울루)

Amazon EventBridge 글로벌 엔드포인트 작업에 대한 모범 사례

글로벌 엔드포인트를 설정할 때 권장되는 모범 사례는 다음과 같습니다.

주제

- 이벤트 복제 활성화
- 이벤트 제한 방지
- Amazon Route 53 상태 확인에서 구독자 지표 사용

이벤트 복제 활성화

복제 기능을 설정하고 글로벌 엔드포인트에 할당한 보조 리전에서 이벤트를 처리하는 것이 좋습니다. 이렇게 하면 보조 리전의 애플리케이션이 올바르게 구성됩니다. 또한 문제가 완화된 후 기본 리전으로 자동 복구되도록 복제 기능을 설정해야 합니다.

이벤트 ID는 API 직접 호출에 따라 변경될 수 있으므로 리전 간에 이벤트를 상호 연결하려면 변경이 불가능한 고유 식별자가 필요합니다. 또한 소비자는 멱등성을 염두에 두고 설계되어야 합니다. 이렇게 하면 이벤트를 복제하거나 아카이브에서 재생하는 경우 두 리전에서 이벤트가 처리되더라도 부작용이 발생하지 않습니다.

이벤트 제한 방지

이벤트가 제한되는 것을 방지하려면 리전 간에 일관성을 유지하도록 `PutEvents` 및 대상 제한을 업데이트하는 것이 좋습니다.

Amazon Route 53 상태 확인에서 구독자 지표 사용

Amazon Route 53 상태 확인에서 구독자 지표를 포함하지 마세요. 이 지표를 포함하면 기본 리전에서 다른 모든 구독자가 정상 상태임에도 구독자에게 문제가 발생할 경우, 게시자가 보조 리전으로 장애 조치될 수 있습니다. 구독자 중 한 명이 기본 리전의 이벤트를 처리하지 못하는 경우 보조 리전의 구독자가 이벤트를 성공적으로 처리할 수 있도록 복제 기능을 설정해야 합니다.

Route 53 상태 확인 설정을 위한 AWS CloudFormation 템플릿

글로벌 엔드포인트를 사용하는 경우 리전 상태를 모니터링하려면 Route 53 상태 확인이 필요합니다. 다음 템플릿은 Amazon CloudWatch 경보를 정의하고 이를 사용하여 Route 53 상태 확인을 정의합니다.

주제

- Route 53 상태 확인을 정의하기 위한 AWS CloudFormation 템플릿
- CloudWatch 경보 템플릿 속성
- Route 53 상태 확인 템플릿 속성

Route 53 상태 확인을 정의하기 위한 AWS CloudFormation 템플릿

다음 템플릿을 사용하여 Route 53 상태 확인을 정의합니다.

```
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
```

이벤트 ID는 API 직접 호출에 따라 변경될 수 있으므로 리전 간에 이벤트를 상호 연결하려면 변경이 불가능한 고유 식별자가 필요합니다. 또한 소비자는 멱등성을 염두에 두고 설계되어야 합니다. 이렇게 하면 이벤트를 복제하거나 아카이브에서 재생하는 경우 두 리전에서 이벤트가 처리되더라도 부작용이 발생하지 않습니다.

CloudWatch 경보 템플릿 속성

> **Note**
>
> 모든 **`editable`** 필드에 대해 초당 처리량을 고려합니다. 이벤트를 간헐적으로만 전송하는 경우 평가 기간을 늘리거나 누락된 데이터를 `breaching` 대신 `missing`으로 처리하도록 상태 확인을 변경하는 것이 좋습니다.

템플릿의 CloudWatch 경보 섹션에는 다음과 같은 속성이 사용됩니다.

지표	설명
`AlarmDescription`	경보에 대한 설명입니다. 기본값: **`High Latency in Amazon EventBridge`**
`MetricName`	경보와 연결된 지표의 이름입니다. 지표 기반의 경보에 필수적입니다. 수학 표현식을 기반으로 경보를 생성하면 `Metrics`를 대신 사용해야 하며, `MetricName` 을 지정할 수 없습니다. 기본값: IngestionToInvocationStartLatency
`Namespace`	경보와 연결된 지표의 네임스페이스입니다. 지표 기반의 경보에 필수적입니다. 수학 표현식을 기반으로 경보를 생성하면 `Namespace` 를 지정할 수 없으며 대신 `Metrics`를 사용해야 합니다. 기본값: `AWS/Events`
`Statistic`	경보와 연결된 백분위수 이외의 지표 통계입니다. 기본값: 평균
`Period`	통계가 적용되는 기간(초)입니다. 지표 기반의 경보에 필수적입니다. 유효한 값은 10, 30, 60 및 60의 배수입니다.

지표	설명
	기본값: **60**
EvaluationPeriods	지정 임계값에 대한 데이터 비교가 이루어지는 기간의 일수입니다. 경보를 트리거하기 위해 연속된 여러 데이터 요소를 위반해야 하는 경보를 설정하는 경우 이 값은 해당 일수를 지정합니다. “N 중 M” 경보를 설정하는 경우 이 값은 N이고 DatapointsToAlarm 은 M입니다. 기본값: **5**
Threshold	지정된 통계와 비교할 값입니다. 기본값: **30,000**
ComparisonOperator	지정한 통계와 임계값을 비교하는 경우 사용하는 산술 연산입니다. 지정된 통계 값은 첫 번째 피연산자로 사용됩니다. 기본값: GreaterThanThreshold
TreatMissingData	경보가 누락된 데이터 요소를 처리하는 방법을 설정합니다. 유효한 값: breaching , notBreaching , ignore 및 missing 기본값: breaching

Route 53 상태 확인 템플릿 속성

> **Note**
>
> 모든 **editable** 필드에 대해 초당 처리량을 고려합니다. 이벤트를 간헐적으로만 전송하는 경우 평가 기간을 늘리거나 누락된 데이터를 breaching 대신 missing으로 처리하도록 상태 확인을 변경하는 것이 좋습니다.

템플릿의 Route 53 상태 확인 섹션에는 다음과 같은 속성이 사용됩니다.

지표	설명
HealthCheckName	상태 확인의 이름입니다. 기본값: **LatencyFailuresHealthCheck**
InsufficientDataHealthStatus	CloudWatch가 경보 상태를 확인하기에 지표 데이터가 부족한 경우, Amazon Route 53이 상태 확인에 할당할 상태입니다. 유효한 값: • Healthy: Route 53이 상태 확인을 정상으로 간주합니다. • Unhealthy : Route 53이 상태 확인을 비정상으로 간주합니다. • LastKnownStatus : Route 53은 CloudWatch가 경보 상태를 확인하기에 충분한 데이터를 갖고 있던 마지막 시점에 수행한 상태 확인의 상태를 사용합니다. 마지막으로 알려진 상태가 없는 새로운 상태 확인의 경우 상태 확인의 기본 상태는 정상입니다. 기본값: 비정상 **Note** 이 필드는 TreatMissingData 필드 입력에 따라 업데이트됩니다. TreatingMissingData 를 Missing으로 설정하면 LastKnownStatus 로 업데이트됩니다. TreatingMissingData 를 Breaching 으로 설정하면 Unhealthy 로 업데이트됩니다.

아마존 EventBridge 스키마

스키마는 전송되는 이벤트의 구조를 정의합니다. EventBridge EventBridge AWS 서비스에서 생성되는 모든 이벤트에 대한 스키마를 제공합니다. 사용자 지정 스키마를 생성 또는 업로드하거나 이벤트 버스의 이벤트에서 직접 스키마를 유추할 수도 있습니다. 이벤트에 대한 스키마가 있으면 인기 있는 프로그래밍 언어에 대한 코드 바인딩을 다운로드하고 개발 속도를 높일 수 있습니다. API를 사용하여 EventBridge 콘솔에서 또는 툴킷을 사용하여 IDE에서 직접 스키마용 코드 바인딩으로 작업하고 스키마를 관리할 수 있습니다. AWS 이벤트를 사용하는 서버리스 앱을 빌드하려면 AWS Serverless Application Model을 사용하세요.

> **Note**
>
> 입력 변환기 기능을 사용하는 경우 대상으로 전송되는 변환된 이벤트가 아닌 스키마 검색을 통해 원래 이벤트를 유추합니다.

EventBridge OpenAPI 3 및 JSON스키마 드래프트4 형식을 모두 지원합니다.

VS Code용AWSAWS Toolkit JetBrains 및 Toolkit용 Toolkit의 경우 IDE에서 직접 스키마를 찾아보거나 검색하고 스키마용 코드 바인딩을 다운로드할 수 있습니다.

다음 동영상은 스키마와 스키마 레지스트리에 대한 개요를 제공합니다. 스키마 레지스트리 사용

주제

- 스키마 레지스트리 API 속성 값 마스킹
- 아마존 EventBridge 스키마 찾기
- Amazon EventBridge 스키마 레지스트리
- 아마존 EventBridge 스키마 생성
- 아마존 EventBridge 코드 바인딩

스키마 레지스트리 API 속성 값 마스킹

스키마 레지스트리를 생성하는 데 사용되는 이벤트의 일부 속성 값에는 민감한 고객 정보가 포함될 수 있습니다. 고객 정보를 보호하기 위해 해당 값은 별표(*)로 마스킹됩니다. 이러한 값은 마스킹되므로 다

음 속성이나 해당 값에 명시적으로 의존하는 응용 프로그램은 빌드하지 않는 것이 EventBridge 좋습니다.

- CreateSchema— 신체의 `Content` 속성 `requestParameters`
- GetDiscoveredSchema— `requestParameters` 신체의 `Events` 재산과 `responseElements` 신체의 `Content` 재산
- SearchSchemas— `keywords` 재산의 재산 `requestParameters`
- UpdateSchema— 의 `Content` 재산 `requestParameters`

아마존 EventBridge 스키마 찾기

EventBridge 이벤트를 생성하는 모든 AWS 서비스의 스키마를 포함합니다. 이러한 스키마는 EventBridge 콘솔에서 찾거나 API 작업을 사용하여 찾을 수 있습니다. `SearchSchemas`

콘솔에서 AWS 서비스의 스키마를 찾으려면 EventBridge

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 Schemas(스키마)를 선택합니다.
3. 스키마 페이지에서 AWS 이벤트 스키마 레지스트리를 선택합니다.
   <result>

   사용 가능한 스키마의 첫 페이지가 표시됩니다.
   </result>
4. 스키마를 찾으려면 AWS 이벤트 스키마 검색에 검색어를 입력합니다.

   검색 시 사용 가능한 스키마의 이름과 내용 모두에 대해 일치하는 항목이 반환된 후 일치 항목이 포함된 스키마의 버전이 표시됩니다.
5. 스키마 이름을 선택하여 이벤트 스키마를 엽니다.

Amazon EventBridge 스키마 레지스트리

스키마 레지스트리는 스키마의 컨테이너입니다. 스키마 레지스트리는 스키마가 논리적 그룹에 있도록 스키마를 수집하고 구성합니다. 기본 스키마 레지스트리는 다음과 같습니다.

- 모든 스키마 — AWS 이벤트의 모든 스키마, 검색된 스키마 및 사용자 지정 스키마 레지스트리.
- AWS 이벤트 스키마 레지스트리 - 기본 제공 스키마입니다.
- 검색된 스키마 레지스트리 - 스키마 검색으로 검색된 스키마입니다.

사용자 지정 레지스트리를 생성하여 생성하거나 업로드하는 스키마를 구성할 수 있습니다.

사용자 지정 레지스트리를 생성하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 스키마를 선택한 다음, 레지스트리 생성을 선택합니다.
3. 레지스트리 세부 정보 페이지에서 이름을 입력합니다.
4. (선택 사항) 새 레지스트리에 대한 설명을 입력합니다.
5. Create를 선택합니다.

새 레지스트리에 사용자 지정 스키마를 만들려면 사용자 지정 스키마 생성을 선택합니다. 레지스트리에 스키마를 추가하려면 새 스키마를 생성할 때 해당 레지스트리를 선택합니다.

API를 사용하여 레지스트리를 생성하려면 `CreateRegistry`를 사용합니다. 자세한 내용은 Amazon EventBridge 스키마 레지스트리 API 참조를 참조하십시오.

EventBridge 스키마 레지스트리를 AWS CloudFormation사용하는 방법에 대한 자세한 내용은 의 EventSchemas 리소스 유형 참조를 참조하십시오 AWS CloudFormation.

아마존 EventBridge 스키마 생성

OpenAPI 사양 또는 JSONSchema Draft4 사양이 포함된 JSON 파일을 사용해 스키마를 생성합니다. 템플릿을 사용하거나 이벤트의 JSON을 기반으로 스키마를 EventBridge 생성하여 자체 스키마를 만들거나 업로드할 수 있습니다. 이벤트 버스의 이벤트에서 스키마를 유추할 수도 있습니다. 스키마 레지스트리 API를 사용하여 EventBridge 스키마를 만들려면 API 작업을 사용하십시오. `CreateSchema`

OpenAPI 3과 JSONSchema Draft4 형식 중에서 선택할 때 다음과 같은 차이점을 고려하세요.

- JSONSchema 형식은 OpenAPI에서 지원되지 않는 추가 키워드(예: `$schema`, `additionalItems`)를 지원합니다.
- 키워드 처리 방식에는 `type` 및 `format`과 같이 사소한 차이가 있습니다.
- OpenAPI는 JSON 문서의 JSONSchema 하이퍼-스키마 하이퍼링크를 지원하지 않습니다.
- OpenAPI용 도구는 대체로 빌드 시간에 중점을 두는 반면, JSONSchema용 도구는 스키마 검증을 위한 클라이언트 도구와 같이 런타임 작업에 중점을 두는 경향이 있습니다.

JSONSchema 형식을 사용하여 클라이언트측 유효성 검사를 구현하여 전송된 이벤트가 스키마를 EventBridge 준수하도록 하는 것이 좋습니다. JSONSchema를 사용하여 유효한 JSON 문서에 대한 계약을 정의한 다음, 관련 이벤트를 보내기 전에 JSON 스키마 유효성 검사기를 사용할 수 있습니다.

새 스키마를 만든 후에는 코드 바인딩을 다운로드하여 해당 스키마를 통해 이벤트용 애플리케이션을 생성할 수 있습니다.

주제

- 템플릿을 사용하여 스키마 생성
- 콘솔에서 직접 스키마 템플릿 편집
- 이벤트의 JSON에서 스키마 생성
- 이벤트 버스의 이벤트에서 스키마 생성

템플릿을 사용하여 스키마 생성

템플릿에서 또는 콘솔에서 직접 템플릿을 편집하여 스키마를 생성할 수 있습니다. EventBridge 템플릿을 가져오려면 콘솔에서 다운로드합니다. 스키마가 이벤트와 일치하도록 템플릿을 편집할 수 있습니다. 그런 다음, 콘솔을 통해 새 템플릿을 업로드합니다.

템플릿 파일을 다운로드하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 Schema registry(스키마 레지스트리)를 선택합니다.
3. 스키마 템플릿 아래의 시작하기 섹션에서 다운로드를 선택합니다.

또는 다음 코드 예제에서 JSON 템플릿을 복사할 수 있습니다.

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

스키마 템플릿을 업로드하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 스키마를 선택한 다음, 스키마 생성을 선택합니다.
3. (선택 사항) 스키마 레지스트리를 선택하거나 생성합니다.
4. 스키마 세부 정보에서 스키마의 이름을 입력합니다.
5. (선택 사항) 스키마에 대한 설명을 입력합니다.
6. 스키마 유형에는 OpenAPI 3.0 또는 JSON Schema Draft 4를 선택합니다.
7. 생성 탭의 텍스트 상자에서 스키마 파일을 텍스트 상자로 끌어오거나 스키마 소스를 붙여넣습니다.
8. [생성(Create)]을 선택합니다.

콘솔에서 직접 스키마 템플릿 편집

콘솔에서 스키마를 편집하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 스키마를 선택한 다음, 스키마 생성을 선택합니다.
3. (선택 사항) 스키마 레지스트리를 선택하거나 생성합니다.
4. 스키마 세부 정보에서 스키마의 이름을 입력합니다.
5. 스키마 유형에는 OpenAPI 3.0 또는 JSON Schema Draft 4를 선택합니다.
6. (선택 사항) 생성할 스키마에 대한 설명을 입력합니다.
7. 생성 탭에서 템플릿 로드를 선택합니다.
8. 텍스트 상자에서 스키마가 이벤트와 일치하도록 템플릿을 편집합니다.
9. [생성(Create)]을 선택합니다.

이벤트의 JSON에서 스키마 생성

이벤트의 JSON이 있는 경우 해당 이벤트 유형에 대한 스키마를 자동으로 생성할 수 있습니다.

이벤트의 JSON을 기반으로 스키마를 생성하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 스키마를 선택한 다음, 스키마 생성을 선택합니다.
3. (선택 사항) 스키마 레지스트리를 선택하거나 생성합니다.
4. Schema details(스키마 세부 정보)에서 스키마의 이름을 입력합니다.
5. (선택 사항) 생성한 스키마에 대한 설명을 입력합니다.
6. 스키마 유형에는 OpenAPI 3.0을 선택합니다.

   이벤트의 JSON에서 스키마를 생성할 때는 JSONSchema를 사용할 수 없습니다.
7. Discover from JSON(JSON에서 검색)을 선택합니다.
8. JSON 아래의 텍스트 상자에서 이벤트의 JSON 소스를 붙여넣거나 끌어옵니다.

   예를 들어, 실행 실패의 경우 이 AWS Step Functions 이벤트의 소스를 붙여넣을 수 있습니다.

```
{
    "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. Discover schema(스키마 발견)를 선택합니다.
10. EventBridge 이벤트에 대한 OpenAPI 스키마를 생성합니다. 예를 들어 이전 Step Functions 이벤트에 대해 다음 스키마가 생성됩니다.

```
{
  "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. 스키마가 생성된 후 생성을 선택합니다.

이벤트 버스의 이벤트에서 스키마 생성

EventBridge 이벤트를 발견하여 스키마를 유추할 수 있습니다. 스키마를 유추하기 위해 이벤트 버스에서 이벤트 검색을 켜면 교차 계정 이벤트용 스키마를 포함하여 모든 고유 스키마가 스키마 레지스트리에 추가됩니다. 에서 검색한 스키마는 스키마 페이지의 검색된 스키마 레지스트리에 EventBridge 나타납니다.

이벤트 버스의 이벤트 내용이 변경되면 관련 스키마의 새 버전이 EventBridge 생성됩니다. EventBridge

> **Note**
>
> 이벤트 버스에서 이벤트 검색을 사용하도록 설정하면 비용이 발생할 수 있습니다. 매월 처음 5백만 건의 처리 이벤트는 무료입니다.

> **Note**
>
> EventBridge 기본적으로 계정 간 이벤트에서 스키마를 유추하지만 속성을 업데이트하여 스키마를 비활성화할 수 있습니다. `cross-account` 자세한 내용은 스키마 레지스트리 API 참조의 Discoverers를 EventBridge 참조하십시오.

이벤트 버스에서 스키마 검색을 사용하도록 설정하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 이벤트 버스를 선택합니다.
3. 다음 중 하나를 수행하십시오.
   - 기본 이벤트 버스에서 검색을 사용하도록 설정하려면 검색 시작을 선택합니다.

- 사용자 지정 이벤트 버스에서 검색을 사용하도록 설정하려면 사용자 지정 이벤트 버스의 라디오 버튼을 선택하고 검색 시작을 선택합니다.

아마존 EventBridge 코드 바인딩

이벤트 스키마에 대한 코드 바인딩을 생성하여 Golang, Java, Python 등에서 개발 속도를 높일 수 있습니다. TypeScript 코드 바인딩은 AWS 서비스 이벤트, 사용자가 만든 스키마, 이벤트 버스의 이벤트를 기반으로 생성한 스키마에 대해 사용할 수 있습니다. EventBridge 콘솔, 스키마 레지스트리 API를 사용하거나 툴킷이 있는 IDE에서 EventBridge 스키마에 대한 코드 바인딩을 생성할 수 있습니다. AWS

스키마에서 코드 바인딩을 생성하려면 EventBridge

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 Schemas(스키마)를 선택합니다.
3. 스키마 레지스트리를 살펴보거나 스키마를 검색하여 코드 바인딩이 필요한 스키마를 찾습니다.
4. 스키마 이름을 선택합니다.
5. 스키마 세부 정보 페이지의 버전 섹션에서 코드 바인딩 다운로드를 선택합니다.
6. Download code bindings(코드 바인딩 다운로드) 페이지에서 다운로드할 코드 바인딩의 언어를 선택합니다.
7. 다운로드를 선택합니다.

   다운로드가 시작되는 데 몇 초가 걸릴 수 있습니다. 다운로드 파일은 선택한 언어에 대한 코드 바인딩의 zip 파일입니다.

Amazon EventBridge 관련 서비스 및 도구

Amazon EventBridge는 다른 AWS 서비스 및 도구와 함께 작동하여 이벤트를 처리하거나 리소스를 규칙의 대상으로 간접 호출합니다. 다른 AWS 서비스와의 EventBridge 통합에 대한 자세한 내용은 다음을 참조하십시오.

주제

- 인터페이스 VPC 엔드포인트에서 Amazon EventBridge 사용
- AWS X-Ray와 Amazon EventBridge 통합
- AWS 통합 어플리케이션 테스트 키트와 EventBridge 함께 사용
- AWS CloudFormation 스택에 Amazon EventBridge 리소스 포함

인터페이스 VPC 엔드포인트에서 Amazon EventBridge 사용

Amazon Virtual Private Cloud(VPC)를 사용하여 AWS 리소스를 호스트하는 경우, VPC와 EventBridge 간에 프라이빗 연결을 설정할 수 있습니다. VPC의 리소스는 이 연결을 사용하여 EventBridge와 통신할 수 있습니다.

VPC를 사용하여 IP 주소 범위, 서브넷, 라우팅 테이블, 네트워크 게이트웨이 등의 네트워크 설정을 제어할 수 있습니다. VPC를 EventBridge에 연결하려면 EventBridge에 대한 인터페이스 VPC 엔드포인트를 정의합니다. 이 엔드포인트를 이용하면 인터넷 게이트웨이나 NAT(네트워크 주소 변환) 인스턴스 또는 VPN 연결 없이도 EventBridge에 안정적이고 확장 가능하게 연결됩니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon VPC란 무엇입니까? 섹션을 참조하세요.

인터페이스 VPC 엔드포인트는 프라이빗 IP 주소와 함께 탄력적 네트워크 인터페이스를 사용하여 AWS 서비스 간 프라이빗 통신을 사용할 수 있는 기술인 AWS PrivateLink에 의해 구동됩니다. 자세한 내용은 AWS PrivateLink 및 VPC 엔드포인트를 참조하십시오.

프라이빗 인터페이스 VPC 엔드포인트를 사용하는 경우 VPC가 EventBridge로 전송하는 사용자 지정 이벤트는 해당 엔드포인트를 사용합니다. 그러면 EventBridge는 사용자가 구성한 규칙 및 대상에 따라 해당 이벤트를 다른 AWS 서비스에 보냅니다. 이벤트가 다른 서비스로 전송되면 해당 서비스의 퍼블릭 엔드포인트 또는 VPC 엔드포인트를 통해 이벤트를 수신할 수 있습니다. 예를 들어 Amazon SQS 대기열로 이벤트를 전송하는 규칙을 생성하는 경우, 퍼블릭 엔드포인트를 사용하지 않고 VPC의 해당 대기열에서 메시지를 수신하도록 Amazon SQS용 인터페이스 VPC 엔드포인트를 구성할 수 있습니다.

가용성

현재 EventBridge가 VPC 엔드포인트를 지원하는 리전은 다음과 같습니다.

- 미국 동부(오하이오)
- 미국 동부(버지니아 북부)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오레곤)
- 아프리카(케이프타운)
- 아시아 태평양(뭄바이)
- 아시아 태평양(하이데라바드)
- 아시아 태평양(홍콩)
- 아시아 태평양(싱가포르)

- 아시아 태평양(시드니)
- 아시아 태평양(자카르타)
- 아시아 태평양(멜버른)
- 아시아 태평양(도쿄)
- 아시아 태평양(서울)
- 아시아 태평양(오사카)
- 캐나다(중부)
- 캐나다 서부(캘거리)
- 중국(베이징)
- 중국(닝샤)
- 유럽(프랑크푸르트)
- 유럽(취리히)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(밀라노)
- 유럽(스페인)
- 유럽(파리)
- 유럽(스톡홀름)
- 중동(UAE)
- 중동(바레인)
- 남아메리카(상파울루)
- 이스라엘(텔아비브)
- AWS GovCloud(미국 서부)
- AWS GovCloud(미국 동부)

EventBridge에 대한 VPC 엔드포인트 생성

VPC에서 EventBridge를 사용하려면 EventBridge용 인터페이스 VPC 엔드포인트를 생성하고 com.amazonaws.***Region***.events를 서비스 이름으로 선택합니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하세요.

EventBridge 파이프 세부 사항

인터페이스 VPC 엔드포인트에 대한 전체 EventBridge 파이프 지원은 제공되지 않습니다. EventBridge 파이프가 있는 VPC 내에서 다음 소스를 사용하려면 다음을 참조하세요.

- 아마존 MSK 네트워크 구성
- 자체 관리형 Apache Kafka 네트워크 구성
- Amazon MQ 네트워크 구성

AWS X-Ray와 Amazon EventBridge 통합

AWS X-Ray를 사용하여 EventBridge를 통과하는 이벤트를 추적할 수 있습니다. EventBridge는 대상 서비스가 추적, 분석 및 디버깅할 수 있도록 원본 추적 헤더를 대상에 전달합니다.

EventBridge는 추적 컨텍스트를 전달한 `PutEvents` 요청에서 이벤트가 발생한 경우에만 이벤트에 대한 추적 헤더를 전달할 수 있습니다. X-Ray는 타사 파트너, 예정된 이벤트 또는 AWS서비스에서 발생한 이벤트를 추적하지 않으며 이러한 이벤트 소스는 X-Ray 서비스 맵에 표시되지 않습니다.

X-Ray는 추적 헤더의 유효성을 검사하고 유효하지 않은 추적 헤더는 삭제됩니다. 하지만 이벤트는 계속 처리됩니다.

> **Important**
>
> 간접 호출 대상에 전달된 이벤트에서는 추적 헤더를 사용할 수 없습니다.
>
> - 이벤트 아카이브가 있는 경우 아카이브된 이벤트에는 추적 헤더를 사용할 수 없습니다. 아카이브된 이벤트를 재생하는 경우 추적 헤더는 포함되지 않습니다.
> - DLQ(Dead Letter Queue)가 있는 경우 이벤트를 DLQ로 보내는 `SendMessage` 요청에 추적 헤더가 포함됩니다. `ReceiveMessage`를 사용하여 DLQ에서 이벤트(메시지)를 검색하는 경우 이벤트와 관련된 추적 헤더는 Amazon SQS 메시지 속성에 포함되지만 이벤트 메시지에는 포함되지 않습니다.

EventBridge 이벤트 노드가 소스 및 대상 서비스를 연결하는 방법에 대한 자세한 내용은 AWS X-Ray 개발자 안내서의 X-Ray 서비스 맵에서 소스 및 대상 보기를 참조하십시오.

EventBridge를 통해 다음과 같은 추적 헤더 정보를 전달할 수 있습니다.

- 기본 HTTP 헤더 — X-Ray SDK는 모든 간접 호출 대상의 `X-Amzn-Trace-Id` HTTP 헤더로 추적 헤더를 자동으로 채웁니다. 기본 HTTP 헤더에 대한 자세한 내용은 AWS X-Ray 개발자 안내서의 추적 헤더를 참조하십시오.
- **`TraceHeader`** 시스템 속성 - `TraceHeader`는 X-Ray 추적 헤더를 대상으로 전달하기 위해 EventBridge에서 예약한 PutEventsRequestEntry 속성입니다. `PutEventsRequestEntry`도 사용하는 경우 `PutEventsRequestEntry`가 HTTP 추적 헤더를 재정의합니다.

> **Note**
>
> 추적 헤더는 `PutEventsRequestEntry` 이벤트 크기에 포함되지 않습니다. 자세한 내용은 Amazon EventBridge PutEvents 이벤트 항목 크기 계산 섹션을 참조하세요.

다음 동영상은 X-Ray와 EventBridge를 함께 사용하는 방법을 보여줍니다. 추적을 위해 AWS X-Ray 사용

AWS 통합 어플리케이션 테스트 키트와 EventBridge 함께 사용

Lambda EventBridge 또는 Step Functions와 같은 서버리스 서비스로 구성된 애플리케이션을 생성하는 경우 아키텍처 구성 요소 중 상당수는 데스크톱에 배포할 수 없고 클라우드에만 존재합니다. AWS 로컬에 배포된 애플리케이션을 사용하는 것과 달리, 이러한 유형의 애플리케이션은 자동화된 테스트를 수행하기 위한 클라우드 기반 전략을 활용할 수 있습니다. AWS 통합 애플리케이션 테스트 키트 (AWS IATK) 를 사용하면 애플리케이션에 이러한 전략 중 일부를 구현할 수 있습니다.

AWS IATK는 클라우드 기반 애플리케이션을 위한 자동화된 테스트를 작성하는 데 도움이 되는 소프트웨어 라이브러리입니다.

EventBridge IATK와의 통합 AWS

AWS IATK와 함께 EventBridge 이벤트 및 이벤트 버스를 사용하여 다음을 포함한 자동 테스트를 구현할 수 있습니다.

테스트 하네스 구현

이벤트 기반 아키텍처를 위한 통합 테스트를 작성하려면 애플리케이션을 하위 시스템으로 분할하여 논리적 경계를 설정하십시오. 하위 시스템을 테스트하는 데 유용한 방법 중 하나는 테스트 하네스, 즉 하위 시스템을 테스트하기 위해 특별히 만든 리소스를 생성하는 것입니다.

예를 들어 통합 테스트는 입력 테스트 이벤트를 하위 시스템 프로세스에 전달하여 하위 시스템 프로세스를 시작할 수 있습니다. AWS IATK는 출력 이벤트를 수신하는 테스트 하네스를 자동으로 만들 수 있습니다. EventBridge (기본적으로 하네스는 출력 이벤트를 Amazon SQS로 전달하는 EventBridge 규칙으로 구성되어 있습니다.) 그런 다음 통합 테스트는 테스트 하네스를 쿼리하여 출력을 검사하고 테스트의 통과 또는 실패 여부를 결정합니다.

모의 이벤트 생성

AWS IATK는 스키마 레지스트리에 저장된 스키마에서 모의 이벤트를 생성할 수 있는 기능을 제공합니다. EventBridge 이를 통해 모의 이벤트를 생성하고 생성된 이벤트로 모든 소비자(예: Lambda 함수 또는 Step Functions 상태 머신)를 간접 호출할 수 있습니다.

자세한 내용은 AWS 통합 애플리케이션 테스트 키트 개요를 참조하십시오. GitHub

AWS CloudFormation 스택에 Amazon EventBridge 리소스 포함

AWS CloudFormation 인프라를 코드로 취급하여 중앙 집중화되고 반복 가능한 방식으로 계정 및 지역 전반의 AWS 리소스를 구성하고 관리할 수 있습니다. CloudFormation 프로비저닝 및 관리하려는 리소스를 정의하는 템플릿을 생성하여 이를 수행할 수 있습니다. 이러한 리소스에는 이벤트 버스 및 규칙, 파이프, 스키마, 일정 등과 같은 EventBridge 아티팩트가 포함될 수 있습니다. 이러한 리소스를 사용하여 프로비저닝하고 관리하는 기술 스택에 EventBridge 기능을 포함시키십시오. CloudFormation

Amazon EventBridge 리소스는 다음에서 사용할 수 있습니다. AWS CloudFormation

EventBridge 다음 리소스 네임스페이스의 CloudFormation 템플릿에 사용할 리소스를 제공합니다.

- AWS::Events

  템플릿 예시는 다음과 같습니다.

  - 에 대한 API 대상을 생성합니다. PagerDuty
  - Slack에 대한 API 대상 생성
  - ApiKey 권한 부여 파라미터를 사용하여 연결 생성
  - OAuth 인증 파라미터를 사용하여 연결 생성
  - 이벤트 복제를 사용하여 글로벌 엔드포인트 생성
  - 여러 보안 주체 및 작업을 사용하는 정책 거부
  - 사용자 지정 이벤트 버스를 사용하여 조직에 권한 부여
  - 크로스 리전 규칙 생성
  - 대상에 대한 DLQ(Dead Letter Queue)를 포함하는 규칙 생성
  - 정기적으로 Lambda 함수 간접 호출
  - 이벤트에 응답하여 Lambda 함수 간접 호출

  - 로그 항목에 응답하여 주제 알림
- AWS::Event스키마
- AWS::Pipes

  템플릿 예시는 다음과 같습니다.

  - 이벤트 필터를 사용하여 파이프 생성
- AWS::Scheduler

AWS CloudFormation 템플릿에 대한 Amazon EventBridge 리소스 정의 생성

CloudFormation 템플릿 개발을 빠르게 시작하는 데 도움이 되는 EventBridge 콘솔을 사용하면 계정의 기존 이벤트 버스, 규칙 및 파이프에서 CloudFormation 템플릿을 생성할 수 있습니다.

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

기본 이벤트 버스를 관리할 수 있게 합니다. AWS CloudFormation

기본 이벤트 버스가 계정에 자동으로 EventBridge 프로비저닝되므로 CloudFormation 스택에 포함하려는 리소스에 대해 일반적으로 사용하는 것처럼 CloudFormation 템플릿을 사용하여 생성할 수 없습니다. 스택에 기본 이벤트 버스를 포함하려면 먼저 CloudFormation 스택으로 기본 이벤트 버스를 가져와야 합니다. 기본 이벤트 버스를 스택으로 가져온 후에는 필요에 따라 이벤트 버스 속성을 업데이트할 수 있습니다.

자세한 내용은 ??? 섹션을 참조하세요.

를 사용하여 AWS CloudFormation 스택 이벤트를 관리합니다. EventBridge

CloudFormation 스택에 EventBridge 리소스를 포함하는 것 외에도 스택 자체에서 생성된 CloudFormation 이벤트를 관리하는 EventBridge 데 사용할 수 있습니다. CloudFormation 스택에서 생성, 업데이트, 삭제 또는 드리프트 감지 작업이 EventBridge 수행될 때마다 이벤트를 전송합니다. CloudFormation 또한 스택 세트 및 스택 세트 인스턴스의 상태 변경 이벤트를 EventBridge 에 전송합니다. EventBridge 규칙을 사용하여 정의된 대상으로 이벤트를 라우팅할 수 있습니다.

자세한 내용은 사용 EventBridge AWS CloudFormation 설명서에서 CloudFormation 이벤트 관리를 참조하십시오.

Amazon EventBridge 자습서

EventBridge는 다양한 AWS 서비스 및 SaaS 파트너와 통합됩니다. 이 자습서는 EventBridge의 기본 사항과 이를 서버리스 아키텍처의 일부로 사용하는 방법을 익히는 데 도움이 되도록 설계되었습니다.

자습서:

- Amazon EventBridge 시작하기 자습서
- 다른 AWS 서비스와의 통합을 위한 Amazon EventBridge 자습서
- SaaS 공급자와의 통합을 위한 Amazon EventBridge 자습서

Amazon EventBridge 시작하기 자습서

다음 자습서를 통해 EventBridge와 그 사용 방법을 살펴볼 수 있습니다.

자습서:

- Amazon EventBridge 이벤트 아카이브 및 재생
- Amazon EventBridge 샘플 애플리케이션 생성
- 자습서: EventBridge 스키마 레지스트리를 사용하여 이벤트용 코드 바인딩 다운로드
- 자습서: 입력 변환기를 사용하여 EventBridge가 이벤트 대상에 전달하는 내용을 사용자 지정

Amazon EventBridge 이벤트 아카이브 및 재생

EventBridge를 사용하면 규칙을 사용하여 이벤트를 특정 AWS Lambda함수로 라우팅할 수 있습니다.

이 자습서에서는 Lambda 콘솔을 사용하여 EventBridge 규칙의 대상으로 사용할 함수를 생성합니다. 그런 다음 EventBridge 콘솔을 사용하여 테스트 이벤트를 보관할 아카이브와 규칙을 생성합니다. 해당 아카이브에 이벤트가 있으면 다시 재생해야 합니다.

단계:

- 1단계: Lambda 함수 생성
- 2단계: 아카이브 생성
- 3단계: 규칙 생성
- 4단계: 테스트 이벤트 보내기
- 5단계: 다시 재생 이벤트
- 6단계: 리소스 정리

1단계: Lambda 함수 생성

먼저 이벤트를 기록하는 Lambda 함수를 생성합니다.

Lambda 함수를 생성하려면

1. https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
2. 함수 생성을 선택합니다.
3. 새로 작성을 선택합니다.
4. Lambda 함수의 이름과 설명을 입력합니다. 예를 들어 함수 이름을 `LogScheduledEvent`로 지정합니다.
5. 나머지 옵션은 기본값으로 두고 함수 생성을 선택합니다.
6. 함수 페이지의 코드 탭에서 index.js를 두 번 클릭합니다.
7. 기존 JavaScript 코드를 다음 코드로 바꿉니다.

```
'use strict';

exports.handler = (event, context, callback) => {
    console.log('LogScheduledEvent');
    console.log('Received event:', JSON.stringify(event, null, 2));
```

```
    callback(null, 'Finished');
};
```

8. 배포를 선택합니다.

2단계: 아카이브 생성

다음으로, 모든 테스트 이벤트를 보관할 아카이브를 생성합니다.

아카이브를 생성하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 아카이브를 선택합니다.
3. 아카이브 생성을 선택합니다.
4. 아카이브 이름 및 설명을 입력합니다. 예를 들어, 아카이브 이름을 `ArchiveTest`로 지정합니다.
5. 나머지 옵션은 기본값으로 두고 다음을 선택합니다.
6. 아카이브 생성을 선택합니다.

3단계: 규칙 생성

이벤트 버스로 전송된 이벤트를 보관하는 규칙을 생성합니다.

규칙을 생성하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 [Rules]를 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙에 대해 이름과 설명을 입력하세요. 예를 들어, 규칙의 이름을 `ARTestRule`로 지정합니다.

   규칙은 동일한 리전과 동일한 이벤트 버스의 다른 규칙과 동일한 이름을 가질 수 없습니다.
5. 이벤트 버스에서 이 규칙과 연결할 이벤트 버스를 선택합니다. 이 규칙이 자신의 계정에서 발생하는 이벤트와 일치하도록 하려면 기본을 선택합니다. 계정의 AWS 서비스가 이벤트를 출력하면 항상 계정의 기본 이벤트 버스로 이동합니다.
6. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다.
7. 다음을 선택합니다.
8. 이벤트 소스에서 기타를 선택합니다.

9. 이벤트 패턴에 다음을 입력합니다.

```
{
  "detail-type": [
    "customerCreated"
  ]
}
```

10. 다음을 선택합니다.
11. 대상 유형에서 AWS서비스를 선택합니다.
12. 대상 선택의 경우 드롭다운 목록에서 Lambda 함수를 선택합니다.
13. 함수의 경우 1단계: Lambda 함수 생성 섹션에서 생성한 Lambda 함수를 선택합니다. 이 예시에서는 `LogScheduledEvent`를 선택합니다.
14. 다음을 선택합니다.
15. 다음을 선택합니다.
16. 규칙의 세부 정보를 검토하고 규칙 생성을 선택합니다.

4단계: 테스트 이벤트 보내기

아카이브와 규칙을 설정했으니 이제 아카이브가 제대로 작동하는지 확인하기 위해 테스트 이벤트를 전송하겠습니다.

> **Note**
>
> 이벤트가 아카이브에 도달하는 데 시간이 걸릴 수 있습니다.

테스트 이벤트를 보내려면(콘솔)

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 이벤트 버스를 선택합니다.
3. 기본 이벤트 버스 타일에서 액션, 이벤트 전송을 선택합니다.
4. 이벤트 소스를 입력합니다. 예: `TestEvent`.
5. 세부 정보 유형에는 `customerCreated`를 입력합니다.
6. 이벤트 세부 정보에는 `{}`를 입력합니다.
7. 전송을 선택합니다.

5단계: 다시 재생 이벤트

테스트 이벤트가 아카이브에 저장되면 다시 재생할 수 있습니다.

보관된 이벤트를 재생하려면(콘솔)

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 재생을 선택합니다.
3. 다시 재생 새로 시작을 선택합니다.
4. 다시 재생의 이름 및 설명을 입력합니다. 예를 들어, 다시 재생의 이름을 `ReplayTest`로 지정합니다.
5. 소스의 경우 2단계: 아카이브 생성 섹션에서 생성한 아카이브를 선택합니다.
6. 다시 재생 기간의 경우 다음과 같이 하세요.
   a. 시작 시간에서 테스트 이벤트를 전송한 날짜와 해당 이벤트를 전송하기 전의 시간을 선택합니다. 예: `2021/08/11` 및 `08:00:00`.
   b. 종료 시간에서 현재 날짜 및 시간을 선택합니다. 예: `2021/08/11` 및 `09:15:00`.
7. 다시 재생 시작을 선택합니다.

6단계: 리소스 정리

이 자습서 용도로 생성한 리소스를 보관하고 싶지 않다면 지금 삭제할 수 있습니다. 더 이상 사용하지 않는 AWS 리소스를 삭제하면 AWS 계정에 불필요한 요금이 발생하는 것을 방지할 수 있습니다.

Lambda 함수를 삭제하려면

1. Lambda 콘솔의 함수 페이지를 엽니다.
2. 생성한 함수를 선택합니다.
3. 작업, 삭제를 선택합니다.
4. 삭제를 선택합니다.

EventBridge 아카이브를 삭제하려면

1. EventBridge 콘솔의 아카이브 페이지를 엽니다.
2. 생성한 아카이브를 선택합니다.
3. 삭제를 선택합니다.

4. 아카이브 이름을 입력하고 삭제를 선택합니다.

EventBridge 규칙을 삭제하려면

1. EventBridge 콘솔의 규칙 페이지를 엽니다.
2. 생성한 규칙을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

Amazon EventBridge 샘플 애플리케이션 생성

EventBridge를 사용하면 규칙을 사용하여 이벤트를 특정 Lambda 함수로 라우팅할 수 있습니다.

이 자습서에서는AWS CLI, Node.js 및 GitHub 리포지토리의 코드를 사용하여 다음을 생성합니다.

- 은행 ATM 거래에 대한 이벤트를 생성하는 AWS Lambda 함수.
- EventBridge 규칙의 대상으로 사용할 세 가지 Lambda 함수.
- 생성된 이벤트를 이벤트 패턴에 따라 올바른 다운스트림 함수로 라우팅하는 규칙.

이 예제에서는 AWS SAM 템플릿을 사용하여 EventBridge 규칙을 정의합니다. EventBridge에서 AWS SAM 템플릿을 사용하는 방법에 대한 자세한 내용은 ??? 섹션을 참조하십시오.

리포지토리의 atmProducer 하위 디렉토리에는 이벤트를 생성하는 ATM 서비스를 나타내는 `handler.js`가 포함되어 있습니다. 이 코드는 Node.js로 작성된 Lambda 핸들러이며, 다음 JavaScript 코드 줄을 사용하여 AWS SDK를 통해 EventBridge에 이벤트를 게시합니다.

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

이 디렉터리에는 Entrites 배열의 여러 테스트 트랜잭션을 나열하는 `events.js`도 포함되어 있습니다. JavaScript에서 단일 이벤트는 다음과 같이 정의됩니다.

```
{
  // 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
  })
```

```
}
```

이벤트의 세부 정보 섹션은 트랜잭션 속성을 지정합니다. 여기에는 ATM의 위치, 금액, 제휴 은행 및 거래 결과가 포함됩니다.

atmConsumer 하위 디렉토리의 `handler.js` 파일에는 다음과 같은 세 가지 함수가 포함되어 있습니다.

```
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))
}
```

각 함수는 트랜잭션 이벤트를 수신하며, 이 이벤트는 `console.log` 명령문을 통해 Amazon CloudWatch Logs에 기록됩니다. 소비자 함수는 생산자와 독립적으로 작동하며 이벤트의 출처를 인식하지 못합니다.

라우팅 로직은 애플리케이션의 AWS SAM 템플릿으로 배포되는 EventBridge 규칙에 포함되어 있습니다. 규칙은 들어오는 이벤트 스트림을 평가하고 일치하는 이벤트를 대상 Lambda 함수로 라우팅합니다.

규칙은 일치하는 이벤트와 동일한 구조의 JSON 객체인 이벤트 패턴을 사용합니다. 규칙 중 하나의 이벤트 패턴은 다음과 같습니다.

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

```
  }
}
```

단계:

- 필수 조건
- 1단계: 애플리케이션 생성
- 2단계: 애플리케이션 실행
- 3단계: 로그 확인 및 애플리케이션 작동 확인
- 4단계: 리소스 정리

필수 조건

이 자습서를 완료하려면 다음 리소스가 필요합니다.

- AWS 계정. 아직 없는 경우 AWS 계정을 생성합니다.
- AWS CLI가 설치되었습니다. AWS CLI를 설치하려면 AWS CLI 버전 2 설치, 업데이트 및 제거를 참조하세요.
- Node.js 12.x가 설치되었습니다. Node.js를 설치하려면 다운로드를 참조하십시오.

1단계: 애플리케이션 생성

예제 애플리케이션을 설정하려면 AWS CLI 및 Git을 사용하여 필요한 AWS 리소스를 생성해야 합니다.

애플리케이션을 생성하려면

1. AWS에 로그인합니다.
2. Git을 설치하고 로컬 시스템에 AWS Serverless Application Model CLI를 설치합니다.
3. 새 디렉터리를 만든 다음 터미널에서 해당 디렉터리로 이동합니다.
4. 명령줄에서 `git clone https://github.com/aws-samples/amazon-eventbridge-producer-consumer-example`을 입력합니다.
5. 명령줄 프롬프트에 다음 명령을 실행합니다.

```
cd ./amazon-eventbridge-producer-consumer-example
sam deploy --guided
```

6. 터미널에서 다음을 수행합니다.

   a. **Stack Name**의 경우 스택의 이름을 입력합니다. 예를 들어, 스택의 이름을 Test로 지정합니다.

   b. **AWS Region**의 경우 리전을 입력합니다. 예: us-west-2.

   c. **Confirm changes before deploy**에 Y를 입력합니다.

   d. **Allow SAM CLI IAM role creation**에 Y를 입력합니다.

   e. **Save arguments to configuration file**에 Y를 입력합니다.

   f. **SAM configuration file**에 samconfig.toml를 입력합니다.

   g. **SAM configuration environment**에 default를 입력합니다.

2단계: 애플리케이션 실행

리소스를 설정했으므로 이제 콘솔을 사용하여 함수를 테스트합니다.

애플리케이션을 실행하려면

1. AWS SAM 애플리케이션을 배포한 동일한 리전에서 Lambda 콘솔을 엽니다.
2. 접두사가 atm-demo인 Lambda 함수가 네 개 있습니다. atmProducerFn 함수를 선택한 다음 액션, 테스트를 선택합니다.
3. 이름에 Test를 입력합니다.
4. 테스트를 선택합니다.

3단계: 로그 확인 및 애플리케이션 작동 확인

이제 애플리케이션을 실행했으니 콘솔을 사용하여 CloudWatch Logs를 확인할 수 있습니다.

로그를 확인하려면

1. AWS SAM 애플리케이션을 실행한 동일한 리전에서 CloudWatch 콘솔을 엽니다.
2. 로그를 선택한 후 로그 그룹을 선택합니다.
3. atmConsumerCase1이 포함된 로그 그룹을 선택합니다. ATM에서 승인한 두 거래를 나타내는 두 개의 스트림이 표시됩니다. 출력을 보려는 로그 스트림을 선택합니다.
4. 로그 그룹 목록으로 돌아간 다음 atmConsumerCase2가 포함된 로그 그룹을 선택합니다. 뉴욕 위치 필터와 일치하는 두 거래를 나타내는 두 개의 스트림이 표시됩니다.

5. 로그 그룹 목록으로 돌아가서 atmConsumerCase3이 포함된 로그 그룹을 선택합니다. 스트림을 열어 거부된 거래를 확인합니다.

4단계: 리소스 정리

이 자습서 용도로 생성한 리소스를 보관하고 싶지 않다면 지금 삭제할 수 있습니다. 더 이상 사용하지 않는 AWS 리소스를 삭제하면 AWS 계정에 불필요한 요금이 발생하는 것을 방지할 수 있습니다.

EventBridge 규칙을 삭제하려면

1. EventBridge 콘솔의 규칙 페이지를 엽니다.
2. 생성한 규칙을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

Lambda 함수를 삭제하려면

1. Lambda 콘솔의 함수 페이지를 엽니다.
2. 생성한 함수를 선택합니다.
3. 작업, 삭제를 선택합니다.
4. 삭제를 선택합니다.

CloudWatch Logs 로그 그룹을 삭제하려면

1. CloudWatch 콘솔을 엽니다.
2. 로그, 로그 그룹을 선택합니다.
3. 이 자습서에서 만든 로그 그룹을 선택합니다.
4. 작업, 로그 그룹 삭제를 선택합니다.
5. 삭제를 선택합니다.

자습서: EventBridge 스키마 레지스트리를 사용하여 이벤트용 코드 바인딩 다운로드

이벤트 스키마에 대한 코드 바인딩을 생성하여 Golang, Java, Python 및 TypeScript 개발 속도를 높일 수 있습니다. 기존 AWS 서비스, 생성한 스키마, 이벤트 버스의 이벤트를 기반으로 생성한 스키마에 대한 코드 바인딩을 가져올 수 있습니다. 다음 방법 중 하나를 사용하여 스키마 코드 바인딩을 생성할 수 있습니다.

- EventBridge 콘솔
- EventBridge 스키마 레지스트리 API
- AWS 툴킷이 포함된 IDE

이 자습서에서는 AWS 서비스 이벤트에 대한 EventBridge 스키마에서 코드 바인딩을 생성하고 다운로드합니다.

EventBridge 스키마에서 코드 바인딩을 생성하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 스키마를 선택합니다.
3. AWS 이벤트 스키마 레지스트리 탭을 선택합니다.
4. 스키마 레지스트리를 탐색하거나 스키마를 검색하여 코드 바인딩을 원하는 AWS 서비스에 대한 스키마를 찾습니다.
5. 스키마 이름을 선택합니다.
6. 스키마 세부 정보 페이지의 버전 섹션에서 코드 바인딩 다운로드를 선택합니다.
7. 코드 바인딩 다운로드 페이지에서 다운로드할 코드 바인딩의 언어를 선택합니다.
8. 다운로드를 선택합니다.

   다운로드가 시작되는 데 몇 초가 걸릴 수 있습니다. 다운로드 파일은 선택한 언어에 대한 코드 바인딩의 .zip 파일입니다.
9. 다운로드한 파일의 압축을 풀고 프로젝트에 추가합니다.

   다운로드한 패키지에는 다양한 프레임워크에서 패키지의 종속성을 구성하는 방법을 설명하는 README 파일이 들어 있습니다.

자체 코드에서 이러한 코드 바인딩을 사용하면 이 EventBridge 이벤트를 사용하여 애플리케이션을 빠르게 빌드할 수 있습니다.

자습서: 입력 변환기를 사용하여 EventBridge가 이벤트 대상에 전달하는 내용을 사용자 지정

EventBridge의 입력 변환기를 사용하여 이벤트를 규칙 대상으로 보내기 전에 이벤트의 텍스트를 사용자 지정할 수 있습니다.

이렇게 하려면 이벤트에서 JSON 경로를 정의하고 그 출력을 다른 변수에 할당합니다. 그런 다음 입력 템플릿에서 이러한 변수를 사용할 수 있습니다. < 및 > 문자는 이스케이프할 수 없습니다. 자세한 정보는 아마존 EventBridge 인풋 트랜스포메이션 섹션을 참조하세요.

> **Note**
>
> 변수를 지정하여 이벤트에 존재하지 않는 JSON 경로를 일치시키는 경우 변수가 생성되지 않기 때문에 출력에 나타나지 않습니다.

이 자습서에서는 `detail-type: "customerCreated"`와 이벤트를 일치시키는 규칙을 만듭니다. 입력 변환기는 `type` 변수를 이벤트의 $.detail-type JSON 경로에 매핑합니다. 그런 다음 EventBridge는 변수를 입력 템플릿 "This event was <type>."에 넣습니다. 결과는 다음 Amazon SNS 메시지입니다.

```
"This event was of customerCreated type."
```

단계:

- 1단계: Amazon SNS 주제 생성
- 2단계: Amazon SNS 구독 생성
- 3단계: 규칙 생성
- 4단계: 테스트 이벤트 보내기
- 5단계: 성공 확인
- 6단계: 리소스 정리

1단계: Amazon SNS 주제 생성

EventBridge에서 이벤트를 수신할 주제를 생성합니다.

주제를 생성하려면

1. https://console.aws.amazon.com/sns/v3/home에서 Amazon SNS 콘솔을 엽니다.
2. 탐색 창에서 주제를 선택합니다.
3. 주제 생성을 선택합니다.
4. 유형에서 표준을 선택합니다.
5. 주제 이름으로 **eventbridge-IT-test**를 입력합니다.
6. 주제 생성을 선택합니다.

2단계: Amazon SNS 구독 생성

변환된 정보가 포함된 이메일을 받기 위한 구독을 생성합니다.

구독을 생성하려면

1. https://console.aws.amazon.com/sns/v3/home에서 Amazon SNS 콘솔을 엽니다.
2. 탐색 창에서 구독을 선택합니다.
3. 구독 생성을 선택합니다.
4. 주제 ARN에서 1단계에서 생성한 주제를 선택합니다. 이 자습서에서는 eventbridge-IT-test를 선택합니다.
5. 프로토콜에서 이메일을 선택합니다.
6. 엔드포인트에 이메일 주소를 입력합니다.
7. 구독 생성을 선택합니다.
8. AWS 알림을 통해 받은 이메일에서 구독 확인을 선택하여 구독을 확인합니다.

3단계: 규칙 생성

입력 변환기를 사용하여 대상으로 이동하는 인스턴스 상태 정보를 사용자 지정하는 규칙을 생성합니다.

규칙을 생성하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 규칙을 선택합니다.
3. 규칙 생성을 선택합니다.

4. 규칙에 대해 이름과 설명을 입력합니다. 예를 들어, 규칙의 이름을 ARTestRule로 지정합니다.
5. 이벤트 버스에서 이 규칙과 연결할 이벤트 버스를 선택합니다. 이 규칙이 자신의 계정에서 발생하는 이벤트와 일치하도록 하려면 기본을 선택합니다. 계정의 AWS 서비스가 이벤트를 출력하면 항상 계정의 기본 이벤트 버스로 이동합니다.
6. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다.
7. 다음을 선택합니다.
8. 이벤트 소스에서 기타를 선택합니다.
9. 이벤트 패턴에 다음을 입력합니다.

```
{
  "detail-type": [
    "customerCreated"
  ]
}
```

10. 다음을 선택합니다.
11. 대상 유형에서 AWS 서비스를 선택합니다.
12. 대상 선택의 경우 드롭다운 목록에서 SNS 주제를 선택합니다.
13. 주제에서 1단계에서 생성한 Amazon SNS 주제를 선택합니다. 이 자습서에서는 eventbridge-IT-test를 선택합니다.
14. 추가 설정에서 다음을 수행합니다.

    a. 대상 입력 구성의 경우 드롭다운 목록에서 입력 변환기를 선택합니다.

    b. 입력 변환기 구성을 선택합니다.

    c. 샘플 이벤트의 경우 다음을 입력합니다.

```
{
  "detail-type": "customerCreated"
}
```

    d. 대상 입력 변환기의 경우 다음을 수행합니다.

        i. 입력 경로에 다음을 입력합니다.

```
{"detail-type":"$.detail-type"}
```

        ii. 입력 템플릿에는 다음 사항을 입력합니다.

```
"This event was of <detail-type> type."
```

e. 확인을 선택합니다.

15. 다음을 선택합니다.
16. 다음을 선택합니다.
17. 규칙의 세부 정보를 검토하고 규칙 생성을 선택합니다.

4단계: 테스트 이벤트 보내기

SNS 주제와 규칙을 설정했으니 이제 규칙이 제대로 작동하는지 확인하기 위해 테스트 이벤트를 전송하겠습니다.

테스트 이벤트를 보내려면(콘솔)

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 이벤트 버스를 선택합니다.
3. 기본 이벤트 버스 타일에서 액션, 이벤트 전송을 선택합니다.
4. 이벤트 소스를 입력합니다. 예: `TestEvent`.
5. 세부 정보 유형에는 `customerCreated`를 입력합니다.
6. 이벤트 세부 정보에는 `{}`를 입력합니다.
7. 전송을 선택합니다.

5단계: 성공 확인

예상 출력과 일치하는 AWS 알림 이메일을 받으면 자습서를 성공적으로 완료한 것입니다.

6단계: 리소스 정리

이 자습서 용도로 생성한 리소스를 보관하고 싶지 않다면 지금 삭제할 수 있습니다. 더 이상 사용하지 않는 AWS 리소스를 삭제하면 AWS 계정에 불필요한 요금이 발생하는 것을 방지할 수 있습니다.

SNS 주제를 삭제하려면

1. SNS 콘솔의 주제 페이지를 엽니다.
2. 생성한 주제를 선택합니다.
3. 삭제를 선택합니다.

4. **`delete me`**을 입력합니다.
5. 삭제를 선택합니다.

SNS 구독을 삭제하려면

1. SNS 콘솔의 구독 페이지를 엽니다.
2. 생성한 구독을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

EventBridge 규칙을 삭제하려면

1. EventBridge 콘솔의 규칙 페이지를 엽니다.
2. 생성한 규칙을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

다른 AWS 서비스와의 통합을 위한 Amazon EventBridge 자습서

Amazon EventBridge는 다른 AWS 서비스와 협력하여 이벤트를 처리하거나 AWS 리소스를 규칙의 대상으로 간접적으로 간접 호출합니다. 다음 자습서에서는 EventBridge를 다른 AWS 서비스와 통합하는 방법을 보여줍니다.

자습서:

- 자습서: EventBridge를 사용하여 Auto Scaling 그룹의 상태 로깅
- 자습서: 사용하여 AWS API 호출 기록 EventBridge
- 자습서: 다음을 사용하여 Amazon EC2 인스턴스의 상태를 기록합니다. EventBridge
- 자습서: CEventBridge를 사용하여 Amazon S3 객체 수준 작업 로깅
- 자습서: EventBridge 및 스키마를 사용하여 Amazon Kinesis 스트림으로 이벤트 전송 aws.events
- 자습서: EventBridge를 사용하여 Amazon EBS 스냅샷 자동화 예약
- 자습서: Amazon S3 객체가 생성되면 알림 보내기
- 자습서: EventBridge를 사용하여 AWS Lambda 함수 예약

자습서: EventBridge를 사용하여 Auto Scaling 그룹의 상태 로깅

Auto Scaling 그룹이 Amazon EC2 인스턴스를 시작하거나 종료할 때마다 이벤트를 로깅하고 이벤트가 성공했는지 여부를 나타내는 AWS Lambda 함수를 실행할 수 있습니다.

Amazon EC2 Auto Scaling 이벤트를 사용하는 추가 시나리오에 대한 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서에서 EventBridge를 사용하여 Auto Scaling 이벤트 처리를 참조하세요.

이 자습서에서는 Lambda 함수를 생성하고 Amazon EC2 Auto Scaling 그룹이 인스턴스를 시작하거나 종료할 때 해당 함수를 호출하는 규칙을 EventBridge 콘솔에 생성합니다.

단계:

- 필수 조건
- 1단계: Lambda 함수 생성
- 2단계: 규칙 생성
- 3단계: 규칙 테스트
- 4단계: 성공 확인
- 5단계: 리소스 정리

필수 조건

이 자습서를 완료하려면 다음 리소스가 필요합니다.

- Auto Scaling 그룹 생성에 대한 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 시작 구성을 사용하여 Auto Scaling 그룹 생성을 참조하세요.

1단계: Lambda 함수 생성

Lambda 함수를 생성하여 Auto Scaling 그룹에서 이벤트의 확장 및 축소를 로그합니다.

Lambda 함수를 만들려면

1. https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
2. 함수 생성을 선택합니다.
3. 새로 작성을 선택합니다.
4. Lambda 함수 이름을 입력합니다. 예를 들어 함수 이름을 `LogAutoScalingEvent`로 지정합니다.

5. 나머지 옵션은 기본값으로 두고 함수 생성을 선택합니다.
6. 함수 페이지의 코드 탭에서 index.js를 두 번 클릭합니다.
7. 기존 코드를 다음 코드로 바꿉니다.

```
'use strict';

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

8. 배포를 선택합니다.

2단계: 규칙 생성

1단계에서 생성한 Lambda 함수를 실행하는 규칙을 생성합니다. 규칙은 Auto Scaling 그룹이 인스턴스를 시작하거나 중지할 때 실행됩니다.

규칙을 생성하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 [Rules]를 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙에 대해 이름과 설명을 입력하세요. 예를 들어, 규칙의 이름을 `TestRule`로 지정합니다.
5. 이벤트 버스에서 이 규칙과 연결할 이벤트 버스를 선택합니다. 이 규칙이 자신의 계정에서 발생하는 이벤트와 일치하도록 하려면 기본을 선택합니다. 계정의 AWS 서비스가 이벤트를 출력하면 항상 계정의 기본 이벤트 버스로 이동합니다.
6. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다.
7. 다음을 선택합니다.
8. 이벤트 소스에서 AWS 서비스를 선택합니다.
9. 이벤트 패턴에서 다음을 수행합니다.
    a. 이벤트 소스의 경우 드롭다운 목록에서 Auto Scaling을 선택합니다.
    b. 이벤트 유형의 경우 드롭다운 목록에서 인스턴스 시작 및 종료를 선택합니다.
    c. 모든 인스턴스 이벤트와 모든 그룹 이름을 선택합니다.

10. 다음을 선택합니다.
11. 대상 유형에서 AWS 서비스를 선택합니다.
12. 대상 선택의 경우 드롭다운 목록에서 Lambda 함수를 선택합니다.
13. 함수의 경우 1단계: Lambda 함수 생성 섹션에서 생성한 Lambda 함수를 선택합니다. 이 예시에서는 `LogAutoScalingEvent`를 선택합니다.
14. 다음을 선택합니다.
15. 다음을 선택합니다.
16. 규칙의 세부 정보를 검토하고 규칙 생성을 선택합니다.

3단계: 규칙 테스트

인스턴스가 시작되도록 Auto Scaling 그룹을 수동으로 크기 조정함으로써 규칙을 테스트할 수 있습니다. 확장 이벤트가 발생할 때까지 몇 분 정도 기다린 후 Lambda 함수가 간접 호출되었는지 확인합니다.

Auto Scaling 그룹을 사용하여 규칙을 테스트하려면

1. Auto Scaling 그룹의 크기를 늘리려면 다음을 수행합니다.
   a. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
   b. 탐색 창에서 Auto Scaling과 Auto Scaling 그룹을 선택합니다.
   c. Auto Scaling 그룹의 확인란을 선택합니다.
   d. 세부 정보 탭에서 편집을 선택합니다. 원하는 용량에서 한 개씩 원하는 용량을 늘립니다. 예를 들어, 현재 값이 2인 경우 3을 입력합니다. 원하는 용량은 그룹의 최대 크기보다 작거나 같아야 합니다. 원하는 용량에 대한 새 값이 최대 용량보다 크면 최대 용량을 업데이트해야 합니다. 작업을 마쳤으면 저장을 선택합니다.
2. Lambda 함수에서 출력을 보려면 다음을 수행합니다.
   a. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
   b. 탐색 창에서 로그를 선택합니다.
   c. Lambda 함수에 대한 로그 그룹 이름을 선택합니다(`/aws/lambda/`*`function-name`*).
   d. 로그 스트림 이름을 선택하여 시작한 인스턴스에서 함수를 통해 제공된 데이터를 확인합니다.
3. (선택 사항) 완료되면 Auto Scaling 그룹이 이전 크기로 돌아가도록 한 개씩 원하는 용량을 줄일 수 있습니다.

4단계: 성공 확인

CloudWatch 로그에 Lambda 이벤트가 표시되면 이 자습서를 성공적으로 완료한 것입니다. 이벤트가 CloudWatch 로그에 없는 경우, 규칙이 성공적으로 생성되었는지 확인하여 문제 해결을 시작하고, 규칙이 올바른 것으로 보이면 Lambda 함수의 코드가 올바른지 확인하십시오.

5단계: 리소스 정리

이 자습서 용도로 생성한 리소스를 보관하고 싶지 않다면 지금 삭제할 수 있습니다. 더 이상 사용하지 않는 AWS 리소스를 삭제하면 AWS 계정에 불필요한 요금이 발생하는 것을 방지할 수 있습니다.

EventBridge 규칙을 삭제하려면

1. EventBridge 콘솔의 규칙 페이지를 엽니다.
2. 생성한 규칙을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

Lambda 함수를 삭제하려면

1. Lambda 콘솔의 함수 페이지를 엽니다.
2. 생성한 함수를 선택합니다.
3. 작업, 삭제를 선택합니다.
4. 삭제를 선택합니다.

자습서: 사용하여 AWS API 호출 기록 EventBridge

Amazon EventBridge 규칙을 사용하여 에서 기록한 AWS 서비스에서 이루어진 API 호출에 응답할 수 AWS CloudTrail있습니다.

이 자습서에서는 콘솔에서 AWS CloudTrail트레일, Lambda 함수 및 규칙을 생성합니다. EventBridge 이 규칙은 Amazon EC2 인스턴스가 중지될 때 Lambda 함수를 간접적으로 간접 호출합니다.

단계:

- 1단계: 트레일 생성 AWS CloudTrail
- 2단계: AWS Lambda 함수 생성
- 3단계: 규칙 생성
- 4단계: 규칙 테스트
- 5단계: 성공 확인
- 6단계: 리소스 정리

1단계: 트레일 생성 AWS CloudTrail

이미 추적을 설정한 경우 2단계로 건너뜁니다.

추적을 생성하려면

1. https://console.aws.amazon.com/cloudtrail/ 에서 CloudTrail 콘솔을 엽니다.
2. 추적, 추적 생성을 선택합니다.
3. 추적 이름에 추적 이름을 입력합니다.
4. 스토리지 위치의 S3 버킷 새로 만들기에서.
5. AWS KMS 별칭에 KMS 키의 별칭을 입력합니다.
6. 다음을 선택합니다.
7. 다음을 선택합니다.
8. 추적 생성을 선택합니다.

2단계: AWS Lambda 함수 생성

Lambda 함수를 생성하여 API 직접 호출 이벤트를 기록합니다.

Lambda 함수를 생성하는 방법

1. https://console.aws.amazon.com/lambda/ 에서 AWS Lambda 콘솔을 엽니다.
2. 함수 생성을 선택합니다.
3. 새로 작성을 선택합니다.
4. Lambda 함수의 이름과 설명을 입력합니다. 예를 들어 함수 이름을 `LogEC2StopInstance`로 지정합니다.
5. 나머지 옵션은 기본값으로 두고 함수 생성을 선택합니다.
6. 함수 페이지의 코드 탭에서 index.js를 두 번 클릭합니다.
7. 기존 코드를 다음 코드로 바꿉니다.

```
'use strict';

exports.handler = (event, context, callback) => {
    console.log('LogEC2StopInstance');
    console.log('Received event:', JSON.stringify(event, null, 2));
    callback(null, 'Finished');
};
```

8. 배포를 선택합니다.

3단계: 규칙 생성

Amazon EC2 인스턴스를 중지할 때마다 2단계에서 생성한 Lambda 함수를 실행하는 규칙을 생성합니다.

규칙을 생성하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 규칙을 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙에 대해 이름과 설명을 입력하십시오. 예를 들어, 규칙의 이름을 `TestRule`로 지정합니다.
5. 이벤트 버스에서 이 규칙과 연결할 이벤트 버스를 선택합니다. 이 규칙이 자신의 계정에서 발생하는 이벤트와 일치하도록 하려면 기본을 선택합니다. 계정의 AWS 서비스가 이벤트를 출력하면 항상 계정의 기본 이벤트 버스로 이동합니다.
6. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다.
7. 다음을 선택합니다.

8. 이벤트 소스에서 AWS 서비스를 선택합니다.
9. 이벤트 패턴에서 다음을 수행합니다.
    a. 이벤트 소스의 경우 드롭다운 목록에서 EC2를 선택합니다.
    b. 이벤트 유형의 경우 드롭다운 CloudTrail 목록에서 AWS API 호출을 선택합니다.
    c. 특정 작업을 선택하고 `StopInstances`를 입력합니다.
10. 다음을 선택합니다.
11. 대상 유형에서 AWS 서비스를 선택합니다.
12. 대상 선택의 경우 드롭다운 목록에서 Lambda 함수를 선택합니다.
13. 함수의 경우 1단계: Lambda 함수 생성 섹션에서 생성한 Lambda 함수를 선택합니다. 이 예시에서는 `LogEC2StopInstance`를 선택합니다.
14. 다음을 선택합니다.
15. 다음을 선택합니다.
16. 규칙의 세부 정보를 검토하고 규칙 생성을 선택합니다.

4단계: 규칙 테스트

Amazon EC2 콘솔을 사용하여 Amazon EC2 인스턴스를 중지함으로써 규칙을 테스트할 수 있습니다. 인스턴스가 중지될 때까지 몇 분 정도 기다린 다음 CloudWatch 콘솔에서 AWS Lambda 측정치를 확인하여 함수가 실행되었는지 확인합니다.

인스턴스를 중지시켜 규칙을 테스트하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 인스턴스 시작. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 시작을 참조하십시오.
3. 인스턴스를 중지합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 중지 및 시작을 참조하십시오.
4. Lambda 함수에서 출력을 보려면 다음을 수행합니다.
    a. https://console.aws.amazon.com/cloudwatch/ 에서 CloudWatch 콘솔을 엽니다.
    b. 탐색 창에서 로그를 선택합니다.
    c. Lambda 함수에 대한 로그 그룹 명칭을 선택합니다(`/aws/lambda/`*`function-name`*).
    d. 로그 스트림 이름을 선택하여 중지한 인스턴스에서 함수를 통해 제공된 데이터를 확인합니다.

5. (선택 사항) 작업이 완료되면 중지된 인스턴스를 종료합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 종료를 참조하십시오.

5단계: 성공 확인

로그에 Lambda 이벤트가 표시되면 CloudWatch 이 자습서를 성공적으로 완료한 것입니다. 이벤트가 CloudWatch 로그에 없는 경우, 규칙이 성공적으로 생성되었는지 확인하여 문제 해결을 시작하고, 규칙이 올바른 것으로 보이면 Lambda 함수의 코드가 올바른지 확인하십시오.

6단계: 리소스 정리

이 자습서 용도로 생성한 리소스를 보관하고 싶지 않다면 지금 삭제할 수 있습니다. 더 이상 사용하지 않는 AWS 리소스를 삭제하면 계정에 불필요한 요금이 부과되는 것을 방지할 수 있습니다. AWS

EventBridge 규칙을 삭제하려면

1. EventBridge 콘솔의 규칙 페이지를 엽니다.
2. 생성한 규칙을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

Lambda 함수를 삭제하려면

1. Lambda 콘솔의 함수 페이지를 엽니다.
2. 생성한 함수를 선택합니다.
3. 작업, 삭제를 선택합니다.
4. 삭제를 선택합니다.

CloudTrail 트레일을 삭제하려면

1. CloudTrail 콘솔의 트레일 페이지를 엽니다.
2. 생성한 추적을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

자습서: 다음을 사용하여 Amazon EC2 인스턴스의 상태를 기록합니다. EventBridge

Amazon EC2 인스턴스의 상태 변경을 로깅하는 AWS Lambda 함수를 생성할 수 있습니다. 상태 전환이 있거나 관심이 있는 하나 이상의 상태로 전환될 때마다 Lambda 함수를 실행하는 규칙을 생성할 수 있습니다. 이 자습서에서는 새 인스턴스의 시작을 기록합니다.

단계:

1단계: AWS Lambda 함수 생성

Lambda 함수를 생성하여 상태 변경 이벤트를 로깅합니다. 2단계에서 규칙을 생성할 때 이 함수를 지정합니다.

Lambda 함수를 생성하는 방법

1. https://console.aws.amazon.com/lambda/ 에서 AWS Lambda 콘솔을 엽니다.
2. 함수 생성을 선택합니다.
3. 새로 작성을 선택합니다.
4. Lambda 함수의 이름과 설명을 입력합니다. 예를 들어 함수 이름을 `LogEC2InstanceStateChange`로 지정합니다.
5. 나머지 옵션은 기본값으로 두고 함수 생성을 선택합니다.
6. 함수 페이지의 코드 탭에서 index.js를 두 번 클릭합니다.
7. 기존 코드를 다음 코드로 바꿉니다.

```
'use strict';

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

```
};
```

8. 배포를 선택합니다.

2단계: 규칙 생성

1단계에서 생성한 Lambda 함수를 실행하는 규칙을 생성합니다. Amazon EC2 인스턴스를 시작할 때 규칙이 실행됩니다.

EventBridge 규칙을 만들려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 규칙을 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙에 대해 이름과 설명을 입력하십시오. 예를 들어, 규칙의 이름을 `TestRule`로 지정합니다.
5. 이벤트 버스에서 이 규칙과 연결할 이벤트 버스를 선택합니다. 이 규칙이 자신의 계정에서 발생하는 이벤트와 일치하도록 하려면 기본을 선택합니다. 계정의 AWS 서비스가 이벤트를 출력하면 항상 계정의 기본 이벤트 버스로 이동합니다.
6. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다.
7. 다음을 선택합니다.
8. 이벤트 소스에서 AWS 서비스를 선택합니다.
9. 이벤트 패턴에서 다음을 수행합니다.
   a. 이벤트 소스의 경우 드롭다운 목록에서 EC2를 선택합니다.
   b. 이벤트 유형의 경우 드롭다운 목록에서 EC2 인스턴스 상태 변경 알림을 선택합니다.
   c. 특정 상태를 선택하고 드롭다운 목록에서 실행을 선택합니다.
   d. 모든 인스턴스를 선택합니다.
10. 다음을 선택합니다.
11. 대상 유형에서 AWS 서비스를 선택합니다.
12. 대상 선택의 경우 드롭다운 목록에서 Lambda 함수를 선택합니다.
13. 함수의 경우 1단계: Lambda 함수 생성 섹션에서 생성한 Lambda 함수를 선택합니다. 이 예시에서는 `LogEC2InstanceStateChange`를 선택합니다.
14. 다음을 선택합니다.
15. 다음을 선택합니다.

16. 규칙의 세부 정보를 검토하고 규칙 생성을 선택합니다.

3단계: 규칙 테스트

Amazon EC2 콘솔을 사용하여 Amazon EC2 인스턴스를 중지함으로써 규칙을 테스트할 수 있습니다. 인스턴스가 중지될 때까지 몇 분 정도 기다린 다음 CloudWatch 콘솔에서 AWS Lambda 지표를 확인하여 함수가 실행되었는지 확인하십시오.

인스턴스를 중지시켜 규칙을 테스트하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 인스턴스 시작. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 시작을 참조하십시오.
3. 인스턴스를 중지합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 중지 및 시작을 참조하십시오.
4. Lambda 함수에서 출력을 보려면 다음을 수행합니다.
   a. https://console.aws.amazon.com/cloudwatch/ 에서 CloudWatch 콘솔을 엽니다.
   b. 탐색 창에서 로그를 선택합니다.
   c. Lambda 함수에 대한 로그 그룹 명칭을 선택합니다(`/aws/lambda/function-name`).
   d. 로그 스트림 이름을 선택하여 중지한 인스턴스에서 함수를 통해 제공된 데이터를 확인합니다.
5. (선택 사항) 작업이 완료되면 중지된 인스턴스를 종료합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 종료를 참조하십시오.

4단계: 성공 확인

로그에 Lambda 이벤트가 표시되면 CloudWatch 이 자습서를 성공적으로 완료한 것입니다. 이벤트가 CloudWatch 로그에 없는 경우, 규칙이 성공적으로 생성되었는지 확인하여 문제 해결을 시작하고, 규칙이 올바른 것으로 보이면 Lambda 함수의 코드가 올바른지 확인하십시오.

5단계: 리소스 정리

이 자습서 용도로 생성한 리소스를 보관하고 싶지 않다면 지금 삭제할 수 있습니다. 더 이상 사용하지 않는 AWS 리소스를 삭제하면 계정에 불필요한 요금이 부과되는 것을 방지할 수 있습니다. AWS

EventBridge 규칙을 삭제하려면

1. EventBridge 콘솔의 규칙 페이지를 엽니다.
2. 생성한 규칙을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

Lambda 함수를 삭제하려면

1. Lambda 콘솔의 함수 페이지를 엽니다.
2. 생성한 함수를 선택합니다.
3. 작업, 삭제를 선택합니다.
4. 삭제를 선택합니다.

자습서: CEventBridge를 사용하여 Amazon S3 객체 수준 작업 로깅

Amazon S3 버킷에서 객체 수준 API 작업을 로깅할 수 있습니다. Amazon EventBridge를 이러한 이벤트와 일치시키려면 먼저 AWS CloudTrail을 사용하여 이러한 이벤트를 수신하기 위한 추적을 설정하고 구성해야 합니다.

이 자습서에서는 CloudTrail 추적을 생성하고 AWS Lambda 함수를 생성한 다음 EventBridge 콘솔에서 S3 데이터 이벤트에 대한 응답으로 해당 함수를 간접 호출하는 규칙을 생성합니다.

단계:

- 1단계: AWS CloudTrail 추적 구성
- 2단계: AWS Lambda 함수 생성
- 3단계: 규칙 생성
- 4단계: 규칙 테스트
- 5단계: 성공 확인
- 6단계: 리소스 정리

1단계: AWS CloudTrail 추적 구성

AWS CloudTrail 및 EventBridge에 S3 버킷의 데이터 이벤트를 로깅하려면 먼저 추적을 생성합니다. 추적은 계정에서 API 직접 호출과 관련 이벤트를 캡처하고 지정된 S3 버킷에 로그 파일을 전달합니다. 기존 추적을 업데이트하거나 생성할 수 있습니다.

자세한 내용을 알아보려면 AWS CloudTrail 사용 설명서의 데이터 이벤트를 참조하세요.

추적을 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudTrail 콘솔을 엽니다.
2. 추적, 추적 생성을 선택합니다.
3. 추적 이름에 추적 이름을 입력합니다.
4. 스토리지 위치의 S3 버킷 새로 만들기에서.
5. AWS KMS 별칭에 KMS 키의 별칭을 입력합니다.
6. 다음을 선택합니다.
7. 이벤트 유형에서는 데이터 이벤트를 선택합니다.

8. 데이터 이벤트에서 다음 중 하나를 수행합니다.

   - 버킷의 모든 Amazon S3 객체에 대한 데이터 이벤트를 로그하려면 S3 버킷과 빈 접두사를 지정합니다. 이벤트가 해당 버킷의 개체에서 발생하면 추적이 해당 이벤트를 처리하고 기록합니다.
   - 버킷의 특정 Amazon S3 객체에 대한 데이터 이벤트를 로깅하려면 S3 버킷과 객체 접두사를 지정합니다. 이벤트가 해당 버킷의 개체에서 발생하고 개체가 지정된 접두사로 시작하면 추적이 이벤트를 처리하고 기록합니다.

9. 각 리소스에 대해 읽기 이벤트를 로깅할지, 쓰기 이벤트를 로깅할지 또는 둘 다 로깅할지를 선택합니다.
10. 다음을 선택합니다.
11. 추적 생성을 선택합니다.

2단계: AWS Lambda 함수 생성

Lambda 함수를 생성하여 S3 버킷에 대한 데이터 이벤트를 로그합니다.

Lambda 함수를 만들려면

1. https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
2. 함수 생성을 선택합니다.
3. 새로 작성을 선택합니다.
4. Lambda 함수의 이름과 설명을 입력합니다. 예를 들어 함수 이름을 `LogS3DataEvents`로 지정합니다.
5. 나머지 옵션은 기본값으로 두고 함수 생성을 선택합니다.
6. 함수 페이지의 코드 탭에서 index.js를 두 번 클릭합니다.
7. 기존 코드를 다음 코드로 바꿉니다.

```
'use strict';

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

8. 배포를 선택합니다.

3단계: 규칙 생성

2단계에서 생성한 Lambda 함수를 실행하는 규칙을 생성합니다. 이 규칙은 Amazon S3 데이터 이벤트에 대한 응답으로 실행됩니다.

규칙을 생성하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 [Rules]를 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙에 대해 이름과 설명을 입력하세요. 예를 들어, 규칙의 이름을 `TestRule`로 지정합니다.
5. 이벤트 버스에서 이 규칙과 연결할 이벤트 버스를 선택합니다. 이 규칙이 자신의 계정에서 발생하는 이벤트와 일치하도록 하려면 기본을 선택합니다. 계정의 AWS 서비스가 이벤트를 출력하면 항상 계정의 기본 이벤트 버스로 이동합니다.
6. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다.
7. 다음을 선택합니다.
8. 이벤트 소스에서 AWS 서비스를 선택합니다.
9. 이벤트 패턴에서 다음을 수행합니다.
    a. 이벤트 소스의 경우 드롭다운 목록에서 Simple Storage Service(S3)를 선택합니다.
    b. 이벤트 유형의 경우 드롭다운 목록에서 CloudTrail을 통한 객체 레벨 API 직접 호출을 선택합니다.
    c. 특정 작업을 선택한 후 PutObject를 선택합니다.
    d. 기본적으로 규칙은 리전의 모든 버킷에 대한 데이터 이벤트와 일치합니다. 특정 버킷에 대한 데이터 이벤트와 일치시키려면 이름 기준 특정 버킷을 선택하고 버킷을 하나 이상 입력합니다.
10. 다음을 선택합니다.
11. 대상 유형에서 AWS서비스를 선택합니다.
12. 대상 선택의 경우 드롭다운 목록에서 Lambda 함수를 선택합니다.
13. 함수에서, 1단계에서 생성한 `LogS3DataEvents` Lambda 함수를 선택합니다.
14. 다음을 선택합니다.
15. 다음을 선택합니다.
16. 규칙의 세부 정보를 검토하고 규칙 생성을 선택합니다.

4단계: 규칙 테스트

규칙을 테스트하려면 S3 버킷에 개체를 배치합니다. Lambda 함수가 호출되었는지 확인할 수 있습니다.

Lambda 함수에 대한 로그를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택합니다.
3. Lambda 함수에 대한 로그 그룹 이름을 선택합니다(`/aws/lambda/function-name`).
4. 로그 스트림 이름을 선택하여 시작한 인스턴스에서 함수를 통해 제공된 데이터를 확인합니다.

추적에 지정한 S3 버킷의 CloudTrail 로그를 확인할 수도 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서에서 CloudTrail 로그 파일 가져오기 및 보기를 참조하세요.

5단계: 성공 확인

CloudWatch 로그에 Lambda 이벤트가 표시되면 이 자습서를 성공적으로 완료한 것입니다. 이벤트가 CloudWatch 로그에 없는 경우, 규칙이 성공적으로 생성되었는지 확인하여 문제 해결을 시작하고, 규칙이 올바른 것으로 보이면 Lambda 함수의 코드가 올바른지 확인하십시오.

6단계: 리소스 정리

이 자습서 용도로 생성한 리소스를 보관하고 싶지 않다면 지금 삭제할 수 있습니다. 더 이상 사용하지 않는 AWS 리소스를 삭제하면 AWS 계정에 불필요한 요금이 발생하는 것을 방지할 수 있습니다.

EventBridge 규칙을 삭제하려면

1. EventBridge 콘솔의 규칙 페이지를 엽니다.
2. 생성한 규칙을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

Lambda 함수를 삭제하려면

1. Lambda 콘솔의 함수 페이지를 엽니다.
2. 생성한 함수를 선택합니다.

3. 작업, 삭제를 선택합니다.
4. 삭제를 선택합니다.

CloudTrail 추적을 삭제하려면

1. CloudTrail 콘솔의 추적 페이지를 엽니다.
2. 생성한 추적을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

자습서: EventBridge 및 스키마를 사용하여 Amazon Kinesis 스트림으로 이벤트 전송 `aws.events`

Amazon Kinesis EventBridge 스트림으로 AWS API 호출 이벤트를 보내고, Kinesis Data Streams 애플리케이션을 생성하고, 대량의 데이터를 처리할 수 있습니다. 이 자습서에서는 Kinesis 스트림을 생성한 다음 Amazon EC2 인스턴스가 중지될 때 해당 스트림으로 이벤트를 전송하는 규칙을 EventBridge 콘솔에서 생성합니다.

단계:

필수 조건

이 자습서에서는 다음을 사용합니다.

- 를 AWS CLI 사용하여 Kinesis 스트림을 사용할 수 있습니다.

  를 AWS CLI설치하려면 AWS CLI 버전 2 설치, 업데이트 및 제거를 참조하십시오.

> **Note**
>
> 이 자습서에서는 AWS 이벤트와 기본 제공 `aws.events` 스키마 레지스트리를 사용합니다. 사용자 지정 이벤트를 사용자 지정 스키마 레지스트리에 수동으로 추가하거나 스키마 검색을 사용하여 사용자 지정 이벤트의 스키마를 기반으로 EventBridge 규칙을 만들 수도 있습니다. 스키마에 대한 자세한 내용은 ??? 단원을 참조하십시오. 다른 이벤트 패턴 옵션을 사용하여 규칙을 생성하는 방법에 대한 자세한 내용은 ??? 단원을 참조하세요.

1단계: Amazon Kinesis 스트림 생성

스트림을 생성하려면 명령 프롬프트에서 `create-stream` AWS CLI 명령을 사용합니다.

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

스트림 상태가 ACTIVE이면 스트림이 준비되었다는 뜻입니다. 스트림 상태를 확인하려면 describe-stream 명령을 사용합니다.

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

2단계: 규칙 생성

Amazon EC2 인스턴스를 중지할 때 스트림으로 이벤트를 전송하는 규칙을 생성합니다.

규칙을 생성하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 규칙을 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙에 대해 이름과 설명을 입력하십시오. 예를 들어, 규칙의 이름을 TestRule로 지정합니다.
5. 이벤트 버스의 경우 기본값을 선택합니다.
6. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다.
7. 다음을 선택합니다.
8. 이벤트 소스에서AWS 이벤트 또는 EventBridge 파트너 이벤트를 선택합니다.
9. 생성 방법에서 스키마 사용을 선택합니다.
10. 이벤트 패턴에서 다음을 수행합니다.

    a. 스키마 유형의 경우 스키마 레지스트리에서 스키마 선택을 선택합니다.

    b. 스키마 레지스트리의 경우 드롭다운 목록에서 aws.events를 선택합니다.

    c. 스키마의 경우 드롭다운 목록에서 aws.ec2 @EC2 InstanceStateChangeNotification 를 선택합니다.

       EventBridge 모델 아래에 이벤트 스키마를 표시합니다.

       EventBridge 이벤트 패턴이 아닌 이벤트에 필요한 속성 옆에 빨간색 별표를 표시합니다.

    d. 모델에서 다음과 같은 이벤트 필터 속성을 설정합니다.

       i. 상태 속성 옆의 + 편집을 선택합니다.

          관계를 비워 둡니다. 값에 running을 입력합니다. Set를 선택합니다.

ii. 소스 속성 옆의 + 편집을 선택합니다.

관계를 비워 둡니다. 값에 aws.ec2을 입력합니다. Set를 선택합니다.

iii. detail-type 속성 옆의 + 편집을 선택합니다.

관계를 비워 둡니다. 값에 EC2 Instance State-change Notification을 입력합니다. Set를 선택합니다.

e. 구성한 이벤트 패턴을 보려면 JSON으로 이벤트 패턴 생성을 선택합니다.

EventBridge 이벤트 패턴을 JSON으로 표시합니다.

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

11. 다음을 선택합니다.
12. 대상 유형에서 AWS 서비스를 선택합니다.
13. 대상 선택의 경우 드롭다운 목록에서 Kinesis 스트림을 선택합니다.
14. 스트림의 경우 1단계: Amazon Kinesis 스트림 생성 섹션에서 생성한 Kinesis 스트림을 선택합니다. 이 예시에서는 test를 선택합니다.
15. 실행 역할에서 이 특정 리소스에 대해 새 역할 생성을 선택합니다.
16. 다음을 선택합니다.
17. 다음을 선택합니다.
18. 규칙의 세부 정보를 검토하고 규칙 생성을 선택합니다.

3단계: 규칙 테스트

규칙을 테스트하려면 Amazon EC2 인스턴스를 중지합니다. 인스턴스가 중지될 때까지 몇 분 정도 기다린 다음 CloudWatch 측정치를 확인하여 함수가 실행되었는지 확인하십시오.

인스턴스를 중지시켜 규칙을 테스트하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.

2. 인스턴스 시작. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 시작을 참조하십시오.
3. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
4. 탐색 창에서 규칙을 선택합니다.

   생성한 규칙의 이름을 선택한 후 규칙에 대한 지표를 선택합니다.
5. (선택 사항) 작업이 완료되면 인스턴스를 종료합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 종료를 참조하십시오.

4단계: 이벤트가 전송되었는지 확인

를 사용하여 스트림에서 레코드를 가져와 이벤트가 전송되었는지 확인할 수 있습니다. AWS CLI

레코드를 가져오려면

1. Kinesis 스트림에서 읽기를 시작하려면 명령 프롬프트에서 `get-shard-iterator` 명령을 사용합니다.

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

   출력의 예제는 다음과 같습니다.

   ```
   {
       "ShardIterator": "AAAAAAAAAAHSywljv0zEgPX4NyKdZ5wryMzP9yALs8NeKbUjp1IxtZs1Sp
   +KEd9I6AJ9ZG4lNR1EMi+9Md/nHvtLyxpfhEzYvkTZ4D9DQVz/mBYWRO6OTZRKnW9gd
   +efGN2aHFdkH1rJl4BL9Wyrk+ghYG22D2T1Da2EyNSH1+LAbK33gQweTJADBdyMwlo5r6PqcP2dzhg="
   }
   ```
2. 레코드를 얻으려면 다음 `get-records` 명령을 사용합니다. 이전 단계의 출력에서 샤드 반복기를 사용합니다.

   ```
   aws kinesis get-records --shard-
   iterator AAAAAAAAAAHSywljv0zEgPX4NyKdZ5wryMzP9yALs8NeKbUjp1IxtZs1Sp
   +KEd9I6AJ9ZG4lNR1EMi+9Md/nHvtLyxpfhEzYvkTZ4D9DQVz/mBYWRO6OTZRKnW9gd
   +efGN2aHFdkH1rJl4BL9Wyrk+ghYG22D2T1Da2EyNSH1+LAbK33gQweTJADBdyMwlo5r6PqcP2dzhg=
   ```

   이 명령이 성공하면 지정된 샤드에서 스트림에서 나온 레코드를 요청합니다. 0개 이상의 레코드를 수신할 수 있습니다. 반환된 레코드가 스트림의 모든 레코드를 나타내는 것은 아닙니다. 기대한 데이터를 수신하지 못한 경우에는 계속해서 `get-records`를 호출합니다.

3. Kinesis의 레코드는 Base64로 인코딩됩니다. Base64 디코더를 사용하여 데이터를 디코딩하여 해당 이벤트가 JSON 형식으로 스트림에 전송되었는지 확인할 수 있습니다.

5단계: 리소스 정리

이 자습서 용도로 생성한 리소스를 보관하고 싶지 않다면 지금 삭제할 수 있습니다. 더 이상 사용하지 않는 AWS 리소스를 삭제하면 AWS 계정에 불필요한 요금이 청구되는 것을 방지할 수 있습니다.

EventBridge 규칙을 삭제하려면

1. EventBridge 콘솔의 규칙 페이지를 엽니다.
2. 생성한 규칙을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

Kinesis 스트림을 삭제하려면

1. Kinesis 콘솔의 데이터 스트림 페이지를 엽니다.
2. 생성한 스트림을 선택합니다.
3. 작업, 삭제를 선택합니다.
4. 필드에 삭제를 입력하고 삭제를 선택합니다.

자습서: EventBridge를 사용하여 Amazon EBS 스냅샷 자동화 예약

일정에 따라 EventBridge 규칙을 실행할 수 있습니다. 이 자습서에서는 일정에 따라 기존 Amazon Elastic Block Store(Amazon EBS) 볼륨의 스냅샷을 생성합니다. 고정된 속도를 선택하여 몇 분마다 스냅샷을 생성하거나 cron 표현식을 사용하여 특정 시간대에 스냅샷이 생성할 수 있습니다.

> **Important**
>
> 기본 제공 대상이 있는 규칙을 생성하려면 AWS Management Console을 사용해야 합니다.

단계:

- 1단계: 규칙 생성
- 2단계: 규칙 테스트
- 3단계: 성공 확인
- 4단계: 리소스 정리

1단계: 규칙 생성

일정에 따라 스냅샷을 가져오는 규칙을 생성합니다. rate 표현식이나 cron 표현식을 사용하여 예약을 지정할 수 있습니다. 자세한 내용은 일정에 따라 실행되는 Amazon EventBridge 규칙 생성 섹션을 참조하세요.

규칙을 생성하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 [Rules]를 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙에 대해 이름과 설명을 입력합니다.

   규칙은 동일한 리전과 동일한 이벤트 버스의 다른 규칙과 동일한 이름을 가질 수 없습니다.
5. 이벤트 버스(Event bus)에서 이 규칙과 연결할 이벤트 버스를 선택합니다. 이 규칙이 자신의 계정에서 발생하는 이벤트와 일치하도록 하려면 AWS 기본 이벤트 버스를 선택합니다. 계정의 AWS 서비스가 이벤트를 출력하면 항상 계정의 기본 이벤트 버스로 이동합니다.
6. 규칙 유형에서 스케줄을 선택합니다.
7. 다음을 선택합니다.

8. 일정 패턴의 경우 일정한 간격(예: 10분마다)으로 실행되는 일정을 선택하고 **5**를 입력하고 드롭다운 목록에서 분을 선택합니다.
9. 다음을 선택합니다.
10. 대상 유형에서 AWS 서비스를 선택합니다.
11. 대상 선택의 경우 드롭다운 목록에서 EBS 생성 스냅샷을 선택합니다.
12. 볼륨 ID에 Amazon EBS 볼륨의 ID를 입력합니다.
13. 실행 역할에서 이 특정 리소스에 대해 새 역할 생성을 선택합니다.
14. 다음을 선택합니다.
15. 다음을 선택합니다.
16. 규칙의 세부 정보를 검토하고 규칙 생성을 선택합니다.

2단계: 규칙 테스트

생성된 첫 번째 스냅샷을 보면 규칙이 작동하는지 확인할 수 있습니다.

규칙을 테스트하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 Elastic Block Store와 스냅샷을 선택합니다.
3. 목록에 첫 번째 스냅샷이 나타나는지 확인합니다.

3단계: 성공 확인

목록에 스냅샷이 보이면 이 자습서를 성공적으로 완료한 것입니다. 스냅샷이 목록에 없는 경우 규칙이 성공적으로 생성되었는지 확인하여 문제 해결을 시작하십시오.

4단계: 리소스 정리

이 자습서 용도로 생성한 리소스를 보관하고 싶지 않다면 지금 삭제할 수 있습니다. 더 이상 사용하지 않는 AWS 리소스를 삭제하면 AWS 계정에 불필요한 요금이 발생하는 것을 방지할 수 있습니다.

EventBridge 규칙을 삭제하려면

1. EventBridge 콘솔의 규칙 페이지를 엽니다.
2. 생성한 규칙을 선택합니다.
3. 삭제를 선택합니다.

4. 삭제를 선택합니다.

자습서: Amazon S3 객체가 생성되면 알림 보내기

Amazon EventBridge 및 Amazon SNS를 사용하여 Amazon Simple Storage Service(S3) 객체가 생성되면 이메일 알림을 보낼 수 있습니다. 이 자습서에서는 SNS 주제 및 구독을 생성합니다. 그런 다음 Amazon S3 `Object Created` 이벤트가 수신되면 EventBridge 콘솔에서 해당 주제에 이벤트를 보내는규칙을 생성합니다.

단계:

- 필수 조건
- 1단계: Amazon SNS 주제 생성
- 2단계: Amazon SNS 구독 생성
- 3단계: 규칙 생성
- 4단계: 규칙 테스트
- 5단계: 리소스 정리

필수 조건

EventBridge에서 Amazon S3 이벤트를 수신하려면 Amazon S3 콘솔에서 EventBridge를 활성화해야 합니다. 이 자습서에서는 EventBridge가 활성화되어 있다고 가정합니다. 자세한 내용은 S3 콘솔에서 Amazon EventBridge 활성화를 참조하십시오.

1단계: Amazon SNS 주제 생성

EventBridge에서 이벤트를 수신할 주제를 생성합니다.

주제 생성

1. https://console.aws.amazon.com/sns/v3/home에서 Amazon SNS 콘솔을 엽니다.
2. 탐색 창에서 주제를 선택합니다.
3. 주제 생성을 선택합니다.
4. 유형에서 표준을 선택합니다.
5. 주제 이름으로 **`eventbridge-test`**를 입력합니다.
6. 주제 생성을 선택합니다.

2단계: Amazon SNS 구독 생성

구독을 생성하면 주제에서 이벤트를 수신할 때 Amazon S3로부터 이메일 알림을 받을 수 있습니다.

구독 생성

1. https://console.aws.amazon.com/sns/v3/home에서 Amazon SNS 콘솔을 엽니다.
2. 탐색 창에서 구독을 선택합니다.
3. 구독 생성을 선택합니다.
4. 주제 ARN에서 1단계에서 생성한 주제를 선택합니다. 이 자습서에서는 eventbridge-test를 선택합니다.
5. 프로토콜에서 이메일을 선택합니다.
6. 엔드포인트에 이메일 주소를 입력합니다.
7. 구독 생성을 선택합니다.
8. AWS 알림을 통해 받은 이메일에서 구독 확인을 선택하여 구독을 확인합니다.

3단계: 규칙 생성

Amazon S3 객체 생성 시 주제에 이벤트를 전송하는 규칙을 생성합니다.

규칙을 생성하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 [Rules]를 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙에 대해 이름과 설명을 입력하세요. 예를 들어, 규칙의 이름을 `s3-test`로 지정합니다.
5. 이벤트 버스의 경우 기본값을 선택합니다.
6. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다.
7. 다음을 선택합니다.
8. 이벤트 소스에서 AWS 이벤트 또는 EventBridge 파트너 이벤트를 선택합니다.
9. 생성 방법에서 패턴 양식 사용을 선택합니다.
10. 이벤트 패턴에서 다음을 수행합니다.
    a. 이벤트 소스의 경우 드롭다운 목록에서 AWS 서비스를 선택합니다.
    b. AWS 서비스의 경우 드롭다운 목록에서 Simple Storage Service(S3)를 선택합니다.

c. 이벤트 유형의 경우 드롭다운 목록에서 Amazon S3 이벤트 알림을 선택합니다.

d. 특정 이벤트를 선택하고 드롭다운 목록에서 객체 생성됨을 선택합니다.

e. 버킷 추가를 선택합니다.

11. 다음을 선택합니다.
12. 대상 유형에서 AWS 서비스를 선택합니다.
13. 대상 선택의 경우 드롭다운 목록에서 SNS 주제를 선택합니다.
14. 주제에서는 1단계: SNS 주제 생성 섹션에서 생성한 Amazon SNS 주제를 선택합니다. 이 예시에서는 `eventbridge-test`를 선택합니다.
15. 다음을 선택합니다.
16. 다음을 선택합니다.
17. 규칙의 세부 정보를 검토하고 규칙 생성을 선택합니다.

4단계: 규칙 테스트

규칙을 테스트하려면 EventBridge가 활성화된 버킷에 파일을 업로드하여 Amazon S3 객체를 생성하십시오. 그런 다음 몇 분 기다렸다가 AWS 알림에서 이메일을 수신하는지 확인합니다.

5단계: 리소스 정리

이 자습서 용도로 생성한 리소스를 보관하고 싶지 않다면 지금 삭제할 수 있습니다. 더 이상 사용하지 않는 AWS 리소스를 삭제하면 AWS 계정에 불필요한 요금이 발생하는 것을 방지할 수 있습니다.

SNS 주제를 삭제하려면

1. SNS 콘솔의 주제 페이지를 엽니다.
2. 생성한 주제를 선택합니다.
3. 삭제를 선택합니다.
4. **`delete me`**을 입력합니다.
5. 삭제를 선택합니다.

SNS 구독을 삭제하려면

1. SNS 콘솔의 구독 페이지를 엽니다.
2. 생성한 구독을 선택합니다.

3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

EventBridge 규칙을 삭제하려면

1. EventBridge 콘솔의 규칙 페이지를 엽니다.
2. 생성한 규칙을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

자습서: EventBridge를 사용하여 AWS Lambda 함수 예약

예약된 일정에 따라 AWS Lambda 함수를 실행하도록 규칙을 설정할 수 있습니다. 이 자습서에서는 AWS Management Console 또는 AWS CLI를 사용하여 규칙을 생성하는 방법을 보여줍니다. AWS CLI를 사용하고 싶지만 아직 설치하지 않은 경우 AWS CLI 버전 2 설치, 업데이트 및 제거를 참조하십시오.

일정의 경우 EventBridge는 일정 표현식에 초 단위의 정밀성을 제공하지 않습니다. cron 표현식을 사용해 가장 정밀하게 설정할 수 있는 단위가 1분입니다. EventBridge와 대상 서비스가 분산되어 있기 때문에 예약된 규칙이 트리거되는 시간과 대상 서비스가 대상 리소스 실행하는 시간 사이에 몇 초의 지연이 있을 수 있습니다.

단계:

- 1단계: Lambda 함수 생성
- 2단계: 규칙 생성
- 3단계: 규칙 확인
- 4단계: 성공 확인
- 5단계: 리소스 정리

1단계: Lambda 함수 생성

Lambda 함수를 생성하여 예약된 이벤트를 기록합니다.

Lambda 함수를 만들려면

1. https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
2. 함수 생성(Create function)을 선택합니다.
3. 새로 작성을 선택합니다.
4. Lambda 함수의 이름과 설명을 입력합니다. 예를 들어 함수 이름을 `LogScheduledEvent`로 지정합니다.
5. 나머지 옵션은 기본값으로 두고 함수 생성을 선택합니다.
6. 함수 페이지의 코드 탭에서 index.js를 두 번 클릭합니다.
7. 기존 코드를 다음 코드로 바꿉니다.

```
'use strict';
```

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

8. 배포를 선택합니다.

2단계: 규칙 생성

일정에 따라 1단계에서 생성한 Lambda 함수를 실행하도록 규칙을 생성합니다.

콘솔 또는 AWS CLI를 사용하여 규칙을 생성할 수 있습니다. AWS CLI를 사용하려면 먼저 Lambda 함수를 간접 호출할 수 있는 권한을 규칙에 부여해야 합니다. 그런 다음 규칙을 생성하고 Lambda 함수를 대상으로서 추가할 수 있습니다.

규칙을 생성하는 방법(콘솔)

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 규칙을 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙에 대해 이름과 설명을 입력합니다.

   규칙은 동일한 리전과 동일한 이벤트 버스의 다른 규칙과 동일한 이름을 가질 수 없습니다.
5. 이벤트 버스(Event bus)에서 이 규칙과 연결할 이벤트 버스를 선택합니다. 이 규칙이 자신의 계정에서 발생하는 이벤트와 일치하도록 하려면 AWS 기본 이벤트 버스를 선택합니다. 계정의 AWS 서비스가 이벤트를 출력하면 항상 계정의 기본 이벤트 버스로 이동합니다.
6. 규칙 유형에서 스케줄을 선택합니다.
7. 다음을 선택합니다.
8. 일정 패턴의 경우 일정한 간격(예: 10분마다)으로 실행되는 일정을 선택하고 **5**를 입력하고 드롭다운 목록에서 분을 선택합니다.
9. 다음을 선택합니다.
10. 대상 유형에서 AWS 서비스를 선택합니다.
11. 대상 선택의 경우 드롭다운 목록에서 Lambda 함수를 선택합니다.
12. 함수의 경우 1단계: Lambda 함수 생성 섹션에서 생성한 Lambda 함수를 선택합니다. 이 예시에서는 `LogScheduledEvent`를 선택합니다.
13. 다음을 선택합니다.

14. 다음을 선택합니다.
15. 규칙의 세부 정보를 검토하고 규칙 생성을 선택합니다.

규칙(AWS CLI)을 생성하려면

1. 일정에 따라 실행되는 규칙을 만들려면 `put-rule` 명령을 사용합니다.

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

이 규칙이 실행되면 이벤트를 만든 다음 대상으로 전송합니다. 다음은 이벤트 예제입니다.

```
{
    "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. EventBridge 서비스 주체(`events.amazonaws.com`)에 규칙을 실행할 권한을 부여하려면 `add-permission` 명령을 사용합니다.

```
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. 다음 콘텐츠가 포함된 `targets.json` 파일을 생성합니다.

```
[
  {
```

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

4. 1단계에서 생성한 Lambda 함수를 규칙에 추가하려면 `put-targets` 명령을 사용합니다.

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

3단계: 규칙 확인

2단계 완료 후 최소 5분 정도 기다리면 Lambda 함수가 간접 호출되었는지 확인할 수 있습니다.

Lambda 함수의 출력 보기

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택합니다.
3. Lambda 함수에 대한 로그 그룹 이름을 선택합니다(`/aws/lambda/function-name`).
4. 로그 스트림 이름을 선택하여 시작한 인스턴스에서 함수를 통해 제공된 데이터를 확인합니다.

4단계: 성공 확인

CloudWatch 로그에 Lambda 이벤트가 표시되면 이 자습서를 성공적으로 완료한 것입니다. 이벤트가 CloudWatch 로그에 없는 경우, 규칙이 성공적으로 생성되었는지 확인하여 문제 해결을 시작하고, 규칙이 올바른 것으로 보이면 Lambda 함수의 코드가 올바른지 확인하십시오.

5단계: 리소스 정리

이 자습서 용도로 생성한 리소스를 보관하고 싶지 않다면 지금 삭제할 수 있습니다. 더 이상 사용하지 않는 AWS 리소스를 삭제하면 AWS 계정에 불필요한 요금이 발생하는 것을 방지할 수 있습니다.

EventBridge 규칙을 삭제하려면

1. EventBridge 콘솔의 규칙 페이지를 엽니다.
2. 생성한 규칙을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

Lambda 함수를 삭제하려면

1. Lambda 콘솔의 함수 페이지를 엽니다.
2. 생성한 함수를 선택합니다.
3. 작업, 삭제를 선택합니다.
4. 삭제를 선택합니다.

SaaS 공급자와의 통합을 위한 Amazon EventBridge 자습서

EventBridge는 SaaS 파트너 애플리케이션 및 서비스와 직접 협력하여 이벤트를 보내고 받을 수 있습니다. 다음 자습서에서는 SaaS 파트너와 EventBridge를 통합하는 방법을 보여줍니다.

자습서:

- 자습서: API 대상으로 Datadog에 대한 연결 생성
- 자습서: API 대상으로 Salesforce에 대한 연결 생성
- 자습서: API 대상으로 Zendesk에 대한 연결 생성

자습서: API 대상으로 Datadog에 대한 연결 생성

EventBridge를 사용하여 Datadog와 같은 타사 서비스로 이벤트를 라우팅할 수 있습니다.

이 자습서에서는 EventBridge 콘솔을 사용하여 Datadog에 대한 연결, Datadog을 가리키는 API 대상 및 이벤트를 Datadog로 라우팅하는규칙을 생성합니다.

단계:

- 필수 조건
- 1단계: 연결 생성
- 2단계: API 대상 생성
- 3단계: 규칙 생성
- 4단계: 규칙 테스트
- 5단계: 리소스 정리

필수 조건

이 자습서를 완료하려면 다음 리소스가 필요합니다.

- Datadog 계정.
- Datadog API 키.
- EventBridge 지원 Amazon Simple Storage Service(S3) 버킷.

1단계: 연결 생성

Datadog에 이벤트를 보내려면 먼저 Datadog API에 대한 연결을 설정해야 합니다.

연결을 생성하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 API 대상을 선택합니다.
3. 연결 탭을 선택한 다음 연결 생성을 선택합니다.
4. 연결의 이름과 설명을 입력합니다. 예를 들어 **`Datadog`**를 이름으로 **`Datadog API Connection`**을 설명으로 입력합니다.
5. 권한 부여 유형에서 API 키를 선택합니다.

6. API 키 이름에 **DD-API-KEY**를 입력합니다.
7. 값 에 Datadog 비밀 API 키를 붙여넣습니다.
8. 생성을 선택합니다.

2단계: API 대상 생성

연결을 만들었으니 이제 규칙의 대상으로 사용할 API 대상을 생성해 보겠습니다.

API 대상을 생성하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 API 대상을 선택합니다.
3. API 대상 생성을 선택합니다.
4. API 대상에 대한 이름과 설명을 입력합니다. 예시에서는 이름에 **DatadogAD**를 입력하고 설명에 **Datadog API Destination**을 입력합니다.
5. API 대상 엔드포인트에 **https://http-intake.logs.datadoghq.com/api/v2/logs**를 입력합니다.
6. HTTP 메서드에 대해 POST를 선택합니다.
7. 간접 호출 속도 제한에는 **300**를 입력합니다.
8. 연결에서 기존 연결 사용을 선택하고 1단계에서 만든 Datadog 연결을 선택합니다.
9. 생성을 선택합니다.

3단계: 규칙 생성

다음으로 Amazon S3 객체 생성 시 이벤트를 Datadog로 전송하는 규칙을 생성합니다.

규칙을 생성하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 [Rules]를 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙에 대해 이름과 설명을 입력합니다. 이 예에서는 이름에 **DatadogRule**을 입력하고 설명에 **Rule to send events to Datadog for S3 object creation**을 입력합니다.
5. 이벤트 버스에서 기본값을 선택합니다.
6. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다.

7. 다음을 선택합니다.
8. 이벤트 소스에서 기타를 선택합니다.
9. 이벤트 패턴에 다음을 입력합니다.

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

10. 다음을 선택합니다.
11. 대상 유형에서는 EventBridge API 대상을 선택합니다.
12. API 대상의 경우 기존 API 대상 사용을 선택한 다음 2단계에서 생성한 DatadogAD 대상을 선택합니다.
13. 실행 역할에서 이 특정 리소스에 대해 새 역할 생성을 선택합니다.
14. 추가 설정에서 다음을 수행합니다.
    a. 대상 입력 구성의 경우 드롭다운 목록에서 입력 변환기를 선택합니다.
    b. 입력 변환기 구성을 선택합니다.
    c. 샘플 이벤트의 경우 다음을 입력합니다.

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

    d. 대상 입력 변환기의 경우 다음을 수행합니다.
        i. 입력 경로에 다음을 입력합니다.

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

        ii. 입력 템플릿에는 다음 사항을 입력합니다.

```
{"message": <detail>}
```

    e. 확인을 선택합니다.
15. 다음을 선택합니다.
16. 다음을 선택합니다.
17. 규칙의 세부 정보를 검토하고 규칙 생성을 선택합니다.

4단계: 규칙 테스트

규칙을 테스트하려면 EventBridge가 활성화된 버킷에 파일을 업로드하여 Amazon S3 객체를 생성하십시오. 생성된 객체는 Datadog Logs 콘솔에 기록됩니다.

5단계: 리소스 정리

이 자습서 용도로 생성한 리소스를 보관하고 싶지 않다면 지금 삭제할 수 있습니다. 더 이상 사용하지 않는 AWS 리소스를 삭제하면 AWS 계정에 불필요한 요금이 발생하는 것을 방지할 수 있습니다.

EventBridge 연결을 삭제하려면

1. EventBridge 콘솔의 API 대상 페이지를 엽니다.
2. 연결 탭을 선택합니다.
3. 생성한 연결을 선택합니다.
4. 삭제를 선택합니다.
5. 연결 이름을 입력하고 삭제를 선택합니다.

EventBridge API 대상을 삭제하려면

1. EventBridge 콘솔의 API 대상 페이지를 엽니다.
2. 생성한 API 대상을 선택합니다.
3. 삭제를 선택합니다.
4. API 대상의 이름을 입력하고 삭제를 선택합니다.

EventBridge 규칙을 삭제하려면

1. EventBridge 콘솔의 규칙 페이지를 엽니다.
2. 생성한 규칙을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

자습서: API 대상으로 Salesforce에 대한 연결 생성

를 EventBridge 사용하여 이벤트를 타사 서비스 (예:) 로 라우팅할 수 Salesforce있습니다.

이 자습서에서는 EventBridge 콘솔을 사용하여 연결Salesforce, 를 가리키는 API 대상Salesforce, 이벤트를 라우팅하는 규칙을 생성합니다Salesforce.

단계:

- 필수 조건
- 1단계: 연결 생성
- 2단계: API 대상 생성
- 3단계: 규칙 생성
- 4단계: 규칙 테스트
- 5단계: 리소스 정리

필수 조건

이 자습서를 완료하려면 다음 리소스가 필요합니다.

- Salesforce 계정.
- Salesforce 연결된 앱.
- Salesforce 보안 토큰.
- Salesforce 사용자 지정 플랫폼 이벤트.
- EventBridge활성화된 아마존 심플 스토리지 서비스 (Amazon S3) 버킷.

1단계: 연결 생성

Salesforce에 이벤트를 보내려면 먼저 Salesforce API에 대한 연결을 설정해야 합니다.

연결을 생성하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 API 대상을 선택합니다.
3. 연결 탭을 선택한 다음 연결 생성을 선택합니다.

4. 연결의 이름과 설명을 입력합니다. 예를 들어 **Salesforce**를 이름으로 **Salesforce API Connection**을 설명으로 입력합니다.
5. 대상 유형으로는 파트너를 선택하고 파트너 대상의 경우 드롭다운 목록에서 Salesforce를 선택합니다.
6. 권한 부여 엔드포인트에 다음 중 하나를 입력합니다.
   - 프로덕션 조직을 사용하는 경우 다음을 입력하십시오. **https://*MyDomainName*.my.salesforce.com./services/oauth2/token**
   - 향상된 도메인이 없는 샌드박스를 사용하는 경우 다음을 입력하십시오. **https://*MyDomainName*--*SandboxName*.my. salesforce.com/services /oauth2/token**
   - 향상된 도메인이 있는 샌드박스를 사용하는 경우 다음을 입력하십시오. **https://*MyDomainName*-- *SandboxName*.sandbox.my.salesforce.com/services/oauth2/token**
7. HTTP 메서드의 경우 드롭다운 목록에서 POST를 선택합니다.
8. 클라이언트 ID에는 Salesforce 연결된 앱의 클라이언트 ID를 입력합니다.
9. 클라이언트 비밀번호에는Salesforce 연결된 앱의 클라이언트 비밀번호를 입력합니다.
10. OAuth Http 파라미터의 경우 다음 키/값 쌍을 입력합니다.

Key(키)	값
grant_type	client_credentials

11. 생성을 선택하세요.

2단계: API 대상 생성

연결을 만들었으니 이제 규칙의 대상으로 사용할 API 대상을 생성해 보겠습니다.

API 대상을 생성하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 API 대상을 선택합니다.
3. API 대상 생성을 선택합니다.
4. API 대상에 대한 이름과 설명을 입력합니다. 예시에서는 이름에 **SalesforceAD**를 입력하고 설명에 **Salesforce API Destination**을 입력합니다.

5. API 대상 엔드포인트에 **https://*MyDomainName*.my.salesforce.com/services/data/v54.0/sobjects/*MyEvent__e***를 입력합니다. 여기서 Myevent__e는 정보를 보내려는 플랫폼 이벤트입니다.
6. HTTP 메서드의 경우 드롭다운 목록에서 POST를 선택합니다.
7. 간접 호출 속도 제한에는 **300**을 입력합니다.
8. 연결에서 기존 연결 사용을 선택하고 1단계에서 만든 Salesforce 연결을 선택합니다.
9. 생성을 선택하세요.

3단계: 규칙 생성

다음으로 Amazon S3 객체 생성 시 이벤트를 Salesforce로 전송하는 규칙을 생성합니다.

규칙을 생성하려면

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 규칙을 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙에 대해 이름과 설명을 입력하십시오. 이 예에서는 이름에 **SalesforceRule**을 입력하고 설명에 **Rule to send events to Salesforce for S3 object creation**을 입력합니다.
5. 이벤트 버스에서 기본값을 선택합니다.
6. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다.
7. 다음을 선택합니다.
8. 이벤트 소스에서 기타를 선택합니다.
9. 이벤트 패턴에 다음을 입력합니다.

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

10. 다음을 선택합니다.
11. 대상 유형에서는 EventBridge API 대상을 선택합니다.
12. API 대상의 경우 기존 API 대상 사용을 선택한 다음 2단계에서 생성한 SalesforceAD 대상을 선택합니다.
13. 실행 역할에서 이 특정 리소스에 대해 새 역할 생성을 선택합니다.
14. 추가 설정에서 다음을 수행합니다.

a. 대상 입력 구성의 경우 드롭다운 목록에서 입력 변환기를 선택합니다.
b. 입력 변환기 구성을 선택합니다.
c. 샘플 이벤트의 경우 다음을 입력합니다.

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

d. 대상 입력 변환기의 경우 다음을 수행합니다.

i. 입력 경로에 다음을 입력합니다.

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

ii. 입력 템플릿에는 다음 사항을 입력합니다.

```
{"message": <detail>}
```

e. 확인을 선택합니다.

15. 다음을 선택합니다.
16. 다음을 선택합니다.
17. 규칙의 세부 정보를 검토하고 규칙 생성을 선택합니다.

4단계: 규칙 테스트

규칙을 테스트하려면 EventBridge -enabled 버킷에 파일을 업로드하여 Amazon S3 객체를 생성하십시오. 생성된 객체에 대한 정보는 Salesforce 플랫폼 이벤트로 전송됩니다.

5단계: 리소스 정리

이 자습서 용도로 생성한 리소스를 보관하고 싶지 않다면 지금 삭제할 수 있습니다. 더 이상 사용하지 않는 AWS 리소스를 삭제하면 계정에 불필요한 요금이 청구되는 것을 방지할 수 AWS 있습니다.

EventBridge 연결을 삭제하려면

1. EventBridge 콘솔의 API 대상 페이지를 엽니다.
2. 연결 탭을 선택합니다.
3. 생성한 연결을 선택합니다.

4. 삭제를 선택합니다.
5. 연결 이름을 입력하고 삭제를 선택합니다.

EventBridge API 대상을 삭제하려면

1. EventBridge 콘솔의 API 대상 페이지를 엽니다.
2. 생성한 API 대상을 선택합니다.
3. 삭제를 선택합니다.
4. API 대상의 이름을 입력하고 삭제를 선택합니다.

EventBridge 규칙을 삭제하려면

1. EventBridge 콘솔의 규칙 페이지를 엽니다.
2. 생성한 규칙을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

자습서: API 대상으로 Zendesk에 대한 연결 생성

EventBridge를 사용하여 Zendesk와 같은 타사 서비스로 이벤트를 라우팅할 수 있습니다.

이 자습서에서는 EventBridge 콘솔을 사용하여 Zendesk에 대한 연결, Zendesk을 가리키는 API 대상 및 이벤트를 Zendesk로 라우팅하는규칙을 생성합니다.

단계:

필수 조건

이 자습서를 완료하려면 다음 리소스가 필요합니다.

- Zendesk 계정.
- EventBridge 지원 Amazon Simple Storage Service(S3) 버킷.

1단계: 연결 생성

Zendesk에 이벤트를 보내려면 먼저 Zendesk API에 대한 연결을 설정해야 합니다.

연결을 생성하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 API 대상을 선택합니다.
3. 연결 탭을 선택한 다음 연결 생성을 선택합니다.
4. 연결의 이름과 설명을 입력합니다. 이 예에서는 이름에 **Zendesk**을 입력하고 설명에 **Connection to Zendesk API**을 입력합니다.
5. 인증 유형에서 기본(사용자 이름/암호)을 선택합니다.
6. 사용자 이름에 Zendesk 사용자 이름을 입력합니다.

7. 암호에 Zendesk 암호를 입력합니다.
8. 생성을 선택합니다.

2단계: API 대상 생성

연결을 만들었으니 이제 규칙의 대상으로 사용할 API 대상을 생성해 보겠습니다.

API 대상을 생성하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 API 대상을 선택합니다.
3. API 대상 생성을 선택합니다.
4. API 대상에 대한 이름과 설명을 입력합니다. 이 예에서는 이름에 **ZendeskAD**을 입력하고 설명에 **Zendesk API destination**을 입력합니다.
5. API 대상 엔드포인트에 **https://*your-subdomain*.zendesk.com/api/v2/tickets.json**을 입력합니다. 여기서 *your-subdomain*은 Zendesk 계정과 연결된 하위 도메인입니다.
6. HTTP 메서드에 대해 POST를 선택합니다.
7. 간접 호출 속도 제한에는 **10**을 입력합니다.
8. 연결에서 기존 연결 사용을 선택하고 1단계에서 만든 Zendesk 연결을 선택합니다.
9. 생성을 선택합니다.

3단계: 규칙 생성

다음으로 Amazon S3 객체 생성 시 이벤트를 Zendesk로 전송하는 규칙을 생성합니다.

규칙을 생성하려면

1. https://console.aws.amazon.com/events/에서 Amazon EventBridge 콘솔을 엽니다.
2. 탐색 창에서 [Rules]를 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙에 대해 이름과 설명을 입력하세요. 이 예에서는 이름에 **ZendeskRule**을 입력하고 설명에 **Rule to send events to Zendesk when S3 objects are created**을 입력합니다.
5. 이벤트 버스에서 기본값을 선택합니다.
6. 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다.

7. 다음을 선택합니다.
8. 이벤트 소스에서 기타를 선택합니다.
9. 이벤트 패턴에 다음을 입력합니다.

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

10. 다음을 선택합니다.
11. 대상 유형에서는 EventBridge API 대상을 선택합니다.
12. API 대상의 경우 기존 API 대상 사용을 선택한 다음 2단계에서 생성한 ZendeskAD 대상을 선택합니다.
13. 실행 역할에서 이 특정 리소스에 대해 새 역할 생성을 선택합니다.
14. 추가 설정에서 다음을 수행합니다.
    a. 대상 입력 구성의 경우 드롭다운 목록에서 입력 변환기를 선택합니다.
    b. 입력 변환기 구성을 선택합니다.
    c. 샘플 이벤트의 경우 다음을 입력합니다.

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

    d. 대상 입력 변환기의 경우 다음을 수행합니다.
        i. 입력 경로에 다음을 입력합니다.

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

        ii. 입력 템플릿에는 다음 사항을 입력합니다.

```
{"message": <detail>}
```

    e. 확인을 선택합니다.
15. 다음을 선택합니다.
16. 다음을 선택합니다.
17. 규칙의 세부 정보를 검토하고 규칙 생성을 선택합니다.

4단계: 규칙 테스트

규칙을 테스트하려면 EventBridge가 활성화된 버킷에 파일을 업로드하여 Amazon S3 객체를 생성하십시오. 이벤트가 규칙과 일치하면 EventBridge는 Zendesk Create Ticket API를 호출합니다. 새 티켓이 Zendesk 대시보드에 나타납니다.

5단계: 리소스 정리

이 자습서 용도로 생성한 리소스를 보관하고 싶지 않다면 지금 삭제할 수 있습니다. 더 이상 사용하지 않는 AWS 리소스를 삭제하면 AWS 계정에 불필요한 요금이 발생하는 것을 방지할 수 있습니다.

EventBridge 연결을 삭제하려면

1. EventBridge 콘솔의 API 대상 페이지를 엽니다.
2. 연결 탭을 선택합니다.
3. 생성한 연결을 선택합니다.
4. 삭제를 선택합니다.
5. 연결 이름을 입력하고 삭제를 선택합니다.

EventBridge API 대상을 삭제하려면

1. EventBridge 콘솔의 API 대상 페이지를 엽니다.
2. 생성한 API 대상을 선택합니다.
3. 삭제를 선택합니다.
4. API 대상의 이름을 입력하고 삭제를 선택합니다.

EventBridge 규칙을 삭제하려면

1. EventBridge 콘솔의 규칙 페이지를 엽니다.
2. 생성한 규칙을 선택합니다.
3. 삭제를 선택합니다.
4. 삭제를 선택합니다.

EventBridge AWS SDK와 함께 사용

AWS 소프트웨어 개발 키트 (SDK) 는 널리 사용되는 여러 프로그래밍 언어에 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예시 및 설명서를 제공합니다.

SDK 설명서	코드 예시
AWS SDK for C++	AWS SDK for C++ 코드 예제
AWS CLI	AWS CLI 코드 예제
AWS SDK for Go	AWS SDK for Go 코드 예제
AWS SDK for Java	AWS SDK for Java 코드 예제
AWS SDK for JavaScript	AWS SDK for JavaScript 코드 예제
AWS SDK for Kotlin	AWS SDK for Kotlin 코드 예제
AWS SDK for .NET	AWS SDK for .NET 코드 예제
AWS SDK for PHP	AWS SDK for PHP 코드 예제
AWS Tools for PowerShell	PowerShell 코드 예제를 위한 도구
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예제
AWS SDK for Ruby	AWS SDK for Ruby 코드 예제
AWS SDK for Rust	AWS SDK for Rust 코드 예제
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP 코드 예제
AWS SDK for Swift	AWS SDK for Swift 코드 예제

특정 예제는 EventBridge 을 참조하십시오AWS SDK EventBridge 사용을 위한 코드 예제.

가용성 예제

필요한 예제를 찾을 수 없습니까? 이 페이지 하단의 피드백 제공 링크를 사용하여 코드 예시를 요청하세요.

AWS SDK EventBridge 사용을 위한 코드 예제

다음 코드 예제는 AWS 소프트웨어 개발 키트 (SDK) EventBridge 와 함께 사용하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

교차 서비스 예시는 여러 AWS 서비스전반에서 작동하는 샘플 애플리케이션입니다.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. EventBridge AWS SDK와 함께 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

시작하기

안녕하세요. EventBridge

다음 코드 예제는 사용을 시작하는 방법을 보여줍니다 EventBridge.

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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();
        }
    }
}
```

- API 세부 정보는 AWS SDK for .NET API ListEventBuses참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
```

```
 * 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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API ListEventBuses참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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}")
        }
    }
}
```

- API 세부 정보는 Kotlin APIAWS SDK 레퍼런스를 참조하세요 ListEventBuses.

코드 예시

- SDK 사용을 위한 조치 EventBridge AWS
  - AWS SDK 또는 DeleteRule CLI와 함께 사용
  - AWS SDK 또는 DescribeRule CLI와 함께 사용

- AWS SDK 또는 DisableRule CLI와 함께 사용
- AWS SDK 또는 EnableRule CLI와 함께 사용
- AWS SDK 또는 ListRuleNamesByTarget CLI와 함께 사용
- AWS SDK 또는 ListRules CLI와 함께 사용
- AWS SDK 또는 ListTargetsByRule CLI와 함께 사용
- AWS SDK 또는 PutEvents CLI와 함께 사용
- AWS SDK 또는 PutRule CLI와 함께 사용
- AWS SDK 또는 PutTargets CLI와 함께 사용
- AWS SDK 또는 RemoveTargets CLI와 함께 사용
- SDK EventBridge 사용 AWS 시나리오
  - AWS SDK를 EventBridge 사용하여 Amazon에서 규칙을 생성하고 트리거하기
  - SDK를 사용하여 EventBridge 규칙 및 대상으로 시작하세요. AWS
- SDK 사용을 위한 EventBridge 크로스 서비스 예제 AWS
  - 예약된 이벤트를 사용하여 Lambda 함수 호출

SDK 사용을 위한 조치 EventBridge AWS

다음 코드 예제는 AWS SDK를 사용하여 개별 EventBridge 작업을 수행하는 방법을 보여줍니다. 이 발췌문은 EventBridge API를 호출하며 컨텍스트에서 실행해야 하는 대규모 프로그램에서 발췌한 코드입니다. 각 예제에는 코드 설정 및 실행 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 Amazon EventBridge API 참조를 참조하십시오.

예제

- AWS SDK 또는 DeleteRule CLI와 함께 사용
- AWS SDK 또는 DescribeRule CLI와 함께 사용
- AWS SDK 또는 DisableRule CLI와 함께 사용
- AWS SDK 또는 EnableRule CLI와 함께 사용
- AWS SDK 또는 ListRuleNamesByTarget CLI와 함께 사용
- AWS SDK 또는 ListRules CLI와 함께 사용
- AWS SDK 또는 ListTargetsByRule CLI와 함께 사용
- AWS SDK 또는 PutEvents CLI와 함께 사용

- AWS SDK 또는 PutRule CLI와 함께 사용
- AWS SDK 또는 PutTargets CLI와 함께 사용
- AWS SDK 또는 RemoveTargets CLI와 함께 사용

AWS SDK 또는 **DeleteRule** CLI와 함께 사용

다음 코드 예제는 DeleteRule의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 규칙 및 목표 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

이름을 기준으로 규칙을 삭제합니다.

```
/// <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;
}
```

- API 세부 정보는 AWS SDK for .NET API DeleteRule참조를 참조하십시오.

CLI

AWS CLI

CloudWatch 이벤트 규칙 삭제하기

이 예제에서는 InstanceStateChanges EC2라는 규칙을 삭제합니다.

```
aws events delete-rule --name "EC2InstanceStateChanges"
```

- API 세부 정보는 AWS CLI 명령 DeleteRule참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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");
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API DeleteRule참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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")
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 DeleteRule.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. EventBridge AWS SDK와 함께 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **DescribeRule** CLI와 함께 사용

다음 코드 예제는 DescribeRule의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 규칙 및 목표 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

규칙 설명을 사용하여 규칙의 상태를 가져옵니다.

```
    /// <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;
    }
```

- API 세부 정보는 AWS SDK for .NET API DescribeRule참조를 참조하십시오.

CLI

AWS CLI

CloudWatch 이벤트 규칙에 대한 정보를 표시하려면

이 예제에서는 DailyLambdaFunction 다음과 같은 규칙에 대한 정보를 표시합니다.

```
aws events describe-rule --name "DailyLambdaFunction"
```

- API 세부 정보는 AWS CLI 명령 DescribeRule참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API DescribeRule참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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")
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 DescribeRule.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. EventBridge AWS SDK와 함께 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **DisableRule** CLI와 함께 사용

다음 코드 예제는 DisableRule의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 규칙 및 목표 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

규칙 이름을 사용하여 규칙을 비활성화합니다.

```
/// <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;
}
```

- API 세부 정보는 AWS SDK for .NET API DisableRule참조를 참조하십시오.

CLI

AWS CLI

CloudWatch 이벤트 규칙을 비활성화하려면

이 예제에서는 이름이 DailyLambdaFunction 지정된 규칙을 비활성화합니다. 규칙이 삭제되지는 않습니다.

```
aws events disable-rule --name "DailyLambdaFunction"
```

- API 세부 정보는 AWS CLI 명령 DisableRule참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

규칙 이름을 사용하여 규칙을 비활성화합니다.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API DisableRule참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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)
        }
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 DisableRule.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. EventBridge AWS SDK와 함께 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **EnableRule** CLI와 함께 사용

다음 코드 예제는 EnableRule의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 규칙 및 목표 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

규칙 이름을 사용하여 규칙을 활성화합니다.

```
/// <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;
}
```

- API 세부 정보는 AWS SDK for .NET API EnableRule참조를 참조하십시오.

CLI

AWS CLI

CloudWatch 이벤트 규칙을 활성화하려면

이 예제에서는 이전에 비활성화되었던 이름이 지정된 DailyLambdaFunction 규칙을 활성화합니다.

```
aws events enable-rule --name "DailyLambdaFunction"
```

- API 세부 정보는 AWS CLI 명령 EnableRule참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

규칙 이름을 사용하여 규칙을 활성화합니다.

```
   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);
       }
```

```
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API EnableRule참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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)
        }
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 EnableRule.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. EventBridge AWS SDK와 함께 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **ListRuleNamesByTarget** CLI와 함께 사용

다음 코드 예제는 ListRuleNamesByTarget의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 규칙 및 목표 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

대상을 사용하여 모든 규칙 이름을 나열합니다.

```
    /// <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;
    }
```

- API 세부 정보는 AWS SDK for .NET API ListRuleNamesByTarget참조를 참조하십시오.

CLI

AWS CLI

지정된 대상이 있는 모든 규칙을 표시하는 방법

이 예제는 이름이 MyFunctionName ""인 Lambda 함수를 대상으로 하는 모든 규칙을 표시합니다.

```
aws events list-rule-names-by-target --target-arn "arn:aws:lambda:us-
east-1:123456789012:function:MyFunctionName"
```

- API 세부 정보는 AWS CLI 명령 ListRuleNamesByTarget참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

대상을 사용하여 모든 규칙 이름을 나열하세요.

```
   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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API ListRuleNamesByTarget참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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")
        }
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 ListRuleNamesByTarget.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. EventBridge AWS SDK와 함께 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **ListRules** CLI와 함께 사용

다음 코드 예제는 ListRules의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 규칙 및 목표 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

이벤트 버스의 모든 규칙을 나열합니다.

```
   /// <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;
    }
```

- API 세부 정보는 AWS SDK for .NET API ListRules참조를 참조하십시오.

CLI

AWS CLI

모든 CloudWatch 이벤트 규칙 목록을 표시하려면

이 예제에서는 해당 지역의 모든 CloudWatch 이벤트 규칙을 표시합니다.

```
aws events list-rules
```

특정 문자열로 시작하는 CloudWatch 이벤트 규칙 목록을 표시하려면

이 예제에서는 이름이 “Daily”로 시작하는 지역의 모든 CloudWatch 이벤트 규칙을 표시합니다.

```
aws events list-rules --name-prefix "Daily"
```

- API 세부 정보는 AWS CLI 명령 ListRules참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

규칙 이름을 사용하여 규칙을 활성화합니다.

```
    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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API ListRules참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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}")
        }
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 ListRules.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. EventBridge AWS SDK와 함께 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **ListTargetsByRule** CLI와 함께 사용

다음 코드 예제는 ListTargetsByRule의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 규칙 및 목표 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

규칙 이름을 사용하여 규칙의 모든 대상을 나열합니다.

```
    /// <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;
    }
```

- API 세부 정보는 AWS SDK for .NET API ListTargetsByRule참조를 참조하십시오.

CLI

AWS CLI

CloudWatch 이벤트 규칙의 모든 대상을 표시하려면

이 예제에서는 DailyLambdaFunction 다음과 같은 이름의 규칙의 모든 대상을 표시합니다.

```
aws events list-targets-by-rule --rule  "DailyLambdaFunction"
```

- API 세부 정보는 AWS CLI 명령 ListTargetsByRule참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

규칙 이름을 사용하여 규칙의 모든 대상을 나열하세요.

```
    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());
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API ListTargetsByRule참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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}")
        }
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 ListTargetsByRule.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. EventBridge AWS SDK와 함께 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **PutEvents** CLI와 함께 사용

다음 코드 예제는 PutEvents의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 규칙 생성 및 트리거
- 규칙 및 목표 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

규칙의 사용자 지정 패턴과 일치하는 이벤트를 전송합니다.

```
    /// <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;
    }
```

- API 세부 정보는 AWS SDK for .NET API PutEvents참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#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>
```

이벤트를 전송합니다.

```
        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;
        }
```

- API 세부 정보는 AWS SDK for C++ API PutEvents참조를 참조하십시오.

CLI

AWS CLI

커스텀 이벤트를 이벤트에 보내려면 CloudWatch

이 예시에서는 커스텀 이벤트를 CloudWatch Events에 전송합니다. 이벤트는 putevents.json 파일 내에 포함되어 있습니다.

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

putevents.json file 파일의 콘텐츠는 다음과 같습니다.

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

- API 세부 정보는 AWS CLI 명령 PutEvents참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API PutEvents참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
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;
};
```

- API 세부 정보는 AWS SDK for JavaScript API PutEvents참조를 참조하십시오.

JavaScript (v2) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// 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);
  }
});
```

- API 세부 정보는 AWS SDK for JavaScript API PutEvents참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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)
        }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 PutEvents.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. EventBridge AWS SDK와 함께 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **PutRule** CLI와 함께 사용

다음 코드 예제는 PutRule의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 규칙 생성 및 트리거
- 규칙 및 목표 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon Simple Storage Service 버킷에 객체가 추가될 때 트리거되는 규칙을 생성합니다.

```
    /// <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;
    }
```

사용자 지정 패턴을 사용하는 규칙을 생성합니다.

```
    /// <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;
    }
```

- API 세부 정보는 AWS SDK for .NET API PutRule참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#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>
```

규칙을 생성합니다.

```
        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;
        }
```

- API 세부 정보는 AWS SDK for C++ API PutRule참조를 참조하십시오.

CLI

AWS CLI

CloudWatch 이벤트 규칙을 만들려면

이 예시에서는 매일 오전 09:00(UTC)에 트리거되는 규칙을 생성합니다. put-targets를 사용하여 Lambda 함수를 이 규칙의 대상으로 추가하는 경우 매일 지정된 시간에 Lambda 함수를 실행할 수 있습니다.

```
aws events put-rule --name "DailyLambdaFunction" --schedule-expression "cron(0 9
 * * ? *)"
```

이 예시에서는 리전 내 EC2 인스턴스가 상태가 변경될 때 트리거되는 규칙을 생성합니다.

```
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"
```

이 예시에서는 리전 내 EC2 인스턴스가 정지 또는 종료될 때 트리거되는 규칙을 생성합니다.

```
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"
```

- API 세부 정보는 AWS CLI 명령 PutRule참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

예약된 규칙을 생성합니다.

```
    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);
        }
    }
```

객체가 Amazon Simple Storage Service 버킷에 추가될 때 트리거되는 규칙을 생성하세요.

```
    // 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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API PutRule참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
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;
};
```

- API 세부 정보는 AWS SDK for JavaScript API PutRule참조를 참조하십시오.

JavaScript (v2) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// 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);
  }
});
```

- API 세부 정보는 AWS SDK for JavaScript API PutRule참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

예약된 규칙을 생성합니다.

```
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}")
    }
}
```

객체가 Amazon Simple Storage Service 버킷에 추가될 때 트리거되는 규칙을 생성하세요.

```
// 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}")
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 PutRule.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. EventBridge AWS SDK와 함께 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **PutTargets** CLI와 함께 사용

다음 코드 예제는 PutTargets의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 규칙 및 목표 시작하기

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon SNS 주제를 규칙의 대상으로 추가합니다.

```
    /// <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;
    }
```

규칙의 대상에 입력 변환기를 추가합니다.

```
    /// <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;
    }
```

- API 세부 정보는 AWS SDK for .NET API PutTargets참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#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>
```

대상을 추가합니다.

```
        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;
        }
```

- API 세부 정보는 AWS SDK for C++ API PutTargets참조를 참조하십시오.

CLI

AWS CLI

CloudWatch 이벤트 규칙에 대상을 추가하려면

다음 예시에서는 Lambda 함수를 규칙 대상으로 추가합니다.

```
aws events put-targets --rule DailyLambdaFunction --targets
 "Id"="1","Arn"="arn:aws:lambda:us-east-1:123456789012:function:MyFunctionName"
```

이 예시에서는 Amazon Kinesis 스트림을 대상으로 설정하여 이 규칙에 의해 포착된 이벤트가 스트림으로 전달되도록 합니다.

```
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"
```

이 예시에서는 두 개의 Amazon Kinesis 스트림을 하나의 규칙 대상으로 설정합니다.

```
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"
```

- API 세부 정보는 AWS CLI 명령 PutTargets참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon SNS 주제를 규칙의 대상으로 추가합니다.

```
    // 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 + ".");
    }
```

규칙의 대상에 입력 변환기를 추가합니다.

```
    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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API PutTargets참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
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;
};
```

- API 세부 정보는 AWS SDK for JavaScript API PutTargets참조를 참조하십시오.

JavaScript (v2) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// 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);
  }
});
```

- API 세부 정보는 AWS SDK for JavaScript API PutTargets참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// 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.")
    }
}
```

규칙의 대상에 입력 변환기를 추가합니다.

```
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)
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 PutTargets.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. EventBridge AWS SDK와 함께 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **RemoveTargets** CLI와 함께 사용

다음 코드 예제는 RemoveTargets의 사용 방법을 보여줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

규칙 이름을 사용하여 규칙의 모든 대상을 제거합니다.

```
    /// <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;
    }
```

- API 세부 정보는 AWS SDK for .NET API RemoveTargets참조를 참조하십시오.

CLI

AWS CLI

이벤트 대상을 제거하는 방법

이 예제에서는 이름이 MyStream 1인 Amazon Kinesis 스트림을 규칙의 대상에서 제거합니다. DailyLambdaFunction 이 DailyLambdaFunction 스트림은 생성 당시 ID가 Target1인 대상으로 설정되었습니다.

```
aws events remove-targets --rule "DailyLambdaFunction" --ids "Target1"
```

- API 세부 정보는 AWS CLI 명령 RemoveTargets참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

규칙 이름을 사용하여 규칙의 모든 대상을 제거합니다.

```
   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");
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API RemoveTargets참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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")
            }
        }
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 RemoveTargets.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. EventBridge AWS SDK와 함께 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

SDK EventBridge 사용 AWS 시나리오

다음 코드 예제는 AWS SDK를 사용하여 일반적인 시나리오를 구현하는 방법을 보여줍니다. EventBridge 이 시나리오는 내에서 여러 함수를 호출하여 특정 작업을 수행하는 방법을 보여줍니다. EventBridge 각 시나리오에는 코드 설정 및 실행 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

예제

- AWS SDK를 EventBridge 사용하여 Amazon에서 규칙을 생성하고 트리거하기
- SDK를 사용하여 EventBridge 규칙 및 대상으로 시작하세요. AWS

AWS SDK를 EventBridge 사용하여 Amazon에서 규칙을 생성하고 트리거하기

다음 코드 예제는 Amazon에서 규칙을 생성하고 트리거하는 방법을 보여줍니다 EventBridge.

Ruby

SDK for Ruby

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

함수를 올바른 순서로 직접적으로 호출합니다.

```
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"
```

지정된 Amazon Simple Notification Service(SNS) 주제가 이 함수에 제공된 주제 중에 있는지 확인합니다.

```
# 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
```

Amazon SNS에서 호출자가 사용할 수 있는 주제 중에 지정된 주제가 있는지 확인합니다.

```
# 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
```

Amazon SNS에서 주제를 생성한 다음, 이메일 주소를 구독하여 해당 주제에 대한 알림을 받습니다.

```
# 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
```

이 함수에 제공된 역할 중에 지정된 AWS Identity and Access Management (IAM) 역할이 존재하는지 확인하세요.

```
# 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
```

IAM에서 호출자가 사용할 수 있는 역할 중에 지정된 역할이 있는지 확인합니다.

```
# 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
```

IAM에서 역할을 생성합니다.

```
# 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
```

이 함수에 제공된 EventBridge 규칙 중에 지정된 규칙이 존재하는지 확인합니다.

```
# 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
```

호출자가 사용할 수 있는 규칙 중에 지정된 규칙이 존재하는지 확인합니다. 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
```

에서 EventBridge 규칙을 생성합니다.

```
# 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
```

Amazon Logs에서 호출자가 사용할 수 있는 그룹 중에 지정된 CloudWatch 로그 그룹이 존재하는지 확인하십시오.

```
# 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
```

CloudWatch Logs에서 로그 그룹을 생성합니다.

```
# 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
```

Logs의 로그 스트림에 이벤트를 CloudWatch 기록합니다.

```
# 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
```

Amazon Elastic Compute Cloud (Amazon EC2) 인스턴스를 다시 시작하고 관련 활동에 대한 정보를 Logs의 로그 스트림에 추가합니다. CloudWatch

```
# 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
```

규칙의 EventBridge 활동에 대한 정보를 표시합니다.

```
# 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
```

로그 로그 그룹의 모든 로그 스트림에 대한 CloudWatch 로그 정보를 표시합니다.

```
# 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
```

발신자에게 더 이상 필요하지 않은 관련 AWS 리소스를 수동으로 정리하라는 알림을 표시합니다.

```
# 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__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 다음 주제를 참조하십시오.
  - PutEvents
  - PutRule

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. EventBridge AWS SDK와 함께 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

SDK를 사용하여 EventBridge 규칙 및 대상으로 시작하세요. AWS

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 규칙을 생성하고 규칙에 대상을 추가합니다.
- 규칙을 활성화 및 비활성화합니다.
- 규칙과 대상을 나열하고 업데이트합니다.
- 이벤트를 전송하고 리소스를 정리합니다.

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
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;
    }
}
```

EventBridge 작업을 래핑하는 클래스를 만드세요.

```
/// <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;
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 주제를 참조하십시오.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * 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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하십시오.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules

- ListTargetsByRule
- PutEvents
- PutRule
- PutTargets

Kotlin

SDK for Kotlin

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/*
 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
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API reference의 다음 주제를 참조하세요.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules

- ListTargetsByRule
- PutEvents
- PutRule
- PutTargets

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오 EventBridge AWS SDK와 함께 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

SDK 사용을 위한 EventBridge 크로스 서비스 예제 AWS

다음 샘플 애플리케이션은 AWS SDK를 사용하여 다른 EventBridge 애플리케이션과 결합합니다. AWS 서비스각 예제에는 애플리케이션 설정 및 실행 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

예제

- 예약된 이벤트를 사용하여 Lambda 함수 호출

예약된 이벤트를 사용하여 Lambda 함수 호출

다음 코드 예제는 Amazon EventBridge 예약 이벤트에 의해 호출되는 AWS Lambda 함수를 생성하는 방법을 보여줍니다.

Java

SDK for Java 2.x

AWS Lambda 함수를 호출하는 Amazon EventBridge 예약 이벤트를 생성하는 방법을 보여 줍니다. cron 표현식을 사용하여 Lambda 함수가 호출되는 시기를 EventBridge 스케줄링하도록 구성합니다. 이 예제에서는 Lambda Java 런타임 API를 사용하여 Lambda 함수를 생성합니다. 이 예제는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 1주년 기념일에 직원에게 축하하는 모바일 문자 메시지를 전송하는 앱을 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 전체 예제를 참조하십시오. GitHub

이 예제에서 사용되는 서비스

- DynamoDB

- EventBridge
- Lambda
- Amazon SNS

JavaScript

JavaScript (v3) 용 SDK

AWS Lambda 함수를 호출하는 Amazon EventBridge 예약 이벤트를 생성하는 방법을 보여 줍니다. cron 표현식을 사용하여 Lambda 함수가 호출되는 시기를 EventBridge 스케줄링하도록 구성합니다. 이 예시에서는 Lambda 런타임 API를 사용하여 Lambda 함수를 생성합니다. JavaScript 이 예제는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 1주년 기념일에 직원에게 축하하는 모바일 문자 메시지를 전송하는 앱을 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 전체 예제를 참조하십시오. GitHub

이 예시는 AWS SDK for JavaScript v3 개발자 안내서에서도 확인할 수 있습니다.

이 예제에서 사용되는 서비스

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK for Python(Boto3)

이 예제에서는 AWS Lambda 함수를 예약된 Amazon EventBridge 이벤트의 대상으로 등록하는 방법을 보여줍니다. Lambda 핸들러는 나중에 검색할 수 있도록 CloudWatch Amazon Logs에 친숙한 메시지와 전체 이벤트 데이터를 기록합니다.

- Lambda 함수를 배포합니다.
- EventBridge 예약된 이벤트를 생성하고 Lambda 함수를 대상으로 설정합니다.
- Lambda EventBridge 함수를 호출할 수 있는 권한을 부여합니다.
- CloudWatch Logs의 최신 데이터를 인쇄하여 예약된 호출의 결과를 표시합니다.

- 데모 중에 생성된 모든 리소스를 정리합니다.

이 예시는 에서 가장 잘 보입니다. GitHub 전체 소스 코드와 설정 및 실행 방법에 대한 지침은 의 전체 예제를 참조하십시오 GitHub.

이 예시에서 사용되는 서비스

- CloudWatch 로그
- EventBridge
- Lambda

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오 EventBridge AWS SDK와 함께 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon EventBridge 보안

Amazon EventBridge 다른 AWS 서비스 및 리소스에 대한 액세스를 제어하는 데 사용합니다 AWS Identity and Access Management . IAM의 작동 방식에 대한 개요는 IAM 사용 설명서의 액세스 관리 개요를 참조하세요. 보안 인증 정보에 대한 개요는 Amazon Web Services 일반 참조의 AWS 보안 인증 정보를 참조하세요.

주제

아마존에서의 데이터 보호 EventBridge

AWS 공동 책임 모델 의 데이터 보호에 적용됩니다 Amazon EventBridge. 이 모델에 설명된 대로 AWS 는 모든 데이터를 실행하는 글로벌 인프라를 보호하는 역할을 AWS 클라우드합니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 AWS IAM Identity Center OR AWS Identity and Access Management (IAM) 을 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 멀티 팩터 인증 설정(MFA)을 사용하세요.
- SSL/TLS를 사용하여 리소스와 통신할 수 있습니다. AWS TLS 1.2는 필수이며 TLS 1.3를 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다. AWS CloudTrail
- 포함된 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용하십시오 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 AWS 통해 액세스할 때 FIPS 140-2로 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용하십시오. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 FIPS(Federal Information Processing Standard) 140-2를 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 콘솔, API EventBridge 또는 AWS 서비스 SDK를 사용하거나 다른 방법으로 작업하는 경우가 포함됩니다. AWS CLI AWS 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함해서는 안 됩니다.

EventBridge 이벤트 버스의 데이터 암호화

EventBridge 이벤트 데이터를 보호하기 위해 저장 중 암호화와 전송 중 암호화를 모두 제공합니다.

- 저장 중 암호화

EventBridge AWS Key Management Service (KMS) 와 통합하여 이벤트 버스에 저장된 이벤트 데이터를 암호화합니다. 기본적으로 EventBridge 는 AWS 소유 키 an을 사용하여 이벤트 데이터를 암호화합니다. 사용자 지정 및 파트너 이벤트에 대신 EventBridge 고객 관리형 키 for를 사용하도록 지정할 수도 있습니다.

- 전송 중 암호화

  EventBridge 전송 계층 보안 (TLS) EventBridge 을 사용하여 다른 서비스 간에 전달되는 데이터를 암호화합니다. 이벤트 버스의 경우 여기에는 이벤트가 전송되는 동안과 규칙 대상으로 이벤트를 EventBridge 보내는 시기가 포함됩니다. EventBridge

이벤트 버스의 미사용 암호화

EventBridge AWS Key Management Service (KMS) 와 통합하여 투명한 서버 측 암호화를 제공합니다. 저장 데이터를 기본적으로 암호화하면 민감한 데이터 보호와 관련된 운영 오버헤드와 복잡성을 줄이는 데 도움이 됩니다. 동시에 엄격한 암호화 규정 준수 및 규제 요구 사항을 충족하는 안전한 애플리케이션을 구축할 수 있습니다.

저장된 이벤트 버스 데이터 EventBridge 암호화에는 다음이 포함됩니다.

- 이벤트 데이터 AWS, 커스텀 이벤트, 파트너 이벤트.

  이벤트 버스의 경우 이벤트 데이터에는 이벤트 ??? 요소에 포함된 모든 필드가 포함됩니다.

  EventBridge 이벤트 메타데이터를 암호화하지 않습니다. 이벤트 메타데이터에 대한 자세한 내용은 을 참조하십시오???.
- 이벤트 패턴
- 입력 트랜스포머

기본적으로 AWS 소유 키 an을 EventBridge 사용하여 이벤트 데이터를 암호화합니다. 사용자 지정 및 파트너 이벤트에 대신 EventBridge 고객 관리형 키 for를 사용하도록 지정할 수도 있습니다.

이벤트 버스 암호화에 대한 보안 고려 사항

기밀 또는 민감한 정보는 유휴 시 암호화되지 않으므로 다음 필드에 입력하지 않는 것이 좋습니다.

- 이벤트 버스 이름
- 규칙 이름

- 공유 리소스 (예: 태그)

KMS key 이벤트 버스 암호화 옵션

EventBridge AWS 소유 키 an을 사용하여 이벤트 버스에 저장된 AWS 서비스 이벤트를 암호화합니다.

각 이벤트 버스에 대해 해당 버스에 저장된 사용자 지정 및 파트너 이벤트를 암호화하는 데 사용할 유형을 선택할 수 있습니다. KMS key EventBridge

- AWS 소유 키

  기본적으로, EventBridge 무단 액세스로부터 데이터를 보호하는 데 도움이 되는 256비트 고급 암호화 표준 (AES-256) 을 사용하여 데이터를 암호화합니다. AWS 소유 키

  데이터 사용을 AWS 소유 키보거나, 관리하거나, 사용하거나 감사할 수 없습니다. 하지만 데이터를 암호화하는 키를 보호하기 위해 어떤 작업을 수행하거나 어떤 프로그램을 변경할 필요가 없습니다.

  일반적으로 리소스를 보호하는 암호화 키를 감사하거나 제어해야 하는 경우가 아니라면 이 방법을 AWS 소유 키 사용하는 것이 좋습니다. AWS 소유 키 완전히 무료이며 (월 사용료 또는 사용료 없음) 계정 AWS KMS 할당량에 포함되지 않습니다. 키 또는 키 정책을 만들거나 유지하지 않아도 됩니다.

  자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS 소유 키를 참조하십시오.
- 고객 관리형 키

  EventBridge 직접 만들고, 소유하고, 관리하는 고객 관리형 키 대칭을 사용할 수 있습니다. 이 유형을 완전히 제어할 수 있으므로 다음과 같은 작업을 수행할 수 있습니다. KMS key

  - 키 정책 수립 및 유지
  - IAM 정책 및 권한 수립 및 유지
  - 키 정책 활성화 및 비활성화
  - 키 암호화 자료 교체
  - 태그 추가
  - 키 별칭 생성
  - 삭제를 위한 스케줄 키

  자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 관리형 키을 참조하세요.

  EventBridge 다중 지역 키 및 키에 대한 계정 간 액세스를 지원합니다.

고객 관리형 키 월 사용료가 발생합니다. 자세한 내용은 개발자 AWS Key Management Service 안내서의 가격 및 할당량을 참조하십시오.AWS Key Management Service

> **Note**
>
> EventBridge 를 사용하여 암호화된 이벤트 버스에서 다음 기능을 지원하지 않습니다. 고객 관리형 키
>
> - 아카이브
> - 스키마 검색

자세한 내용은 ??? 섹션을 참조하세요.

를 사용하여 이벤트를 암호화합니다. 고객 관리형 키

AWS 소유 키 as를 기본값으로 EventBridge 사용하는 대신 a를 사용하여 이벤트 버스에 저장된 데이터 (커스텀 및 파트너 이벤트) 를 AWS KMS 고객 관리형 키 암호화하도록 지정할 수 있습니다. 이벤트 버스를 생성하거나 업데이트할 고객 관리형 키 때 를 지정할 수 있습니다. 사용자 지정 및 파트너 이벤트에도 사용하도록 기본 이벤트 버스를 업데이트할 수도 있습니다. 고객 관리형 키 자세한 정보는 ???을 참조하세요.

이벤트 고객 관리형 키 버스에 a를 지정하는 경우 이벤트 버스의 데드레터 큐 (DLQ) 를 지정할 수 있습니다. EventBridge 그런 다음 암호화 또는 암호 해독 오류를 생성하는 모든 사용자 지정 또는 파트너 이벤트를 해당 DLQ에 전달합니다. 자세한 정보는 ???을 참조하세요.

이벤트 버스를 생성할 때 암호화를 고객 관리형 키 위해 a 지정 (콘솔 사용)

- 다음 지침들을 따릅니다.

  ???.

이벤트 버스 생성 시 암호화를 고객 관리형 키 위한 a 지정 (CLI 사용)

- `create-event-bus`호출할 때는 `kms-key-identifier` 옵션을 사용하여 이벤트 버스의 암호화에 사용할 형식을 지정합니다. 고객 관리형 키 EventBridge

  선택적으로 데드레터 큐 (DLQ) 를 지정하는 `dead-letter-config` 데 사용합니다.

암호화에 고객 관리형 키 a를 사용하도록 이벤트 버스 업데이트 (콘솔 사용)

- 다음 지침들을 따릅니다.

  ???.

암호화에 고객 관리형 키 a를 사용하도록 이벤트 버스 업데이트 (CLI 사용)

- `update-event-bus`호출할 때는 `kms-key-identifier` 옵션을 사용하여 이벤트 버스의 고객 관리형 키 암호화에 사용할 양식을 지정하십시오. EventBridge

  선택적으로 데드레터 큐 (DLQ) 를 지정하는 `dead-letter-config` 데 사용합니다.

를 사용하여 암호화에 고객 관리형 키 a를 사용하도록 기본 이벤트 버스 업데이트 CloudFormation

기본 이벤트 버스가 계정에 자동으로 EventBridge 프로비저닝되므로 CloudFormation 스택에 포함하려는 리소스에 대해 일반적으로 사용하는 것처럼 CloudFormation 템플릿을 사용하여 기본 이벤트 버스를 생성할 수 없습니다. 스택에 기본 이벤트 버스를 포함하려면 먼저 CloudFormation 스택으로 기본 이벤트 버스를 가져와야 합니다. 기본 이벤트 버스를 스택으로 가져온 후에는 필요에 따라 이벤트 버스 속성을 업데이트할 수 있습니다.

- 다음 지침들을 따릅니다.

  ???.

사용 EventBridge 권한 부여 고객 관리형 키

고객 관리형 키 계정에서 EventBridge 이벤트 버스를 보호하는 데 사용하는 경우 해당 정책에 따라 사용자를 대신하여 사용할 수 있는 EventBridge 권한을 KMS key 부여해야 합니다. 키 정책에서 이러한 권한을 제공합니다.

EventBridge 기본값을 사용하여 AWS 계정의 EventBridge 리소스를 보호하는 AWS 소유 키 데는 추가 승인이 필요하지 않습니다.

EventBridge a에는 다음과 같은 권한이 필요합니다 고객 관리형 키.

- `kms:DescribeKey`

  EventBridge 제공된 키 ID에 대한 KMS key ARN을 검색하고 키가 대칭인지 확인하려면 이 권한이 필요합니다.

- kms:GenerateDataKey

  EventBridge 이벤트 데이터의 암호화 키로 데이터 키를 생성하려면 이 권한이 필요합니다.

- kms:Decrypt

  EventBridge 암호화된 이벤트 데이터와 함께 암호화되고 저장된 데이터 키를 해독하려면 이 권한이 필요합니다.

  EventBridge 이를 규칙 일치에 사용하므로 사용자는 데이터에 액세스할 수 없습니다.

다음 예제 키 정책은 필요한 권한을 제공합니다.

```
{
  "Sid": "Allow EventBridge to encrypt events",
  "Effect": "Allow",
  "Principal": {
    "Service": "events.amazonaws.com"
  },
  "Action": [
    "kms:DescribeKey",
    "kms:GenerateDataKey",
    "kms:Decrypt"
  ]
  "Resource": "*",
  "Condition": {
    "StringEquals": {
        "kms:EncryptionContext:aws:events:event-bus:arn":
 "arn:aws:events:region:account-id:event-bus/event-bus-arn",
        "aws:SourceArn": "arn:aws:events:region:account-id:event-bus/event-bus-name"
    }
  }
}
```

EventBridge 이벤트 버스 고객 관리형 키 암호화에 사용할 때의 보안

보안 모범 사례로 aws:SourceArnaws:sourceAccount, 또는 kms:EncryptionContext:aws:events:event-bus:arn 조건 키를 키 정책에 추가하십시오. AWS KMS IAM 글로벌 조건 키는 지정된 버스 또는 계정에만 KMS 키를 EventBridge 사용하도록 하는 데 도움이 됩니다.

다음 예는 정책에서 이 모범 사례를 따르는 방법을 보여줍니다. IAM

```
{
      "Sid": "Allow the use of key",
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "*",
      "Condition" : {
        "StringEquals": {
          "aws:SourceAccount": "arn:aws:events:region:account-id",
          "aws:SourceArn": "arn:aws:events:region:account-id:event-bus/event-bus-name",
          "kms:EncryptionContext:aws:events:event-bus:arn":
 "arn:aws:events:region:account-id:event-bus/event-bus-arn"
        }
      }
```

EventBridge 이벤트 버스 암호화 관리 고객 관리형 키

필요한 고객 관리형 키정보에 EventBridge 항상 액세스할 수 있도록 하려면:

- a로 암호화된 모든 이벤트가 처리되었는지 확인하기 고객 관리형 키 전에는 a를 삭제하지 마십시오.

  다음 작업을 수행할 때는 이전에 암호화된 이벤트에 계속 사용할 EventBridge 수 있도록 이전 키 자료를 보관해 두십시오.

  - 자동 키 교체
  - 수동 키 로테이션
  - 키 별칭 업데이트

  일반적으로 키를 삭제하려는 경우 먼저 AWS KMS 키를 비활성화하고 암호화된 데이터를 해독하는 데 키를 사용할 필요가 없도록 CloudWatch 경보 또는 유사한 메커니즘을 설정하십시오.
- 키 사용 권한을 제공하는 EventBridge 키 정책을 삭제하지 마십시오.

기타 고려 사항은 다음과 같습니다.

- 고객 관리형 키 규칙 대상을 적절하게 지정하십시오.

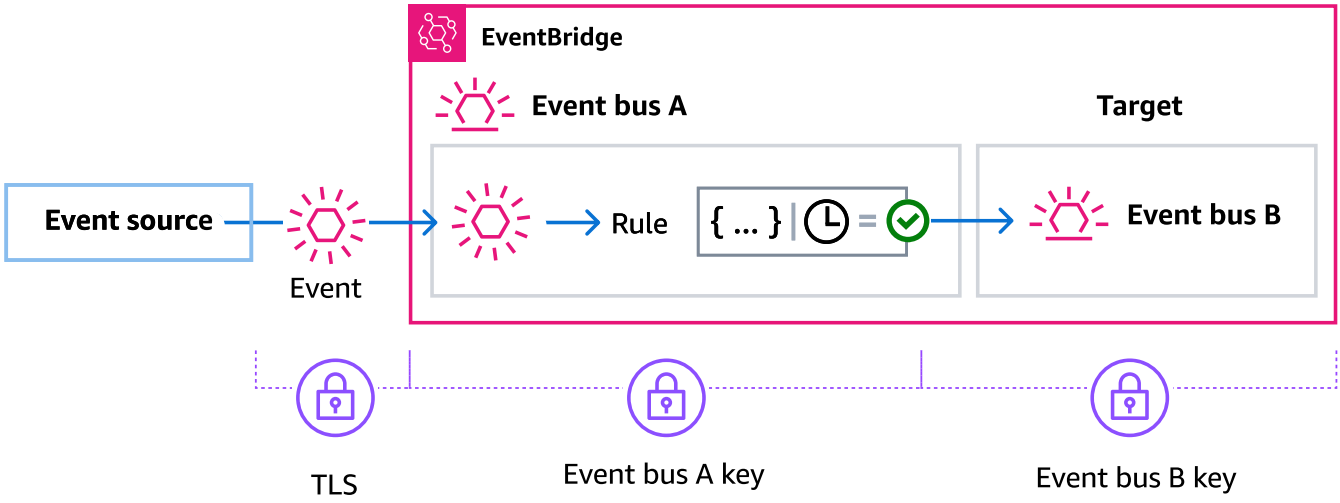
규칙 대상으로 이벤트를 EventBridge 전송할 때 이벤트는 전송 계층 보안 (TLS) 을 사용하여 전송됩니다. 하지만 이벤트가 대상에 저장될 때 적용되는 암호화는 대상 자체에 구성한 암호화에 따라 달라집니다.

이벤트 암호화: 이벤트 버스가 규칙 대상인 경우의 이벤트 암호화

사용자 지정 또는 파트너 이벤트가 이벤트 버스로 전송되면 해당 이벤트 버스의 EventBridge 저장 중 암호화 KMS 키 구성 (기본값 AWS 소유 키 또는 지정된 경우) 에 따라 해당 이벤트를 암호화합니다. 고객 관리형 키이벤트가 규칙과 일치하는 경우, 규칙 대상이 다른 이벤트 버스가 아닌 한 이벤트가 규칙 대상으로 전송될 때까지 해당 이벤트 버스의 KMS 키 구성을 사용하여 이벤트를 EventBridge 암호화합니다.

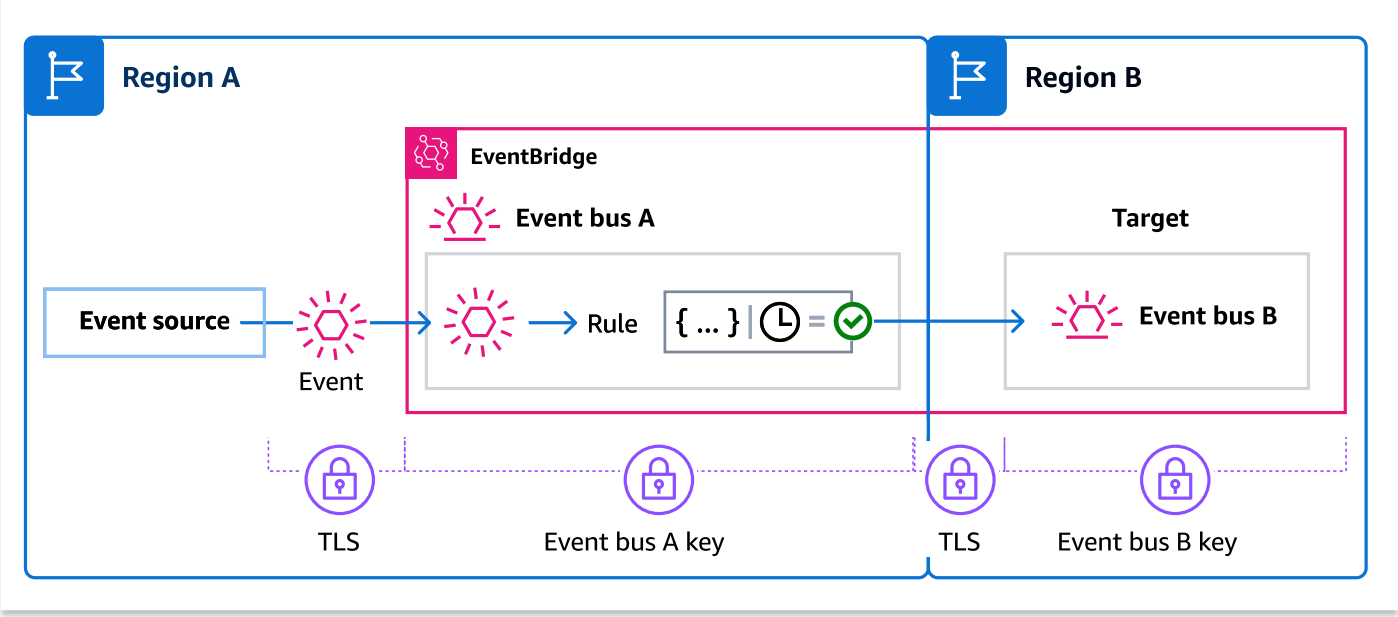
- 규칙의 대상이 같은 AWS 지역의 다른 이벤트 버스인 경우:

  대상 이벤트 버스에 지정된 고객 관리형 키이벤트 버스가 있는 경우 대신 대상 이벤트 고객 관리형 키 버스의 이벤트를 EventBridge 암호화하여 전달합니다.



- 규칙의 대상이 다른 AWS 지역의 다른 이벤트 버스인 경우:

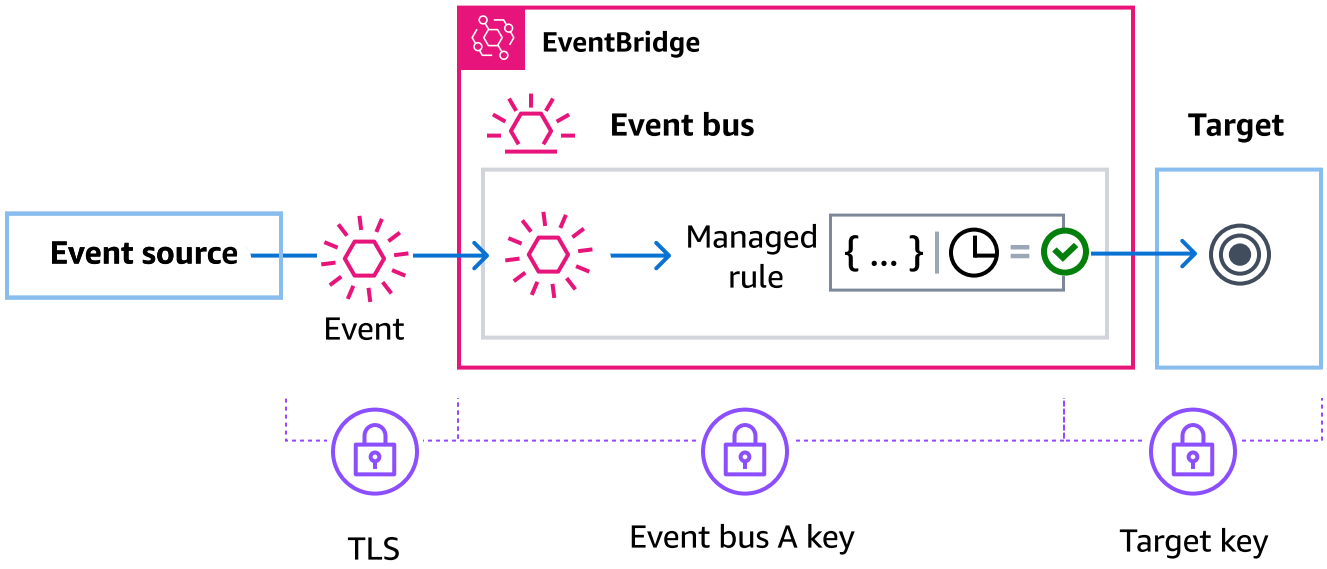
  EventBridge 첫 번째 이벤트 버스의 KMS 키 구성에 따라 미사용 이벤트를 암호화합니다. EventBridge TLS를 사용하여 다른 지역의 두 번째 이벤트 버스로 이벤트를 보낸 다음 대상 이벤트 버스에 지정된 KMS 키 구성에 따라 암호화됩니다.



관리형 규칙의 이벤트 암호화

AWS 서비스는 해당 서비스의 특정 기능에 필요한 이벤트 버스 규칙을 AWS 계정에서 생성하고 관리할 수 있습니다. 관리형 규칙의 일부로 AWS 서비스는 규칙 대상에 고객 관리형 키 지정된 규칙을 EventBridge 사용하도록 지정할 수 있습니다. 이렇게 하면 규칙 대상을 기반으로 사용할 항목을 고객 관리형 키 유연하게 지정할 수 있습니다.

이러한 경우 사용자 지정 또는 파트너 이벤트가 관리형 규칙과 일치하면 관리형 규칙에 고객 관리형 키 지정된 대상을 EventBridge 사용하여 규칙 대상으로 전송될 때까지 이벤트를 암호화합니다. 이는 이벤트 버스가 암호화에 자체 고객 관리형 키 이벤트 버스를 사용하도록 구성되었는지 여부와 무관합니다. 이는 관리형 규칙의 대상이 다른 이벤트 버스이고 해당 이벤트 버스에 암호화를 위한 자체 이벤트 버스가 고객 관리형 키 지정된 경우에도 마찬가지입니다. EventBridge 이벤트가 이벤트 버스가 아닌 대상으로 전송될 때까지 관리 규칙에 고객 관리형 키 지정된 대상을 계속 사용합니다.



규칙 대상이 다른 지역의 이벤트 버스인 경우 다중 지역 키를 제공해야 합니다. 첫 번째 지역의 이벤트 버스는 관리형 규칙에 고객 관리형 키 지정된 것을 사용하여 이벤트를 암호화합니다. 그런 다음 두 번째 지역의 대상 이벤트 버스로 이벤트를 전송합니다. 해당 이벤트 버스는 이벤트를 타겟으로 전송할 고객 관리형 키 때까지 를 계속 사용할 수 있어야 합니다.

EventBridge 이벤트 버스 암호화 컨텍스트

암호화 컨텍스트는 보안되지 않은 임의의 데이터를 포함하는 키-값 페어 세트입니다. 데이터 암호화 요청에 암호화 컨텍스트를 포함하는 경우 AWS KMS 는 암호화된 데이터에 암호화 컨텍스트를 암호 방식으로 바인딩합니다. 따라서 동일한 암호화 컨텍스트로 전달해야 이 데이터를 해독할 수 있습니다.

또한 암호화 컨텍스트를 정책 및 권한 부여의 권한 부여 조건으로 사용할 수 있습니다.

이벤트 버스의 경우 모든 AWS KMS 암호화 작업에서 동일한 암호화 컨텍스트를 EventBridge 사용합니다. 고객 관리 키를 사용하여 EventBridge 리소스를 보호하는 경우 암호화 컨텍스트를 사용하여 감사 레코드 및 KMS key 로그에서의 사용을 식별할 수 있습니다. 또한 AWS CloudTrail 및 Amazon CloudWatch Logs 같은 로그에서 일반 텍스트에 나타나기도 합니다.

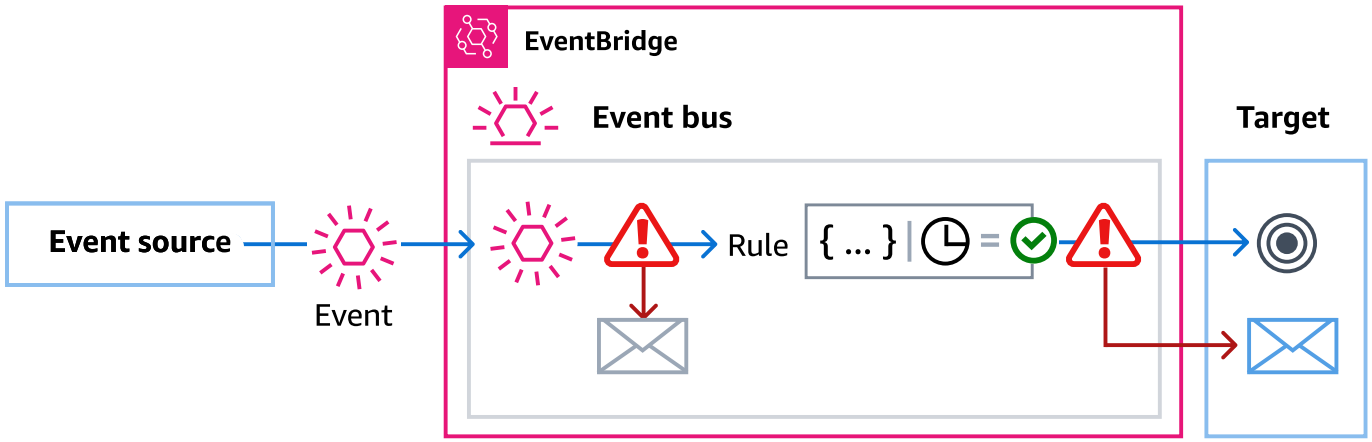
요청에서 는 이벤트 AWS KMS버스 ARN을 포함하는 단일 키-값 쌍을 사용한 암호화 컨텍스트를 EventBridge 사용합니다.

```
"encryptionContext": {
    "kms:EncryptionContext:aws:events:event-bus:arn": "event-bus-arn"
}
```

데드레터 대기열을 사용하여 암호화된 이벤트 오류를 캡처합니다.

이벤트 버스에 고객 관리형 키 암호화를 구성하는 경우 해당 이벤트 버스에 DLQ (데드레터 큐) 를 지정하는 것이 좋습니다. EventBridge 이벤트 버스에서 이벤트를 처리하는 동안 복구할 수 없는 오류가 발생하는 경우 사용자 지정 및 파트너 이벤트를 이 DLQ로 보냅니다. 복구할 수 없는 오류는 지정된 오류가 비활성화되거나 누락되는 등 근본적인 문제를 해결하기 위해 사용자 작업이 필요한 오류입니다. 고객 관리형 키

- 이벤트 버스에서 이벤트를 처리하는 동안 EventBridge 복구할 수 없는 암호화 또는 암호 해독 오류가 발생하는 경우 해당 이벤트는 이벤트 버스의 DLQ (지정된 경우) 로 전송됩니다.
- 대상으로 이벤트를 보내려고 시도하는 동안 EventBridge 복구할 수 없는 암호화 또는 암호 해독 오류가 발생하면 이벤트는 대상의 DLQ (지정된 경우) 로 전송됩니다.



DLQ 사용 시 고려 사항 및 권한 설정 지침을 비롯한 자세한 내용은 을 참조하십시오. ???

데드레터 대기열의 이벤트 암호 해독 EventBridge

복구할 수 없는 오류를 일으키는 근본적인 문제를 해결했으면 이벤트 버스 또는 대상 DLQ로 전송된 이벤트를 처리할 수 있습니다. 암호화된 이벤트의 경우 이벤트를 처리하려면 먼저 이벤트 암호를 해독해야 합니다.

다음 예제는 이벤트 버스 또는 대상 DLQ에 EventBridge 전달된 이벤트를 해독하는 방법을 보여줍니다.

```
// You will receive an encrypted event in the following json format.
        // ```
        //   {
        //     "version": "0",
```

```
        //     "id": "053afa53-cdd7-285b-e754-b0dfd0ac0bfb",  // New event id not the
 same as the original one
        //     "account": "123456789012",
        //     "time": "2020-02-10T10:22:00Z",
        //     "resources": [ ],
        //     "region": "us-east-1",
        //     "source": "aws.events",
        //     "detail-type": "Encrypted Events",
        //     "detail": {
        //       "event-bus-arn": "arn:aws:events:region:account:event-bus/bus-name",
        //       "rule-arn": "arn:aws:events:region:account:event-bus/bus-name/rule-
name",
        //       "kms-key-arn": "arn:aws:kms:region:account:key/key-arn",
        //       "encrypted-payload": "AgR4qiru/XNwTUyCgRHqP7rbbHn/
xpmVeVeRIAd12TDYYVwAawABABRhd3M6ZXZlbnRzOmV2ZW50LWJ1cwB
        //
 RYXJuOmF3czpldmVudHM6dXMtZWFzdC0xOjE0NjY4NjkwNDY3MzpldmVudC1idXMvY21rbXMtZ2EtY3Jvc3
        //
 MtYWNjb3VudC1zb3VyY2UtYnVzAAEAB2F3cy1rbXMAS2Fybjphd3M6a21zOnVzLWVhc3QtMToxNDY2ODY5"
        //     }
        //   }
        // ```

        // Construct an AwsCrypto object with the encryption algorithm
 `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` which
        // is used by EventBridge for encryption operation. This object is an entry
 point for decryption operation.
        // It can later use decryptData(MasterKeyProvider, byte[]) method to decrypt
 data.
        final AwsCrypto crypto = AwsCrypto.builder()

 .withEncryptionAlgorithm(CryptoAlgorithm.ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY)
                .build();

        // Construct AWS KMS master key provider with AWS KMS Client Supplier and AWS
 KMS Key ARN. The KMS Client Supplier can
        // implement a RegionalClientSupplier interface. The AWS KMS Key ARN can be
 fetched from kms-key-arn property in
        // encrypted event json detail.
        final KmsMasterKeyProvider kmsMasterKeyProvider =
 KmsMasterKeyProvider.builder()
                .customRegionalClientSupplier(...)
                .buildStrict(KMS_KEY_ARN);
```

```
        // The string of encrypted-payload is base64 encoded. Decode it into byte
 array, so it can be furthur
        // decrypted. The encrypted payload can be fetched from encrypted-payload field
 in encrypted event json detail.
        byte[] encryptedByteArray = Base64.getDecoder().decode(ENCRYPTED_PAYLOAD);

        // The decryption operation. It retrieves the encryption context and encrypted
 data key from the cipher
        // text headers, which is parsed from byte array encrypted data. Then it
 decrypts the data key, and
        // uses it to finally decrypt event payload. This encryption/decryption
 strategy is called envelope
        // encryption, https://docs.aws.amazon.com/kms/latest/developerguide/
concepts.html#enveloping
        final CryptoResult<byte[], KmsMasterKey> decryptResult =
 crypto.decryptData(kmsMasterKeyProvider, encryptedByteArray);

        final byte[] decryptedByteArray = decryptResult.getResult();

        // Decode the event json plaintext from byte array into string with UTF_8
 standard.
        String eventJson = new String(decryptedByteArray, StandardCharsets.UTF_8);
```

태그 기반 정책

Amazon EventBridge에서는 태그를 기반으로 하는 정책을 사용하여 리소스에 대한 액세스를 제어할 수 있습니다.

예를 들어 `environment` 키 및 `production` 값이 있는 태그가 포함된 리소스에 대한 액세스를 제한할 수 있습니다. 다음 예시 정책은 이 태그가 있는 리소스가 `environment/production` 태그가 지정된 리소스에 대한 태그, 규칙 또는 이벤트 버스를 생성, 삭제 또는 수정하는 기능을 거부합니다.

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

태그 지정에 대한 자세한 내용은 다음을 참조하십시오.

- 아마존 EventBridge 태그
- IAM 태그를 사용한 액세스 제어

아마존 EventBridge 및 AWS Identity and Access Management

EventBridgeAmazon에 액세스하려면 요청을 인증하는 데 사용할 AWS 수 있는 자격 증명이 필요합니다. 이 자격 증명에는 다른 AWS 리소스에서의 이벤트 데이터 검색과 같이 AWS 리소스에 액세스할 수 있는 권한이 포함되어야 합니다. 다음 섹션에서는 사용 방법 AWS Identity and Access Management(IAM) 에 대한 세부 정보를 제공하고 리소스에 액세스할 수 있는 사용자를 제어하여 리소스를 보호하는 EventBridge 데 도움이 됩니다.

주제

- 인증
- 액세스 제어
- Amazon EventBridge 리소스에 대한 액세스 권한 관리
- Amazon의 자격 증명 기반 정책 (IAM 정책) 사용 EventBridge
- Amazon EventBridge에 리소스 기반 정책 사용
- 교차 서비스 혼동된 대리자 예방
- Amazon EventBridge 스키마에 대한 리소스 기반 정책
- Amazon EventBridge 권한 참조
- IAM 정책 조건을 사용하여 세분화된 액세스 제어 구현
- EventBridge에 서비스 연결 역할 사용

인증

다음과 같은 유형의 자격 증명으로 AWS에 액세스할 수 있습니다.

- AWS 계정 루트 사용자 – AWS에 가입할 때 계정과 연결된 이메일 주소 및 암호를 지정합니다. 이 두 가지가 루트 보안 인증 정보이며 모든 AWS 리소스에 대한 전체 액세스 권한을 제공합니다.

> **Important**
>
> 보안상 관리자, 즉 계정에 대한 전체 권한이 있는 IAM 사용자를 만들 때에만 루트 자격 증명을 사용하는 것이 좋습니다. 그러면 이 관리자를 사용하여 제한된 권한이 있는 다른 사용자 및 역할을 만들 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 모범 사례 및 관리자 사용자 및 그룹 생성 단원을 참조하세요.

- IAM 사용자 - IAM 사용자는 특정 권한 (예: 이벤트 데이터를 대상으로 전송할 수 있는 권한) 을 가진 계정 내의 자격 증명입니다. EventBridge IAM 로그인 보안 인증 정보를 사용하여 보안 AWS 웹 페이지(예: AWS Management Console, AWS 토론 포럼, AWS Support 센터)에 로그인할 수 있습니다.

  로그인 보안 인증 정보 외에도 각 사용자마다 액세스 키를 생성할 수도 있습니다. 여러 SDK 중 하나를 통해 또는 AWS Command Line Interface(AWS CLI)를 사용하여 AWS 서비스에 프로그래밍 방식으로 액세스하여 요청에 암호화 방식으로 서명할 때 이러한 키를 사용할 수 있습니다. AWS 도구를 사용하지 않으면 인바운드 API 요청을 인증하기 위한 프로토콜인 서명 버전 4를 사용하여 요청에 직접 서명해야 합니다. 요청 인증에 대한 자세한 내용은 Amazon Web Services 일반 참조의 서명 버전 4 서명 프로세스를 참조하십시오.

- IAM 역할 – IAM 역할은 계정에 만들 수 있는, 특정 권한을 지닌 또 하나의 IAM 자격 증명입니다. 이 역할은 IAM 사용자와 유사하지만 특정 개인과 연결되지 않습니다. IAM 역할을 사용하면 AWS 서비스 및 리소스에 액세스하기 위한 임시 액세스 키를 얻을 수 있습니다. 임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

  - 페더레이션 사용자 액세스 – 사용자를 생성하는 대신 AWS Directory Service, 엔터프라이즈 사용자 디렉터리 또는 웹 ID 제공업체(IdP)의 자격 증명을 사용할 수 있습니다. 이 사용자를 페더레이션 사용자라고 합니다. AWS에서는 사용자가 ID 제공업체를 통해 액세스를 요청하면 페더레이션 사용자에게 역할을 할당합니다. 페더레이션 사용자에 대한 자세한 정보는 IAM 사용 설명서의 페더레이션 사용자 및 역할을 참조하세요.
  - 크로스 계정 액세스 – 계정의 IAM 역할을 사용하여 계정의 리소스에 액세스할 수 있는 권한을 다른 계정에 부여할 수 있습니다. 사용 예는 IAM 사용 설명서의 자습서: IAM 역할을 사용한 AWS 계정 간 액세스 권한 위임 단원을 참조하세요.
  - AWS 서비스 액세스 – 계정의 IAM 역할을 사용하여 AWS 서비스에 계정의 리소스에 액세스할 권한을 부여할 수 있습니다. 예를 들어, Amazon Redshift가 Amazon S3 버킷에 저장된 데이터를 Amazon Redshift 클러스터로 로드하도록 허용하는 역할을 생성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성 단원을 참조하세요.
  - Amazon EC2에서 실행되는 애플리케이션 — EventBridge 액세스가 필요한 Amazon EC2 애플리케이션의 경우, EC2 인스턴스에 액세스 키를 저장하거나 IAM 역할을 사용하여 임시 자격 증명을 관리할 수 있습니다. EC2 인스턴스에 AWS 역할을 할당하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 애플리케이션에 임시 자격 증명을 제공합니다. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html자세한 정보는 IAM 사용 설명서의 Amazon EC2에서 실행되는 애플리케이션의 역할 사용하기를 참조하세요.

액세스 제어

EventBridge 리소스를 생성하거나 액세스하려면 유효한 자격 증명과 권한이 모두 필요합니다. 예를 들어, AWS Lambda, Amazon Simple Notification Service(SNS) 및 Amazon Simple Queue Service(Amazon SQS) 대상을 간접적으로 간접 호출하려면 해당 서비스에 대한 권한이 있어야 합니다.

Amazon EventBridge 리소스에 대한 액세스 권한 관리

자격 증명 기반 또는 리소스 기반 정책을 사용하여 규칙이나 이벤트와 같은 EventBridge 리소스에 대한 액세스를 관리합니다.

EventBridge 리소스

EventBridge 리소스와 하위 리소스에는 고유한 Amazon 리소스 이름(ARN)이 연결됩니다. EventBridge의 ARN을 사용하여 이벤트 패턴을 생성합니다. ARN에 대한 자세한 내용은 Amazon Web Services 일반 참조의 Amazon 리소스 이름(ARN) 및 AWS 서비스 네임스페이스를 참조하세요.

EventBridge에서 리소스 작업을 위해 제공하는 작업 목록은 Amazon EventBridge 권한 참조 단원을 참조하십시오.

> **Note**
>
> 대부분의 AWS 서비스는 콜론(:) 또는 슬래시(/)를 ARN에서 동일한 문자로 처리합니다. 그러나 EventBridge는 이벤트 패턴 및 규칙에서 정확히 일치해야 합니다. 따라서 이벤트 패턴을 만들 때 일치시키려는 이벤트에서 ARN 구문이 일치하도록 정확한 ARN 문자를 사용해야 합니다.

다음 표에는 EventBridge 리소스가 나와 있습니다.

리소스 유형	ARN 형식
아카이브	`arn:aws:events:` *`region`*`:`*`account`*`:archive/` *`archive-name`*
다시 재생	`arn:aws:events:` *`region`*`:`*`account`*`:replay/`*`replay-name`*
규칙	`arn:aws:events:` *`region`*`:`*`account`*`:rule/`*`[event-bus-name]/rule-name`*
이벤트 버스	`arn:aws:events:` *`region`*`:`*`account`*`:event-bus/` *`event-bus-name`*
모든 EventBridge 리소스	`arn:aws:events:*`

리소스 유형	ARN 형식
지정한 리전에서 지정한 계정이 소유한 모든 EventBridge 리소스	`arn:aws:events: region:account:*`

다음 예에서는 ARN을 사용하여 문에서 특정 규칙(*myRule*)을 나타내는 방법을 보여줍니다.

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

특정 계정에 속하는 모든 규칙을 지정하려면 다음과 같이 별표(*) 와일드카드를 사용합니다.

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

모든 리소스를 지정해야 하거나 특정 API 작업이 ARN을 지원하지 않는 경우 다음과 같이 `Resource` 요소에 별표(*) 와일드카드를 사용합니다.

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

명령문 하나에 여러 리소스 또는 `PutTargets`을 지정하려면 다음과 같이 각 ARN을 쉼표로 구분합니다.

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

리소스 소유권

계정은 리소스를 누가 생성했든 상관없이 계정의 리소스를 소유합니다. 리소스 소유자는 보안 주체 엔터티의 계정, 계정 루트 사용자, 리소스 생성 요청을 인증하는 IAM 사용자 또는 역할입니다. 다음 예에서는 이 계정의 작동 방식을 설명합니다.

- 계정의 루트 사용자 자격 증명을 사용하여 규칙을 생성하면, 계정이 EventBridge 리소스의 소유자가 됩니다.
- 계정에 사용자를 생성하고 이 사용자에게 EventBridge 리소스를 생성할 수 있는 권한을 부여하면 해당 사용자는 EventBridge 리소스를 생성할 수 있습니다. 하지만 EventBridge 리소스는 사용자가 속한 계정이 소유합니다.

- 계정에서 EventBridge 리소스를 생성할 권한이 있는 IAM 역할을 생성하는 경우, 해당 역할을 수임할 수 있는 사람은 누구나 EventBridge 리소스를 생성할 수 있습니다. 이 경우 역할이 속한 계정이 EventBridge 리소스를 소유합니다.

리소스 액세스 관리

권한 정책은 누가 무엇에 액세스할 수 있는지를 나타냅니다. 다음 섹션에서는 권한 정책을 만드는 데 사용 가능한 옵션에 대해 설명합니다.

> **Note**
>
> 이 섹션에서는 EventBridge의 맥락에서 IAM을 사용하는 방법에 대해 설명하며, IAM 서비스에 대한 자세한 정보는 다루지 않습니다. IAM 설명서 전체 내용은 IAM 사용 설명서의 IAM이란 무엇인가요? 단원을 참조하세요. IAM 정책 구문 및 설명에 대한 자세한 내용은 IAM 사용 설명서의 IAM 정책 참조 단원을 참조하세요.

IAM 자격 증명에 연결된 정책을 자격 증명 기반 정책(IAM 정책)이라 하고, 리소스에 연결된 정책을 리소스 기반 정책이라고 합니다. EventBridge에서는 자격 증명 기반(IAM 정책)과 리소스 기반 정책을 모두 사용할 수 있습니다.

주제

- 자격 증명 기반 정책(IAM 정책)
- 리소스 기반 정책(IAM 정책)

자격 증명 기반 정책(IAM 정책)

정책을 IAM 자격 증명에 연결할 수 있습니다. 예를 들면, 다음을 수행할 수 있습니다.

- 계정 내 사용자 또는 그룹에 권한 정책 연결 – 사용자에게 Amazon CloudWatch 콘솔에서 규칙을 볼 수 있는 권한을 부여하려면 사용자가 속한 사용자 또는 그룹에 권한 정책을 연결하면 됩니다.
- 역할에 권한 정책 연결(교차 계정 권한 부여) – 자격 증명 기반 권한 정책을 IAM 역할에 연결하여 교차 계정 권한을 부여할 수 있습니다. 예를 들어 계정 A의 관리자는 다음과 같이 역할을 생성하여 다른 (예: 계정 B) 또는 AWS 서비스에 교차 계정 권한을 부여할 수 있습니다.
  1. 계정 A 관리자는 IAM 역할을 생성하고 계정 A의 리소스에 대한 권한을 부여하는 권한 정책을 역할에 연결합니다.
  2. 계정 A 관리자는 계정 B를 역할을 수임할 보안 주체로 식별하는 역할에 신뢰 정책을 연결합니다.

3. 그런 다음 계정 B 관리자가 계정 B의 사용자에게 역할을 수임할 권한을 위임할 수 있습니다. 그러면 계정 B의 사용자가 계정 에서 리소스를 생성하거나 액세스할 수 있습니다. 신뢰 정책의 보안 주체가 AWS 제품 보안 주체가 되어 역할을 수임하는 데 필요한 권한을 AWS 제품에 부여할 수도 있습니다.

   IAM을 사용하여 권한을 위임하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 액세스 관리 섹션을 참조하세요.

특정 IAM 정책을 생성하여 계정의 사용자가 액세스할 수 있는 직접적 호출 및 리소스를 제한한 다음, 해당 정책을 사용자에게 연결합니다. IAM 역할 생성 방법 및 EventBridge용 IAM 정책 명령문 예제를 살펴보는 방법에 대한 자세한 내용은 Amazon EventBridge 리소스에 대한 액세스 권한 관리 단원을 참조하세요.

리소스 기반 정책(IAM 정책)

EventBridge에서 규칙이 실행되면 규칙과 관련된 모든 대상이 간접 호출됩니다. 즉, 함수를 간접적으로 간접 호출하거나, Amazon SNS 주제에 게시하거나, 이벤트를 Amazon Kinesis 스트림으로 중계합니다. 소유하고 있는 리소스에 대해 API 직접 호출을 수행하려면 EventBridge가 해당되는 권한을 가지고 있어야 합니다. Lambda, Amazon SNS 및 Amazon SQS 리소스의 경우 EventBridge는 리소스 기반 정책을 사용합니다. Kinesis 스트림의 경우 EventBridge는 IAM 역할을 사용합니다.

IAM 역할을 생성하는 방법과 EventBridge에 대한 리소스 기반 정책 문 예시를 살펴보는 방법에 대한 자세한 내용은 Amazon EventBridge에 리소스 기반 정책 사용 단원을 참조하십시오.

정책 요소 지정: 작업, 효과, 보안 주체

각 EventBridge 리소스에 대해 EventBridge는 일련의 API 작업을 정의합니다. 이러한 API 작업에 대한 권한을 부여하기 위해 EventBridge에서는 정책에서 지정할 수 있는 작업을 정의합니다. 일부 API 작업에서는 API 작업을 수행하기 위해 복수의 작업에 대한 권한이 필요합니다. 리소스 및 API 작업에 대한 자세한 내용은 EventBridge 리소스 및 Amazon EventBridge 권한 참조 섹션을 참조하세요.

다음은 기본 정책 요소입니다.

- Resource – Amazon 리소스 이름(ARN)을 사용하여 정책을 적용할 리소스를 식별합니다. 자세한 내용은 EventBridge 리소스 섹션을 참조하세요.
- Action - 키워드를 사용하여 허용 또는 거부할 리소스 작업을 식별합니다. 예를 들어 `events:Describe` 권한은 사용자에게 `Describe` 작업 수행을 허용합니다.

- Effect – 허용 또는 거부를 지정합니다. 리소스에 대한 액세스 권한을 부여(허용)하지 않는 경우, 액세스는 묵시적으로 거부됩니다. 한 리소스에 대한 액세스를 명시적으로 거부할 수도 있으며, 다른 정책에 따라 액세스 권한이 부여되더라도 사용자가 이 리소스에 액세스하지 못하도록 조치를 취합니다.
- Principal – 자격 증명 기반 정책(IAM 정책)에서 정책이 연결되는 사용자는 암시적인 보안 주체입니다. 리소스 기반 정책의 경우 사용자, 계정, 서비스 또는 권한의 수신자인 기타 개체를 지정합니다(리소스 기반 정책에만 해당).

IAM 정책 구문과 설명에 대한 자세한 내용은 IAM User Guide의 IAM JSON Policy Reference를 참조하세요.

EventBridge API 작업 및 해당 작업이 적용되는 리소스에 대한 자세한 내용은 Amazon EventBridge 권한 참조 섹션을 참조하세요.

정책에서 조건 지정

권한을 부여할 때 액세스 정책 언어를 사용하여 조건이 적용되는 조건을 지정할 수 있습니다. 예를 들어, 특정 날짜 이후에만 정책을 적용할 수 있습니다. 정책 언어에서의 조건 지정에 관한 자세한 내용은 IAM 사용 설명서의 조건을 참조하세요.

조건을 표시하려면 미리 정의된 조건 키를 사용합니다. AWS 조건 키와 EventBridge 관련 키를 적절하게 사용할 수 있습니다. AWS 키의 전체 목록은 IAM 사용 설명서의 사용 가능한 조건 키를 참조하세요. EventBridge 관련 키의 전체 목록은 IAM 정책 조건을 사용하여 세분화된 액세스 제어 구현 단원을 참조하십시오.

Amazon의 자격 증명 기반 정책 (IAM 정책) 사용 EventBridge

자격 증명 기반 정책은 IAM 자격 증명에 연결할 수 있는 권한 정책입니다.

주제

- AWS 관리형 정책은 다음과 같습니다. EventBridge
- IAM 역할을 사용하여 대상에 액세스하는 EventBridge 데 필요한 권한
- 고객 관리형 정책 예시: 태그를 사용하여 규칙에 대한 액세스 제어
- Amazon, AWS 관리형 정책 EventBridge 업데이트

AWS 관리형 정책은 다음과 같습니다. EventBridge

AWS 에서 생성하고 관리하는 독립형 IAM 정책을 제공하여 많은 일반적인 사용 사례를 해결합니다. AWS관리형 또는 미리 정의된 정책은 일반 사용 사례에 필요한 권한을 부여하므로 어떤 권한이 필요한지 조사할 필요가 없습니다. 자세한 내용은 IAM 사용자 설명서의 AWS 관리형 정책을 참조하세요.

계정의 사용자에게 연결할 수 있는 다음과 같은 AWS 관리형 정책은 다음과 같습니다. EventBridge

- AmazonEventBridgeFullAccess— EventBridge 파이프, EventBridge 스키마 EventBridge, EventBridge 스케줄러를 포함한 모든 액세스 권한을 부여합니다.
- AmazonEventBridgeReadOnlyAccess— EventBridge 파이프, EventBridge 스키마 EventBridge, 스케줄러를 포함한 읽기 전용 액세스 권한을 부여합니다. EventBridge

AmazonEventBridgeFullAccess 정책

AmazonEventBridgeFullAccess 정책은 모든 EventBridge 작업을 사용할 수 있는 권한과 다음 권한을 부여합니다.

- `iam:CreateServiceLinkedRole`— 계정에서 API 대상의 서비스 역할을 만들려면 이 권한이 EventBridge 필요합니다. 이 권한은 IAM 서비스에만 사용자 계정에서 API 대상의 역할을 생성할 수 있는 권한을 부여합니다.
- `iam:PassRole`— EventBridge 규칙의 대상을 호출하기 위해 호출 역할을 EventBridge 전달하려면 이 권한이 필요합니다.
- Secrets Manager 권한 - 연결 리소스를 통해 자격 증명을 제공하여 API 대상을 승인할 때 계정의 비밀을 관리하려면 이러한 권한이 EventBridge 필요합니다.

다음 JSON은 정책을 보여줍니다. AmazonEventBridgeFullAccess

```
{
    "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": "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**
>
> 이 섹션의 정보는 `CloudWatchEventsFullAccess` 정책에도 적용됩니다. 하지만 Amazon CloudWatch Events EventBridge 대신 Amazon을 사용하는 것이 좋습니다.

AmazonEventBridgeReadOnlyAccess 정책

AmazonEventBridgeReadOnlyAccess 정책은 모든 읽기 EventBridge 작업을 사용할 권한을 부여합니다.

다음 JSON은 AmazonEventBridgeReadOnlyAccess 정책을 보여줍니다.

```
{
    "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**
>
> 이 섹션의 정보는 `CloudWatchEventsReadOnlyAccess` 정책에도 적용됩니다. 하지만 Amazon CloudWatch Events EventBridge 대신 Amazon을 사용하는 것이 좋습니다.

EventBridge 스키마별 관리형 정책

스키마는 전송되는 이벤트의 구조를 정의합니다. EventBridge EventBridge AWS 서비스에서 생성되는 모든 이벤트에 대한 스키마를 제공합니다. EventBridge 스키마와 관련된 다음과 같은 AWS 관리형 정책을 사용할 수 있습니다.

- AmazonEventBridgeSchemasServiceRolePolicy
- AmazonEventBridgeSchemasFullAccess
- AmazonEventBridgeSchemasReadOnlyAccess

EventBridge 스케줄러별 관리형 정책

Amazon EventBridge Scheduler는 하나의 중앙 관리형 서비스에서 작업을 생성, 실행 및 관리할 수 있는 서버리스 스케줄러입니다. 스케줄러와 관련된 AWS 관리형 정책은 EventBridge 스케줄러 사용 설명서의 EventBridge 스케줄러AWS 관리형 정책을 참조하십시오. EventBridge

EventBridge 파이프별 관리 정책

Amazon EventBridge Pipes는 이벤트 소스를 대상에 연결합니다. 파이프는 이벤트 중심 아키텍처를 개발할 때 전문 지식 및 통합 코드의 필요성을 줄여줍니다. 이는 회사 애플리케이션 전반의 일관성을 보장하는 데 도움이 됩니다. EventBridge Pipes와 관련된 다음과 같은 AWS 관리형 정책을 사용할 수 있습니다.

- AmazonEventBridgePipesFullAccess

  Amazon EventBridge Pipes에 대한 전체 액세스 권한을 제공합니다.

  > **Note**
  >
  > 이 정책은 다음을 제공합니다`iam:PassRole`. Pipes가 호출 역할을 전달하여 파이프를 생성하고 EventBridge 시작하려면 이 권한이 필요합니다. EventBridge

- AmazonEventBridgePipesReadOnlyAccess

  Amazon EventBridge Pipes에 대한 읽기 전용 액세스를 제공합니다.

- AmazonEventBridgePipesOperatorAccess

  Amazon EventBridge Pipes에 대한 읽기 전용 및 운영자 (즉, Pipes 실행을 중지하고 시작할 수 있는 기능) 액세스 권한을 제공합니다.

이벤트 전송을 위한 IAM 역할

이벤트를 대상으로 중계하려면 IAM 역할이 EventBridge 필요합니다.

이벤트를 전송하기 위한 IAM 역할을 만들려면 EventBridge

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. IAM 역할을 생성하려면 IAM 사용 설명서의 AWS 서비스에 권한을 위임할 역할 생성의 단계를 따르십시오. 단계를 따르면서 다음을 수행합니다.
   - 역할 이름에 계정 내에서 고유한 이름을 사용합니다.
   - 역할 유형 선택에서 AWS 서비스 역할을 선택한 다음 Amazon을 선택합니다 EventBridge. 이렇게 하면 역할을 수임할 EventBridge 권한이 부여됩니다.
   - [연결 정책] 에서 선택합니다 AmazonEventBridgeFullAccess.

사용자 지정 IAM 정책을 생성하여 EventBridge 작업 및 리소스에 대한 권한을 허용할 수도 있습니다. 해당 권한이 필요한 IAM 사용자 또는 그룹에 이러한 사용자 지정 정책을 연결할 수 있습니다. IAM 정책 작성에 대한 자세한 내용은 IAM 사용 설명서의 IAM 정책 개요를 참조하세요. 사용자 지정 IAM 정책 관리 및 생성에 대한 자세한 내용 IAM 사용 설명서에서 IAM 정책 관리를 참조하세요.

IAM 역할을 사용하여 대상에 액세스하는 EventBridge 데 필요한 권한

EventBridge 대상에는 일반적으로 대상을 호출할 EventBridge 권한을 부여하는 IAM 역할이 필요합니다. 다음은 다양한 AWS 서비스 및 대상에 대한 몇 가지 예입니다. 다른 경우에는 EventBridge 콘솔을 사용하여 규칙을 만든 다음 적절한 범위의 권한이 미리 구성된 정책으로 생성될 새 역할을 만들 수 있습니다.

Amazon SQS, Amazon SNS, Lambda, CloudWatch 로그 및 EventBridge 버스 타겟은 역할을 사용하지 않으므로 리소스 정책을 통해 권한을 EventBridge 부여해야 합니다. API 게이트웨이 대상은 리소스 정책 또는 IAM 역할을 사용할 수 있습니다.

대상이 API 대상인 경우 지정한 역할에 다음 정책이 포함되어야 합니다.

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

대상이 Kinesis 스트림이면 그 대상에게 이벤트 데이터를 전송하는 데 사용된 역할에 다음 정책이 포함되어야 합니다.

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

```
        }
    ]
}
```

대상이 시스템 관리자 실행 명령이고 해당 명령에 하나 이상의 InstanceIds 값을 지정하는 경우에는 지정한 역할에 다음 정책이 반드시 포함되어야 합니다.

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

대상이 시스템 관리자 실행 명령이고 해당 명령에 하나 이상의 태그를 지정하는 경우에는 지정한 역할에 다음 정책이 반드시 포함되어야 합니다.

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

대상이 AWS Step Functions 스테이트 머신인 경우 지정하는 역할에 다음 정책이 포함되어야 합니다.

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

대상이 Amazon ECS 작업인 경우에는 지정한 역할에 다음 정책이 포함되어야 합니다.

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

다음 정책은 내장된 대상이 사용자를 EventBridge 대신하여 Amazon EC2 작업을 수행하도록 허용합니다. 를 사용하여 대상이 내장된 규칙을 AWS Management Console 생성해야 합니다.

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

다음 정책은 계정의 Kinesis 스트림에 이벤트를 릴레이할 수 있도록 허용합니다 EventBridge .

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

```
            "Resource": "*"
        }
    ]
}
```

고객 관리형 정책 예시: 태그를 사용하여 규칙에 대한 액세스 제어

다음 예는 EventBridge 작업에 대한 권한을 부여하는 사용자 정책을 보여줍니다. 이 정책은 EventBridge API, AWS SDK 또는 를 AWS CLI사용할 때 작동합니다.

사용자에게 특정 EventBridge 규칙에 대한 액세스 권한을 부여하면서 다른 규칙에는 액세스하지 못하게 할 수 있습니다. 이렇게 하려면 두 규칙 세트 모두에 태그를 지정한 다음 해당 태그를 참조하는 IAM 정책을 사용합니다. EventBridge 리소스에 태그를 지정하는 방법에 대한 자세한 내용은 을 참조하십시오아마존 EventBridge 태그.

특정 태그가 있는 규칙에만 액세스할 수 있도록 사용자에게 IAM 정책에 대한 권한을 부여할 수 있습니다. 특정 태그로 규칙에 태그를 지정하여 액세스 권한을 부여할 규칙을 선택합니다. 예를 들어 다음 정책은 태그 키 `Stack`의 값이 `Prod`인 규칙에 대한 사용자 액세스 권한을 부여합니다.

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

IAM 정책에 자세한 내용은 IAM 사용 설명서에서 정책을 사용하여 액세스 제어를 참조하세요.

Amazon, AWS 관리형 정책 EventBridge 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 EventBridge 이후의 AWS 관리형 정책 업데이트에 대한 세부 정보를 확인하십시오. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 EventBridge 문서 기록 페이지에서 RSS 피드를 구독하십시오.

변경 사항	설명	날짜
AmazonEventBridgeFullAccess - 정책 업데이트	AWS GovCloud (US) Regions 전용 다음 권한은 사용되지 않으므로 포함되지 않습니다. • `iam:CreateServiceLinkedRole` EventBridge 스키마 레지스트리에 대한 권한	2024년 5월 9일
AmazonEventBridgeSchemasFullAccess - 정책 업데이트	AWS GovCloud (US) Regions 전용 다음 권한은 사용되지 않으므로 포함되지 않습니다. • `iam:CreateServiceLinkedRole` EventBridge 스키마 레지스트리에 대한 권한	2024년 5월 9일
AmazonEventBridgePipesFullAccess— 새 정책 추가	EventBridge EventBridge Pipes 사용에 대한 전체 권한에 대한 관리형 정책이 추가되었습니다.	2022년 12월 1일
AmazonEventBridgePipesReadOnlyAccess— 새 정책 추가	EventBridge EventBridge Pipes 정보 리소스를 볼 수 있는 권한에 대한 관리형 정책이 추가되었습니다.	2022년 12월 1일
AmazonEventBridgePipesOperatorAccess— 새 정책 추가	EventBridge EventBridge 파이프 정보를 볼 수 있는 권한과 파이프 실행을 시작하고 중지할	2022년 12월 1일

변경 사항	설명	날짜
	수 있는 권한에 대한 관리형 정책이 추가되었습니다.	
AmazonEventBridgeFullAccess-기존 정책 업데이트	EventBridge EventBridge Pipes 기능을 사용하는 데 필요한 권한을 포함하도록 정책을 업데이트했습니다.	2022년 12월 1일
AmazonEventBridgeReadOnlyAccess-기존 정책 업데이트	EventBridge EventBridge 파이프 정보 리소스를 보는 데 필요한 권한을 추가했습니다. 다음 작업이 추가되었습니다. • `pipes:DescribePipe` • `pipes:ListPipes` • `pipes:ListTagsForResource`	2022년 12월 1일
CloudWatchEventsReadOnlyAccess-기존 정책 업데이트	일치하도록 업데이트되었습니다 AmazonEventBridgeReadOnlyAccess.	2022년 12월 1일
CloudWatchEventsFullAccess-기존 정책 업데이트	이에 맞게 업데이트되었습니다 AmazonEventBridgeFullAccess.	2022년 12월 1일

변경 사항	설명	날짜
AmazonEventBridgeFullAccess-기존 정책 업데이트	EventBridge 스키마 및 스케줄러 기능을 사용하는 데 필요한 권한을 포함하도록 정책을 업데이트했습니다. 다음 권한이 추가되었습니다. • EventBridge 스키마 레지스트리 작업 • EventBridge 스케줄러 작업 • `iam:CreateServiceLinkedRole` EventBridge 스키마 레지스트리에 대한 권한 • `iam:PassRole` EventBridge 스케줄러에 대한 권한	2022년 11월 10일

변경 사항	설명	날짜
AmazonEventBridgeReadOnlyAccess-기존 정책 업데이트	EventBridge 스키마 및 스케줄러 정보 리소스를 보는 데 필요한 권한이 추가되었습니다. 다음 작업이 추가되었습니다. • 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	2022년 11월 10일

변경 사항	설명	날짜
	• schemas:SearchSchemas • scheduler:GetSchedule • scheduler:GetScheduleGroup • scheduler:ListSchedules • scheduler:ListScheduleGroups • scheduler:ListTagsForResource	
AmazonEventBridgeReadOnlyAccess-기존 정책 업데이트	EventBridge 엔드포인트 정보를 보는 데 필요한 권한을 추가했습니다. 다음 작업이 추가되었습니다. • events:ListEndpoints • events:DescribeEndpoint	2022년 4월 7일

변경 사항	설명	날짜
AmazonEventBridgeReadOnlyAccess-기존 정책 업데이트	EventBridge 연결 및 API 대상 정보를 보는 데 필요한 권한을 추가했습니다. 다음 작업이 추가되었습니다. • `events:DescribeConnection` • `events:ListConnections` • `events:DescribeApiDestination` • `events:ListApiDestinations`	2021년 3월 4일
AmazonEventBridgeFullAccess-기존 정책 업데이트	EventBridge API 대상 사용에 필요한 `iam:CreateServiceLinkedRole` AWS Secrets Manager 권한과 포함하도록 정책을 업데이트했습니다. 다음 작업이 추가되었습니다. • `secretsmanager:CreateSecret` • `secretsmanager:UpdateSecret` • `secretsmanager:DeleteSecret` • `secretsmanager:GetSecretValue` • `secretsmanager:PutSecretValue`	2021년 3월 4일

변경 사항	설명	날짜
EventBridge 변경 내용 추적 시작	EventBridge AWS 관리형 정책의 변경 사항 추적을 시작했습니다.	2021년 3월 4일

Amazon EventBridge에 리소스 기반 정책 사용

EventBridge에서 규칙이 실행되면 이 규칙과 연관된 모든 대상이 간접 호출됩니다. 규칙은 AWS Lambda 함수를 간접 호출하거나, Amazon SNS 주제에 게시하거나, 이벤트를 Kinesis 스트림으로 중계할 수 있습니다. 소유하고 있는 리소스에 대해 API 직접 호출을 수행하려면 EventBridge에 적절한 권한이 필요합니다. Lambda, Amazon SNS, Amazon SQS 및 Amazon CloudWatch Logs 리소스의 경우 EventBridge는 리소스 기반 정책을 사용합니다. Kinesis 스트림의 경우 EventBridge는 자격 증명 기반 정책을 사용합니다.

AWS CLI를 사용하여 대상에 권한을 추가할 수 있습니다. AWS CLI의 설치 및 구성 방법에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS Command Line Interface로 설정을 참조하세요.

주제

- Amazon API Gateway 권한
- CloudWatch Logs 권한
- AWS Lambda 권한
- Amazon SNS 권한
- Amazon SQS 권한
- EventBridge 파이프 세부 사항

Amazon API Gateway 권한

EventBridge 규칙을 사용하여 Amazon API Gateway 엔드포인트를 간접 호출하려면 API Gateway 엔드포인트의 정책에 다음 권한을 추가하십시오.

```
{
  "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 권한

CloudWatch Logs가 규칙의 대상인 경우 EventBridge는 로그 스트림을 생성하며 CloudWatch Logs는 이벤트의 텍스트를 로그 항목으로 저장합니다. EventBridge가 로그 스트림을 생성하고 이벤트를 기록하도록 허용하려면 CloudWatch Logs에 EventBridge가 CloudWatch Logs에 기록할 수 있도록 허용하는 리소스 기반 정책이 포함되어야 합니다.

AWS Management Console을 사용하여 CloudWatch Logs를 규칙의 대상으로 추가하면 리소스 기반 정책이 자동으로 생성됩니다. AWS CLI를 사용하여 대상을 추가하고 정책이 아직 존재하지 않는 경우 정책을 생성해야 합니다.

다음 예에서는 EventBridge가 이름이 `/aws/events/`로 시작하는 모든 로그 그룹에 기록하도록 허용합니다. 이러한 종류의 로그에 대해 다른 명명 정책을 사용하는 경우, 그에 따라 예제를 조정해야 합니다.

```
{
    "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"
}
```

자세한 내용은 CloudWatch Logs API 참조 가이드에서 PutResourcePolicy를 참조하세요.

AWS Lambda 권한

EventBridge 규칙을 사용하여 AWS Lambda 함수를 간접 호출하려면 Lambda 함수의 정책에 다음 권한을 추가합니다.

```
{
  "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"
}
```

EventBridge가 AWS CLI를 사용하여 Lambda 함수를 간접 호출할 수 있도록 하는 위의 권한을 추가하려면

- 명령 프롬프트에서 다음 명령을 입력합니다.

```
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"
```

EventBridge가 Lambda 함수를 간접 호출할 수 있도록 하는 권한 설정에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 AddPermission 및예약된 이벤트와 함께 Lambda 사용을 참조하십시오.

Amazon SNS 권한

EventBridge가 Amazon SNS 주제에 게시하도록 허용하려면 `aws sns get-topic-attributes` 및 `aws sns set-topic-attributes` 명령을 사용합니다.

> **Note**
>
> EventBridge의 Amazon SNS 주제 정책에서는 `Condition` 블록을 사용할 수 없습니다.

EventBridge가 SNS 주제를 게시하도록 허용하는 권한을 추가하려면

1. SNS 주제의 속성을 나열하려면 다음 명령을 사용합니다.

```
aws sns get-topic-attributes --topic-arn "arn:aws:sns:region:account-id:topic-name"
```

   다음 예제는 새 SNS 주제의 결과를 보여줍니다.

```
{
    "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. JSON-문자열 변환기를 사용하여 다음 명령문을 문자열로 변환합니다.

```
{
  "Sid": "PublishEventsToMyTopic",
  "Effect": "Allow",
  "Principal": {
    "Service": "events.amazonaws.com"
```

```
  },
  "Action": "sns:Publish",
  "Resource": "arn:aws:sns:region:account-id:topic-name"
}
```

이 명령문이 문자열로 변환되면 다음 예시와 유사하게 화면에 나타납니다.

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

3. 이전 단계에서 만든 문자열을 "Policy" 속성 내 "Statement" 컬렉션에 추가합니다.
4. aws sns set-topic-attributes 명령을 사용하여 새 정책을 설정합니다.

```
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\"}]}"
```

자세한 내용은 Amazon Simple Notification Service API 참조의 SetTopicAttributes 작업을 참조하십시오.

Amazon SQS 권한

EventBridge 규칙이 Amazon SQS 대기열을 간접 호출하도록 허용하려면 aws sqs get-queue-attributes 및 aws sqs set-queue-attributes 명령을 사용합니다.

SQS 대기열의 정책이 비어 있는 경우 먼저 정책을 생성한 다음 권한 설명을 추가할 수 있습니다. 새 SQS 대기열에는 빈 정책이 있습니다.

SQS 대기열에 이미 정책이 있는 경우 원본 정책을 복사하고 새 명령문과 결합하여 권한 설명을 추가해야 합니다.

EventBridge 규칙이 SQS 대기열을 간접 호출하도록 허용하는 권한을 추가하려면

1. SQS 대기열 속성을 나열하려면 명령 프롬프트에서 다음 명령을 입력합니다.

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

2. 다음 명령문을 추가합니다.

```
{
      "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. JSON-문자열 변환기를 사용하여 앞의 명령문을 문자열로 변환합니다. 정책이 문자열로 변환되면 다음과 유사하게 화면에 나타납니다.

```
{\"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. 다음 콘텐츠를 통해 `set-queue-attributes.json`이라는 파일을 생성합니다.

```
{
    "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. 다음 명령과 같이 방금 생성한 `set-queue-attributes.json` 파일을 입력으로 사용하여 정책 속성을 설정합니다.

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

자세한 내용은 Amazon Simple Queue Service 개발자 안내서의 Amazon SQS 정책 예를 참조하세요.

EventBridge 파이프 세부 사항

EventBridge 파이프는 리소스 기반 정책을 지원하지 않으며 리소스 기반 정책 조건을 지원하는 API도 없습니다.

교차 서비스 혼동된 대리자 예방

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. AWS에서는 교차 서비스 가장으로 인해 혼동된 대리자 문제가 발생할 수 있습니다. 교차 서비스 가장은 한 서비스(호출하는 서비스)가 다른 서비스(호출되는 서비스)를 호출할 때 발생할 수 있습니다. 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

Amazon EventBridge가 리소스에 다른 서비스를 제공하는 권한을 제한하려면 리소스 정책에서 `aws:SourceArn` 및 `aws:SourceAccount` 글로벌 조건 컨텍스트 키를 사용하는 것이 좋습니다. 하나의 리소스만 교차 서비스 액세스와 연결되도록 허용하려는 경우 `aws:SourceArn`을(를) 사용하세요. 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 `aws:SourceAccount`을(를) 사용하세요.

혼동된 대리자 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 `aws:SourceArn` 글로벌 조건 컨텍스트 키를 사용하는 것입니다. 리소스의 전체 ARN을 모르거나 여러 리소스를 지정하는 경우, ARN의 알 수 없는 부분에 대해 와일드카드 문자(*)를 포함한 `aws:SourceArn` 글로벌 조건 컨텍스트 키를 사용합니다. 예: `arn:aws:servicename:*:123456789012:*`.

만약 aws:SourceArn 값에 Amazon S3 버킷 ARN과 같은 계정 ID가 포함되어 있지 않은 경우, 권한을 제한하려면 두 전역 조건 컨텍스트 키를 모두 사용해야 합니다.

이벤트 버스

EventBridge 이벤트 버스 규칙 대상의 경우 aws:SourceArn의 값은 규칙 ARN이어야 합니다.

다음 예는 EventBridge에서 aws:SourceArn 및 aws:SourceAccount 글로벌 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 방지하는 방법을 보여줍니다. 이 예는 EventBridge 규칙에서 사용하는 역할에 대한 역할 신뢰 정책에 사용하기 위한 것입니다.

```
{
  "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

EventBridge 파이프의 경우 aws:SourceArn의 값은 파이프 ARN이어야 합니다.

다음 예는 EventBridge에서 aws:SourceArn 및 aws:SourceAccount 글로벌 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 방지하는 방법을 보여줍니다. 이 예는 EventBridge 파이프에서 사용하는 역할에 대한 역할 신뢰 정책에 사용하기 위한 것입니다.

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

Amazon EventBridge 스키마에 대한 리소스 기반 정책

EventBridge 스키마 레지스트리는 리소스 기반 정책을 지원합니다. 리소스 기반 정책은 IAM 자격 증명이 아니라 리소스에 연결된 정책입니다. 예를 들어 Amazon Simple Storage Service(S3)에서 리소스 정책은 Amazon S3 버킷에 연결됩니다.

EventBridge 스키마 및 리소스 기반 정책에 대한 자세한 내용은 다음을 참조하세요.

- Amazon EventBridge 스키마 REST API 참조
- IAM 사용 설명서의 자격 증명 기반 정책 및 리소스 기반 정책

리소스 기반 정책을 위한 지원되는 API

EventBridge 스키마 레지스트리에 대한 리소스 기반 정책과 함께 다음 API를 사용할 수 있습니다.

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

AWS 계정에 지원되는 모든 작업을 부여하는 정책 예시

EventBridge 스키마 레지스트리의 경우 항상 리소스 기반 정책을 레지스트리에 연결해야 합니다. 스키마에 대한 액세스 권한을 부여하려면 정책에서 스키마 ARN과 레지스트리 ARN을 지정합니다.

EventBridge 스키마에서 사용 가능한 모든 API에 대한 액세스 권한을 사용자에게 부여하려면 다음과 유사한 정책을 사용하여 액세스 권한을 부여하려는 계정의 계정 ID로 "Principal"을 대체하십시오.

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

AWS 계정에 읽기 전용 작업을 부여하는 정책 예시

다음 예시에서는 EventBridge 스키마에 대한 읽기 전용 API에 대해서만 계정에 액세스 권한을 부여합니다.

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

조직에 모든 작업을 부여하는 정책 예시

EventBridge 스키마 레지스트리와 함께 리소스 기반 정책을 사용하여 조직에 액세스 권한을 부여할 수 있습니다. 자세한 정보는 AWS Organizations 사용 설명서를 참조하세요. 다음 예시에서는 ID가 o-a1b2c3d4e5인 조직에 스키마 레지스트리에 대한 액세스 권한을 부여합니다.

```
{
    "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 권한 참조

EventBridge 정책에서 작업을 지정하려면 다음 예와 같이 `events:` 접두사 다음에 API 작업 이름을 사용합니다.

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

명령문 하나에 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

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

여러 작업을 지정하려면 와일드카드를 사용할 수도 있습니다. 예를 들어 다음과 같이 `"Put"`이라는 단어로 시작되는 모든 작업을 지정할 수 있습니다.

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

모든 EventBridge API 작업을 지정하려면 다음과 같이 * 와일드카드를 사용합니다.

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

다음 표에는 IAM 정책에서 지정할 수 있는 EventBridge API 작업 및 해당 작업이 나열되어 있습니다.

EventBridge API 작업	필요한 권한	설명
DeleteRule	`events:DeleteRule`	규칙을 삭제하는 데 필요합니다.
DescribeEventBus	`events:DescribeEventBus`	현재 계정의 이벤트 버스에 이벤트를 기록할 수 있는 계정을 나열해야 합니다.
DescribeRule	`events:DescribeRule`	규칙에 대한 세부 사항을 나열하는 데 필요합니다.
DisableRule	`events:DisableRule`	규칙을 비활성화하는 데 필요합니다.

EventBridge API 작업	필요한 권한	설명
EnableRule	events:EnableRule	규칙을 활성화하는 데 필요합니다.
ListRuleNamesByTarget	events:ListRuleNamesByTarget	대상과 연관된 규칙을 나열하는 데 필요합니다.
ListRules	events:ListRules	계정에서 모든 그룹을 나열하는 데 필요합니다.
ListTagsForResource	events:ListTagsForResource	EventBridge 리소스와 연결된 모든 태그를 나열하는 데 필요합니다. 현재는 규칙에만 태그를 지정할 수 있습니다.
ListTargetsByRule	events:ListTargetsByRule	규칙과 연관된 모든 대상을 나열하는 데 필요합니다.
PutEvents	events:PutEvents	규칙에 일치시킬 수 있는 사용자 지정 이벤트를 추가하는 데 필요합니다.
PutPermission	events:PutPermission	이 계정의 기본 이벤트 버스에 이벤트를 기록할 수 있는 계정 권한을 하나 더 부여해야 합니다.
PutRule	events:PutRule	규칙을 생성 또는 업데이트하는 데 필요합니다.
PutTargets	events:PutTargets	규칙에 대상을 추가하는 데 필요합니다.
RemovePermission	events:RemovePermission	이 계정의 기본 이벤트 버스에 이벤트를 기록할 수 있는 다른 계정의 권한을 취소해야 합니다.

EventBridge API 작업	필요한 권한	설명
RemoveTargets	`events:RemoveTargets`	규칙에서 대상을 제거하는 데 필요합니다.
TestEventPattern	`events:TestEventPattern`	특정 이벤트를 기준으로 이벤트 패턴을 테스트하는 데 필요합니다.

IAM 정책 조건을 사용하여 세분화된 액세스 제어 구현

권한을 부여하려면 정책 설명에서 IAM 정책 언어를 사용하여 정책이 적용되는 조건을 지정합니다. 예를 들어 특정 날짜 이후에만 적용되는 정책이 있을 수 있습니다.

정책 조건은 키-값 페어로 구성됩니다. 조건 키에는 대/소문자가 구분되지 않습니다.

단일 조건에 여러 조건이나 키를 지정하는 경우 EventBridge가 권한을 부여하려면 모든 조건 및 키를 충족해야 합니다. 조건 하나에서 키 하나에 여러 값을 지정하면 EventBridge는 값 중 하나가 충족되면 권한을 부여합니다.

조건을 지정할 때 자리 표시자나 정책 변수를 사용할 수 있습니다. 자세한 내용은 IAM 사용 설명서에서 정책 변수를 참조하십시오. IAM 정책 언어에서의 조건 지정에 관한 자세한 내용은 IAM 사용 설명서의 조건을 참조하세요.

기본적으로 IAM 사용자와 역할은 계정의 이벤트에 전혀 액세스할 수 없습니다. 이벤트에 액세스하려면 `PutRule` API 작업에 대해 허가를 받아야 합니다. IAM 사용자 또는 역할에 `events:PutRule` 작업에 대한 권한이 부여되면 특정 이벤트와 일치하는 규칙을 생성할 수 있습니다. 그러나 규칙이 유용하려면 사용자에게 `events:PutTargets` 작업에 대한 권한도 있어야 합니다. 규칙이 CloudWatch 지표 게시하는 것 이상의 작업을 수행하도록 하려면 규칙에 대상도 추가해야 하기 때문입니다.

사용자나 역할이 특정한 소스 세트 및 세부 유형에만 일치하는 규칙을 생성할 수 있도록 IAM 사용자 또는 역할의 정책 명령문에 조건을 제공할 수 있습니다. 특정 소스 및 유형의 이벤트에 대한 액세스 권한을 부여하려면 `events:source` 및 `events:detail-type` 조건 키를 사용하십시오.

사용자나 역할이 계정의 특정 리소스에만 일치하는 규칙을 생성할 수 있도록 IAM 사용자 또는 역할의 정책 명령문에 조건을 제공할 수 있습니다. 특정 리소스에 대한 액세스 권한을 부여하려면 `events:TargetArn` 조건 키를 사용하십시오.

다음 예제는 사용자가 `PutRule` API 작업에 대한 거부 명령문을 사용하여 EventBridge의 Amazon EC2 이벤트를 제외한 모든 이벤트에 액세스할 수 있도록 허용하는 정책입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyPutRuleForAllEC2Events",
            "Effect": "Deny",
            "Action": "events:PutRule",
            "Resource": "*",
```

```
            "Condition": {
                "StringEquals": {
                    "events:source": "aws.ec2"
                }
            }
        }
    ]
}
```

EventBridge 조건 키

다음 표에는 EventBridge의 정책에서 사용할 수 있는 조건 키와 키-값 페어가 나와 있습니다.

조건 키	키-값 페어:	평가 유형
aws:SourceAccount	`aws:SourceArn` 에서 지정한 규칙이 존재하는 계정입니다.	계정 ID, Null
aws:SourceArn	이벤트를 전송하는 규칙의 ARN입니다.	ARN, Null
events:creatorAccount	`"events:creatorAccount":"` *`creatorAccount`* `"` *`creatorAccount`* 의 경우 규칙을 생성한 계정의 계정 ID를 사용합니다. 이 조건을 사용하면 특정 계정의 규칙에 대한 API 직접 호출을 승인할 수 있습니다.	creatorAccount, Null
events:detail-type	`"events:detail-type":"` *`detail-type`* `"` 여기서 *`detail-type`* 은 `"AWS API Call via CloudTrail"` 및 `"EC2 Instance State-change Notification"` 과 같은 이벤트의 detail-type 필드에 대한 리터럴 문자열입니다.	세부 유형, Null

조건 키	키-값 페어:	평가 유형
events: detail.eventTypeCode	"events:detail.eventTypeCode":" *eventTypeCode* " *eventTypeCode* 의 경우 이벤트의 detail.eventTypeCode 필드에 대한 리터럴 문자열(예: "AWS_ABUSE_DOS_REPORT")을 사용합니다.	eventTypeCode, Null
events: detail.service	"events:detail.service":" *service*" *service*는 이벤트의 detail.service 필드에 대한 리터럴 문자열(예: "ABUSE")을 사용합니다.	service, Null
events: detail.userIdentity.principalId	"events:detail.userIdentity.principalId":" *principal-id* " *principal-id* 의 경우 "AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName." 과 같은 detail-type "AWS API Call via CloudTrail" 이 있는 이벤트의 detail.userIdentity.principalId 필드에 리터럴 문자열을 사용합니다.	보안 주체 ID, Null

조건 키	키-값 페어:	평가 유형
events:eventBusInvocation	`"events:eventBusInvocation":" boolean"` *boolean*의 경우 규칙이 다른 계정의 이벤트 버스인 대상으로 이벤트를 보내는 경우 true를 사용합니다. `PutEvents` API 직접 호출을 사용할 때는 false를 사용하십시오.	eventBusInvocation, Null
events:ManagedBy	AWS 서비스에서 내부적으로 사용합니다. 사용자를 대신하여 AWS 서비스에서 생성된 규칙의 경우 값은 규칙을 생성한 서비스의 보안 주체 이름입니다.	고객 정책에 사용할 수 없습니다.
events:source	`"events:source":" source "` `"aws.ec2"` 또는 `"aws.s3"` 같은 이벤트의 소스 필드를 위한 리터럴 문자열에 *source*를 사용합니다. *source*에 사용 가능한 값을 더 보려면 서비스의 이벤트 AWS 단원에서 예제 이벤트를 참조하십시오.	소스, Null
events:TargetArn	`"events:TargetArn" :" target-arn "` 예를 들어 *target-arn* 의 경우 대상의 ARN을 규칙에 사용합니다(예: `"arn:aws:lambda:*:*:function:*"`).	ArrayOfARN, Null

EventBridge용 예제 정책 명령문은 Amazon EventBridge 리소스에 대한 액세스 권한 관리 단원을 참조하십시오.

주제

EventBridge 파이프 세부 사항

EventBridge 파이프는 추가 IAM 정책 조건 키를 지원하지 않습니다.

예: **`creatorAccount`** 조건 사용

다음 예제 정책 설명은 정책에서 `creatorAccount` 조건을 사용하여 `creatorAccount`로 지정된 계정이 규칙을 만든 계정인 경우에만 규칙을 만들도록 허용하는 방법을 보여줍니다.

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

```
                }
            }
        }
    ]
}
```

예: **eventBusInvocation** 조건 사용

eventBusInvocation는 간접 호출이 교차 계정 대상에서 시작되었는지 아니면 PutEvents API 요청에서 시작되었는지를 나타냅니다. 대상이 다른 계정의 이벤트 버스인 경우와 같이 교차 계정 대상이 포함된 규칙에서 간접 호출이 발생한 경우 이 값은 true입니다. PutEvents API 요청으로 인한 간접 호출 결과인 경우 이 값은 false입니다. 다음 예는 교차 계정 대상으로부터의 간접 호출을 나타냅니다.

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

예제: 특정 소스에 대한 액세스 제한

다음은 IAM 사용자에게 연결할 수 있는 정책의 예제입니다. 정책 A는 모든 이벤트에서 PutRule API 작업을 허용하는 반면, 정책 B는 생성 중인 규칙의 이벤트 패턴이 Amazon EC2 이벤트와 일치하는 경우에만 PutRule을 허용합니다.

정책 A: 모든 이벤트 허용

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

```
            "Sid": "AllowPutRuleForAllEvents",
            "Effect": "Allow",
            "Action": "events:PutRule",
            "Resource": "*"
        }
    ]
    }
```

정책 B: —Amazon EC2의 이벤트만 허용

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

EventPattern은 PutRule에 대한 필수 인수입니다. 따라서 정책 B의 사용자가 다음과 같은 이벤트 패턴을 통해 PutRule을 호출하는 경우에는

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

정책이 이러한 특정 소스(예: "aws.ec2")를 허용하기 때문에 규칙을 생성할 수 있습니다. 그러나 정책 B를 사용하는 사용자가 다음과 같은 이벤트 패턴으로 PutRule을 호출하면 정책이 이 특정 소스(즉, "aws.s3")를 허용하지 않으므로 규칙 생성이 거부됩니다.

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

기본적으로 정책 B의 사용자만 Amazon EC2에서 호출된 이벤트와 일치하는 규칙을 생성할 수 있습니다. 따라서 이들만 Amazon EC2에서 이벤트 액세스가 허용됩니다.

정책 A와 정책 B를 비교하려면 다음 표를 참고하십시오.

이벤트 패턴	정책 A에서 허용	정책 B에서 허용
`{ "source": [ "aws.ec2" ] }`	예	예
`{ "source": [ "aws.ec2", "aws.s3" ] }`	예	아니요(소스 aws.s3이 허용되지 않음)
`{ "source": [ "aws.ec2" ], "detail-type": [ "EC2 Instance State-change Notification" ] }`	예	예
`{ "detail-type": [ "EC2 Instance State-change Notification" ] }`	예	아니요(소스를 지정해야 함)

예제: 이벤트 패턴에서 개별적으로 사용할 수 있는 소스를 여러 개 정의

다음 정책은 IAM 사용자 또는 역할이 `EventPattern`의 소스가 Amazon EC2 또는 Amazon ECS인 규칙을 생성할 수 있도록 허용합니다.

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

다음 표에는 이 정책에서 허용 또는 거부되는 이벤트 패턴의 몇 가지 예를 보여줍니다.

이벤트 패턴	정책에서 허용
`{ "source": [ "aws.ec2" ] }`	예
`{ "source": [ "aws.ecs" ] }`	예
`{ "source": [ "aws.s3" ] }`	아니요
`{ "source": [ "aws.ec2", "aws.ecs" ] }`	아니요

이벤트 패턴	정책에서 허용

<pre>{
"detail-type": ["AWS API Call via CloudTrail"]
}</pre>
	아니요

예제: 이벤트 패턴에서 사용할 수 있는 소스 및 **DetailType** 정의

다음 정책은 DetailType가 EC2 instance state change notification인 aws.ec2 소스에서 나온 이벤트만 허용합니다.

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

다음 표에는 이 정책에서 허용 또는 거부되는 이벤트 패턴의 몇 가지 예를 보여줍니다.

이벤트 패턴	정책에서 허용

<pre>{
"source": ["aws.ec2"]
}</pre>
	아니요

이벤트 패턴	정책에서 허용
`{ "source": [ "aws.ecs" ] }`	아니요
`{ "source": [ "aws.ec2" ], "detail-type": [ "EC2 Instance State-change Notification" ] }`	예
`{ "source": [ "aws.ec2" ], "detail-type": [ "EC2 Instance Health Failed" ] }`	아니요
`{ "detail-type": [ "EC2 Instance State-change Notification" ] }`	아니요

예제: 소스가 이벤트 패턴에 정의되어 있는지 확인

다음 정책은 사용자가 소스 필드가 있는 EventPatterns으로 규칙만 생성하도록 허용합니다. 이 정책을 사용하면 IAM 사용자나 역할은 특정 소스를 제공하지 않는 EventPattern으로는 규칙을 생성할 수 없습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutRuleIfSourceIsSpecified",
            "Effect": "Allow",
            "Action": "events:PutRule",
```

```
            "Resource": "*",
            "Condition": {
                "Null": {
                    "events:source": "false"
                }
            }
        }
    ]
}
```

다음 표에는 이 정책에서 허용 또는 거부되는 이벤트 패턴의 몇 가지 예를 보여줍니다.

이벤트 패턴	정책에서 허용
`{ "source": [ "aws.ec2" ], "detail-type": [ "EC2 Instance State-change Notification" ] }`	예
`{ "source": [ "aws.ecs", "aws.ec2" ] }`	예
`{ "detail-type": [ "EC2 Instance State-change Notification" ] }`	아니요

예제: 소스가 여러 개인 이벤트 패턴에서 허용되는 소스의 목록을 정의

다음 정책은 사용자가 여러 소스가 포함된 EventPatterns으로 규칙을 생성하는 것을 허용합니다. 이벤트 패턴의 각 소스는 해당 조건에서 제공되는 목록의 구성원이어야 합니다. ForAllValues 조건을 사용할 때는 이 조건의 항목 중 적어도 하나가 반드시 정의되어 있어야 합니다.

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

다음 표에는 이 정책에서 허용 또는 거부되는 이벤트 패턴의 몇 가지 예를 보여줍니다.

이벤트 패턴	정책에서 허용
`{ "source": [ "aws.ec2" ] }`	예
`{ "source": [ "aws.ec2", "aws.s3" ] }`	예
`{ "source": [ "aws.ec2", "aws.autoscaling" ] }`	아니요
`{`	아니요

이벤트 패턴	정책에서 허용

<pre>"detail-type": ["EC2 Instance State-change Notification"]
}</pre>	

예: **detail.service**에 의한 **PutRule** 액세스 제한

IAM 사용자 또는 역할에 대해 events:details.service 필드에 특정 값이 있는 이벤트에 대해서만 규칙을 생성하도록 제한할 수 있습니다. events:details.service의 값이 반드시 AWS 서비스의 이름일 필요는 없습니다.

이 정책 조건은 보안 또는 위반과 관련된 AWS Health의 이벤트를 사용할 때 유용합니다. 이 정책 조건을 사용하면 이러한 기밀 경보에 대한 액세스를, 해당 경보를 볼 필요가 있는 사용자로만 제한할 수 있습니다.

예를 들어 다음 정책은 events:details.service 값이 ABUSE인 경우에만 이벤트에 대한 규칙 생성을 허용합니다.

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

예: **detail.eventTypeCode**에 의한 **PutRule** 액세스 제한

IAM 사용자 또는 역할에 대해 events:details.eventTypeCode 필드에 특정 값이 있는 이벤트에 대해서만 규칙을 생성하도록 제한할 수 있습니다. 이 정책 조건은 보안 또는 위반과 관련된 AWS

Health의 이벤트를 사용할 때 유용합니다. 이 정책 조건을 사용하면 이러한 기밀 경보에 대한 액세스를, 해당 경보를 볼 필요가 있는 사용자로만 제한할 수 있습니다.

예를 들어 다음 정책은 `events:details.eventTypeCode` 값이 `AWS_ABUSE_DOS_REPORT`인 경우에만 이벤트에 대한 규칙 생성을 허용합니다.

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

예: 특정 **PrincipalId**의 API 직접 호출에 대해 AWS CloudTrail 이벤트만 허용되도록 설정

모든 AWS CloudTrail 이벤트는 이벤트의 `detail.userIdentity.principalId` 경로에서 API 직접 호출을 수행한 사용자의 PrincipalId를 가지고 있습니다. `events:detail.userIdentity.principalId` 조건 키를 사용하여 IAM 사용자나 역할이 특정 계정에서 들어오는 호출에 대한 CloudTrail 이벤트만 액세스할 수 있도록 제한할 수 있습니다.

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

다음 표에는 이 정책에서 허용 또는 거부되는 이벤트 패턴의 몇 가지 예를 보여줍니다.

이벤트 패턴	정책에서 허용

<pre>{
"detail-type": ["AWS API
Call via CloudTrail"]
}</pre>
	아니요

<pre>{
"detail-type": ["AWS API
Call via CloudTrail"],
"detail.userIdentity.princi
palId": ["AIDAJ45Q7YFFAREXA
MPLE"]
}</pre>
	예

<pre>{
"detail-type": ["AWS API
Call via CloudTrail"],
"detail.userIdentity.princi
palId": ["AROAIDPPEZS35WEXA
MPLE:AssumedRoleSessionName
"]
}</pre>
	아니요

예제: 대상에 대한 액세스 제한

IAM 사용자나 역할이 `events:PutTargets` 권한을 가지고 있는 경우에는 동일한 계정을 가진 모든 대상을 액세스를 허용하는 규칙에 추가할 수 있습니다. 다음 정책은 사용자가 특정 규칙(123456789012 계정 하의 `MyRule`)에만 대상을 추가하도록 제한합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutTargetsOnASpecificRule",
            "Effect": "Allow",
            "Action": "events:PutTargets",
            "Resource": "arn:aws:events:us-east-1:123456789012:rule/MyRule"
        }
    ]
}
```

규칙에 추가할 수 있는 대상을 제한하려면 `events:TargetArn` 조건 키를 사용합니다. 다음 예제에서와 같이 Lambda 함수로만 대상을 제한할 수 있습니다.

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

EventBridge에 서비스 연결 역할 사용

Amazon EventBridge는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 링크 역할은 EventBridge에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 링크 역할은

EventBridge에서 사전 정의하며 서비스에서 다른 AWS 서비스를 자동으로 호출하기 위해 필요한 모든 권한을 포함합니다.

주제

역할을 사용하여 API 대상의 시크릿 생성

Amazon EventBridge은(는) AWS Identity and Access Management(IAM) service-linked roles(서비스 링크 역할)을 사용합니다. 서비스 링크 역할은 EventBridge에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 EventBridge에서 사전 정의하며 서비스에서 다른 AWS 서비스를 자동으로 직접적으로 호출하기 위해 필요한 모든 권한을 포함합니다.

서비스 연결 역할을 통해 EventBridge 설정이 쉬워지는데 필요한 권한을 수동으로 추가할 필요가 없기 때문입니다. EventBridge에서 서비스 연결 역할 권한을 정의하므로, 달리 정의되지 않은 한 EventBridge에서만 해당 역할을 맡을 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 부주의로 삭제할 수 없기 때문에 EventBridge 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는 AWS 서비스를 참조하고 서비스 연결 역할(Service-linked roles) 열에 예(Yes)가 있는 서비스를 찾으세요. 해당 서비스에 대한 서비스 링크 역할 설명서를 보려면 예 링크를 선택합니다.

EventBridge에 대한 서비스 링크 역할 권한

EventBridge는 라는 이름의 AWSServiceRoleForAmazonEventBridgeApiDestinations서비스 연결 역할을 사용합니다. 에서 만든 Secrets Manager 시크릿에 액세스할 수 있게 합니다. EventBridge

AWSServiceRoleForAmazonEventBridgeApiDestinations 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- `apidestinations.events.amazonaws.com`

Policy라는 역할 권한 AmazonEventBridgeApiDestinationsServiceRole정책을 사용하면 지정된 리소스에서 다음 작업을 EventBridge 완료할 수 있습니다.

- 작업: `secrets created for all connections by EventBridge`에 대한 `create, describe, update and delete secrets; get and put secret values`

사용자, 그룹 또는 역할이 서비스 연결 역할을 생성, 편집 또는 삭제할 수 있도록 사용 권한을 구성해야 합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하십시오.

EventBridge에 대한 서비스 링크 역할 생성

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. AWS Management ConsoleAWS CLI, 또는 AWS API에서 연결을 생성하면 서비스 연결 역할이 자동으로 EventBridge 생성됩니다.

> **Important**
>
> 이러한 서비스 연결 역할은 해당 역할이 지원하는 기능을 사용하는 다른 서비스에서 작업을 완료했을 경우 계정에 나타날 수 있습니다. EventBridge서비스 연결 역할을 지원하기 시작한 2021년 2월 11일 이전에 서비스를 사용하고 있었다면 계정에 역할을 EventBridge 생성하십시오 AWSServiceRoleForAmazonEventBridgeApiDestinations. 자세한 내용은 내 AWS 계정에 표시되는 새 역할을 참조하십시오.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 연결을 생성하면 서비스 연결 역할이 다시 EventBridge 생성됩니다.

EventBridge에 대한 서비스 링크 역할 편집

EventBridge에서는 AWSServiceRoleForAmazonEventBridgeApiDestinations 서비스 연결 역할을 편집하도록 허용하지 않습니다. 서비스 링크 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하십시오.

EventBridge에 대한 서비스 링크 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제하는 것이 좋습니다. 따라서 적극적으로 모니터링하거나 유지하지 않는 미사용 엔터티가 없도록 합니다. 단, 서비스 연결 역할을 정리해야 수동으로 삭제할 수 있습니다.

서비스 연결 역할을 정리

IAM을 사용하여 서비스 연결 역할을 삭제하기 전에 먼저 역할에서 사용되는 리소스를 삭제해야 합니다.

> **Note**
>
> 리소스를 삭제하려 할 때 EventBridge 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하십시오.

AWSServiceRoleForAmazonEventBridgeApiDestinations에서 사용하는 EventBridge 리소스를 삭제하려면(콘솔)

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 통합에서 API 대상을 선택한 다음 연결 탭을 선택합니다.
3. 연결을 선택한 다음 삭제를 선택합니다.

AWSServiceRoleForAmazonEventBridgeApiDestinations에서 사용하는 EventBridge 리소스를 삭제하려면(AWS CLI)

- 다음 명령을 `delete-connection` 사용하세요.

AWSServiceRoleForAmazonEventBridgeApiDestinations에서 사용하는 EventBridge 리소스를 삭제하려면(API)

- 다음 명령을 `DeleteConnection` 사용하세요.

수동으로 서비스 연결 역할 삭제

IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 AWSServiceRoleForAmazonEventBridgeApiDestinations 서비스 연결 역할을 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스에 연결 역할 삭제를 참조하십시오.

EventBridge 서비스 링크 역할이 지원되는 리전

EventBridge에서는 서비스를 사용할 수 있는 모든 리전에서 서비스 링크 역할 사용을 지원합니다. 자세한 내용은 AWS 리전 및 엔드포인트 단원을 참조하십시오.

스키마 검색을 위한 역할 사용

Amazon EventBridge은(는) AWS Identity and Access Management(IAM) service-linked roles(서비스 링크 역할)을 사용합니다. 서비스 링크 역할은 EventBridge에 직접 연결된 고유한 유형의 IAM 역할입

니다. 서비스 연결 역할은 EventBridge에서 사전 정의하며 서비스에서 다른 AWS 서비스를 자동으로 직접적으로 호출하기 위해 필요한 모든 권한을 포함합니다.

서비스 연결 역할을 통해 EventBridge 설정이 쉬워지는데 필요한 권한을 수동으로 추가할 필요가 없기 때문입니다. EventBridge에서 서비스 연결 역할 권한을 정의하므로, 달리 정의되지 않은 한 EventBridge에서만 해당 역할을 맡을 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 부주의로 삭제할 수 없기 때문에 EventBridge 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는 AWS 서비스를 참조하고 서비스 연결 역할(Service-linked roles) 열에 예(Yes)가 있는 서비스를 찾으세요. 해당 서비스에 대한 서비스 링크 역할 설명서를 보려면 예 링크를 선택합니다.

EventBridge에 대한 서비스 링크 역할 권한

EventBridgeAWSServiceRoleForSchemas—라는 이름의 서비스 연결 역할을 사용합니다. 스키마에서 만든 Amazon EventBridge 관리형 규칙에 권한을 부여합니다.

AWSServiceRoleForSchemas 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- `schemas.amazonaws.com`

라는 역할 권한 정책을 AmazonEventBridgeSchemasServiceRolePolicy사용하면 지정된 리소스에서 다음 작업을 EventBridge 완료할 수 있습니다.

- 작업: `all managed rules created by EventBridge`에 대한 `put, enable, disable, and delete rules; put and remove targets; list targets per rule`

사용자, 그룹 또는 역할이 서비스 연결 역할을 생성, 편집 또는 삭제할 수 있도록 사용 권한을 구성해야 합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하십시오.

EventBridge에 대한 서비스 링크 역할 생성

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. AWS Management ConsoleAWS CLI, 또는 AWS API에서 스키마 검색을 수행하면 서비스 연결 역할이 자동으로 EventBridge 생성됩니다.

> **Important**
>
> 이러한 서비스 연결 역할은 해당 역할이 지원하는 기능을 사용하는 다른 서비스에서 작업을 완료했을 경우 계정에 나타날 수 있습니다. EventBridge서비스 연결 역할을 지원하기 시작한 2019년 11월 27일 이전에 서비스를 사용하고 있었다면 계정에 역할을 EventBridge 생성하십시오 AWSServiceRoleForSchemas. 자세한 내용은 내 AWS 계정에 표시되는 새 역할을 참조하십시오.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 스키마 검색을 수행하면 서비스 연결 역할이 다시 EventBridge 생성됩니다.

EventBridge에 대한 서비스 링크 역할 편집

EventBridge에서는 AWSServiceRoleForSchemas 서비스 연결 역할을 편집하도록 허용하지 않습니다. 서비스 링크 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하십시오.

EventBridge에 대한 서비스 링크 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제하는 것이 좋습니다. 따라서 적극적으로 모니터링하거나 유지하지 않는 미사용 엔터티가 없도록 합니다. 단, 서비스 연결 역할을 정리해야 수동으로 삭제할 수 있습니다.

서비스 연결 역할을 정리

IAM을 사용하여 서비스 연결 역할을 삭제하기 전에 먼저 역할에서 사용되는 리소스를 삭제해야 합니다.

> **Note**
>
> 리소스를 삭제하려 할 때 EventBridge 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하십시오.

AWSServiceRoleForSchemas에서 사용하는 EventBridge 리소스를 삭제하려면(콘솔)

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.

2. 버스에서 이벤트 버스를 선택한 다음 이벤트 버스를 선택합니다.
3. [스톱 디스커버리] 를 선택합니다.

AWSServiceRoleForSchemas에서 사용하는 EventBridge 리소스를 삭제하려면(AWS CLI)

- 다음 명령을 `delete-discoverer` 사용하세요.

AWSServiceRoleForSchemas에서 사용하는 EventBridge 리소스를 삭제하려면(API)

- 다음 명령을 `DeleteDiscoverer` 사용하세요.

수동으로 서비스 연결 역할 삭제

IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 AWSServiceRoleForSchemas 서비스 연결 역할을 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스에 연결 역할 삭제를 참조하십시오.

EventBridge 서비스 링크 역할이 지원되는 리전

EventBridge에서는 서비스를 사용할 수 있는 모든 리전에서 서비스 링크 역할 사용을 지원합니다. 자세한 내용은 AWS 리전 및 엔드포인트 단원을 참조하십시오.

를 사용하여 Amazon EventBridge API 호출 로깅 AWS CloudTrail

Amazon EventBridge 사용자 AWS CloudTrail, 역할 또는 담당자가 수행한 작업의 기록을 제공하는 서비스와 통합되어 AWS 서비스있습니다. CloudTrail 모든 API 호출을 EventBridge 이벤트로 캡처합니다. 캡처된 호출에는 EventBridge 콘솔에서의 호출 및 EventBridge API 작업에 대한 코드 호출이 포함됩니다. 에서 수집한 CloudTrail 정보를 사용하여 요청을 받은 사람 EventBridge, 요청한 IP 주소, 요청 시기 및 추가 세부 정보를 확인할 수 있습니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 보안 인증 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트 사용자로 했는지 사용자 보안 인증으로 했는지 여부.
- IAM Identity Center 사용자를 대신하여 요청이 이루어졌는지 여부입니다.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 인증 정보를 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

CloudTrail 계정을 만들 AWS 계정 때 활성화되며 자동으로 CloudTrail 이벤트 기록에 액세스할 수 있습니다. CloudTrail 이벤트 기록은 지난 90일간의 기록된 관리 이벤트를 보고, 검색하고, 다운로드할 수 있고, 변경할 수 없는 기록을 제공합니다. AWS 리전자세한 내용은 사용 설명서의 CloudTrail 이벤트 기록 사용을 참조하십시오.AWS CloudTrail 이벤트 기록 조회에는 CloudTrail 요금이 부과되지 않습니다.

AWS 계정 지난 90일 동안 진행 중인 이벤트 기록을 보려면 트레일 또는 CloudTrail호수 이벤트 데이터 저장소를 생성하세요.

CloudTrail 트레일

트레일을 사용하면 CloudTrail Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 를 사용하여 생성된 모든 트레일은 멀티 AWS Management Console 리전입니다. AWS CLI를 사용하여 단일 리전 또는 다중 리전 추적을 생성할 수 있습니다. 계정의 모든 활동을 기록할 수 있으므로 멀티 리전 트레일을 생성하는 것이 좋습니다 AWS 리전 . 단일 리전 추적을 생성하는 경우 추적의 AWS 리전에 로깅된 이벤트만 볼 수 있습니다. 추적에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 Creating a trail for your AWS 계정 및 Creating a trail for an organization을 참조하세요.

트레일을 CloudTrail 생성하여 진행 중인 관리 이벤트의 사본 하나를 Amazon S3 버킷으로 무료로 전송할 수 있지만 Amazon S3 스토리지 요금이 부과됩니다. CloudTrail 요금에 대한 자세한 내용은 AWS CloudTrail 요금을 참조하십시오. Amazon S3 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하세요.

CloudTrail Lake 이벤트 데이터 스토어

CloudTrail Lake를 사용하면 이벤트에 대한 SQL 기반 쿼리를 실행할 수 있습니다. CloudTrail Lake는 행 기반 JSON 형식의 기존 이벤트를 Apache ORC 형식으로 변환합니다. ORC는 빠른 데이터 검색에 최적화된 열 기반 스토리지 형식입니다. 이벤트는 이벤트 데이터 스토어로 집계되며, 이벤트 데이터 스토어는 고급 이벤트 선택기를 적용하여 선택한 기준을 기반으로 하는 변경 불가능한 이벤트 컬렉션입니다. 이벤트 데이터 스토어에 적용하는 선택기는 어떤 이벤트가 지속되고 쿼리할 수 있는지 제어합니다. CloudTrail Lake에 대한 자세한 내용은 사용 설명서의 Lake 사용을 참조하십시오. AWS CloudTrail AWS CloudTrail

CloudTrail Lake 이벤트 데이터 저장 및 쿼리로 인해 비용이 발생합니다. 이벤트 데이터 스토어를 생성할 때 이벤트 데이터 스토어에 사용할 요금 옵션을 선택합니다. 요금 옵션에 따라 이벤트 모으기 및 저장 비용과 이벤트 데이터 스토어의 기본 및 최대 보존 기간이 결정됩니다. CloudTrail 요금에 대한 자세한 내용은 AWS CloudTrail 요금을 참조하십시오.

EventBridge 의 데이터 이벤트 CloudTrail

데이터 이벤트는 리소스 기반 또는 리소스에서 수행된 리소스 작업에 대한 정보를 제공합니다(예: Amazon S3 객체 읽기 또는 쓰기). 이를 데이터 영역 작업이라고도 합니다. 데이터 이벤트가 대량 활동인 경우도 있습니다. 기본적으로 데이터 이벤트를 기록하지 CloudTrail 않습니다. CloudTrail 이벤트 기록에는 데이터 이벤트가 기록되지 않습니다.

데이터 이벤트에는 추가 요금이 적용됩니다. CloudTrail 요금에 대한 자세한 내용은 AWS CloudTrail 요금을 참조하십시오.

CloudTrail 콘솔 또는 CloudTrail API 작업을 사용하여 EventBridge 리소스 유형에 대한 데이터 이벤트를 기록할 수 있습니다. AWS CLI데이터 이벤트를 로깅하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 Logging data events with the AWS Management Console 및 Logging data events with the AWS Command Line Interface를 참조하세요.

다음 표에는 데이터 이벤트를 기록할 수 있는 EventBridge 리소스 유형이 나열되어 있습니다. 데이터 이벤트 유형 (콘솔) 열에는 CloudTrail 콘솔의 데이터 이벤트 유형 목록에서 선택할 수 있는 값이 표시됩니다. resources.type 값 열에는 또는 `resources.type` API를 사용하여 고급 이벤트 선택기를 구성할 때 지정하는 값이 표시됩니다. AWS CLI CloudTrail 데이터 API 로깅 대상 CloudTrail 열에는 해당 리소스 유형에 대해 로깅된 API 호출이 표시됩니다. CloudTrail

데이터 이벤트 유형(콘솔)	resources.type 값	로깅된 데이터 API CloudTrail
이벤트 버스	`AWS::Events::EventBus`	• DescribeEventBus
이벤트 버스 규칙	`AWS::Events::Rule`	• DeleteRule • DescribeRule • DisableRule • EnableRule • ListRuleNamesByTarget • ListRules • ListTargetsByRule • PutRule • PutTargets • RemoveTargets • TestEventPattern
파이프	`AWS::Pipes::Pipe`	• CreatePipe • DeletePipe • DescribePipe • ListPipes • StartPipe • StopPipe • UpdatePipe

`eventName`, `readOnly` 및 `resources.ARN` 필드를 필터링하여 중요한 이벤트만 로깅하도록 고급 이벤트 선택기를 구성할 수 있습니다. 이러한 필드에 대한 자세한 내용은 AWS CloudTrail API 참조의 AdvancedFieldSelector 섹션을 참조하세요.

EventBridge 의 관리 이벤트 CloudTrail

관리 이벤트는 내 리소스에 대해 수행되는 관리 작업에 대한 정보를 제공합니다 AWS 계정. 이를 제어 영역 작업이라고도 합니다. 기본적으로 관리 이벤트를 CloudTrail 기록합니다.

Amazon EventBridge 모든 EventBridge 컨트롤 플레인 작업을 관리 이벤트로 기록합니다. EventBridge 로그되는 Amazon EventBridge 컨트롤 플레인 작업 목록은 Amazon EventBridge API 참조를 참조하십시오. CloudTrail

EventBridge 이벤트 예제

이벤트는 모든 소스의 단일 요청을 나타내며 요청된 API 작업, 작업 날짜 및 시간, 요청 파라미터 등에 대한 정보를 포함합니다. CloudTrail 로그 파일은 공개 API 호출의 정렬된 스택 트레이스가 아니므로 이벤트가 특정 순서로 표시되지 않습니다.

다음 예제는 PutRule 작업을 시연하는 CloudTrail 이벤트를 보여줍니다.

```
{
  "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"
}
```

CloudTrail 레코드 내용에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 레코드 내용을 참조하십시오.

CloudTrail Pipes에서 수행한 EventBridge 작업에 대한 로그 항목

EventBridge Pipes는 소스에서 이벤트를 읽거나, 강화를 호출하거나, 대상을 호출할 때 제공된 IAM 역할을 맡습니다. 모든 보강, 대상 및 Amazon SQS, Kinesis 및 DynamoDB 소스에서 계정에서 수행한 작업과 관련된 CloudTrail 항목의 경우 및 필드에 다음이 포함됩니다. `sourceIPAddress` `invokedBy` `pipes.amazonaws.com`

모든 보강, 대상, Amazon SQS, Kinesis 및 DynamoDB 소스에 대한 샘플 CloudTrail 로그 항목

```
{
  "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"
}
```

다른 모든 소스의 경우 CloudTrail 로그 항목 `sourceIPAddress` 필드에 동적 IP 주소가 포함되므로 통합 또는 이벤트 분류에 의존해서는 안 됩니다. 또한 이러한 항목에는 `invokedBy` 필드가 없습니다.

다른 모든 소스의 샘플 CloudTrail 로그 항목

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    ...
  },
  "eventTime": ",,,",
  "eventName": "...",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "Python-httplib2/0.8 (gzip)",
}
```

Amazon EventBridge의 규정 준수 검증

SOC, PCI, FedRAMP 및 HIPAA와 같은 서드 파티 감사자는 여러 AWS 규정 준수 프로그램의 일환으로 AWS 서비스의 보안 및 규정 준수를 평가합니다.

특정 규정 준수 프로그램의 범위 내에 있는 AWS 서비스 목록은 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하세요. 일반적인 정보는 AWS 규정 준수 프로그램을 참조하세요.

AWS Artifact를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 AWS Artifact에서 보고서 다운로드 를 참조하세요.

EventBridge 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다. AWS는 규정 준수를 지원할 다음과 같은 리소스를 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서 – AWS에서 보안 및 규정 준수에 중점을 둔 기본 환경을 배포하기 위한 아키텍처 고려 사항 및 단계입니다.
- HIPAA 보안 및 규정 준수 기술 백서 설계 - 기업에서 AWS를 사용하여 HIPAA를 준수하는 애플리케이션을 생성하는 방법입니다.
- AWS 규정 준수 리소스 - 워크북 및 가이드 콜렉션 입니다.
- AWS Config 개발자 가이드의 규칙을 사용하여 리소스 평가 – AWS Config를 사용하여 리소스 구성이 내부 사례, 업계 지침, 규정을 얼마나 잘 준수하는지 평가하는 방법에 대한 정보입니다.
- AWS Security Hub – 보안 업계 표준 및 모범 사례 준수 여부를 확인하는 데 도움이 되는 AWS 내 보안 상태에 대한 포괄적인 뷰입니다.

Amazon EventBridge 복원성

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전은 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며 이러한 가용 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크에 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 정보는 AWS 글로벌 인프라를 참조하세요.

Amazon EventBridge의 인프라 보안

관리형 서비스인 Amazon EventBridge는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스와 AWS의 인프라 보호 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안에 대한 모범 사례를 사용하여 AWS 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS에서 게시한 API 직접 호출을 사용하여 네트워크를 통해 EventBridge에 액세스합니다. 고객은 다음을 지원해야 합니다.

- 전송 계층 보안(TLS). TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 보안 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

모든 네트워크 위치에서 이러한 API 작업을 호출할 수 있으며 EventBridge에서 리소스 기반 액세스 정책을 사용할 수 있습니다. 여기에는 소스 IP 주소에 따른 제한이 포함될 수 있습니다. EventBridge 정책을 사용하여 특정 Amazon Virtual Private Cloud(VPC) 엔드포인트 또는 특정 VPC에서 액세스를 제어할 수도 있습니다. 그러면 AWS 네트워크의 특정 VPC에서만 특정 EventBridge 리소스에 대한 네트워크 액세스가 효과적으로 격리됩니다.

Amazon EventBridge의 구성 및 취약성 분석

구성 및 IT 제어는 AWS와 고객 간의 공동 책임입니다. 자세한 내용은 AWS 공동 책임 모델을 참조하세요.

아마존 모니터링 EventBridge

EventBridge 일치하는 이벤트 수부터 규칙에 의해 대상이 호출된 횟수까지 모든 항목에 대한 지표를 CloudWatch 1분마다 Amazon에 전송합니다.

다음 동영상은 CloudWatch 다음을 통한 모니터링 및 감사 EventBridge 행동을 검토합니다. 모니터링 및 감사 이벤트

주제

- EventBridge 메트릭
- EventBridge 지표의 측정기준

EventBridge 메트릭

AWS/Events 네임스페이스에 포함된 지표는 다음과 같습니다.

카운트를 단위로 사용하는 메트릭의 경우 Sum SampleCount and 가 가장 유용한 통계입니다.

RuleName차원만 지정하는 지표는 기본 이벤트 버스를 참조합니다. EventBusName및 RuleName 차원을 모두 지정하는 지표는 커스텀 이벤트 버스를 참조합니다.

지표	설명	차원	단위
DeadLetterInvocations	이벤트에 대한 응답으로 규칙 대상이 간접 호출되지 않은 횟수입니다. 여기에는 동일한 규칙을 다시 실행하여 무한 루프를 초래하는 간접 호출이 포함됩니다.	RuleName	개수
Events	에서 수집한 파트너 이벤트 수입니다. EventBridge	EventSourceName	개수
FailedInvocations	계속해서 실패한 간접 호출 수입니다. 여기에는 재시도된 간접 호출이나 재시도 이후 성공한 간접 호출은 포함되지 않습니다. 또한 DeadLetterInvocations 계산에 포함된 실패 간접 호출은 계산되지 않습니다.	RuleName	개수

지표	설명	차원	단위
	Note EventBridge 0이 아닌 CloudWatch 경우에만 이 지표를 전송합니다.		
`Invocations`	이벤트에 대한 응답으로 규칙에 의해 대상이 간접 호출된 횟수입니다. 여기에는 성공한 간접 호출과 실패한 간접 호출이 모두 포함되지만, 병목 현상에 걸리거나 재시도하여 계속해서 실패한 간접 호출은 포함되지 않습니다. `DeadLetterInvocations` 는 포함되지 않습니다. **Note** EventBridge 0이 아닌 CloudWatch 경우에만 이 지표를 전송합니다.	없음, RuleName	개수
`InvocationAttempts`	대상 호출을 EventBridge 시도한 횟수.	None	개수
`InvocationsCreated`	각 이벤트에 대한 응답으로 생성된 총 간접 호출 수입니다. 이 지표는 초당 EventBridge 트랜잭션 서비스 할당량의 호출 스로틀 한도 사용률을 모니터링하는 데 주로 사용됩니다.	None	개수

지표	설명	차원	단위
`InvocationsFailedToBeSentToDlq`	DLQ(Dead Letter Queue)로 이동할 수 없는 간접 호출 수입니다. DLQ(Dead Letter Queue) 오류는 권한 오류, 사용할 수 없는 리소스 또는 크기 제한으로 인해 발생합니다. **Note** EventBridge 0이 아닌 경우에만 이 지표를 CloudWatch 전송합니다.	RuleName	개수
`IngestionToInvocationCompleteLatency`	이벤트 수집부터 첫 번째 성공적인 간접 호출 시도 완료까지 걸린 시간입니다.	EventBusName, 없음, RuleName	밀리초
`IngestionToInvocationStartLatency`	이벤트를 인제스트한 시점부터 대상을 처음 EventBridge 호출할 때까지의 이벤트 처리 시간입니다.	EventBusName, 없음, RuleName	밀리초
`InvocationsSentToDlq`	DLQ(Dead Letter Queue)로 이동된 간접 호출 수입니다. **Note** EventBridge 이 메트릭을 0이 아닌 CloudWatch 경우에만 전송합니다.	RuleName	개수
`MatchedEvents`	EventBusName 또는 EventSourceName 가 지정된 경우 모든 규칙과 일치하는 이벤트 수입니다. 를 지정하는 RuleName 경우, 특정 규칙과 일치하는 이벤트 수입니다.	EventBusName, EventSourceName, RuleName	개수

지표	설명	차원	단위
RetryInvocationAttempts	대상 간접 호출이 재시도된 횟수입니다. **Note** EventBridge 0이 아닌 CloudWatch 경우에만 이 지표를 전송합니다.	None	개수
SuccessfulInvocationAttempts	대상이 성공적으로 간접 호출된 횟수입니다.	None	개수
ThrottledRules	규칙 실행이 제한된 횟수입니다. 해당 규칙에 대한 간접 호출이 지연될 수 있습니다. 자세한 내용은 ???의 Invocations throttle limit in transactions per second를 참조하세요.	EventBusName, 없음, RuleName	개수
TriggeredRules	실행되고 모든 이벤트와 일치하는 규칙 수입니다. 규칙이 CloudWatch 트리거되기 전까지는 이 지표가 표시되지 않습니다.	EventBusName, 없음, RuleName	개수

EventBridge PutEvents 지표

AWS/Events 네임스페이스는 PutEvents API 요청과 관련된 다음 지표를 포함합니다.

카운트를 단위로 사용하는 메트릭의 경우 Sum SampleCount and 가 가장 유용한 통계입니다.

지표	설명	차원	단위
PutEventsApproxima	수신된 PutEvents 요청의 대략적인 수입니다.	None	개수

지표	설명	차원	단위
teCallCount			
PutEventsApproximateFailedCount	실패한 PutEvents 요청의 대략적인 수입니다.	None	개수
PutEventsApproximateSuccessCount	성공한 PutEvents 요청의 대략적인 수입니다.	None	개수
PutEventsApproximateThrottledCount	제한으로 인해 거부된 PutEvents 요청 수입니다.	None	개수
PutEventsEntriesCount	PutEvents 요청에 포함된 이벤트 항목 수입니다.	None	개수
PutEventsFailedEntriesCount	수집에 실패한 PutEvents 요청에 포함된 이벤트 항목 수입니다.	None	개수
PutEventsLatency	PutEvents 요청당 소요되는 시간입니다.	None	밀리초
PutEventsRequestSize	PutEvents 요청의 크기입니다.	None	바이트

EventBridge PutPartnerEvents 지표

AWS/Events 네임스페이스는 PutPartnerEvents API 요청과 관련된 다음 지표를 포함합니다.

> **Note**
>
> EventBridge 이벤트를 보내는 SaaS 파트너 계정의 PutPartnerEvents요청과 관련된 지표만 포함합니다. 자세한 내용은 ??? 섹션을 참조하세요.

카운트를 단위로 사용하는 메트릭의 경우 합계 및 통계가 가장 유용한 SampleCount 경향이 있습니다.

지표	설명	차원	단위
PutPartnerEventsApproximateCallCount	수신된 PutPartnerEvents 요청의 대략적인 수입니다.	None	개수
PutPartnerEventsApproximateFailedCount	실패한 PutPartnerEvents 요청의 대략적인 수입니다.	None	개수
PutPartnerEventsApproximateThrottledCount	제한으로 인해 거부된 PutPartnerEvents 요청 수입니다.	None	개수
PutPartnerEventsApproximateSuccessCount	성공한 PutPartnerEvents 요청의 대략적인 수입니다.	None	개수

지표	설명	차원	단위
PutPartnerEventsEntriesCount	PutPartnerEvents 요청에 포함된 이벤트 항목 수입니다.	None	개수
PutPartnerEventsFailedEntriesCount	수집에 실패한 PutPartnerEvents 요청에 포함된 이벤트 항목 수입니다.	None	개수
PutPartnerEventsLatency	PutPartnerEvents 요청당 소요되는 시간입니다.	None	밀리초

EventBridge 지표의 측정기준

EventBridge 지표에는 차원 또는 정렬 가능한 속성이 있으며, 이러한 속성은 다음과 같습니다.

측정기준	설명
EventBusName	이벤트 버스 이름을 기준으로 사용할 수 있는 지표를 필터링합니다.
EventSourceName	파트너 이벤트 소스 이름을 기준으로 사용할 수 있는 지표를 필터링합니다.
RuleName	규칙 이름을 기준으로 사용할 수 있는 지표를 필터링합니다.

아마존 문제 해결 EventBridge

이 섹션의 단계를 사용하여 Amazon EventBridge 문제를 해결할 수 있습니다.

주제

규칙이 실행되었지만 Lambda 함수가 간접 호출되지 않음

Lambda 함수가 실행되지 않는 한 가지 이유는 적절한 권한이 없는 경우입니다.

Lambda 함수에 대한 권한을 확인하려면

1. AWS CLI를 사용하여 함수 및 AWS 지역과 함께 다음 명령을 실행합니다.

```
aws lambda get-policy --function-name MyFunction --region us-east-1
```

다음과 같이 출력되어야 합니다.

```
{
    "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. 다음과 같은 메시지가 나타나는 경우

```
A client error (ResourceNotFoundException) occurred when calling the GetPolicy
 operation: The resource you requested does not exist.
```

또는 이러한 출력 화면은 나타났지만 정책에서 신뢰할 수 있는 개체로서 events.amazonaws.com의 위치를 찾을 수 없는 경우에는 다음 명령을 실행합니다.

```
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. 출력에 `SourceAccount` 필드가 포함되어 있으면 이를 제거해야 합니다. `SourceAccount`설정으로 EventBridge 인해 함수를 호출할 수 없습니다.

> **Note**
>
> 정책이 잘못된 경우 규칙을 제거한 다음 다시 추가하여 EventBridge 콘솔에서 규칙을 편집할 수 있습니다. 그러면 EventBridge 콘솔이 대상에 대해 올바른 권한을 설정합니다.
> 특정한 Lambda 별칭 또는 버전을 사용하고 있는 경우 다음 명령과 같이 `aws lambda get-policy` 및 `aws lambda add-permission` 명령에 `--qualifier` 파라미터를 추가합니다.

```
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
```

방금 규칙을 생성 또는 수정했지만, 테스트 이벤트와 일치하지 않음

규칙이나 규칙의 대상을 변경한다고 수신 이벤트가 즉시 새로운 규칙이나 업데이트된 규칙에 대한 매칭을 시작 또는 중지하지는 않습니다. 변경 사항이 적용될 때까지 잠시 기다리세요.

잠시 후에도 이벤트가 일치하지 `TriggeredRules` `Invocations` 않으면 CloudWatch 측정항목 및 `FailedInvocations` 규칙을 확인하세요. 이러한 지표에 대한 자세한 내용은 Amazon 모니터링을 참조하십시오 EventBridge.

규칙이 AWS 서비스의 이벤트와 일치하도록 의도된 경우 다음 중 하나를 수행하십시오.

- `TestEventPattern` 작업을 사용하여 규칙의 이벤트 패턴이 테스트 이벤트와 일치하는지 테스트합니다. 자세한 내용은 Amazon EventBridge API 참조를 참조하십시오 TestEventPattern.
- EventBridge 콘솔에서 샌드박스를 사용하십시오.

`ScheduleExpression`에 지정한 시간에 내 규칙이 실행되지 않음

UTC+0 시간대에 규칙이 예약이 되어 있는지 확인합니다. `ScheduleExpression`이 올바르면 방금 규칙을 생성 또는 수정했지만, 테스트 이벤트와 일치하지 않음의 단계를 따릅니다.

내 규칙이 예상된 시간에 실행되지 않음

EventBridge 설정한 시작 시간으로부터 1분 이내에 규칙을 실행합니다. 규칙이 생성되는 즉시 런타임에 대한 카운트다운이 시작됩니다.

> **Note**
>
> 예약된 규칙의 전송 유형은 guaranteed입니다. 즉, 예상 시간마다 이벤트가 한 번 이상 트리거됩니다.

cron 표현식을 사용하여 지정된 시간에 대상을 간접 호출할 수 있습니다. 4시간마다 0분에 실행되는 규칙을 만들려면 다음 중 하나를 수행합니다.

- EventBridge 콘솔에서는 cron 표현식을 `0 0/4 * * ? *` 사용합니다.
- AWS CLI를 사용하면 `cron(0 0/4 * * ? *)` 표현식을 사용합니다.

예를 들어, 를 사용하여 4시간마다 TestRule 실행되는 규칙을 만들려면 다음 명령을 사용합니다. AWS CLI

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

5분마다 규칙을 실행하려면 다음 cron 표현식을 사용합니다.

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

cron 표현식을 사용하는 EventBridge 규칙의 가장 정확한 해결 방법은 1분입니다. 예약된 규칙은 지정한 시간(분) 이내에 실행되지만 정확한 초 단위로 실행되지는 않습니다.

대상 서비스가 분산되어 있기 때문에 EventBridge 스케줄링된 규칙이 실행되는 시간과 대상 서비스가 대상 리소스에서 작업을 수행하는 시간 사이에 몇 초의 지연이 있을 수 있습니다.

내 규칙은 AWS 글로벌 서비스 API 호출과 일치하지만 실행되지 않았습니다.

AWS IAM 및 Amazon Route 53과 같은 글로벌 서비스는 미국 동부 (버지니아 북부) 지역에서만 사용할 수 있으므로 글로벌 서비스의 AWS API 호출 이벤트는 해당 지역에서만 사용할 수 있습니다. 자세한 정보는 서비스의 이벤트 AWS을 참조하세요.

규칙이 실행될 때 내 규칙과 연관된 IAM 역할이 무시됨

EventBridge Kinesis 스트림으로 이벤트를 보내는 규칙에 IAM 역할만 사용합니다. Lambda 함수 또는 Amazon SNS 주제를 간접 호출하는 규칙에 대해 리소스 기반 권한을 제공해야 합니다.

제공된 IAM 역할을 수임할 때 사용할 EventBridge 수 있도록 지역 AWS STS 엔드포인트가 활성화되어 있는지 확인하십시오. 자세한 내용은 IAM 사용 설명서의 AWS STSAWS 지역에서의 활성화 및 비활성화를 참조하십시오.

내 규칙에는 리소스와 일치해야 하는 이벤트 패턴이 있지만 일치하는 이벤트가 없음

에 있는 대부분의 서비스는 Amazon 리소스 이름 (ARN) 에서 콜론 (:) 또는 슬래시 (/) 를 동일한 문자로 AWS 취급하지만 이벤트 패턴 및 규칙에서 정확히 일치하는 문자를 EventBridge 사용합니다. 따라서 이벤트 패턴을 만들 때 일치시킬 이벤트에서 ARN 구문과 일치하도록 정확한 ARN 문자를 사용해야 합니다.

의 AWS API 호출 이벤트와 같은 일부 이벤트의 CloudTrail 경우 리소스 필드에 아무것도 없습니다.

내 이벤트를 대상에 전달할 때 지연됨

EventBridge 대상 리소스가 제한된 시나리오를 제외하고 최대 24시간 동안 대상에 이벤트를 전달하려고 시도합니다. 이벤트가 이벤트 스트림에 도착하는 즉시 첫 번째 시도가 이루어집니다. 대상 서비스에 문제가 있는 경우 다른 배달 일정을 EventBridge 자동으로 조정합니다. 이벤트 도착 후 24시간이 경과하면 이벤트 전송 시도를 EventBridge 중지하고 지표를 게시합니다. `FailedInvocations` CloudWatch 대상에 성공적으로 전달하지 못한 이벤트를 저장하려면 DLQ를 설정하는 것이 좋습니다. 자세한 내용은 데드레터 대기열을 사용하여 전달되지 않은 이벤트를 처리합니다. 섹션을 참조하세요.

일부 이벤트가 내 대상으로 전달되지 않음

EventBridge 규칙의 대상이 장기간 제한되는 경우 전달을 재시도하지 EventBridge 않을 수 있습니다. 예를 들어 대상이 수신되는 이벤트 트래픽을 처리하도록 프로비저닝되지 않고 대상 서비스가 사용자를 대신하여 보내는 요청을 제한하고 있는 EventBridge 경우 전송을 재시도하지 않을 수 있습니다. EventBridge

한 개의 이벤트에 응답하기 위해 내 규칙이 한 번 이상 실행됨

드문 경우이지만, 단일 이벤트 또는 예약된 시간에서 동일한 규칙을 한 번 이상 실행하거나 트리거된 특정 규칙에서 동일한 대상을 한 번 이상 간접 호출할 수 있습니다.

무한 루프 방지

EventBridge에서는 규칙이 반복적으로 실행되는 무한 루프로 이어지는 규칙을 만들 수 있습니다. 무한 루프를 유발하는 규칙이 있는 경우 규칙을 수행하는 작업이 동일한 규칙과 일치하지 않도록 규칙을 다시 작성하세요.

예를 들어 Amazon S3 버킷에서 ACL이 변경되었음을 감지한 다음, 소프트웨어를 실행하여 ACL을 새 상태로 변경하는 규칙은 무한 루프를 유발합니다. 이 문제를 해결하는 한 가지 방법은 잘못된 상태에 있는 ACL과만 일치하도록 규칙을 다시 작성하는 것입니다.

무한 루프는 예상보다 높은 요금을 빠르게 야기할 수 있습니다. 따라서 요금이 지정한 한도를 초과할 경우 이를 알려줄 수 있는 예산 관리를 사용하는 것이 좋습니다. 자세한 내용은 예산을 통해 비용 관리 단원을 참조하세요.

내 이벤트가 대상 Amazon SQS 대기열에 전달되지 않음

Amazon SQS 대기열이 암호화된 경우 고객 관리형 KMS 키를 생성하고 KMS 키 정책에 다음 권한 섹션을 포함해야 합니다. 자세한 내용은 AWS KMS 권한 구성을 참조하십시오.

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

내 규칙이 실행되지만 내 Amazon SNS 주제에 어떤 메시지도 게시되지 않음

시나리오 1

Amazon SNS 주제에 메시지를 게시하려면 권한이 필요합니다. us-east-1을 해당 AWS CLI지역으로 바꾸고 주제 ARN을 사용하여 다음 명령을 사용합니다.

```
aws sns get-topic-attributes --region us-east-1 --topic-arn "arn:aws:sns:us-
east-1:123456789012:MyTopic"
```

올바른 권한을 사용하려면 정책 속성이 다음과 유사해야 합니다.

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

정책에 Publish 권한이 있는 events.amazonaws.com이 표시되지 않는 경우 먼저 현재 정책을 복사하고 다음 문을 명령문 목록에 추가하세요.

```
{\"Sid\":\"Allow_Publish_Events\",
\"Effect\":\"Allow\",\"Principal\":{\"Service\":\"events.amazonaws.com\"},
\"Action\":\"sns:Publish\",
```

```
\"Resource\":\"arn:aws:sns:us-east-1:123456789012:MyTopic\"}
```

그런 다음 를 사용하여 주제 속성을 설정하고 다음 명령을 AWS CLI사용하십시오.

```
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**
>
> 정책이 잘못된 경우 규칙을 제거한 후 다시 추가하여 EventBridge 콘솔에서 규칙을 편집할 수도 있습니다. EventBridge 대상에 올바른 권한을 설정합니다.

시나리오 2

SNS 주제가 암호화된 경우 KMS 키 정책에 다음 섹션을 포함해야 합니다.

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

Amazon SNS 주제와 관련된 규칙을 삭제한 EventBridge 후에도 Amazon SNS 주제에 대한 권한이 계속 남아 있습니다.

Amazon SNS를 대상으로 하는 규칙을 생성하면 사용자를 대신하여 Amazon SNS 주제에 권한을 EventBridge 추가합니다. 규칙을 생성한 직후 규칙을 삭제하면 Amazon SNS 주제에서 권한이 제거되지 EventBridge 않을 수 있습니다. 이 경우 `aws sns set-topic-attributes` 명령을 사용하여 주제에서 권한을 제거할 수 있습니다. 이벤트 전송을 위한 리소스 기반 권한에 대한 자세한 내용은 Amazon EventBridge에 리소스 기반 정책 사용 섹션을 참조하세요.

어떤 IAM 조건 키와 함께 EventBridge 사용할 수 있습니까?

EventBridge AWS-wide 조건 키 (IAM 사용 설명서의 IAM 및 AWS STS 조건 컨텍스트 키 참조) 와 에 나열된 키를 지원합니다. IAM 정책 조건을 사용하여 세분화된 액세스 제어 구현

EventBridge 규칙이 언제 위반되었는지 어떻게 알 수 있나요?

EventBridge 규칙 위반 시 다음 경보를 사용하여 알림을 받을 수 있습니다.

규칙 위반 시 이를 알리기 위해 경보를 생성하려면

1. https://console.aws.amazon.com/cloudwatch/ 에서 CloudWatch 콘솔을 엽니다.
2. 경보 생성을 선택합니다. 범주별 CloudWatch 지표 창에서 이벤트 지표를 선택합니다.
3. 지표 목록에서 을 선택합니다 FailedInvocations.
4. 그래프 위에서 통계, 합계를 선택합니다.
5. 기간으로 5분 등의 값을 선택합니다. 다음을 선택합니다.
6. 예를 들어, 경보 임계값에서 이름에 경보의 고유한 이름을 입력합니다 myFailedRules. 설명에는 규칙이 대상에 이벤트를 전달하지 않음 등의 경보 설명을 입력합니다.
7. 조건에 대해 >= 및 1을 선택합니다. 기간에는 10을 입력합니다.
8. 작업의 이 경보가 발생할 경우 항상에서 상태가 ALARM입니다를 선택합니다.
9. 알림 보내기에 대해 기존 Amazon SNS 주제를 선택하거나 새로 만듭니다. 주제를 새로 생성하려면 새 목록을 선택합니다. 새 Amazon SNS 주제의 이름을 입력합니다 (예:) myFailedRules.
10. 이메일 목록에서, 경보가 ALARM 상태로 변경될 때 알림을 보낼 이메일 주소 목록을 쉼표로 구분하여 입력합니다.
11. 경보 생성을 선택합니다.

Amazon EventBridge 할당량

EventBridge의 대부분의 측면에는 할당량이 있습니다.

주제

> **Note**
>
> EventBridge 스케줄러의 할당량 목록은 EventBridge 스케줄러 사용 안내서의 EventBridge 스케줄러 할당량을 참조하십시오.

EventBridge 할당량

EventBridge에는 다음과 같은 할당량이 있습니다.

Service Quotas 콘솔은 EventBridge 할당량에 대한 정보를 제공합니다. 기본 할당량을 볼 수 있을 뿐만 아니라 Service Quotas 콘솔을 사용하여 조정 가능한 할당량에 대한 할당량 증가를 요청할 수 있습니다.

명칭	기본값	조정 가능	Description
API 대상	지원되는 각 리전: 3,000	예	리전별 계정당 API 대상의 최대 개수입니다.
연결	지원되는 각 리전: 3,000	예	리전별 계정당 최대 연결 수입니다.

명칭	기본값	조정 가능	Description
초당 트랜잭션의 CreateEndpoint 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	초당 CreateEndpoint API에 대한 요청의 최대 수입니다. 추가 요청은 제한됩니다.
초당 트랜잭션의 DeleteEndpoint 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	초당 DeleteEndpoint API에 대한 요청의 최대 수입니다. 추가 요청은 제한됩니다.
엔드포인트	지원되는 리전: 100	예	리전별 계정당 최대 엔드포인트 수입니다.
이벤트 버스 정책 크기	지원되는 각 리전: 10,240	예	최대 정책 크기(문자 수)입니다. 이 정책 크기는 다른 계정에 대한 액세스 권한을 부여할 때마다 증가합니다. DescribeEventBus API를 사용하여 현재 정책 및 크기를 확인할 수 있습니다.
이벤트 버스	지원되는 리전: 100	예	계정당 최대 이벤트 버스입니다.
이벤트 패턴 크기	지원되는 각 리전: 2,048	예	이벤트 패턴의 최대 크기(문자 수)입니다.

명칭	기본값	조정 가능	Description
간접 호출은 초당 트랜잭션의 한도를 조절합니다.	us-east-1: 초당 18,750 us-east-2: 초당 4,500 us-west-1: 초당 2,250 us-west-2: 초당 18,750 af-south-1: 초당 750 ap-northeast-1: 초당 2,250 ap-northeast-3: 초당 750 ap-southeast-1: 초당 2,250 ap-southeast-2: 초당 2,250 ap-southeast-3: 초당 750 eu-central-1: 초당 4,500 eu-south-1: 초당 750	예	간접 호출은 규칙과 일치하고 규칙의 대상으로 전송되는 이벤트입니다. 한도에 도달하면 간접 호출이 제한됩니다. 즉, 간접 호출이 계속 발생하지만 지연됩니다.

명칭	기본값	조정 가능	Description
	eu-west-1: 초당 18,750 eu-west-2: 초당 2,250 각각의 지원되는 다른 리전: 초당 1,100		
규칙 수	af-south-1: 100 eu-south-1: 100 각각의 지원되는 다른 리전: 300	예	이벤트 버스당 계정당 보유할 수 있는 최대 규칙 수

명칭	기본값	조정 가능	Description
PutEvents는 초당 트랜잭션 한도를 조절합니다.	us-east-1: 초당 10,000 us-east-2: 초당 2,400 us-west-1: 초당 1,200 us-west-2: 초당 10,000 af-south-1: 초당 400 ap-northeast-1: 초당 1,200 ap-northeast-3: 초당 400 ap-southeast-1: 초당 1,200 ap-southeast-2: 초당 1,200 ap-southeast-3: 초당 400 eu-central-1: 초당 2,400 eu-south-1: 초당 400	예	초당 PutEvents API에 대한 요청의 최대 수입니다. 추가 요청은 제한됩니다.

명칭	기본값	조정 가능	Description
	eu-west-1: 초당 10,000 eu-west-2: 초당 1,200 각각의 지원되는 다른 리전: 초당 600		
API 대상별 간접 호출 비율	지원되는 각 리전: 초당 300	예	리전별 계정당 각 API 대상 엔드포인트로 전송할 초당 최대 간접 호출 수. 할당량이 충족되면 해당 API 엔드포인트에 대한 향후 간접 호출이 제한됩니다. 간접 호출은 계속 발생하지만 지연됩니다.
규칙당 대상	지원되는 각 리전: 5	아니요	규칙에 연결할 수 있는 최대 대상 수
초당 트랜잭션의 스로틀 제한	지원되는 각 리전: 초당 50	예	모든 EventBridge API 작업에 대한 초당 최대 요청 수(PutEvents 제외)입니다. 추가 요청은 제한됩니다.
초당 트랜잭션의 UpdateEndpoint 스로틀 제한	지원되는 각 리전: 초당 5개	아니요	초당 UpdateEndpoint API에 대한 요청의 최대 수입니다. 추가 요청은 제한됩니다.

또한 EventBridge에는 Service Quotas 콘솔을 통해 관리되지 않는 다음과 같은 할당량이 있습니다.

명칭	기본값	Description
이벤트 버스	지원되는 각 리전: 100	계정당 최대 이벤트 버스입니다.
이벤트 버스 정책 크기	지원되는 각 리전: 10240	최대 정책 크기(문자 수)입니다. 이 정책 크기는 다른 계정에 대한 액세스 권한을 부여할 때마다 증가합니다. `DescribeEventBus` API를 사용하여 현재 정책 및 크기를 확인할 수 있습니다.
이벤트 패턴 크기	지원되는 각 리전: 2048	이벤트 패턴의 최대 크기(문자 수)입니다. 최대 4096자까지 조정할 수 있습니다. 더 높은 최대 한도에 대한 요구 사항이 있는 경우 지원팀에 문의하세요.
와일드카드가 포함된 규칙	지원되는 각 리전: 이벤트 버스당 규칙 30개	와일드카드가 포함된 이벤트 필터를 포함할 수 있는 최대 규칙 수 (계정당 이벤트 버스당). 이 할당량은 조정할 수 없습니다. 이벤트 패턴에서 와일드카드를 사용하는 방법에 대한 자세한 내용은 ??? 단원을 참조하십시오.
스키마 검색 레벨	지원되는 각 리전: 255개 레벨	스키마 검색이 중첩된 이벤트를 유추하는 최대 레벨 수입니다. 255 레벨을 넘는 모든 이벤트는 무시됩니다.

리전별 PutPartnerEvents 할당량

더 높은 최대 한도에 대한 요구사항이 있는 경우 지원팀에 문의하세요.

리전	초당 트랜잭션 수
• AWS GovCloud(미국 서부) • AWS GovCloud(미국 동부) • 미국 동부(버지니아 북부)	PutPartnerEvents에는 모든 리전에서 기본적으로 초당 1,400개의 처리량 요청과 초당 3,600개의 버스트 요청이라는 소프트 제한이 있습니다.

리전	초당 트랜잭션 수
• 미국 동부(오하이오) • 미국 서부(캘리포니아 북부) • 미국 서부(오레곤) • 아프리카(케이프타운) • 아시아 태평양(홍콩) • 아시아 태평양(뭄바이) • 아시아 태평양(오사카) • 아시아 태평양(서울) • 아시아 태평양(싱가포르) • 아시아 태평양(시드니) • 아시아 태평양(도쿄) • 캐나다(중부) • 유럽(프랑크푸르트) • 유럽(아일랜드) • 유럽(런던) • 유럽(밀라노) • 유럽(파리) • 유럽(스톡홀름) • 유럽(밀라노) • 남아메리카(상파울루) • 중국(닝샤) • 중국(베이징)	

EventBridge 스키마 레지스트리 할당량

EventBridge 스키마 레지스트리에는 다음과 같은 할당량이 있습니다.

Service Quotas 콘솔은 EventBridge 할당량에 대한 정보를 제공합니다. 기본 할당량을 볼 수 있을 뿐만 아니라 Service Quotas 콘솔을 사용하여 조정 가능한 할당량에 대한 할당량 증가를 요청할 수 있습니다.

명칭	기본값	조정 가능	Description
검색된 스키마	지원되는 각 리전: 200	예	현재 리전에서 생성할 수 있는 검색된 스키마 레지스트리의 최대 스키마 수
검색기	지원되는 각 리전: 10	예	현재 리전에서 생성할 수 있는 최대 검색기 수입니다.
레지스트리	지원되는 각 리전: 10	예	현재 리전에서 생성할 수 있는 최대 레지스트리 수입니다.
SchemaVersions	지원되는 각 리전: 100	예	현재 리전에서 생성할 수 있는 스키마당 최대 버전 수입니다.
스키마	지원되는 각 리전: 100	예	현재 리전에서 생성할 수 있는 레지스트리당 최대 스키마 수입니다. (검색된 스키마 레지스트리 제외)

EventBridge 파이프 할당량

EventBridge 파이프 EventBridge Pipes는 다음과 같은 할당량이 있습니다. 더 높은 최대 한도에 대한 요구사항이 있는 경우 지원팀에 문의하세요.

Resource	리전	기본 한도
계정별 동시 파이프 실행 수	• AWS GovCloud(미국 서부) • AWS GovCloud(미국 동부) • 중국(닝샤)	1000

Resource	리전	기본 한도
	• 중국(베이징) • 아시아 태평양(오사카) • 아프리카(케이프타운) • 유럽(밀라노) • 미국 동부(오하이오) • 유럽(프랑크푸르트) • 미국 서부(캘리포니아 북부) • 유럽(런던) • 아시아 태평양(시드니) • 아시아 태평양(도쿄) • 아시아 태평양(싱가포르) • 캐나다(중부) • 유럽(파리) • 유럽(스톡홀름) • 남아메리카(상파울루) • 아시아 태평양(서울) • 아시아 태평양(뭄바이) • 아시아 태평양(홍콩) • 중동(바레인) • 중국(닝샤) • 중국(베이징) • 아시아 태평양(오사카) • 아프리카(케이프타운) • 유럽(밀라노)	
계정별 동시 파이프 실행 수	• 미국 동부(버지니아 북부) • 미국 서부(오레곤) • 유럽(아일랜드)	3000
계정당 파이프 수	모두	1000

아마존 EventBridge 태그

태그는 사용자가 또는 리소스에 AWS 할당하는 사용자 지정 속성 레이블입니다 AWS . EventBridge에서는 규칙 및 이벤트 버스에 태그를 할당할 수 있습니다. 각 리소스는 최대 50개의 태그를 보유할 수 있습니다.

태그를 사용하여 AWS 리소스를 식별하고 구성할 수 있습니다. 많은 AWS 서비스가 태그 지정을 지원하므로 서로 다른 서비스의 리소스에 동일한 태그를 할당하여 리소스가 관련되어 있음을 나타낼 수 있습니다. 예를 들어, EC2 인스턴스에 할당하는 EventBridge 규칙에 동일한 태그를 할당할 수 있습니다.

태그는 두 부분으로 구성됩니다.

- 태그 키(예: `CostCenter`, `Environment` 또는 `Project`)입니다.
  - 태그 키는 대소문자를 구별합니다.
  - 태그 키의 최대 길이는 UTF-8 형식의 유니코드 문자 128자입니다.
  - 각 리소스에 대해 각 태그 키는 고유해야 합니다.
  - 허용되는 문자는 UTF-8로 표현할 수 있는 문자, 숫자, 공백 및 . : + = @ _ / -(하이픈) 문자도 있습니다.
  - `aws:`접두사는 사용하도록 예약되어 있으므로 태그에는 사용할 AWS 수 없습니다. 이 접두사가 지정된 태그 키나 값은 편집하거나 삭제할 수 없습니다. 이 접두사가 지정된 태그는 리소스당 태그 수 제한에 포함되지 않습니다.
- 선택적 태그 값 필드(예: `111122223333` 또는 `Production`).
  - 각 태그 키는 하나의 값만 가질 수 있습니다.
  - 태그 값은 대소문자를 구분합니다.
  - 태그 값을 생략하는 것은 빈 문자열을 사용하는 것과 같습니다.
  - 태그 값의 최대 길이는 UTF-8 형식의 유니코드 문자 256자입니다.
  - 허용되는 문자는 UTF-8로 표현할 수 있는 문자, 숫자, 공백 및 . : + = @ _ / -(하이픈) 문자도 있습니다.

> **Tip**
>
> 모범 사례는 태그를 대문자로 사용할 것을 전략으로 결정하고 모든 리소스 유형에 대해 일관되게 해당 전략을 구현하는 것입니다. 예를 들어, `Costcenter`, `costcenter` 또는 `CostCenter`를 사용할지 결정하고 모든 태그에 대해 동일한 규칙을 사용합니다.

EventBridge 콘솔, EventBridge API 또는 를 사용하여 태그를 추가, 편집 또는 삭제할 수 있습니다. AWS CLI 자세한 내용은 다음 자료를 참조하십시오.

- TagResourceUntagResource, 및 아마존 EventBridge API ListTagsForResource레퍼런스에서
- 리소스 태그 지정, 리소스 태그 해제, 참조 내용 list-tags-for-resourceAWS CLI
- Resource Groups 사용 설명서의 Tag Editor 작업

문서 기록

다음 표에는 2019년 7월부터 출시되는 Amazon EventBridge 사용 설명서의 각 릴리스에서 변경된 주요 내용이 설명되어 있습니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

변경 사항	설명	릴리스 날짜
AWS 관리형 정책이 업데이트되었습니다.	AWS GovCloud (US) Regions 전용 `AmazonEventBridgeFullAccess` 사용되지 않으므로 `AmazonEventBridgeSchemasFullAccess` 정책에 포함되지 `iam:CreateServiceLinkedRole` 않습니다. • the section called “정책 업데이트”	2024년 5월 9일
이벤트 버스와 규칙에서 AWS CloudFormation 템플릿을 생성합니다.	이제 기존 Amazon EventBridge 이벤트 버스 및 규칙에서 AWS CloudFormation 템플릿을 생성할 수 있습니다. • Amazon EventBridge 이벤트 버스에서 AWS CloudFormation 템플릿 생성	2022년 11월 18일
EventBridge Pipes 설명서를 시작했습니다.	이제 선택적 필터링 및 보강 기능을 사용하여 소스를 대상에 연결하는 파이프를 생성할 수 있습니다. • 파이프	2022년 12월 1일
이벤트 버스와 규칙에서 AWS CloudFormation 템플릿을 생성합니다.	이제 기존 Amazon EventBridge 이벤트 버스 및 규칙에서 AWS CloudFormation 템플릿을 생성할 수 있습니다. • Amazon EventBridge 이벤트 버스에서 AWS CloudFormation 템플릿 생성	2022년 11월 18일
AmazonEventBridgePipesFullAccess	Amazon EventBridge Pipes에 대한 전체 액세스 권한을 제공합니다.	2022년 12월 1일

변경 사항	설명	릴리스 날짜
정책을 추가했습니다.	• EventBridge 파이프별 관리 정책	
AmazonEventBridgePipesReadOnlyAccess 정책이 추가되었습니다.	Amazon EventBridge Pipes에 대한 읽기 전용 액세스를 제공합니다. • EventBridge 파이프별 관리 정책	2022년 12월 1일
AmazonEventBridgePipesOperatorAccess 정책을 추가했습니다.	Amazon EventBridge Pipes에 대한 읽기 전용 및 운영자 (즉, Pipes 실행을 중지하고 시작할 수 있는 기능) 액세스 권한을 제공합니다. • EventBridge 파이프별 관리 정책	2022년 12월 1일
CloudWatchEventsFullAccess 정책을 업데이트했습니다.	`AmazonEventBridgeFullAccess` 와 일치하도록 업데이트되었습니다. • AmazonEventBridgeFullAccess 정책	2022년 12월 1일
CloudWatchEventsReadOnlyAccess 정책을 업데이트했습니다.	`AmazonEventBridgeReadOnlyAccess` 와 일치하도록 업데이트되었습니다. • AmazonEventBridgeReadOnlyAccess 정책	2022년 12월 1일
이벤트 패턴의 콘텐츠 필터링이 업데이트되었습니다.	이제 `suffix`, `equals-ignore-case` 및 `$or` 필터링 옵션을 사용해 이벤트 패턴을 생성할 수 있습니다. • Amazon EventBridge 이벤트 패턴의 콘텐츠 필터링	2022년 11월 14일

변경 사항	설명	릴리스 날짜
AmazonEventBridgeFullAccess 정책을 업데이트했습니다.	EventBridge 스키마 레지스트리 및 EventBridge 스케줄러를 사용하는 데 필요한 권한이 추가되었습니다. • AmazonEventBridgeFullAccess 정책	2022년 11월 10일
AmazonEventBridgeReadOnlyAccess 정책을 업데이트했습니다.	이제 EventBridge 스키마 레지스트리 및 EventBridge 스케줄러 정보를 볼 수 있습니다. • AmazonEventBridgeReadOnlyAccess 정책	2022년 11월 10일
이벤트 패턴의 콘텐츠 필터링이 업데이트되었습니다.	이제 `suffix`, `equals-ignore-case` 및 $or 필터링 옵션을 사용해 이벤트 패턴을 생성할 수 있습니다. • Amazon EventBridge 이벤트 패턴의 콘텐츠 필터링	2022년 11월 14일
AmazonEventBridgeFullAccess 정책을 업데이트했습니다.	EventBridge 스키마 레지스트리 및 EventBridge 스케줄러를 사용하는 데 필요한 권한이 추가되었습니다. • AmazonEventBridgeFullAccess 정책	2022년 11월 10일
AmazonEventBridgeReadOnlyAccess 정책을 업데이트했습니다.	이제 EventBridge 스키마 레지스트리 및 EventBridge 스케줄러 정보를 볼 수 있습니다. • AmazonEventBridgeReadOnlyAccess 정책	2022년 11월 10일
AmazonEventBridgeReadOnlyAccess 정책을 업데이트했습니다.	이제 엔드포인트 정보를 볼 수 있습니다. • AmazonEventBridgeReadOnlyAccess 정책	2022년 4월 7일

변경 사항	설명	릴리스 날짜
글로벌 엔드포인트에 대한 지원이 추가되었습니다.	Amazon은 EventBridge 이제 추가 비용 없이 애플리케이션이 지역별 내결함성을 유지할 수 있도록 글로벌 엔드포인트 사용을 지원합니다. 자세한 내용은 다음을 참조하세요. • 글로벌 엔드포인트 및 이벤트 복제를 통해 애플리케이션의 리전별 내결함성 설정 • `CreateEndpoint`	2022년 4월 7일
아카이브 및 이벤트 재생에 대한 지원이 추가되었습니다.	Amazon은 EventBridge 이제 아카이브를 사용하여 이벤트를 저장하고 이벤트 재생을 사용하여 아카이브에서 이벤트를 재생할 수 있도록 지원합니다. 자세한 내용은 다음을 참조하세요. • 아마존 EventBridge 이벤트 보관. • `CreateArchive` • `StartReplay`	2020년 11월 5일
DLQ(Dead Letter Queue)에 대한 지원과 대상에 대한 재시도 정책이 추가되었습니다.	Amazon은 EventBridge 이제 데드레터 대기열 사용 및 대상에 대한 재시도 정책 정의를 지원합니다. 자세한 내용은 다음을 참조하세요. • 데드레터 대기열을 사용하여 전달되지 않은 이벤트를 처리합니다.. • `PutTargets`	2020년 10월 12일
JSONSchema Draft4 형식 스키마에 대한 지원이 추가되었습니다.	Amazon은 EventBridge 이제 JSON/스키마 드래프트 4 형식의 스키마를 지원합니다. 이제 API를 사용하여 스키마를 내보낼 수도 있습니다. EventBridge 자세한 내용은 다음을 참조하세요. • 아마존 EventBridge 스키마 • `Export` EventBridge 스키마 레지스트리 API 참조에서	2020년 9월 28일

변경 사항	설명	릴리스 날짜
스키마 레지스트리의 EventBridge 리소스 기반 정책	Amazon EventBridge 스키마 레지스트리는 이제 리소스 기반 정책을 지원합니다. 추가 정보는 다음을 참조하세요. • Amazon EventBridge 스키마에 대한 리소스 기반 정책 • `Policy` EventBridge 스키마 레지스트리 API 참조에서 • RegistryPolicy AWS CloudFormation 사용 설명서의 리소스 유형	2020년 4월 30일
이벤트 버스 태그	이 릴리스에서는 이벤트 버스에 대한 태그를 만들고 관리할 수 있습니다. 이벤트 버스를 만들 때 태그를 추가하고 관련 API를 호출하여 기존 태그를 추가하거나 관리할 수 있습니다. 추가 정보는 다음을 참조하세요. • 아마존 EventBridge 태그 • 태그 기반 정책 • `TagResource` • `UntagResource` • `ListTagsForResource`	2020년 2월 24일
서비스 할당량 증가	EventBridge Amazon은 호출 및 요청에 대한 할당량을 늘렸습니다. `PutEvents` 할당량은 리전에 따라 다르며 필요한 경우 늘릴 수 있습니다.	2020년 2월 11일

변경 사항	설명	릴리스 날짜
대상 입력 변환에 대한 새로운 주제를 추가하고 Application Auto Scaling 이벤트에 대한 링크를 추가했습니다.	입력 변환기에 대한 설명서가 개선되었습니다. • 아마존 EventBridge 인풋 트랜스포메이션 • 입력 변환기를 사용하여 이벤트에서 데이터를 추출하고 해당 데이터를 대상에 입력 • 자습서: 입력 변환기를 사용하여 EventBridge가 이벤트 대상에 전달하는 내용을 사용자 지정 Application Auto Scaling Events에 대한 링크를 추가했습니다. • Application Auto Scaling 이벤트 및 EventBridge • 서비스의 이벤트 AWS	2019년 12월 20일
콘텐츠 기반 필터링		2019년 12월 19일
Amazon Augmented AI 이벤트 예제에 대한 링크가 추가되었습니다.	Amazon 개발자 안내서에 Amazon Augmented AI에 대한 예제 이벤트를 제공하는 SageMaker Amazon Augmented AI 주제 링크를 추가했습니다. 추가 정보는 다음을 참조하세요. • Amazon Augmented AI에서 이벤트 사용 • 서비스의 이벤트 AWS	2019년 12월 13일
Amazon Chime 이벤트 예제에 대한 링크가 추가되었습니다.	해당 서비스에 대한 예제 이벤트를 제공하는 Amazon Chime 주제에 대한 링크가 추가되었습니다. 추가 정보는 다음을 참조하세요. • 다음을 통한 Amazon Chime 자동화 EventBridge • 서비스의 이벤트 AWS	2019년 12월 12일

변경 사항	설명	릴리스 날짜
아마존 EventBridge 스키마	이제 Amazon에서 스키마를 관리하고 이벤트에 대한 코드 바인딩을 생성할 수 있습니다. EventBridge 추가 정보는 다음을 참조하세요. • 아마존 EventBridge 스키마 • EventBridge 스키마 API 레퍼런스 • EventSchemas 의 리소스 유형 참조 AWS CloudFormation	2019년 12월 1일
AWS CloudFormation 이벤트 버스 지원	AWS CloudFormation 이제 EventBus 리소스를 지원합니다. 또한 Rule EventBusPolicy 리소스와 Rule 리소스 모두에서 EventBusName 파라미터를 지원합니다. 자세한 내용은 Amazon EventBridge 리소스 유형 참조를 참조하십시오.	2019년 10월 7일
새로운 서비스	아마존의 첫 출시 EventBridge.	2019년 7월 11일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.