aws

ユーザーガイド

Amazon EventBridge

Amazon EventBridge: ユーザーガイド

Table of Contents

Amazon EventBridge とは

EventBridge は、イベントを使用してアプリケーションコンポーネント同士を接続するサーバーレスサービスです。これにより、スケーラブルなイベント駆動型アプリケーションを簡単に構築できます。イベント駆動型アーキテクチャとは、イベントの発信と応答によって連携する、ゆるやかに結合されたソフトウェアシステムを構築するスタイルです。イベント駆動型アーキテクチャは、俊敏性を高め、信頼性が高くスケーラブルなアプリケーションを構築するのに役立ちます。

EventBridge を使用して、自社開発アプリケーション、AWS サービス、サードパーティソフトウェアなどのソースから組織全体のコンシューマアプリケーションにイベントをルーティングできます。EventBridge では、イベントの取り込み、フィルタリング、変換、配信をシンプルかつ一貫性のある方法で行うことができるため、アプリケーションをすばやく構築できます。

以下の動画では Amazon EventBridge の機能を簡単に紹介しています。

EventBridge では次の 2 つの方法でイベントを処理できます: イベントバスとパイプ。

- イベントバスは、イベントを受信するルーターであり、ゼロ個以上のターゲットに配信します。イベントバスは、イベントをさまざまなソースから多数のターゲットにルーティングするのに適しており、オプションでターゲットに配信する前にイベントを変換できます。

  次の動画でイベントバスの概要を示します。

- パイプ EventBridge Pipes はポイントツーポイント統合を目的としています。各パイプは単一のソースからイベントを受け取り、処理して単一のターゲットに配信します。パイプでは、高度な変換やターゲットへの配信前のイベントのエンリッチメントもサポートされています。

パイプとイベントバスはよく一緒に使用されます。一般的なユースケースは、イベントバスをターゲットとするパイプを作成することです。パイプはイベントをイベントバスに送信し、イベントバスはそれらのイベントを複数のターゲットに送信します。たとえば、ソースに DynamoDB ストリームを使用し、ターゲットとしてイベントバスを含むパイプを作成できます。パイプは DynamoDB ストリームからイベントを受け取り、イベントバスに送信します。イベントバスは、イベントバスで指定したルールに従ってイベントを複数のターゲットに送信します。

EventBridge は、Amazon CloudWatch Events の進化形です。

EventBridge は、以前は Amazon CloudWatch Events と呼ばれていました。CloudWatch Events で作成したデフォルトのイベントバスとルールも EventBridge コンソールに表示されます。EventBridge では同じ CloudWatch Events API を使用するため、CloudWatch Events API を使用するコードに変化はありません。

EventBridge は CloudWatch イベントの機能を基に、パートナーイベント、スキーマレジストリ、EventBridge Pipes などの機能を備えています。EventBridge に追加された新機能は、CloudWatch Events には追加されません。詳細については、「???」を参照してください。

CloudWatch イベントで使い慣れている機能はすべて EventBridge にも搭載されています。これには次のようなものがあります。

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

イベントの機能をさらに発展させ、拡張する EventBridge の機能には以下が含まれます:

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

Amazon EventBridge のセットアップと前提条件

Amazon を使用するには EventBridge、 AWS アカウントが必要です。アカウントでは、Amazon EC2 などの のサービスを使用して、 EventBridge コンソールに表示されるイベントを生成できます。コマンドラインインターフェイスを使用してイベントを表示するように AWS Command Line Interface（ AWS CLI) をインストールして設定することもできます。

トピック

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. https://portal.aws.amazon.com/billing/signup を開きます。
2. オンラインの手順に従います。

   サインアップ手順の一環として、通話呼び出しを受け取り、電話キーパッドで検証コードを入力するように求められます。

   にサインアップすると AWS アカウント、 AWS アカウントのルートユーザーが作成されます。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用してルートユーザーアクセスが必要なタスクを実行してください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。https://aws.amazon.com/ の [アカウント] をクリックして、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理することができます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、 を保護し AWS アカウントのルートユーザー、 を有効にして AWS IAM Identity Center、日常的なタスクにルートユーザーを使用しないように管理ユーザーを作成します。

のセキュリティ保護 AWS アカウントのルートユーザー

1. ルートユーザーを選択し、 AWS アカウント E メールアドレスを入力して、アカウント所有者AWS Management Consoleとして にサインインします。次のページでパスワードを入力します。

   ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの「ルートユーザーとしてサインインする」を参照してください。
2. ルートユーザーの多要素認証 (MFA) を有効にします。

   手順については、「IAM ユーザーガイド」の AWS アカウント 「ルートユーザーの仮想 MFA デバイスを有効にする (コンソール)」を参照してください。

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「AWS IAM Identity Centerの有効化」を参照してください。
2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

   を ID ソース IAM アイデンティティセンターディレクトリ として使用する方法のチュートリアルについては、「 ユーザーガイド」の「デフォルト でユーザーアクセス IAM アイデンティティセンターディレクトリを設定するAWS IAM Identity Center 」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

  IAM Identity Center ユーザーを使用してサインインする方法については、「 AWS サインイン ユーザーガイド」の AWS 「 アクセスポータルへのサインイン」を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するというベストプラクティスに従ったアクセス許可セットを作成します。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」を参照してください。
2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「グループの参加」を参照してください。

Amazon EventBridge コンソールにサインインする

Amazon EventBridge コンソールにサインインするには

- にサインイン AWS Management Console し、https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。

アカウント認証情報

ルートユーザーの認証情報を使用して にアクセスできますが EventBridge、代わりに AWS Identity and Access Management (IAM) アカウントを使用することをお勧めします。IAM アカウントを使用して にアクセスする場合は EventBridge、次のアクセス許可が必要です。

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

詳細については、「認証」を参照してください。

のセットアップ AWS Command Line Interface

を使用してオペレーション AWS CLI を実行できます EventBridge 。

をインストールして設定する方法については AWS CLI、「 ユーザーガイド」の「 のセットアップ AWS Command Line InterfaceAWS Command Line Interface 」を参照してください。

リージョンのエンドポイント

を使用するには、デフォルトのリージョンエンドポイントを有効にする必要があります EventBridge。詳細については、IAM ユーザーガイドの「 AWS リージョン AWS STS での のアクティブ化と非アクティブ化」を参照してください。

Amazon の開始方法 EventBridge

の基礎 EventBridge は、イベントをターゲット にルーティングするルールを作成することです。このセクションでは、基本的なルールを作成します。特定のシナリオと特定のターゲットのチュートリアルについては、「Amazon EventBridge チュートリアル」を参照してください。

Amazon でルールを作成する EventBridge

イベントのルールを作成するには、 がルールのイベントパターンに一致するイベント EventBridge を受信したときに実行するアクションを指定します。イベントが一致すると、 は指定されたターゲットにイベント EventBridge を送信し、ルールで定義されたアクションをトリガーします。

AWS アカウントの AWS サービスがイベントを発行すると、常にアカウントのデフォルトのイベントバスに移動します。アカウントの AWS サービスからのイベントに一致するルールを作成するには、デフォルトのイベントバスに関連付ける必要があります。

AWS サービスのルールを作成するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで Rules] (ルール) を選択します。
3. ルールの作成 を選択します。
4. ルールの名前と説明を入力します。

   ルールには、同じリージョン内および同じイベントバス上の別のルールと同じ名前を付けることはできません。
5. Event bus] (イベントバス) では、このルールに関連付けるイベントバスを選択します。このルールをアカウントからのイベントと一致させるには、 AWS のデフォルトのイベントバスを選択します。アカウントの AWS サービスがイベントを発行すると、常にアカウントのデフォルトのイベントバスに移動します。
6. [Rule type] (ルールタイプ) では、[Rule with an event pattern] (イベントパターンを持つルール) を選択します。
7. 次へ をクリックします。
8. [Event source] (イベントソース) では、AWS [services] (サービス) を選択します。
9. (オプション) [Sample events] (イベント例) では、イベントのタイプを選択します。
10. [Event pattern] (イベントパターン) の場合は、次のいずれかを実行します。

- テンプレートを使用してイベントパターンを作成するには、[Event pattern form] (イベントパターンフォーム) を選択し、さらに [Event source] (イベントソース) および [Event type] (イベントタイプ) を選択します。イベントタイプとしてすべてのイベントを選択すると、この AWS サービスによって出力されるすべてのイベントがルールと一致します。

  テンプレートをカスタマイズするには、[Custom pattern (JSON editor)] (カスタムパターン (JSON エディタ)) を選択して変更します。
- カスタムイベントパターンを使用するには、[Custom pattern (JSON editor)] (カスタムパターン (JSON エディタ)) を選択し、イベントパターンを作成します。

11. 次へ をクリックします。
12. ターゲットタイプ] では、AWS サービス] を選択します。
13. ターゲットを選択 で、 がイベントパターンに一致するイベント EventBridge を検出したときに情報を送信する AWS サービスを選択します。
14. 表示されるフィールドは、選択したサービスによって異なります。必要に応じて、このターゲットタイプに固有の情報を入力します。
15. 多くのターゲットタイプでは、 はターゲットにイベントを送信するためのアクセス許可 EventBridge が必要です。このような場合は、ルールの実行に必要な IAM ロール EventBridge を作成できます。次のいずれかを行います。

    - 自動的に IAM ロールを作成するには、この特定のリソースに対して新しいロールを作成するを選択します。
    - 以前に作成した IAM ロールを使用するには、[Use existing role] (既存のロールの使用) をクリックし、ドロップダウンリストから既存のロールを選択します。
16. (オプション) [Additional settings] (追加設定) では、以下を実行します。

    a. Maximum age of event (最大イベント有効期間) に、1 分 (00:01) から 24 時間 (24:00) の間の値を入力します。

    b. 再試行 で、0 ~ 185 の数値を入力します。

    c. デッドレターキュー では、標準の Amazon SQS キューをデッドレターキューとして使用するかどうかを選択します。 は、このルールに一致する EventBridge イベントがターゲットに正常に配信されない場合、デッドレターキューに送信します。次のいずれかを行います。

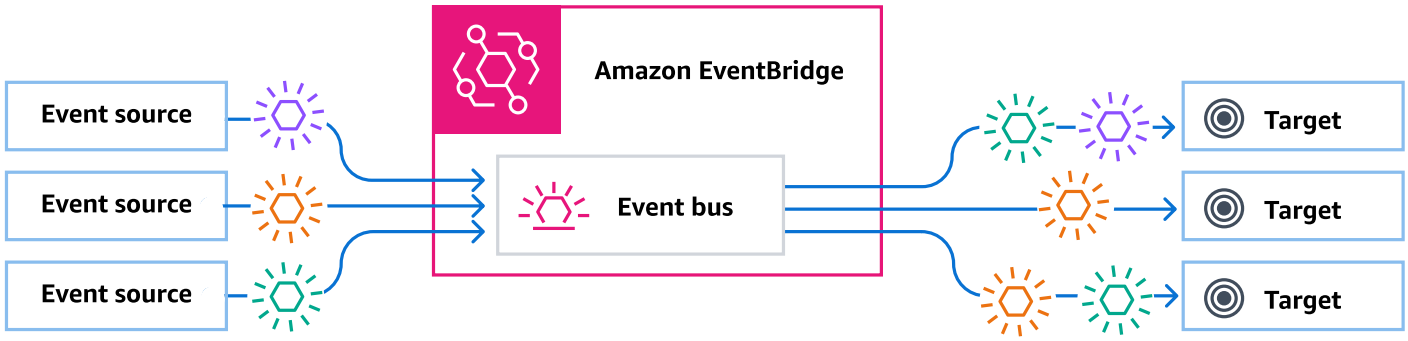
       - デッドレターキューを使用しない場合は、[None] (なし) を選択します。
       - Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue(デッドレターキューとして使用する現在の アカウントの Amazon SQS キューを選択) を選択し、ドロップダウンリストから使用するキューを選択します。

- 他の AWS アカウントの Amazon SQS キューをデッドレターキューとして選択を選択し、使用するキューの ARN を入力します。メッセージを送信する EventBridge アクセス許可を付与するリソースベースのポリシーをキューにアタッチする必要があります。詳細については、「デッドレターキューへのアクセス許可の付与」を参照してください。

17. (オプション) [Add another target] (別のターゲットを追加) を選択して、このルールに別のターゲットを追加します。
18. 次へ をクリックします。
19. (オプション) ルールに 1 つ以上のタグを入力します。詳細については、「Amazon EventBridge タグ」を参照してください。
20. 次へ をクリックします。
21. ルールの詳細を確認し、ルールの作成 を選択します。

Amazon EventBridge イベントバス

イベントバスは、イベントを受信するルーターであり、ゼロ個以上の送信先やターゲットに配信します。イベントバスは、イベントをさまざまなソースから多数のターゲットにルーティングするのに適しており、オプションでターゲットに配信する前にイベントを変換できます。



イベントバスに関連付けられたルールによって、受信したイベントが評価されます。各ルールは、イベントがルールのパターンに一致するかどうかをチェックします。イベントが一致した場合、EventBridge はイベントを送信します。

ルールを特定のイベントバスに関連付けると、そのルールはそのイベントバスで受信したイベントにのみ適用されます。

> **Note**
>
> EventBridge Pipes. EventBridge Pipes を使用してイベントを処理することもできます。point-to-point 各パイプは、単一のソースからイベントを受信し、単一のターゲットに処理および配信します。Pipes では、高度な変換やターゲットへの配信前のイベントのエンリッチメントもサポートされています。詳細については、「???」を参照してください。

トピック

- イベントバスの仕組み
- Amazon EventBridge Event Bus の概念
- Amazon EventBridge イベントバスの作成
- Amazon EventBridge イベントバスの更新
- Amazon EventBridge イベントバスの削除
- Amazon EventBridge イベントバスのアクセス許可

- Amazon EventBridge イベントバスから AWS CloudFormation テンプレートを生成する

イベントバスの仕組み

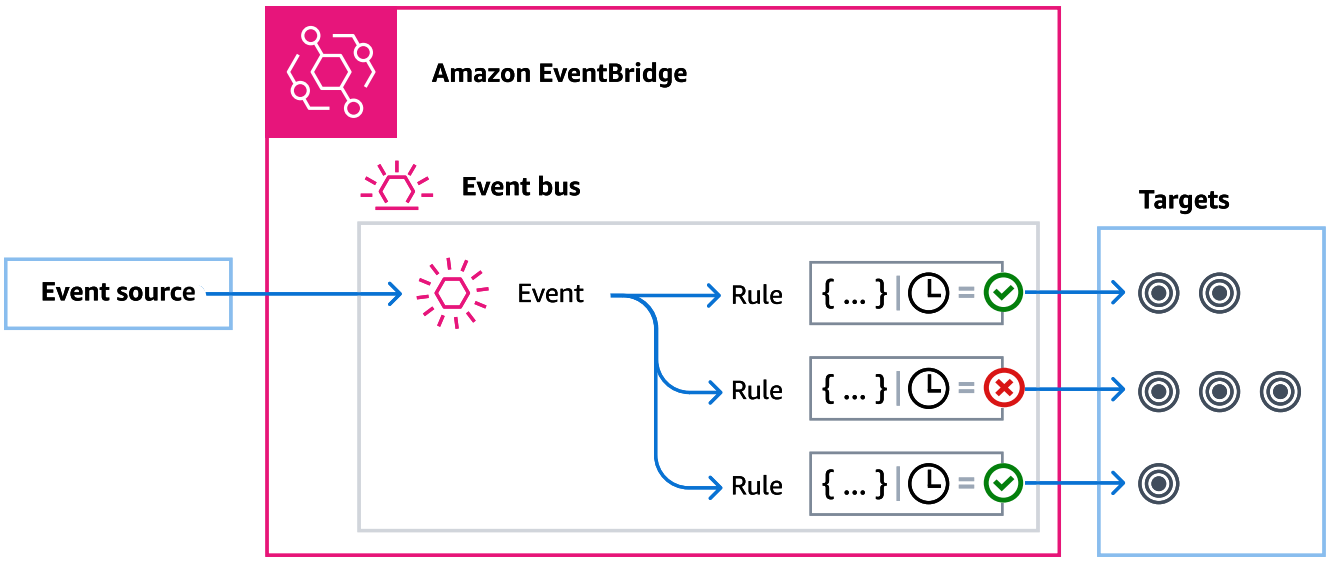
イベントバスを使用すると、複数のソースから複数の送信先またはターゲットにイベントをルーティングできるようになります。

大まかに言うと、その仕組みは次のとおりです。

1. サービス、独自のカスタムアプリケーション AWS 、または SaaS プロバイダーであるイベントソースは、イベントをイベントバスに送信します。
2. EventBridge 次に、 はそのイベントバスに定義された各ルールに対してイベントを評価します。

   ルールに一致するイベントごとに、 EventBridge はそのルールに指定されたターゲットにイベントを送信します。オプションで、ルールの一部として、 EventBridge がターゲット (複数可) に送信する前にイベントを変換する方法を指定することもできます。

   1 つのイベントが複数のルールに一致する場合があるため、各ルールには最大 5 つのターゲットを指定できます。(イベントはどのルールとも一致しない場合があり、その場合、 EventBridge はアクションを実行しません)。



AWS サービスからイベントを自動的に受信する EventBridge デフォルトのイベントバスを使用した例を考えてみましょう。

1. EC2 Instance State-change Notification イベントのデフォルト イベントバスにルールを作成します。
   - Amazon EC2 インスタンスが state から running に変更されたイベントにルールが一致するように指定します。

     そのためには、ルールをトリガーするためにイベントが一致する必要がある属性と値を定義する JSON を指定します。これを「イベントパターン」と呼びます。

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

   - ルールのターゲットを特定の Lambda 関数に指定します。
2. Amazon EC2 インスタンスの状態が変わるたびに、Amazon EC2 (イベントソース) はそのイベントをデフォルトのイベントバスに自動的に送信します。
3. EventBridge は、デフォルトのイベントバスに送信されたすべてのイベントを、作成したルールに対して評価します。

   イベントがルールと一致する場合 (つまり、イベントが Amazon EC2 インスタンスの状態が に変わった場合running)、 EventBridge は指定されたターゲットにイベントを送信します。この場合、それが Lambda 関数です。

次の動画では、イベントバスとは何か、その機能について説明しています。イベントバスとは

次のビデオでは、さまざまなイベントバスと、これらのバスをいつ使用するかについて説明します。イベントバスの違い

Amazon EventBridge Event Bus の概念

ここでは、イベントバスをベースに構築されたイベント駆動型アーキテクチャの主要コンポーネントを詳しく見ていきます。

イベントバス

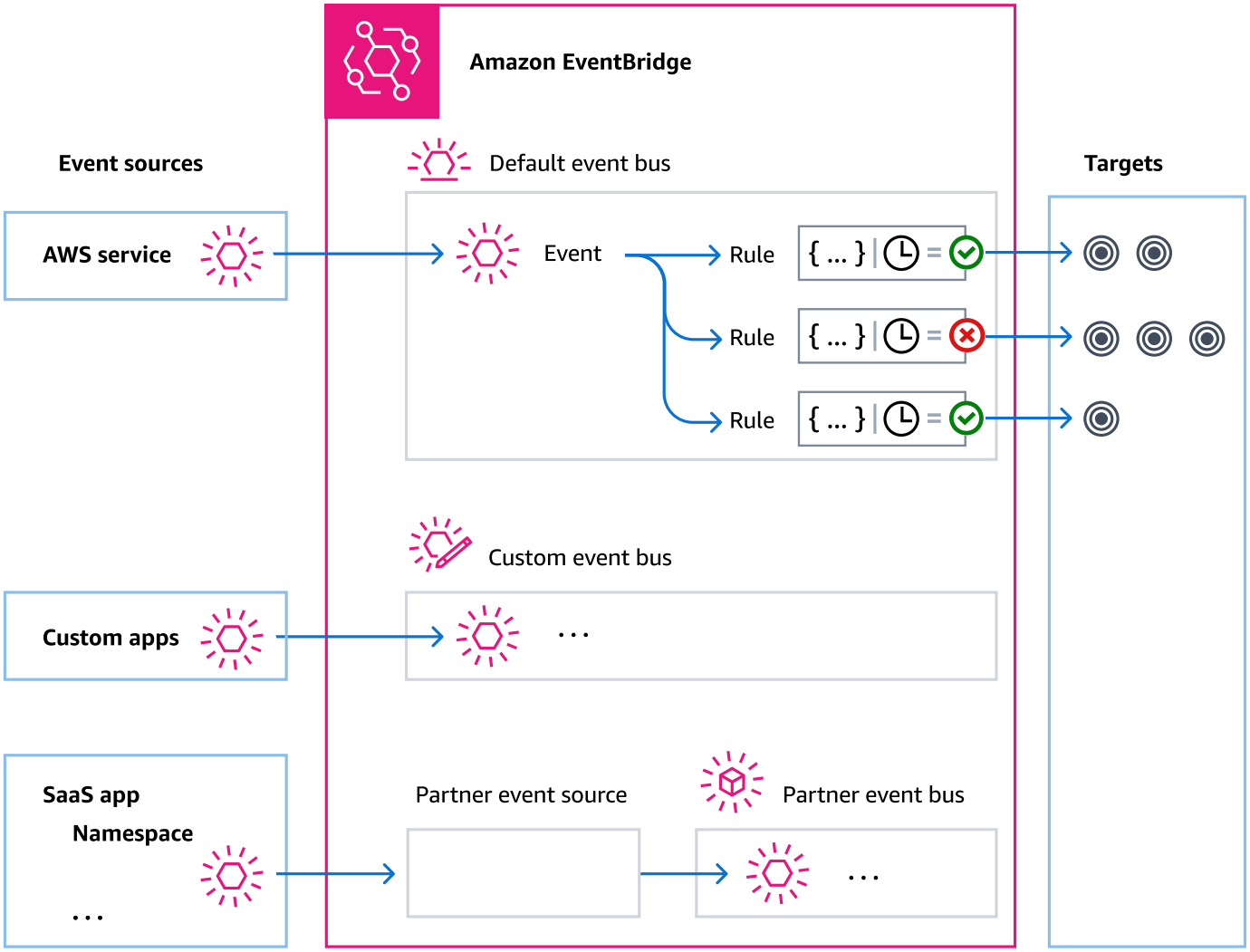
イベントバスは、イベントを受信するルーターであり、ゼロ個以上の送信先やターゲットに配信します。イベントバスは、イベントをさまざまなソースから多数のターゲットにルーティングし、オプションでターゲットに配信する前にイベントを変換する必要がある際に使用します。

アカウントには、 AWS サービスからイベントを自動的に受信するデフォルトのイベントバスが含まれています。以下の操作も可能です。

- カスタムイベントバスという名のイベントバスを追加作成し、受信するイベントを指定します。
- パートナーイベントバスを作成します。これは、SaaS パートナーからイベントを受信します。

イベントバスの一般的な使用例には以下が含まれます。

- イベントバスを異なるワークロード、サービス、またはシステム間の仲介役として使用します。
- アプリケーション内で複数のイベントバスを使用してイベントトラフィックを分割します。たとえば、個人識別情報 (PII) を含むイベントを処理するバスを作成し、個人識別情報 (PII) を含まないイベント用に別のバスを作成する場合などです。
- 複数のイベントバスから一元化されたイベントバスにイベントを送信してイベントを集約します。この集中型バスは、他のバスと同じアカウントにあっても、別のアカウントやリージョンにあってもかまいません。



イベント

最も単純なイベントは、 EventBridge イベントバスまたはパイプに送信される JSON オブジェクトです。

イベント駆動型アーキテクチャ (EDA) のコンテキストでは、イベントはリソースや環境の変化を示す指標となることがよくあります。

詳細については、「???」を参照してください。

イベントソース

EventBridge は、次のようなイベントソースからイベントを受信できます。

- AWS サービス
- カスタムアプリケーション
- Software as a Service (SaaS)

ルール

ルールは、受信したイベントを受信し、処理のためにターゲットに適切に送信します。各ルールがターゲットを呼び出す方法は、次のいずれかに基づいて指定できます。

- イベントパターン。イベントに一致する 1 つ以上のフィルターが含まれています。イベントパターンには、次の条件に一致するフィルターを含めることができます。
  - イベントメタデータ — イベント (イベントソースなど)、またはイベントが発生したアカウントまたはリージョンについてのデータ。
  - イベントデータ — イベント自体のプロパティ。これらのプロパティはイベントによって異なります。
  - イベントコンテンツ — イベントデータの実際のプロパティ値。
- ターゲットを定期的に呼び出すスケジュール。

  スケジュールされたルールは、 内で指定することも EventBridge、スEventBridge ケジューラ を使用して指定することもできます。

> **Note**
>
> EventBridge は、1 つの中央マネージドサービスからタスクを作成、実行、管理できるサーバーレス EventBridge ケジューラである Amazon Scheduler を提供します。EventBridge スケジューラは高度にカスタマイズ可能で、より広範なターゲット API オペレーションと AWS サービスセットにより、スケジュールされたルールよりも EventBridgeスケーラビリティが向上します。
> スケジュールに従ってターゲットを呼び出すには、ス EventBridge ケジューラを使用することをお勧めします。詳細については、「???」を参照してください。

各ルールは特定のイベントバスに対して定義され、そのイベントバス上のイベントにのみ適用されます。

1 つのルールで、最大 5 つのターゲットにイベントを送信できます。

デフォルトでは、イベントバスあたり最大 300 のルールを設定できます。このクォータは、Service Quotas コンソール内で数千のルールまで増やすことができます。ルール制限は各バスに適用されるため、さらに多くのルールが必要な場合は、アカウントに追加のカスタムイベントバスを作成できます。

サービスごとに異なるアクセス許可を持つイベントバスを作成することで、アカウントでのイベントの受信方法をカスタマイズできます。

がターゲットに EventBridge 渡す前にイベントの構造または日付をカスタマイズするには、入力トランスフォーマーを使用して、ターゲットに移動する前に情報を編集します。

詳細については、「???」を参照してください。

ターゲット

ターゲットは、イベントがルールに定義されたイベントパターンと一致するときに がイベント EventBridge を送信するリソースまたはエンドポイントです。

ターゲットは複数のイベントバスから複数のイベントを受信できます。

詳細については、「???」を参照してください。

イベントバス用の高度な機能

EventBridge には、イベントバスの開発、管理、使用に役立つ以下の機能が含まれています。

API 送信先を使用して、サービス間の REST API コールを有効にする

EventBridge API 送信先は、 AWS サービスまたはリソースにイベントデータを送信するのと同じ方法で、ルールのターゲットとして設定できる HTTP エンドポイントです。API 送信先を使用すると、API コールを使用して、 AWS サービス、統合された SaaS アプリケーション、および AWS外のアプリケーションの間でイベントをルーティングできます。API 送信先を作成するときは、その送信先に使用する接続を指定します。各接続 には、API 送信先エンドポイントでの認証に使用する認証タイプとパラメータに関する詳細が含まれます。

開発とディザスタリカバリに役立つイベントの Archive and Replay

イベントをアーカイブして、後でそのアーカイブからイベントを再生することができます。アーカイブは次のような場合に役立ちます。

- 新しいイベントを待つのではなく、イベントを保存して利用できるので、アプリケーションのテストに便利です。

- 新しいサービスが初めてオンラインになったら、ハイドレートします。
- イベント駆動型アプリケーションの耐久性を高めます。

スキーマレジストリを使用してイベントパターンの作成をすぐに開始する

を使用するサーバーレスアプリケーションを構築する場合 EventBridge、イベントを生成しなくても、一般的なイベントの構造を把握しておくと便利です。イベント構造はスキーマ で説明されています。スキーマ は、 のサービスによって生成されたすべてのイベント AWS で使用できます EventBridge。

AWS サービスから来ないイベントの場合、次のことができます。

- カスタムスキーマを作成またはアップロードします。
- Schema Discovery を使用して、 がイベントバスに送信されるイベントのスキーマ EventBridge を自動的に作成します。

イベントのスキーマがあれば、一般的なプログラミング言語のコードバインディングをダウンロードできます。

ポリシーによるリソースへのアクセスの管理

AWS リソースを整理したり、 のコストを追跡したりするには EventBridge、カスタムラベル またはタグ を AWS リソースに割り当てることができます。タグベースのポリシー を使用すると、 内で実行できるリソースとできないリソースを制御できます EventBridge。

タグベースのポリシーに加えて、 はアイデンティティベースのポリシーとリソースベースのポリシー EventBridge をサポートし、 へのアクセスを制御します EventBridge。アイデンティティ ベースのポリシーを使用して、グループ、ロール、またはユーザーのアクセス許可を制御します。リソースベースのポリシーを使用して、Lambda 関数や Amazon SNS トピックなど各リソースに特定のアクセス許可を付与します。

Amazon EventBridge イベントバスの作成

カスタムイベントバスを作成して、アプリケーションからイベントを受信できます。アプリケーションは、デフォルトのイベントバスにイベントを送信することもできます。イベントバスを作成すると、リソースベースのポリシーをアタッチして、他のアカウントにアクセス許可を付与できます。そうすると、他のアカウントで、現在のアカウントのイベントバスにイベントを送信できます。

次のビデオでは、イベントバスの作成について説明します：イベントバスの作成

カスタムイベントバスを作成するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインの [Event Buses] (イベントバス) を選択します。
3. [イベントバスの作成 (Create event bus)] を選択します。
4. 新しいイベントバスの名前を入力します。
5. イベントバス EventBridge に保存されているイベントデータを暗号化するときに使用する KMS key を選択します。

> **Note**
>
> アーカイブとスキーマ検出は、 を使用して暗号化されたイベントバスではサポートされていません カスタマー管理キー。イベントバスでアーカイブまたはスキーマ検出を有効にするには、 の使用を選択します AWS 所有のキー。詳細については、「???」を参照してください。

- を使用してデータを暗号化 EventBridge するには、 AWS 所有のキーに使用する を選択します AWS 所有のキー。

  AWS 所有のキー これは、 KMS key が複数の AWS アカウントで使用するために EventBridge 所有および管理する です。一般に、リソースを保護する暗号化キーを監査または制御する必要がない限り、 AWS 所有のキー が適しています。

  これがデフォルトです。
- で使用 カスタマー管理キー を選択して EventBridge 、 カスタマー管理キー 指定した または作成した を使用してデータを暗号化します。

  カスタマーマネージドキー は、ユーザーが作成、所有、管理する AWS アカウント KMS keys にあります。これらの は完全に制御できます KMS keys。

  a. 既存の を指定するか カスタマー管理キー、新しい を作成する KMS keyを選択します。

     EventBridge は、指定された に関連付けられているキーステータスとキーエイリアスを表示します カスタマー管理キー。

b. このイベントバスのデッドレターキュー (DLQ) として使用する Amazon SQS キューがあれば選択します。

EventBridge は、設定されている場合、正常に暗号化されていないイベントを DLQ に送信し、後で処理できるようにします。

6. オプションのイベントバス機能を設定します。

- 次のいずれかを実行して、リソースベースのポリシーを指定します。
  - イベントバスに付与するアクセス許可を含むポリシーを入力します。別のソースからポリシーを貼り付けることも、ポリシーの JSON を入力することもできます。サンプルポリシーのいずれかを使用して、環境に合わせて変更できます。
  - ポリシーにテンプレートを使用するには、[Load template] (テンプレートのロード) を選択します。環境に応じてポリシーを変更し、使用するポリシーでプリンシパルを認可するアクションを追加します。

  リソースベースのポリシーによるイベントバスへのアクセス許可の付与の詳細については、「」を参照してください???。
- アーカイブを有効にする (オプション)

  イベントのアーカイブを作成して、後で簡単に再生できます。例えば、イベントを再生して、エラーから回復したり、アプリケーションの新機能を検証したりする場合があります。詳細については、「???」を参照してください。

  a. アーカイブ で、有効化 を選択します。

  b. アーカイブの名前と説明を指定します。

  > **Note**
  >
  > アーカイブとスキーマ検出は、 を使用して暗号化されたイベントバスではサポートされていません カスタマー管理キー。イベントバスでアーカイブまたはスキーマ検出を有効にするには、 の使用を選択します AWS 所有のキー。詳細については、「???」を参照してください。

- スキーマ検出を有効にする (オプション)

  スキーマ検出を有効にして、 EventBridge このイベントバスで実行されているイベントから直接スキーマを自動的に推測します。詳細については、「???」を参照してください。

  a. スキーマ検出 で、有効 を選択します。

> **Note**
>
> アーカイブとスキーマ検出は、 を使用して暗号化されたイベントバスではサポートされていません カスタマー管理キー。イベントバスでアーカイブまたはスキーマ検出を有効にするには、 の使用を選択します AWS 所有のキー。詳細については、「???」を参照してください。

- タグを指定する (オプション)

  タグは、 AWS リソースに割り当てるカスタム属性ラベルです。タグを使用して、 AWS リソースを識別して整理します。多くの AWS のサービスではタグ付けがサポートされているため、異なる サービスのリソースに同じタグを割り当てて、リソースが関連していることを示すことができます。詳細については、「???」を参照してください。

  a. [タグ] で [タグを追加] を選択します。

  b. キーを指定し、オプションで新しいタグの値を指定します。

7. [作成] を選択します。

Amazon EventBridge イベントバスの更新

イベントバスは、作成後に設定を更新できます。これには、 がアカウントに自動的に EventBridge 作成するデフォルトのイベントバスが含まれます。

暗号化に使用される の更新 KMS key

> **Note**
>
> アーカイブとスキーマ検出は、 を使用して暗号化されたイベントバスではサポートされていません カスタマー管理キー。イベントバスでアーカイブまたはスキーマ検出を有効にするには、 の使用を選択します AWS 所有のキー。詳細については、「???」を参照してください。

EventBridge コンソールを使用してイベントバスの保管時の暗号化 KMS key に使用される を変更するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。

2. ナビゲーションペインの [Event Buses] (イベントバス) を選択します。
3. 更新するイベントバスを選択します。
4. イベントバスの詳細ページで、暗号化タブを選択します。
5. イベントバス KMS key EventBridge に保存されているイベントデータを暗号化するときに使用する を選択します。
   - を使用してデータを暗号化 EventBridge するには、 AWS 所有のキーに使用する を選択します AWS 所有のキー。

     AWS 所有のキー これは、 KMS key が複数の AWS アカウントで使用するために EventBridge 所有および管理する です。一般に、リソースを保護する暗号化キーを監査または制御する必要がない限り、 AWS 所有のキー が適しています。

     これがデフォルトです。
   - で使用 カスタマー管理キー を選択して EventBridge 、 カスタマー管理キー 指定した または作成した を使用してデータを暗号化します。

     カスタマーマネージドキー は、ユーザーが作成、所有、管理する AWS アカウント KMS keys にあります。これらの は完全に制御できます KMS keys。

     a. 既存の を指定するか カスタマー管理キー、新しい を作成する KMS keyを選択します。

        EventBridge は、指定された に関連付けられているキーステータスとキーエイリアスを表示します カスタマー管理キー。

     b. このイベントバスのデッドレターキュー (DLQ) として使用する Amazon SQS キューがあれば選択します。

        EventBridge は、設定されている場合、正常に暗号化されていないイベントを DLQ に送信し、後で処理できるようにします。

イベントバスのアクセス許可の更新

イベントバスに追加のアクセス許可を付与するには、リソースベースのポリシーをアタッチします。イベントバスのアクセス許可を更新する詳細な手順については、「イベントバスのアクセス許可の管理」を参照してください。

イベントバスでのアーカイブの追加または削除

アーカイブを使用すると、後で簡単に再生できるようにイベントをキャプチャできます。例えば、イベントを再生して、エラーから回復したり、アプリケーションの新機能を検証したりする場合があります。詳細については、EventBridge 「アーカイブとリプレイ」を参照してください。

> **Note**
>
> アーカイブとスキーマ検出は、 を使用して暗号化されたイベントバスではサポートされていません カスタマー管理キー。イベントバスでアーカイブまたはスキーマ検出を有効にするには、 の使用を選択します AWS 所有のキー。詳細については、「???」を参照してください。

コンソールを使用して EventBridge イベントバスからアーカイブを追加または削除するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインの [Event Buses] (イベントバス) を選択します。
3. 更新するイベントバスを選択します。
4. イベントバスの詳細ページで、アーカイブタブを選択します。
5. 次のいずれかを行います。
   - アーカイブを追加するには：
     a. [Create archive] (アーカイブの作成) を選択します。
     b. アーカイブの属性を指定します。
     c. [次へ] をクリックします。
     d. アーカイブのイベントに適用するイベントパターンを選択します。
     e. [Create archive] (アーカイブの作成) を選択します。
   - アーカイブを削除するには：
     a. 削除するタグで、削除を選択します。
     b. アーカイブの名前を入力し、「削除」を選択します。

        アーカイブは完全に削除されます。このオペレーションは元に戻すことができません。

を使用してイベントバスのアーカイブを作成または削除するには AWS CLI

- アーカイブを作成するには、create-archive を使用します。

  アーカイブを完全に削除するには、delete-archive を使用します。

イベントバスでのスキーマ検出の開始または停止

スキーマ検出の詳細については、EventBridge 「 スキーマ」を参照してください。

> **Note**
>
> アーカイブとスキーマ検出は、 を使用して暗号化されたイベントバスではサポートされていません カスタマー管理キー。イベントバスでアーカイブまたはスキーマ検出を有効にするには、 の使用を選択します AWS 所有のキー。詳細については、「???」を参照してください。

コンソールを使用して EventBridge イベントバスでスキーマ検出を開始または停止するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインの [Event Buses] (イベントバス) を選択します。
3. 更新するイベントバスを選択します。
4. 次のいずれかを行います。
   - スキーマ検出を開始するには、検出の開始 を選択します。
   - スキーマ検出を停止するには、検出の削除 を選択します。

を使用してイベントバスでスキーマ検出を開始または停止するには AWS CLI

- スキーマ検出を開始するには、create-discoverer を使用します。

  スキーマ検出を停止するには、delete-discoverer を使用します。

イベントバスでのタグの追加または削除

タグは、 AWS リソース AWS に割り当てるカスタム属性ラベルです。タグを使用して、 AWS リソースを識別して整理します。詳細については、EventBridge 「tags」を参照してください。

コンソールを使用して EventBridge イベントバスからタグを追加または削除するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインの [Event Buses] (イベントバス) を選択します。
3. 更新するイベントバスを選択します。
4. イベントバスの詳細ページで、タグ タブを選択し、タグの管理 を選択します。
5. 次のいずれかを行います。
   - タグを追加するには：
     a. [新しいタグを追加] をクリックします。
     b. タグのキーと値を指定する
     c. [更新] を選択します。
   - タグを削除するには：
     a. 削除するタグで、 の削除 を選択します。
     b. [更新] を選択します。

を使用してイベントバスからタグを追加または削除するには AWS CLI

- タグを追加するには、tag-resource を使用します。

  タグを削除するには、untag-resource を使用します。

を使用したデフォルトのイベントバスの更新 AWS CloudFormation

AWS CloudFormation では、インフラストラクチャを AWS code として扱うことで、アカウントとリージョン間でリソースを一元的かつ反復可能な方法で設定および管理できます。 CloudFormation これにより、プロビジョニングおよび管理したいリソースを定義するテンプレート を作成できます。

はデフォルトのイベントバスを自動的にアカウントに EventBridge プロビジョニングするため、スタックに含める CloudFormation リソースの場合と同様に、 CloudFormation テンプレートを使用して作成することはできません。 CloudFormation スタックにデフォルトのイベントバスを含めるには、まずスタックにインポートする必要があります。デフォルトのイベントバスをスタックにインポートしたら、必要に応じてイベントバスのプロパティを更新できます。

既存のリソースを新規または既存の CloudFormation スタックにインポートするには、次の情報が必要です。

- インポートするリソースの一意の識別子。

  デフォルトのイベントバスの場合、識別子は Nameで、識別子値は ですdefault。

- 既存のリソースの現在のプロパティを正確に記述するテンプレート。

  以下のテンプレートスニペットには、デフォルトのイベントバスの現在のプロパティを記述するAWS::Events::EventBusリソースが含まれています。この例では、イベントバスは保管時の暗号化に カスタマー管理キー と DLQ を使用するように設定されています。

  また、インポートするデフォルトのイベントバスを記述するAWS::Events::EventBusリソースには、 に設定されたDeletionPolicyプロパティが含まれている必要があります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"
                }
            }
        }
    }
}
```

詳細については、「 ユーザーガイド」の「既存のリソース CloudFormation を管理に取り込む」を参照してください。 CloudFormation

Amazon EventBridge イベントバスの削除

カスタムイベントバスまたはパートナーイベントバスを削除できます。デフォルトのイベントバスは削除できません。イベントバスを削除すると、そのイベントバスに関連付けられたルールが削除されます。

EventBridge コンソールを使用してイベントバスを削除するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインの [Event Buses] (イベントバス) を選択します。
3. 削除するイベントバスを選択します。
4. 次のいずれかを行います。
   - [削除] を選択します。
   - イベントバスの名前を選択します。

     イベントバスの詳細ページで、「削除」を選択します。

Amazon EventBridge イベントバスのアクセス許可

AWS アカウントのデフォルトのイベントバスでは、1 つのアカウントからのイベントしか許可されません。イベントバスに追加のアクセス許可を付与するには、リソースベースのポリシーをアタッチします。リソースベースのポリシーを使用すると、別のアカウントからの `PutEvents`、`PutRule`、および `PutTargets` API コールを許可できます。また、ポリシーで IAM 条件を使用して組織にアクセス許可を付与するには、タグを適用するか、特定のルールまたはアカウントからのイベントにのみイベントをフィルタリングします。イベントバスのリソースベースのポリシーは、イベントバスの作成時または作成後に設定できます。

イベントバスの `Name` パラメータを受け入れる EventBridge API (`PutRule`、`PutTargets`、`DeleteRule`、`RemoveTargets`、`DisableRule`、`EnableRule` など) は、イベントバス ARN も受け入れます。これらのパラメータを使用し、API を介してクロスアカウントまたはクロスリージョンのイベントバスを参照します。例えば、`PutRule` を呼び出すと、ロールを引き受けなくても、別のアカウントのイベントバスにルールを作成することができます。

このトピックのポリシー例を IAM ロールに添付して、別のアカウントまたはリージョンにイベントを送信するアクセス許可を付与することができます。IAM ロールを使用して、アカウントから他のアカウントにイベントを送信できるユーザーに関する組織のコントロールポリシーと境界を設定します。ルールのターゲットがイベントバスの場合は、常に IAM ロールを使用することをお勧めしま

す。IAM ロールは、PutTarget 呼び出しを使用して添付できます。別のアカウントまたはリージョンにイベントを送信するルールの作成については、AWS アカウント間での Amazon EventBridge イベントの送受信 を参照してください。

トピック

イベントバスのアクセス許可の管理

既存のイベントバスのアクセス許可を変更するには、次の手順を実行します。AWS CloudFormation を使用してイベントバスポリシーを作成する方法については、AWS::Events::EventBusPolicy を参照してください。

既存のイベントバスのアクセス許可を管理するには

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. 左側のナビゲーションペインで [Event Buses] (イベントバス) を選択します。
3. [Name] (名前) で、アクセス許可を管理するイベントバスの名前を選択します。

   リソースポリシーがイベントバスにアタッチされている場合は、ポリシーが表示されます。
4. [Manage permissions] (アクセス許可の管理) を選択して、次のいずれかを実行します。
   - イベントバスに付与するアクセス許可を含むポリシーを入力します。別のソースからポリシーを貼り付けることも、ポリシーの JSON を入力することもできます。
   - ポリシーにテンプレートを使用するには、[Load template] (テンプレートのロード) を選択します。環境に応じてポリシーを変更し、使用するポリシーでプリンシパルを認可する追加のアクションを追加します。
5. [Update] (更新) を選択します。

テンプレートには、お客様のアカウントや環境に合わせてカスタマイズできるポリシー文の例があります。テンプレートは有効なポリシーではありません。ユースケースに応じてテンプレートを変更することも、ポリシー例の 1 つをコピーしてカスタマイズすることもできます。

テンプレートは、PutEvents アクションを使用するためにアカウントにアクセス許可を付与する方法、組織にアクセス許可を付与する方法、アカウントでルールを管理するアカウントにアクセス許可を付与する方法の例などのポリシーをロードします。特定のアカウントのテンプレートをカスタマイズして、テンプレートから他のセクションを削除できます。このトピックの後半で、さらに多くのポリシー例を紹介しています。

バスの権限を更新しようとしても、ポリシーにエラーが含まれている場合は、ポリシーの特定の問題を示すエラーメッセージが表示されます。

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

ポリシーの例: 別のアカウントのデフォルトバスにイベントを送信する

以下のポリシー例では、アカウント 123456789012 のデフォルトのイベントバスに対してイベントを発行するアクセス許可をアカウント 111122223333 に付与します。

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

ポリシーの例: 別のアカウントのカスタムバスにイベントを送信する

以下のポリシー例では、アカウント 123456789012 の `central-event-bus` にイベントを発行するアクセス許可をアカウント 111122223333 に付与します。ただし、ソースの値が `com.exampleCorp.webStore` に設定され、`detail-type` が `newOrderCreated` に設定されているイベントのみが対象となります。

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

ポリシーの例: 同じアカウントのイベントバスにイベントを送信する

CustomBus1 という名前のイベントバスにアタッチされた以下のポリシー例では、イベントバスは同じアカウントとリージョンからイベントを受信できます。

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

ポリシーの例: イベントを同じアカウントに送信し、更新を制限する

以下のポリシー例では、アカウント 123456789012 にルールの作成、削除、更新、無効化、有効化、およびターゲットの追加と削除を実行できるアクセス許可を付与します。これらのルールは、com.exampleCorp.webStore をソースとするイベントに一致するように制限されており、"events:creatorAccount": "${aws:PrincipalAccount}" を使用して、アカウント 123456789012 のみが、作成後のルールとターゲットを変更できることを保証します。

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

ポリシーの例: 特定のルールからのイベントのみを別のリージョンのバスに送信する

以下のポリシー例では、中東 (バーレーン) および米国西部 (オレゴン) のリージョンでの SendToUSE1AnotherAccount という名前のルールに一致するイベントを、アカウント 123456789012 の 米国東部 (バージニア北部) の CrossRegionBus というイベントバスに送信するアクセス許可をアカウント 111122223333 に付与します。アカウント 123456789012 の CrossRegionBus というイベントバスに、この例のポリシーを追加します。ポリシーは、イベントがアカウント 111122223333 のイベントバスに指定されたルールに一致する場合にのみイベントを許可します。Condition ステートメントは、指定されたルール ARN を持つルールに一致するイベントのみにイベントを制限します。

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

ポリシーの例: 特定のリージョンからのイベントのみを別のリージョンに送信する

以下のポリシー例では、中東 (バーレーン) および米国西部 (オレゴン) のリージョンでの生成されるすべてのイベントを、米国東部 (バージニア北部) リージョンのアカウント 123456789012 の CrossRegionBus というイベントバスに送信するアクセス許可を、アカウント 111122223333 に付与します。アカウント 111122223333 には、他のリージョンで生成されたイベントを送信するアクセス許可はありません。

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

ポリシーの例: 特定のリージョンからのイベントの送信を拒否する

アカウント 123456789012 の CrossRegionBus という名前のイベントバスにアタッチされている以下のポリシー例は、イベントバスがアカウント 111122223333 からのイベントを受信するアクセス許可を付与しますが、米国西部 (オレゴン) リージョンで生成されたイベントは受信しません。

```
{
  "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 イベントバスから AWS CloudFormation テンプレートを生成する

AWS CloudFormation では、インフラストラクチャをコードとして扱うことで、複数のアカウントやリージョンの AWS リソースを一元的かつ繰り返し可能な方法で設定および管理できます。CloudFormation では、これを実現するために、プロビジョニングおよび管理するリソースを定義するテンプレートを作成できます。

EventBridge では、CloudFormation テンプレートの開発をすぐに開始するための補助として、アカウント内の既存のイベントバスからテンプレートを生成できます。さらに、EventBridge には、そのイベントバスに関連するルールをテンプレートに含めるオプションもあります。次に、これらのテンプレートに基づいて CloudFormation で管理するリソースのスタックを作成できます。

CloudFormation の詳細については、「AWS CloudFormation ユーザーガイド」を参照してください。

> **Note**
>
> EventBridge は、生成されたテンプレートにマネージドルールを含みません。

また、選択したイベントバスに含まれている 1 つ以上のルールからテンプレートを生成することもできます。

イベントバスから CloudFormation テンプレートを生成するには

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインの [Event Buses] (イベントバス) を選択します。
3. CloudFormation テンプレートを生成する元のイベントバスを選択します。
4. アクションメニューから「CloudFormation テンプレート」を選択し、EventBridge でテンプレートを生成したいフォーマット (JSON または YAML) を選択します。

   EventBridge には、選択した形式で生成されたテンプレートが表示されます。デフォルトでは、イベントバスに関連するすべてのルールがテンプレートに含まれています。

- ルールを含めずにテンプレートを生成するには、[Include rules on this EventBus] (この EventBus にルールを含める) を選択解除します。

5. EventBridge では、テンプレートファイルをダウンロードするか、テンプレートをクリップボードにコピーするかを選択できます。

   - テンプレートファイルをダウンロードするには、[Download] (ダウンロード) を選択します。
   - テンプレートをクリップボードにコピーするには、[Copy] (コピー) を選択します。

6. テンプレートを終了するには、[Cancel] (キャンセル) を選択します。

ユースケースに合わせて AWS CloudFormation テンプレートをカスタマイズしたら、これを CloudFormation で使用してスタックを作成できます。

Amazon EventBridge から生成した CloudFormation テンプレートを使用する際の考慮事項

イベントバスから生成した CloudFormation テンプレートを使用するときは、以下の点を考慮してください。

- EventBridge は、生成テンプレートにパスワードを含みません。

  テンプレートを編集してテンプレートパラメータを含めると、ユーザーがテンプレートを使用して CloudFormation スタックを作成または更新するときに、パスワードやその他の機密情報を指定できるようになります。

  さらに、ユーザーは Secrets Manager を使用して目的のリージョンにシークレットを作成し、生成されたテンプレートを編集して動的パラメーターを使用できます。
- 生成されたテンプレートのターゲットは、元のイベントバスで指定されていたものとまったく同じままです。テンプレートを使用して他のリージョンにスタックを作成する前に、テンプレートを適切に編集しないと、リージョン間の問題が発生する可能性があります。

  また、生成されたテンプレートは下流のターゲットを自動的に作成しません。

Amazon EventBridge イベント

イベントとは、 AWS 環境、SaaS パートナーサービスやアプリケーション、またはお客様のアプリケーションやサービスなどの環境での変化を示します。以下は、イベントの例です。

- インスタンスの状態が保留中から実行中に変更されると、Amazon EC2 はイベントを生成します。
- Amazon EC2 Auto Scaling は、インスタンスを起動または終了するときにイベントを生成します。
- AWS CloudTrail は、API コールを行うときにイベントを発行します。

定期的に生成される予定されたイベントをセットアップすることもできます。

各サービスのサンプルイベントなど、イベントを生成するサービスの一覧については、「AWS サービスからのイベント」を参照し、表のリンクに従ってください。

イベントは JSON オブジェクトとして表現され、同じような構造をしています。トップレベルのフィールドも同じです。

[detail] の最上位のフィールドの内容は、どのサービスがイベントを生成したか、そのイベントが何であるかによって異なります。[source] フィールドと [detail-type] フィールドの組み合わせは、[detail] フィールドで見つかるフィールドと値を識別するために役立ちます。 AWS サービスによって生成されるイベントの例については、「」を参照してくださいAWS サービスからのイベント。

トピック

- イベント構造リファレンス
- を使用した Amazon EventBridge イベントの追加 PutEvents
- AWS サービスからのイベント
- Amazon との SaaS パートナーからのイベントの受信 EventBridge
- イベント配信のデバッグ

次のビデオでは、イベントの基本について説明します：イベントとは

次の動画では、イベントが に到達する方法について説明しています EventBridge。イベントはどこから来るのか

イベント構造リファレンス

次のフィールドは、イベントバスに配信されるすべてのイベントに表示され、イベントのメタデータを構成します。

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

version

デフォルトでは、これはすべてのイベントで 0 (ゼロ) に設定されます。

id

イベントごとに生成されるバージョン 4 UUID。id を使用すると、ルールからターゲットに移動するときのイベントをトレースできます。

detail-type (ディテールタイプ)

[source] フィールドと組み合わせて、[detail] フィールドに表示されるフィールドと値を識別します。

によって配信されるイベントは CloudTrail 、 の値AWS API Call via CloudTrailとして を持ちますdetail-type。

ソース

イベントを発生させたサービスを識別します。 AWS サービスからのイベントはすべて、「AWS」で始まります。顧客から発生したイベントは、「aws」で始まらない限り、このフィールドに値があります。Java パッケージ名のスタイルには逆ドメイン名の文字列を使用することをお勧めします。

AWS サービスの の正しい値を見つけるには、「条件キーテーブルsource」を参照し、リストからサービスを選択し、サービスプレフィックス を探します。例えば、Amazon のsource値は CloudFront ですaws.cloudfront。

アカウント

AWS アカウントを識別する 12 桁の番号。

time

イベントを発生したサービスによって指定できるイベントのタイムスタンプ。イベントが時間間隔にまたがる場合、サービスは開始時間をレポートできるため、この値は、イベントが受け取られるより前の時間になることがあります。

region

イベントが発生した AWS リージョンを識別します。

resources

JSON 配列に、イベントにかかわるリソースを識別する ARN が格納されます。これらの ARN を含めるかどうかは、イベントを生成するサービスによって決まります。例えば、Amazon EC2 インスタンスの状態変更では、Amazon EC2 インスタンス ARN が格納され、Auto Scaling イベントでは、インスタンスと Auto Scaling グループの両方の ARN が格納されますが、 AWS CloudTrail での API コールでは、リソース ARN は格納されません。

detail

イベントに関する情報を含む JSON オブジェクト。このフィールドの内容は、イベントを生成するサービスによって決まります。"{}" とすることができます。

AWS API コールイベントには詳細オブジェクトがあり、約 50 個のフィールドが複数のレベルの深さにネストされています。

> **Note**
>
> PutEvents は JSON 形式のデータを受け入れます。JSON 番号 (整数) データタイプの場合、制約は、最小値が -9,223,372,036,854,775,808、最大値が 9,223,372,036,854,775,807 です。

Example 例: Amazon EC2 インスタンスの状態変更通知

Amazon の次のイベントは、Amazon EC2 インスタンスが終了される EventBridge ことを示します。

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

有効なカスタムイベントに必要な最小限の情報

カスタムイベントを作成する場合、次のフィールドが含まれている必要があります。

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

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

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

を使用した Amazon EventBridge イベントの追加 PutEvents

PutEvents アクションは、1 回のリクエスト EventBridge で複数のイベントを に送信します。詳細については、「Amazon EventBridge API リファレンスPutEvents」の「」とAWS CLI 「 コマンドリファレンス」の「put-events」を参照してください。

各 PutEvents リクエストでサポートされているエントリの数は制限されています。詳細については、「Amazon EventBridge クォータ」を参照してください。PutEvents オペレーションでは、リクエストの自然な順序ですべてのエントリを処理するように試みます。を呼び出すとPutEvents、は各イベントに一意の ID を EventBridge 割り当てます。

トピック

- PutEvents での失敗の処理
- を使用したイベントの送信 AWS CLI
- Amazon EventBridge PutEvents イベントエントリサイズの計算

次の Java コードの例では、2 つの同じイベントを に送信します EventBridge。

AWS SDK for Java Version 2.x

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

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

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

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

PutEventsResponse result = eventBridgeClient.putEvents(eventsRequest);

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

AWS SDK for Java Version 1.0

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

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

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

PutEventsResult result = awsEventsClient.putEvents(request);

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

このコードの実行後、PutEvents の結果には応答配列のエントリが含まれます。応答配列の各エントリは、リクエストとレスポンスの最初から最後までの順番に、リクエスト配列のエントリと対応します。応答 Entries 配列には、常にリクエスト配列と同じ数のエントリが含まれます。

PutEvents での失敗の処理

デフォルトでは、リクエスト内の個々のエントリが失敗した場合、 はリクエスト内の残りのエントリの処理 EventBridge を続行します。レスポンスの Entries 配列には、成功したエントリと失敗したエントリの両方を含めることができます。成功しなかったエントリを検出し、それ以降の呼び出しに含める必要があります。

成功した結果エントリには、Id 値が含まれ、失敗した結果エントリには ErrorCode および ErrorMessage の値が含まれます。ErrorCode はエラーのタイプを示します。ErrorMessage にはエラーに関する詳細が指定されます。以下の例では、PutEvents リクエストに対して 3 つの結果エントリがあります。2 番目のエントリは成功していません。

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

> **Note**
>
> PutEvents を使用して、存在しないイベントバスにイベントを発行する場合、 EventBridge イベントマッチングは対応するルールを見つけられず、イベントは削除されます。EventBridge は200レスポンスを送信しますが、リクエストに失敗したり、リクエストレスポンスFailedEntryCountの値にイベントを含めたりすることはありません。

失敗したエントリは、以降の PutEvents リクエストに含めることができます。最初に、失敗したエントリがあるかどうかを調べるために、PutEventsResult の FailedRecordCount パラメータを確認します。ゼロでない場合、Null でない ErrorCode を持つ各 Entry を後続のリクエストに追加することができます。以下は、エラーハンドラーの例です。

```
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);
    }
```

を使用したイベントの送信 AWS CLI

を使用してカスタムイベント AWS CLI を に送信 EventBridge し、処理できます。次の例では、1 つのカスタムイベントを に配置します EventBridge。

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

カスタムイベントを含む JSON ファイルを作成することもできます。

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

次に、 を使用してこのファイルからエントリを AWS CLI 読み取り、イベントを送信するには、コマンドプロンプトで次のように入力します。

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

Amazon EventBridge PutEvents イベントエントリサイズの計算

カスタムイベントは、 PutEventsアクション EventBridge を使用して に送信できます。効率化のために複数のイベントエントリを 1 つのリクエストにまとめることができます。合計エントリサイズは 256 KB 未満でなければなりません。イベントを送信する前に、エントリサイズを計算できます。

> **Note**
>
> サイズ制限はエントリに適用されます。エントリがサイズ制限を下回っていても、イベントの JSON 表現に必要な文字とキーのため、 のイベント EventBridge は常にエントリサイズよりも大きくなります。詳細については、「Amazon EventBridge イベント」を参照してください。

EventBridge は次のようにPutEventsRequestEntryサイズを計算します。

- 指定されている場合、Time パラメータは 14 バイトです。
- Source および DetailType パラメータは、UTF-8 エンコード形式のバイト数です。
- 指定されている場合、Detail パラメータが UTF-8 エンコード形式のバイト数です。
- 指定されている場合は、Resources パラメータの各エントリは UTF-8 エンコード形式のバイト数です。

以下の Java コード例は、指定された PutEventsRequestEntry オブジェクトのサイズを計算します。

```
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**
>
> エントリサイズが 256 KB より大きい場合は、イベントを Amazon S3 バケットにアップロードし、`Object URL` を `PutEvents` エントリに含めることをお勧めします。

AWS サービスからのイベント

多くの AWS サービスは、 が EventBridge 受信するイベントを生成します。アカウントの AWS サービスがイベントを発行すると、アカウントのデフォルトのイベントバスに送られます。

AWS サービスからのイベント配信

イベントを生成する各 AWS サービスは、ベストエフォートまたは永続的な配信試行 EventBridge として に送信します。

- ベストエフォート配信とは、サービスがすべてのイベントを に送信しようとしますが EventBridge、まれにイベントが配信されない場合があります。
- 永続的な配信とは、サービスが EventBridge 少なくとも 1 回イベントを配信しようとすることを意味します。

  EventBridge は、通常の条件下ですべての有効なイベントを受け入れます。 EventBridge サービスの中断によりイベントを配信できない場合、後で AWS サービスによって最大 24 時間再試行されます。

イベントが に配信されると EventBridge、 はそれをルールと EventBridge 照合し、再試行ポリシーとイベントターゲットに指定されたデッドレターキューに従います (複数可) 。

イベントを生成する AWS サービスのリストについては、「」を参照してください???。

による AWS サービスイベントへのアクセス AWS CloudTrail

AWS CloudTrail は、 AWS API コールなどのイベントを自動的に記録するサービスです。 からの情報を使用する EventBridge ルールを作成できます CloudTrail。の詳細については CloudTrail、「 AWS CloudTrailとは」を参照してください。

によって配信されるすべてのイベントは CloudTrail 、 の値`AWS API Call via CloudTrail`として を持ち`detail-type`ます。

`detail-type` 値が のイベントを記録するには`AWS API Call via CloudTrail`、ログ記録が有効になっている CloudTrail 証跡が必要です。

Amazon S3 CloudTrail で を使用する場合は、データイベントをログ CloudTrail に記録するように を設定する必要があります。詳細については、S3 バケットとオブジェクトの CloudTrail イベントログ記録の有効化」を参照してください。

AWS サービス内の一部の出現は、サービス自体と EventBridge の両方で報告できます CloudTrail。例えば、インスタンスを開始または停止する Amazon EC2 API コールは、 を介して EventBridge イベントだけでなく、イベントも生成します CloudTrail。

CloudTrail は、API 発信者とリソース所有者の両方が証跡を作成して Amazon S3 バケットでイベントを受信し、 を介して API 発信者にイベントを配信します EventBridge。API 発信者に加えてリソース所有者は、 を通じてクロスアカウント API コールをモニタリングできます EventBridge。CloudTrailの と の統合 EventBridge により、 イベントに応じてルールベースの自動ワークフローを簡単に設定できます。

256 KB AWS を超える Put*Events API コールイベントは、Put*Events リクエストの最大サイズが 256 KB であるため、イベントパターンとして使用することはできません。使用できる API コールの詳細については、CloudTrail 「サポートされているサービスと統合」を参照してください。

AWS サービスからの読み取り専用管理イベントの受信

経由で AWS のサービスから読み取り専用の管理イベントを受信するように、デフォルトまたはカスタムイベントバスにルールを設定できます CloudTrail。管理イベントは、 AWS アカウント内のリソースで実行される管理オペレーションを可視化します。これらのイベントは、コントロールプレーンオペレーションとも呼ばれます。詳細については、「CloudTrail ユーザーガイド」の「管理イベントのログ記録」を参照してください。

デフォルトまたはカスタムのイベントバスのルールごとに、ルールの状態を設定して、受信するイベントの種類を制御できます。

- がルールとイベントを照合 EventBridge しないように、ルールを無効にします。
- を介して配信される読み取り専用 AWS の管理イベントを除き、 がイベントをルールと EventBridge 照合するようにルールを有効にします CloudTrail。
- を通じて配信される読み取り専用の管理イベントを含め、 がすべてのイベントをルールと EventBridge 照合するようにルールを有効にします CloudTrail。

パートナーイベントバスは AWS イベントを受信しません。

読み取り専用の管理イベントを受信するかどうかを決定する際に考慮すべき点がいくつかあります。

- や などの AWS Key Management Service `GetKeyPolicy`特定の読み取り専用管理イベント`DescribeKey`、または IAM `GetPolicy`や `GetRole`イベントは、一般的な変更イベントよりもはるかに多くのボリュームで発生します。
- イベントが `Describe`、`Get`、または `List` で始まらない場合は、既に読み取り専用の管理イベントを受信している可能性があります。例えば、次の AWS STS APIs からのイベントは、動詞 で始まるとしても、変更イベントです`Get`。
  - `GetFederationToken`
  - `GetSessionToken`

  AWS 、`Describe`、`Get`または のList命名規則に準拠していない読み取り専用の管理イベントのリストについては、「」を参照してください???。

AWS CLI を使用して読み取り専用の管理イベントを受信するルールを作成するには

- `put-rule` コマンドを使用してルールを作成または更新し、パラメータを使用して次のことを行います。
  - ルールがデフォルトのイベントバスに属するか、特定のカスタムイベントバスに属するかを指定します。
  - ルールの状態を `ENABLED_WITH_ALL_CLOUDTRAIL_MANAGEMENT_EVENTS` として設定します。

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

> **Note**
>
> CloudWatch 管理イベントのルールの有効化は、 AWS CLI および AWS CloudFormation テンプレートでのみサポートされます。

Example

次の例は、特定のイベントと照合する方法を示しています。ベストプラクティスは、イベント別に専用のルールを定義し、わかりやすく編集しやすいようにすることです。

この場合、専用ルールは `AssumeRole`の管理イベントと一致します AWS Security Token Service。

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

AWS イベントを生成する サービス

次の表は、イベントを生成する AWS サービスを示しています。サービス名を選択すると、そのサービスと EventBridge 連携する方法の詳細が表示されます。

イベントを生成する各 AWS サービスは、ベストエフォートまたは永続的な配信試行 EventBridge として に送信します。詳細については、「???」を参照してください。

このテーブルには、イベントを に送信する AWS サービスの表示が含まれていますが EventBridge、すべてのサービスが含まれるわけではありません。にイベントを送信するリストにないサービスの場合は EventBridge、ベストエフォート配信を引き受けます。

サービス	試行タイプ
Alexa for Business	ベストエフォート
AWS Account Management	ベストエフォート
Amazon API Gateway	ベストエフォート

サービス	試行タイプ
AWS AppConfig	ベストエフォート
Amazon AppFlow	ベストエフォート
Application Auto Scaling	ベストエフォート
AWS Application Cost Profiler	ベストエフォート
AWS Application Migration Service	ベストエフォート
Amazon Athena	ベストエフォート
AWS Backup	ベストエフォート
AWS Batch	保証付き
Amazon Braket	保証付き
AWS Certificate Manager	ベストエフォート
Amazon Chime	ベストエフォート
Amazon Cloud Directory	ベストエフォート
AWS CloudFormation	保証付き
Amazon CloudFront	ベストエフォート
AWS CloudHSM	ベストエフォート
Amazon CloudSearch	ベストエフォート
AWS CloudShell	ベストエフォート
からのイベント AWS CloudTrail	ベストエフォート
Amazon CloudWatch	保証付き
Amazon CloudWatch Application Insights	ベストエフォート

サービス	試行タイプ
Amazon CloudWatch Internet Monitor	ベストエフォート
Amazon CloudWatch Logs	ベストエフォート
Amazon CloudWatch Synthetics	ベストエフォート
AWS CodeArtifact	保証付き
AWS CodeBuild	ベストエフォート
AWS CodeCommit	ベストエフォート
AWS CodeDeploy	ベストエフォート
Amazon CodeGuru Profiler	ベストエフォート
AWS CodePipeline	ベストエフォート
AWS CodeStar	ベストエフォート
CodeConnections	ベストエフォート
Amazon Cognito ID	ベストエフォート
Amazon Cognito ユーザープール	ベストエフォート
Amazon Cognito Sync	ベストエフォート
AWS Config	ベストエフォート
Amazon Connect	ベストエフォート
Amazon Connect Voice ID	ベストエフォート
AWS Control Tower	ベストエフォート
AWS Database Migration Service	ベストエフォート
AWS Data Exchange	ベストエフォート

サービス	試行タイプ
Amazon Data Lifecycle Manager	ベストエフォート
AWS Data Pipeline	ベストエフォート
AWS DataSync	ベストエフォート
AWS Device Farm	ベストエフォート
Amazon DevOpsGuru	ベストエフォート
AWS Direct Connect	ベストエフォート
AWS Directory Service	ベストエフォート
Amazon DynamoDB	ベストエフォート
AWS Elastic Beanstalk	ベストエフォート
Amazon Elastic Block Store	ベストエフォート
Amazon Elastic Block Store ボリュームへの変更	ベストエフォート
Amazon ElastiCache	ベストエフォート
Amazon Elastic Compute Cloud (Amazon EC2)	ベストエフォート
Amazon EC2 Auto Scaling	ベストエフォート
Amazon EC2 フリート	ベストエフォート
Amazon EC2 スポットインスタンスの中断	ベストエフォート
Amazon Elastic Container Registry	ベストエフォート
Amazon Elastic Container Service	保証付き
AWS Elastic Disaster Recovery	ベストエフォート
Amazon Elastic File System	ベストエフォート

サービス	試行タイプ
Amazon Elastic Kubernetes Service	ベストエフォート
Elastic Load Balancing	ベストエフォート
Amazon Elastic MapReduce	ベストエフォート
Amazon Elastic Transcoder	ベストエフォート
AWS Elemental MediaConnect	ベストエフォート
AWS Elemental MediaConvert	保証付き
AWS Elemental MediaLive	ベストエフォート
AWS Elemental MediaPackage	ベストエフォート
AWS Elemental MediaStore	保証付き
Amazon EMR	ベストエフォート
Amazon EMR on EKS	ベストエフォート
Amazon EMR Serverless	ベストエフォート
Amazon EventBridge スケジュールされたルール	保証付き
Amazon EventBridge スキーマ	ベストエフォート
AWS Fault Injection Service	ベストエフォート
Forecast	ベストエフォート
Amazon GameLift	ベストエフォート
AWS Glue	ベストエフォート
AWS Glue DataBrew	ベストエフォート
AWS Ground Station	ベストエフォート

サービス	試行タイプ
Amazon GuardDuty	ベストエフォート
AWS Health	ベストエフォート
AWS HealthLake	保証付き
AWS Identity and Access Management (IAM)	ベストエフォート
IAM Access Analyzer	ベストエフォート
Amazon Inspector Classic	ベストエフォート
Amazon Inspector	ベストエフォート
AWS IoT	ベストエフォート
AWS IoT Analytics	保証付き
AWS IoT Greengrass V1	ベストエフォート
AWS IoT Greengrass V2	ベストエフォート
Amazon Interactive Video Service	ベストエフォート
Amazon Kinesis	ベストエフォート
Amazon Data Firehose	ベストエフォート
AWS Key Management Service CMK 削除	保証付き
AWS Key Management Service CMK ローテーション	ベストエフォート
AWS Key Management Service インポートされたキーマテリアルの有効期限	ベストエフォート
AWS Lambda	ベストエフォート
Amazon Location Service	保証付き

サービス	試行タイプ
Amazon Machine Learning	ベストエフォート
Amazon Macie	ベストエフォート
Amazon Managed Blockchain	ベストエフォート
AWS Managed Services	ベストエフォート
AWS Management Console サインイン	ベストエフォート
AWS Metering Marketplace	ベストエフォート
AWS Migration Hub	ベストエフォート
AWS Migration Hub Refactor Spaces	ベストエフォート
AWS モニタリング	ベストエフォート
AWS Network Manager	ベストエフォート
Amazon OpenSearch サービス	ベストエフォート
AWS OpsWorks	保証付き
AWS OpsWorks CM	ベストエフォート
AWS Organizations	ベストエフォート
Amazon Polly	ベストエフォート
AWS Private Certificate Authority	ベストエフォート
AWS Proton	ベストエフォート
Amazon QLDB	保証付き
Amazon QuickSight	ベストエフォート
Amazon RDS	ベストエフォート

サービス	試行タイプ
AWS ごみ箱	ベストエフォート
Amazon Redshift	保証付き
Amazon Redshift Data API	ベストエフォート
Amazon Redshift Serverless	ベストエフォート
AWS Resource Access Manager	ベストエフォート
AWS Resource Groups	ベストエフォート
AWS Resource Groups Tagging API	ベストエフォート
Amazon Route 53	ベストエフォート
Amazon Route 53 Recovery 準備状況	ベストエフォート
Amazon SageMaker	ベストエフォート
Savings Plans	ベストエフォート
AWS Secrets Manager	ベストエフォート
AWS Security Hub	保証付き
AWS Security Token Service	ベストエフォート
AWS Server Migration Service	ベストエフォート
AWS Service Catalog	ベストエフォート
AWS Signer	保証付き
Amazon Simple Email Service	ベストエフォート
Amazon Simple Storage Service (Amazon S3)	保証付き
Amazon S3 Glacier	ベストエフォート

サービス	試行タイプ
Amazon S3 on Outposts	ベストエフォート
Amazon Simple Queue Service	ベストエフォート
Amazon Simple Notification Service	ベストエフォート
Amazon Simple Workflow Service	ベストエフォート
AWS Step Functions	ベストエフォート
AWS Storage Gateway	保証付き
AWS Support	ベストエフォート
AWS Systems Manager	ベストエフォート
Amazon Transcribe	ベストエフォート
AWS Transfer Family	ベストエフォート
AWS Transit Gateway	ベストエフォート
Amazon Translate	保証付き
AWS Trusted Advisor	ベストエフォート
AWS WAF	ベストエフォート
AWS WAF リージョン	ベストエフォート
AWS Well-Architected Tool	ベストエフォート
Amazon WorkDocs	ベストエフォート
Amazon WorkSpaces	ベストエフォート
AWS X-Ray	ベストエフォート

AWS サービスによって生成される管理イベント

一般に、管理 (または読み取り専用) イベントを生成する API は、動詞 Describe、Get、または List で始まります。次の表に、この命名規則に従わない AWS サービスと、それらが生成する管理イベントを示します。管理イベントの詳細については、「???」を参照してください。

Describe、Get、または List で始まらない管理イベント

次の表は、、Describe、Getまたは で始まる一般的な命名規則に従わない AWS サービスと、それらが生成する管理イベントの一覧ですList。

サービス	イベント名	イベントタイプ
Alexa for Business	ResolveRoom	API コール
Alexa for Business	SearchAddressBooks	API コール
Alexa for Business	SearchContacts	API コール
Alexa for Business	SearchDevices	API コール
Alexa for Business	SearchProfiles	API コール
Alexa for Business	SearchRooms	API コール
Alexa for Business	SearchSkillGroups	API コール
Alexa for Business	SearchUsers	API コール
IAM Access Analyzer	ValidatePolicy	API コール
AWS AdSpace クリーンルーム	BatchGetSchema	API コール
AWS Amplify UI ビルダー	ExportComponents	API コール
AWS Amplify UI ビルダー	ExportForms	API コール
AWS Amplify UI ビルダー	ExportThemes	API コール

サービス	イベント名	イベントタイプ
Amazon OpenSearch サービス	BatchGetCollection	API コール
Amazon API Gateway	ExportApi	API コール
AWS AppConfig	ValidateConfiguration	API コール
Amazon AppFlow	RetrieveConnectorData	API コール
Amazon CloudWatch Application Insights	UpdateApplicationDashboardConfiguration	API コール
Amazon Athena	BatchGetNamedQuery	API コール
Amazon Athena	BatchGetPreparedStatement	API コール
Amazon Athena	BatchGetQueryExecution	API コール
Amazon Athena	CheckQueryCompatibility	API コール
Amazon Athena	ExportNotebook	API コール
AWS Auto Scaling	AreScalableTargetsRegistered	API コール
AWS Auto Scaling	テスト	API コール
AWS Marketplace	SearchAgreements	API コール
AWS Backup	CreateLegalHold	API コール
AWS Backup	ExportBackupPlanTemplate	API コール
AWS Backup gateway	TestHypervisorConfiguration	API コール
AWS Billing and Cost Management	AWSPaymentInstrumentGateway.取得	コンソールアクション

サービス	イベント名	イベントタイプ
AWS Billing and Cost Management	AWSPaymentPortalService.DescribeMakePaymentPage	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.DescribePaymentsDashboard	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetAccountPreferences	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetAdvancePaySummary	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetAsoBulkDownload	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetBillingContactAddress	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetDocuments	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetEligiblePaymentInstruments	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetEntitiesByIds	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetFundingDocuments	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetKybcValidationStatus	コンソールアクション

サービス	イベント名	イベントタイプ
AWS Billing and Cost Management	AWSPaymentPortalService.GetOneTimePasswordStatus	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentHistory	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentProfileByArn	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentProfileCurrencies	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentProfiles	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentProfileServiceProviders	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetPaymentsDue	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetRemittanceInformation	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetTaxInvoiceMetadata	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetTermsAndConditionsForProgramGroup	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetTransactionsHistory	コンソールアクション

サービス	イベント名	イベントタイプ
AWS Billing and Cost Management	AWSPaymentPortalService.GetUnappliedFunds	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPortalService.GetUnpaidInvoices	コンソールアクション
AWS Billing and Cost Management	AWSPaymentPreferenceGateway.取得	コンソールアクション
AWS Billing and Cost Management	CancelBulkDownload	コンソールアクション
AWS Billing and Cost Management	DownloadCommercialInvoice	コンソールアクション
AWS Billing and Cost Management	DownloadCsv	コンソールアクション
AWS Billing and Cost Management	DownloadDoc	コンソールアクション
AWS Billing and Cost Management	DownloadECSVForBillingPeriod	コンソールアクション
AWS Billing and Cost Management	DownloadPaymentHistory	コンソールアクション
AWS Billing and Cost Management	DownloadRegistrationDocument	コンソールアクション
AWS Billing and Cost Management	DownloadTaxInvoice	コンソールアクション
AWS Billing and Cost Management	FindBankRedirectPaymentInstruments	コンソールアクション
AWS Billing and Cost Management	FindECSVForBillingPeriod	コンソールアクション

サービス	イベント名	イベントタイプ
AWS Billing and Cost Management	ValidateReportDestination	コンソールアクション
AWS Billing and Cost Management	VerifyChinaPaymentEligibility	コンソールアクション
Amazon Braket	SearchCompilations	API コール
Amazon Braket	SearchDevices	API コール
Amazon Braket	SearchQuantumTasks	API コール
Amazon Connect Cases	BatchGetField	API コール
Amazon Connect Cases	SearchCases	API コール
Amazon Connect Cases	SearchRelatedItems	API コール
Amazon Chime	RetrieveDataExports	API コール
Amazon Chime	SearchChannels	API コール
Amazon Chime SDK Identity	DeleteProfile	サービスイベント
Amazon Chime SDK Identity	DeleteWorkTalkAccount	サービスイベント
AWS クリーンルーム	BatchGetSchema	API コール
Amazon Cloud Directory	BatchRead	API コール
Amazon Cloud Directory	LookupPolicy	API コール
AWS CloudFormation	DetectStackDrift	API コール
AWS CloudFormation	DetectStackResourceDrift	API コール
AWS CloudFormation	DetectStackSetDrift	API コール
AWS CloudFormation	EstimateTemplateCost	API コール

サービス	イベント名	イベントタイプ
AWS CloudFormation	ValidateTemplate	API コール
AWS CloudShell	RedeemCode	API コール
AWS CloudTrail	LookupEvents	API コール
AWS CodeArtifact	ReadFromRepository	API コール
AWS CodeArtifact	SearchPackages	API コール
AWS CodeArtifact	VerifyResourcesExistForTagris	API コール
AWS CodeBuild	BatchGetBuildBatches	API コール
AWS CodeBuild	BatchGetBuilds	API コール
AWS CodeBuild	BatchGetProjects	API コール
AWS CodeBuild	BatchGetReportGroups	API コール
AWS CodeBuild	BatchGetReports	API コール
AWS CodeBuild	BatchPutCodeCoverages	API コール
AWS CodeBuild	BatchPutTestCases	API コール
AWS CodeBuild	RequestBadge	サービスイベント
AWS CodeCommit	BatchDescribeMergeConflicts	API コール
AWS CodeCommit	BatchGetCommits	API コール
AWS CodeCommit	BatchGetPullRequests	API コール
AWS CodeCommit	BatchGetRepositories	API コール
AWS CodeCommit	EvaluatePullRequestApprovalRules	API コール

サービス	イベント名	イベントタイプ
AWS CodeCommit	GitPull	API コール
AWS CodeDeploy	BatchGetApplicationRevisions	API コール
AWS CodeDeploy	BatchGetApplications	API コール
AWS CodeDeploy	BatchGetDeploymentGroups	API コール
AWS CodeDeploy	BatchGetDeployment Instances	API コール
AWS CodeDeploy	BatchGetDeployments	API コール
AWS CodeDeploy	BatchGetDeploymentTargets	API コール
AWS CodeDeploy	BatchGetOnPremises Instances	API コール
Amazon CodeGuru Profiler	BatchGetFrameMetricData	API コール
Amazon CodeGuru Profiler	SubmitFeedback	API コール
AWS CodePipeline	PollForJobs	API コール
AWS CodePipeline	PollForThirdPartyJobs	API コール
CodeConnections	StartAppRegistrationHandshake	API コール
CodeConnections	StartOAuthHandshake	API コール
CodeConnections	ValidateHostWebhook	API コール
Amazon CodeWhisperer	CreateCodeScan	API コール
Amazon CodeWhisperer	CreateProfile	API コール
Amazon CodeWhisperer	CreateUploadUrl	API コール
Amazon CodeWhisperer	GenerateRecommendations	API コール

サービス	イベント名	イベントタイプ
Amazon CodeWhisperer	UpdateProfile	API コール
Amazon Cognito ID	LookupDeveloperIdentity	API コール
Amazon Cognito ユーザープール	AdminGetDevice	API コール
Amazon Cognito ユーザープール	AdminGetUser	API コール
Amazon Cognito ユーザープール	AdminListDevices	API コール
Amazon Cognito ユーザープール	AdminListGroupsForUser	API コール
Amazon Cognito ユーザープール	AdminListUserAuthEvents	API コール
Amazon Cognito ユーザープール	Beta_Authorize_GET	サービスイベント
Amazon Cognito ユーザープール	Confirm_GET	サービスイベント
Amazon Cognito ユーザープール	ConfirmForgotPassword_GET	サービスイベント
Amazon Cognito ユーザープール	Error_GET	サービスイベント
Amazon Cognito ユーザープール	ForgotPassword_GET	サービスイベント
Amazon Cognito ユーザープール	IntrospectToken	API コール

サービス	イベント名	イベントタイプ
Amazon Cognito ユーザープール	Login_Error_POST	サービスイベント
Amazon Cognito ユーザープール	Login_GET	サービスイベント
Amazon Cognito ユーザープール	Mfa_GET	サービスイベント
Amazon Cognito ユーザープール	MfaOption_GET	サービスイベント
Amazon Cognito ユーザープール	ResetPassword_GET	サービスイベント
Amazon Cognito ユーザープール	Signup_GET	サービスイベント
Amazon Cognito ユーザープール	UserInfo_GET	サービスイベント
Amazon Cognito ユーザープール	UserInfo_POST	サービスイベント
Amazon Cognito Sync	BulkPublish	API コール
Amazon Comprehend	BatchContainsPiiEntities	API コール
Amazon Comprehend	BatchDetectDominantLanguage	API コール
Amazon Comprehend	BatchDetectEntities	API コール
Amazon Comprehend	BatchDetectKeyPhrases	API コール
Amazon Comprehend	BatchDetectPiiEntities	API コール
Amazon Comprehend	BatchDetectSentiment	API コール

サービス	イベント名	イベントタイプ
Amazon Comprehend	BatchDetectSyntax	API コール
Amazon Comprehend	BatchDetectTargetedSentiment	API コール
Amazon Comprehend	ClassifyDocument	API コール
Amazon Comprehend	ContainsPiiEntities	API コール
Amazon Comprehend	DetectDominantLanguage	API コール
Amazon Comprehend	DetectEntities	API コール
Amazon Comprehend	DetectKeyPhrases	API コール
Amazon Comprehend	DetectPiiEntities	API コール
Amazon Comprehend	DetectSentiment	API コール
Amazon Comprehend	DetectSyntax	API コール
Amazon Comprehend	DetectTargetedSentiment	API コール
Amazon Comprehend	DetectToxicContent	API コール
AWS Compute Optimizer	ExportAutoScalingGroupRecommendations	API コール
AWS Compute Optimizer	ExportEBSVolumeRecommendations	API コール
AWS Compute Optimizer	ExportECInstanceRecommendations	API コール
AWS Compute Optimizer	ExportECSServiceRecommendations	API コール
AWS Compute Optimizer	ExportLambdaFunctionRecommendations	API コール

サービス	イベント名	イベントタイプ
AWS Compute Optimizer	ExportRDSInstanceRecommendations	API コール
AWS Config	BatchGetAggregateResourceConfig	API コール
AWS Config	BatchGetResourceConfig	API コール
AWS Config	SelectAggregateResourceConfig	API コール
AWS Config	SelectResourceConfig	API コール
Amazon Connect	AdminGetEmergencyAccessToken	API コール
Amazon Connect	SearchQueues	API コール
Amazon Connect	SearchRoutingProfiles	API コール
Amazon Connect	SearchSecurityProfiles	API コール
Amazon Connect	SearchUsers	API コール
AWS Glue DataBrew	SendProjectSessionAction	API コール
AWS Data Pipeline	EvaluateExpression	API コール
AWS Data Pipeline	QueryObjects	API コール
AWS Data Pipeline	ValidatePipelineDefinition	API コール
AWS DataSync	VerifyResourcesExistForTagris	API コール
AWS DeepLens	BatchGetDevice	API コール
AWS DeepLens	BatchGetModel	API コール

サービス	イベント名	イベントタイプ
AWS DeepLens	BatchGetProject	API コール
AWS DeepLens	CreateDeviceCertificates	API コール
AWS DeepRacer	AdminGetAccountConfig	API コール
AWS DeepRacer	AdminListAssociatedUsers	API コール
AWS DeepRacer	TestRewardFunction	API コール
AWS DeepRacer	VerifyResourcesExistForTagris	API コール
Amazon Detective	BatchGetGraphMemberDatasources	API コール
Amazon Detective	BatchGetMembershipDatasources	API コール
Amazon Detective	SearchGraph	API コール
Amazon DevOpsGuru	SearchInsights	API コール
Amazon DevOpsGuru	SearchOrganizationInsights	API コール
AWS Database Migration Service	BatchStartRecommendations	API コール
AWS Database Migration Service	ModifyRecommendation	API コール
AWS Database Migration Service	StartRecommendations	API コール
AWS Database Migration Service	VerifyResourcesExistForTagris	API コール
AWS Directory Service	VerifyTrust	API コール

サービス	イベント名	イベントタイプ
Amazon Elastic Compute Cloud	ConfirmProductInstance	API コール
Amazon Elastic Compute Cloud	ReportInstanceStatus	API コール
Amazon Elastic Container Registry	BatchCheckLayerAvailability	API コール
Amazon Elastic Container Registry	BatchGetImage	API コール
Amazon Elastic Container Registry	BatchGetImageReferrer	API コール
Amazon Elastic Container Registry	BatchGetRepository ScanningConfiguration	API コール
Amazon Elastic Container Registry	DryRunEvent	サービスイベント
Amazon Elastic Container Registry	PolicyExecutionEvent	サービスイベント
Amazon Elastic Container Registry Public	BatchCheckLayerAvailability	API コール
Amazon Elastic Container Service	DiscoverPollEndpoint	API コール
Amazon Elastic Container Service	FindSubfleetRoute	API コール
Amazon Elastic Container Service	ValidateResources	API コール
Amazon Elastic Container Service	VerifyTaskSetsExist	API コール

サービス	イベント名	イベントタイプ
Amazon Elastic Kubernetes Service	AccessKubernetesApi	API コール
AWS Elastic Beanstalk	CheckDNSAvailability	API コール
AWS Elastic Beanstalk	RequestEnvironmentInfo	API コール
AWS Elastic Beanstalk	RetrieveEnvironmentInfo	API コール
AWS Elastic Beanstalk	ValidateConfigurationSettings	API コール
Amazon Elastic File System	NewClientConnection	サービスイベント
Amazon Elastic File System	UpdateClientConnection	サービスイベント
Amazon Elastic Transcoder	ReadJob	API コール
Amazon Elastic Transcoder	ReadPipeline	API コール
Amazon Elastic Transcoder	ReadPreset	API コール
Amazon EventBridge	TestEventPattern	API コール
Amazon EventBridge	TestScheduleExpression	API コール
Amazon FinSpace API	BatchListCatalogNodesByDataset	API コール
Amazon FinSpace API	BatchListNodesByDataset	API コール
Amazon FinSpace API	BatchValidateAccess	API コール
Amazon FinSpace API	CreateAuditRecordsQuery	API コール
Amazon FinSpace API	SearchDatasets	API コール
Amazon FinSpace API	SearchDatasetsV	API コール
Amazon FinSpace API	ValidateIdToken	API コール

サービス	イベント名	イベントタイプ
AWS Firewall Manager	DisassociateAdminAccount	API コール
Amazon Forecast	InvokeForecastEndpoint	API コール
Amazon Forecast	QueryFeature	API コール
Amazon Forecast	QueryForecast	API コール
Amazon Forecast	QueryWhatIfForecast	API コール
Amazon Forecast	VerifyResourcesExistForTagris	API コール
Amazon Fraud Detector	BatchGetVariable	API コール
Amazon Fraud Detector	VerifyResourcesExistForTagris	API コール
FreeRTOS	VerifyEmailAddress	API コール
Amazon GameLift	RequestUploadCredentials	API コール
Amazon GameLift	ResolveAlias	API コール
Amazon GameLift	SearchGameSessions	API コール
Amazon GameLift	ValidateMatchmakingRuleSet	API コール
Amazon GameSparks	ExportSnapshot	API コール
Amazon Location Service	BatchGetDevicePosition	API コール
Amazon Location Service	CalculateRoute	API コール
Amazon Location Service	CalculateRouteMatrix	API コール
Amazon Location Service	SearchPlaceIndexForPosition	API コール
Amazon Location Service	SearchPlaceIndexForSuggestions	API コール

サービス	イベント名	イベントタイプ
Amazon Location Service	SearchPlaceIndexForText	API コール
Amazon S3 Glacier	InitiateJob	API コール
AWS Glue	BatchGetBlueprints	API コール
AWS Glue	BatchGetColumnStatisticsForTable	API コール
AWS Glue	BatchGetCrawlers	API コール
AWS Glue	BatchGetCustomEntityTypes	API コール
AWS Glue	BatchGetDataQualityResult	API コール
AWS Glue	BatchGetDevEndpoints	API コール
AWS Glue	BatchGetJobs	API コール
AWS Glue	BatchGetMLTransform	API コール
AWS Glue	BatchGetPartition	API コール
AWS Glue	BatchGetTriggers	API コール
AWS Glue	BatchGetWorkflows	API コール
AWS Glue	QueryJobRuns	API コール
AWS Glue	QueryJobRunsAggregated	API コール
AWS Glue	QueryJobs	API コール
AWS Glue	QuerySchemaVersionMetadata	API コール
AWS Glue	SearchTables	API コール
AWS HealthLake	ReadResource	API コール

サービス	イベント名	イベントタイプ
AWS HealthLake	SearchWithGet	API コール
AWS HealthLake	SearchWithPost	API コール
AWS Identity and Access Management	GenerateCredentialReport	API コール
AWS Identity and Access Management	GenerateOrganizationsAccessReport	API コール
AWS Identity and Access Management	GenerateServiceLastAccessedDetails	API コール
AWS Identity and Access Management	SimulateCustomPolicy	API コール
AWS Identity and Access Management	SimulatePrincipalPolicy	API コール
AWS ID ストア	IsMemberInGroups	API コール
AWS Identity Store 認証	BatchGetSession	API コール
Amazon Inspector Classic	PreviewAgents	API コール
Amazon Inspector Classic	BatchGetAccountStatus	API コール
Amazon Inspector Classic	BatchGetFreeTrialInfo	API コール
Amazon Inspector Classic	BatchGetMember	API コール
AWS Invoicing	ValidateDocumentDeliveryS3LocationInfo	API コール
AWS IoT	SearchIndex	API コール
AWS IoT	TestAuthorization	API コール
AWS IoT	TestInvokeAuthorizer	API コール

サービス	イベント名	イベントタイプ
AWS IoT	ValidateSecurityProfileBehaviors	API コール
AWS IoT Analytics	SampleChannelData	API コール
AWS IoT SiteWise	GatewaysVerifyResourcesExistForTagrisInternal	API コール
AWS IoT Things Graph	SearchEntities	API コール
AWS IoT Things Graph	SearchFlowExecutions	API コール
AWS IoT Things Graph	SearchFlowTemplates	API コール
AWS IoT Things Graph	SearchSystemInstances	API コール
AWS IoT Things Graph	SearchSystemTemplates	API コール
AWS IoT Things Graph	SearchThings	API コール
AWS IoT TwinMaker	ExecuteQuery	API コール
AWS IoT Wireless	CreateNetworkAnalyzerConfiguration	API コール
AWS IoT Wireless	DeleteNetworkAnalyzerConfiguration	API コール
AWS IoT Wireless	DeregisterWirelessDevice	API コール
Amazon Interactive Video Service	BatchGetChannel	API コール
Amazon Interactive Video Service	BatchGetStreamKey	API コール
Amazon Kendra	BatchGetDocumentStatus	API コール
Amazon Kendra	Query	API コール

サービス	イベント名	イベントタイプ
Amazon Managed Service for Apache Flink	DiscoverInputSchema	API コール
AWS Key Management Service	Decrypt	API コール
AWS Key Management Service	暗号化	API コール
AWS Key Management Service	GenerateDataKey	API コール
AWS Key Management Service	GenerateDataKeyPair	API コール
AWS Key Management Service	GenerateDataKeyPairWithoutPlaintext	API コール
AWS Key Management Service	GenerateDataKeyWithoutPlaintext	API コール
AWS Key Management Service	GenerateMac	API コール
AWS Key Management Service	GenerateRandom	API コール
AWS Key Management Service	ReEncrypt	API コール
AWS Key Management Service	Sign	API コール
AWS Key Management Service	検証	API コール
AWS Key Management Service	VerifyMac	API コール

サービス	イベント名	イベントタイプ
AWS Lake Formation	SearchDatabasesByLFTags	API コール
AWS Lake Formation	SearchTablesByLFTags	API コール
AWS Lake Formation	StartQueryPlanning	API コール
Amazon Lex	BatchCreateCustomVocabularyItem	API コール
Amazon Lex	BatchDeleteCustomVocabularyItem	API コール
Amazon Lex	BatchUpdateCustomVocabularyItem	API コール
Amazon Lex	DeleteCustomVocabulary	API コール
Amazon Lex	SearchAssociatedTranscripts	API コール
Amazon Lightsail	CreateGUISessionAccessDetails	API コール
Amazon Lightsail	DownloadDefaultKeyPair	API コール
Amazon Lightsail	IsVpcPeered	API コール
Amazon CloudWatch Logs	FilterLogEvents	API コール
Amazon Macie	BatchGetCustomDataIdentifiers	API コール
Amazon Macie	UpdateFindingsFilter	API コール
AWS Elemental MediaConnect	ManagedDescribeFlow	API コール
AWS Elemental MediaConnect	PrivateDescribeFlowMeta	API コール

サービス	イベント名	イベントタイプ
AWS Application Migration Service	OperationalDescribeJobLogItems	API コール
AWS Application Migration Service	OperationalDescribeJobs	API コール
AWS Application Migration Service	OperationalDescribeReplicationConfigurationTemplates	API コール
AWS Application Migration Service	OperationalDescribeSourceServer	API コール
AWS Application Migration Service	OperationalGetLaunchConfiguration	API コール
AWS Application Migration Service	OperationalListSourceServers	API コール
AWS Application Migration Service	VerifyClientRoleForMgn	API コール
AWS HealthOmics	VerifyResourceExists	API コール
AWS HealthOmics	VerifyResourcesExistForTagris	API コール
Amazon Polly	SynthesizeLongSpeech	API コール
Amazon Polly	SynthesizeSpeech	API コール
Amazon Polly	SynthesizeSpeechGet	API コール
AWS マネージドプライベートネットワークを提供する サービス	Ping	API コール
AWS Proton	DeleteEnvironmentTemplateVersion	API コール

サービス	イベント名	イベントタイプ
AWS Proton	DeleteServiceTemplateVersion	API コール
Amazon QLDB	ShowCatalog	API コール
Amazon QuickSight	GenerateEmbedUrlForAnonymousUser	API コール
Amazon QuickSight	GenerateEmbedUrlForRegisteredUser	API コール
Amazon QuickSight	QueryDatabase	サービスイベント
Amazon QuickSight	SearchAnalyses	API コール
Amazon QuickSight	SearchDashboards	API コール
Amazon QuickSight	SearchDataSets	API コール
Amazon QuickSight	SearchDataSources	API コール
Amazon QuickSight	SearchFolders	API コール
Amazon QuickSight	SearchGroups	API コール
Amazon QuickSight	SearchUsers	API コール
Amazon Relational Database Service	DownloadCompleteDBLogFile	API コール
Amazon Relational Database Service	DownloadDBLogFilePortion	API コール
Amazon Rekognition	CompareFaces	API コール
Amazon Rekognition	DetectCustomLabels	API コール
Amazon Rekognition	DetectFaces	API コール

サービス	イベント名	イベントタイプ
Amazon Rekognition	DetectLabels	API コール
Amazon Rekognition	DetectModerationLabels	API コール
Amazon Rekognition	DetectProtectiveEquipment	API コール
Amazon Rekognition	DetectText	API コール
Amazon Rekognition	RecognizeCelebrities	API コール
Amazon Rekognition	SearchFaces	API コール
Amazon Rekognition	SearchFacesByImage	API コール
Amazon Rekognition	SearchUsers	API コール
Amazon Rekognition	SearchUsersByImage	API コール
AWS Resource Explorer	BatchGetView	API コール
AWS Resource Explorer	検索	API コール
AWS Resource Groups	SearchResources	API コール
AWS Resource Groups	ValidateResourceSharing	API コール
AWS RoboMaker	BatchDescribeSimulationJob	API コール
Amazon Route 53	TestDNSAnswer	API コール
Amazon Route 53 Domains	checkAvailabilities	API コール
Amazon Route 53 Domains	CheckDomainAvailability	API コール
Amazon Route 53 Domains	checkDomainTransferability	API コール
Amazon Route 53 Domains	CheckDomainTransferability	API コール
Amazon Route 53 Domains	isEmailReachable	API コール

サービス	イベント名	イベントタイプ
Amazon Route 53 Domains	searchDomains	API コール
Amazon Route 53 Domains	sendVerificationMessage	API コール
Amazon Route 53 Domains	ViewBilling	API コール
Amazon Route 53 Domains	viewBilling	API コール
Amazon CloudWatch RUM	BatchGetRumMetricDefinitions	API コール
Amazon Simple Storage Service	echo	API コール
Amazon Simple Storage Service	GenerateInventory	サービスイベント
Amazon SageMaker	BatchDescribeModelPackage	API コール
Amazon SageMaker	DeleteModelCard	API コール
Amazon SageMaker	QueryLineage	API コール
Amazon SageMaker	RenderUiTemplate	API コール
Amazon SageMaker	検索	API コール
Amazon EventBridge スキーマ	ExportSchema	API コール
Amazon EventBridge スキーマ	SearchSchemas	API コール
Amazon SimpleDB	DomainMetadata	API コール
AWS Secrets Manager	ValidateResourcePolicy	API コール
AWS Service Catalog	ScanProvisionedProducts	API コール
AWS Service Catalog	SearchProducts	API コール
AWS Service Catalog	SearchProductsAsAdmin	API コール

サービス	イベント名	イベントタイプ
AWS Service Catalog	SearchProvisionedProducts	API コール
Amazon SES	BatchGetMetricData	API コール
Amazon SES	TestRenderEmailTemplate	API コール
Amazon SES	TestRenderTemplate	API コール
Amazon Simple Notification Service	CheckIfPhoneNumber IsOptedOut	API コール
AWS SQL Workbench	BatchGetNotebookCell	API コール
AWS SQL Workbench	ExportNotebook	API コール
Amazon EC2 Systems Manager	ExecuteApi	API コール
AWS Systems Manager Incident Manager	DeleteContactChannel	API コール
AWS IAM Identity Center	IsMemberInGroup	API コール
AWS IAM Identity Center	SearchGroups	API コール
AWS IAM Identity Center	SearchUsers	API コール
AWS STS	AssumeRole	API コール
AWS STS	AssumeRoleWithSAML	API コール
AWS STS	AssumeRoleWithWebIdentity	API コール
AWS STS	DecodeAuthorizationMessage	API コール
AWS 税金設定	BatchGetTaxExemptions	API コール
AWS WAFV2	CheckCapacity	API コール

サービス	イベント名	イベントタイプ
AWS WAFV2	GenerateMobileSdkReleaseUrl	API コール
AWS Well-Architected Tool	ExportLens	API コール
AWS Well-Architected Tool	TagResource	API コール
AWS Well-Architected Tool	UntagResource	API コール
AWS Well-Architected Tool	UpdateGlobalSettings	API コール
Amazon Connect Wisdom	QueryAssistant	API コール
Amazon Connect Wisdom	SearchContent	API コール
Amazon Connect Wisdom	SearchSessions	API コール
Amazon WorkDocs	AbortDocumentVersionUpload	API コール
Amazon WorkDocs	AddUsersToGroup	API コール
Amazon WorkDocs	BatchGetUsers	API コール
Amazon WorkDocs	CheckAlias	API コール
Amazon WorkDocs	CompleteDocumentVersionUpload	API コール
Amazon WorkDocs	CreateAnnotation	API コール
Amazon WorkDocs	CreateComment	API コール
Amazon WorkDocs	CreateFeedbackRequest	API コール
Amazon WorkDocs	CreateFolder	API コール
Amazon WorkDocs	CreateGroup	API コール
Amazon WorkDocs	CreateShare	API コール

サービス	イベント名	イベントタイプ
Amazon WorkDocs	CreateUser	API コール
Amazon WorkDocs	DeleteAnnotation	API コール
Amazon WorkDocs	DeleteComment	API コール
Amazon WorkDocs	DeleteDocument	API コール
Amazon WorkDocs	DeleteFeedbackRequest	API コール
Amazon WorkDocs	DeleteFolder	API コール
Amazon WorkDocs	DeleteFolderContents	API コール
Amazon WorkDocs	DeleteGroup	API コール
Amazon WorkDocs	DeleteOrganizationShare	API コール
Amazon WorkDocs	DeleteUser	API コール
Amazon WorkDocs	DownloadDocumentVersion	API コール
Amazon WorkDocs	DownloadDocumentVersionUnderlays	API コール
Amazon WorkDocs	InitiateDocumentVersionUpload	API コール
Amazon WorkDocs	LogoutUser	API コール
Amazon WorkDocs	PaginatedOrganizationActivity	API コール
Amazon WorkDocs	PublishAnnotations	API コール
Amazon WorkDocs	PublishComments	API コール
Amazon WorkDocs	RestoreDocument	API コール
Amazon WorkDocs	RestoreFolder	API コール

サービス	イベント名	イベントタイプ
Amazon WorkDocs	SearchGroups	API コール
Amazon WorkDocs	SearchOrganizationUsers	API コール
Amazon WorkDocs	TransferUserResources	API コール
Amazon WorkDocs	UpdateAnnotation	API コール
Amazon WorkDocs	UpdateComment	API コール
Amazon WorkDocs	UpdateDocument	API コール
Amazon WorkDocs	UpdateDocumentVersion	API コール
Amazon WorkDocs	UpdateFolder	API コール
Amazon WorkDocs	UpdateGroup	API コール
Amazon WorkDocs	UpdateOrganization	API コール
Amazon WorkDocs	UpdateUser	API コール
Amazon WorkMail	AssumeImpersonationRole	API コール
Amazon WorkMail	QueryDnsRecords	API コール
Amazon WorkMail	SearchMembers	API コール
Amazon WorkMail	TestAvailabilityConfiguration	API コール
Amazon WorkMail	TestInboundMailFlowRules	API コール
Amazon WorkMail	TestOutboundMailFlowRules	API コール

EventBridge イベント詳細リファレンス

EventBridge 自体は、次のイベントを発行します。これらのイベントは、他の AWS サービスと同様に、デフォルトのイベントバスに自動的に送信されます。

すべてのイベントに含まれるメタデータフィールドの定義については、「」を参照してくださいthe section called “イベント構造リファレンス”。

トピック

- スケジュールされたイベント
- スキーマが作成されました
- スキーマバージョンが作成されました

スケジュールされたイベント

以下は、Scheduled Eventイベントの詳細フィールドです。

source フィールドと detail-typeフィールドは、 EventBridge イベントに特定の値が含まれているため含まれます。すべてのイベントに含まれる他のメタデータフィールドの定義については、「」を参照してくださいthe section called “イベント構造リファレンス”。

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

detail-type

イベントのタイプを示します。

このイベントの場合、この値は ですScheduled Event。

必須: はい

source

イベントを発生させたサービスを識別します。イベントの場合 EventBridge 、この値は ですaws.events。

必須: はい

`detail`

イベントに関する情報を含む JSON オブジェクト。このフィールドの内容は、イベントを生成するサービスによって決まります。

必須: はい

このオブジェクトには`Scheduled Event`、イベントに必須のフィールドはありません。

Example スケジュールされたイベントイベントの例

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

スキーマが作成されました

以下は、`Schema Created`イベントの詳細フィールドです。

スキーマが作成されると、 は `Schema Created`と `Schema Version Created`イベントの両方 EventBridge を送信します。

`source` フィールドと `detail-type`フィールドは、 EventBridge イベントに特定の値が含まれているため含まれます。すべてのイベントに含まれる他のメタデータフィールドの定義については、「」を参照してくださいthe section called “イベント構造リファレンス”。

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

```
    "SchemaType" : "String",
    "RegistryName" : "String",
    "CreationDate" : "DateTime",
    "Version" : "Number"
  }
}
```

`detail-type`

イベントのタイプを示します。

このイベントの場合、この値は ですSchema Created。

必須: はい

`source`

イベントを発生させたサービスを識別します。イベントの場合 EventBridge 、この値は ですaws.schemas。

必須: はい

`detail`

イベントに関する情報を含む JSON オブジェクト。このフィールドの内容は、イベントを生成するサービスによって決まります。

必須: はい

このイベントの場合、このデータには以下が含まれます。

`SchemaName`

スキーマの名前。

必須: はい

`SchemaType`

スキーマのタイプ。

有効な値: `OpenApi3` | `JSONSchemaDraft4`

必須: はい

`RegistryName`

スキーマを含むレジストリの名前。

必須: はい

`CreationDate`

スキーマが作成された日付。

必須: はい

`Version`

スキーマのバージョン。

`Schema Created` イベントの場合、この値は常に になります`1`。

必須: はい

Example スキーマ作成イベントの例

```
{
  "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"
  }
}
```

スキーマバージョンが作成されました

以下は、`Schema Version Created`イベントの詳細フィールドです。

スキーマが作成されると、 は Schema Createdと Schema Version Createdイベントの両方 EventBridge を送信します。

source フィールドと detail-typeフィールドは、 EventBridge イベントに特定の値が含まれているため含まれます。すべてのイベントに含まれる他のメタデータフィールドの定義については、「」を参照してくださいthe section called “イベント構造リファレンス”。

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

detail-type

イベントのタイプを示します。

このイベントの場合、この値は ですSchema Version Created。

必須: はい

source

イベントを発生させたサービスを識別します。イベントの場合 EventBridge 、この値は ですaws.schemas。

必須: はい

detail

イベントに関する情報を含む JSON オブジェクト。このフィールドの内容は、イベントを生成するサービスによって決まります。

必須: はい

このイベントの場合、このデータには以下が含まれます。

`SchemaName`

スキーマの名前。

必須: はい

`SchemaType`

スキーマのタイプ。

有効な値: `OpenApi3` | `JSONSchemaDraft4`

必須: はい

`RegistryName`

スキーマを含むレジストリの名前。

必須: はい

`CreationDate`

スキーマバージョンが作成された日付。

必須: はい

`Version`

スキーマのバージョン。

必須: はい

Example スキーマバージョン作成イベントの例

```
{
  "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"
  }
}
```

Amazon との SaaS パートナーからのイベントの受信 EventBridge

SaaS パートナーアプリケーションおよびサービスからイベントを受信するには、そのパートナーからのパートナーイベントソースが必要です。その後、パートナーイベントバスを作成し、対応するパートナーイベントソースに一致させることができます。

次の動画では、 との SaaS 統合について説明します EventBridge: Software as a Service (SaaS) パートナー

トピック

- サポートされる SaaS パートナー統合
- SaaS 統合からイベントを受信する EventBridge ように Amazon を設定する
- SaaS パートナーイベントに一致するルールの作成
- AWS Lambda 関数 URLsを使用したイベントの受信
- Salesforce からのイベントの受信

サポートされる SaaS パートナー統合

EventBridge は、以下の SaaS パートナー統合をサポートしています。

- Adobe
- Auth0
- Blitline
- BUIDLHub
- Buildkite

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

- TriggerMesh
- Whispir
- Zendesk
- [Amazon Seller Partner API] (Amazon セラーパートナー API)

パートナーイベントソースは、次のリージョンで利用できます。

Code	名前
us-east-1	米国東部 (バージニア北部)
us-east-2	米国東部 (オハイオ)
us-west-1	米国西部 (北カリフォルニア)
us-west-2	米国西部 (オレゴン)
ca-central-1	カナダ (中部)
eu-central-1	欧州 (フランクフルト)
eu-central-2	欧州 (チューリッヒ)
eu-west-1	欧州 (アイルランド)
eu-west-2	欧州 (ロンドン)
eu-west-3	欧州 (パリ)
eu-north-1	欧州 (ストックホルム)
eu-south-1	欧州 (ミラノ)
eu-south-2	欧州 (スペイン)
af-south-1	アフリカ (ケープタウン)
ap-south-1	アジアパシフィック (ムンバイ)
ap-south-2	アジアパシフィック (ハイデラバード)

Code	名前
ap-east-1	アジアパシフィック (香港)
ap-northeast-1	アジアパシフィック (東京)
ap-northeast-2	アジアパシフィック (ソウル)
ap-northeast-3	アジアパシフィック (大阪)
ap-southeast-1	アジアパシフィック (シンガポール)
ap-southeast-2	アジアパシフィック (シドニー)
ap-southeast-3	アジアパシフィック (ジャカルタ)
ap-southeast-4	アジアパシフィック (メルボルン)
cn-north-1	中国 (北京)
cn-northwest-1	中国 (寧夏)
me-central-1	中東 (アラブ首長国連邦)
me-south-1	中東 (バーレーン)
sa-east-1	南米 (サンパウロ)
il-central-1	イスラエル (テルアビブ)

SaaS 統合からイベントを受信する EventBridge ように Amazon を設定する

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで、[Partner event sources (パートナーイベントソース)] を選択します。
3. 目的のパートナーを見つけ、そのパートナーの [Set up] (設定) を選択します。
4. アカウント ID をクリップボードにコピーするには、 [Copy] (コピー) を選択します。
5. ナビゲーションペインで、[Partner event sources (パートナーイベントソース)] を選択します。

6. パートナーのウェブサイトにアクセスして手順に従い、アカウント ID を使用してパートナーイベントソースを作成します。作成したイベントソースは、アカウントでのみ使用できます。
7. EventBridge コンソールに戻り、ナビゲーションペインでパートナーイベントソースを選択します。
8. パートナーイベントソースの横にあるボタンを選択し、[Associate with event bus] (イベントバスと関連付ける) を選択します。

   そのイベントソースのステータスが `Pending` から `Active` に変わり、イベントバスの名前がパートナーイベントソース名と一致するように更新されます。これで、パートナーイベントソースからのイベントに一致するルールの作成を開始できます。詳細については、「SaaS パートナーイベントに一致するルールの作成」を参照してください。

   > **Note**
   >
   > パートナーがパートナーイベントソースに公開したイベントのうち、イベントバスに関連付けられていないものはすぐに削除されます。これらのイベントは、 に保管中に保持されません EventBridge。

SaaS パートナーイベントに一致するルールの作成

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで Rules] (ルール) を選択します。
3. ルールの作成 を選択します。
4. ルールの名前と説明を入力します。

   ルールには、同じリージョン内および同じイベントバス上の別のルールと同じ名前を付けることはできません。
5. Event bus] (イベントバス) では、このルールに関連付けるイベントバスを選択します。このルールをアカウントからのイベントと一致させるには、 AWS のデフォルトのイベントバスを選択します。アカウントの AWS サービスがイベントを発行すると、常にアカウントのデフォルトのイベントバスに移動します。
6. [Rule type] (ルールタイプ) では、[Rule with an event pattern] (イベントパターンを持つルール) を選択します。
7. 次へ をクリックします。
8. [Event source] (イベントソース) では、[Other] (その他) を選択します。

9. (オプション) [Sample events] (イベント例) では、イベントのタイプを選択します。
10. [Event pattern] (イベントパターン) では、JSON イベントパターンを入力します。
11. 次へ をクリックします。
12. ターゲットタイプ] では、AWS サービス] を選択します。
13. ターゲットを選択 で、 がイベントパターンに一致するイベント EventBridge を検出したときに情報を送信する AWS サービスを選択します。
14. 表示されるフィールドは、選択したサービスによって異なります。必要に応じて、このターゲットタイプに固有の情報を入力します。
15. 多くのターゲットタイプでは、 はターゲットにイベントを送信するためのアクセス許可 EventBridge が必要です。このような場合は、ルールの実行に必要な IAM ロール EventBridge を作成できます。次のいずれかを行います。

    - 自動的に IAM ロールを作成するには、この特定のリソースに対して新しいロールを作成する を選択します。
    - 以前に作成した IAM ロールを使用するには、[Use existing role] (既存のロールの使用) をクリックし、ドロップダウンリストから既存のロールを選択します。

16. (オプション) [Additional settings] (追加設定) では、以下を実行します。

    a. Maximum age of event (最大イベント有効期間) に、1 分 (00:01) から 24 時間 (24:00) の間の値を入力します。

    b. 再試行 で、0 ~ 185 の数値を入力します。

    c. デッドレターキュー では、標準の Amazon SQS キューをデッドレターキューとして使用するかどうかを選択します。 は、このルールに一致する EventBridge イベントがターゲットに正常に配信されない場合、デッドレターキューに送信します。次のいずれかを行います。

        - デッドレターキューを使用しない場合は、[None] (なし) を選択します。
        - Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue(デッドレターキューとして使用する現在の アカウントの Amazon SQS キューを選択) を選択し、ドロップダウンリストから使用するキューを選択します。
        - 他の AWS アカウントの Amazon SQS キューをデッドレターキューとして選択を選択し、使用するキューの ARN を入力します。メッセージを送信する EventBridge アクセス許可を付与するリソースベースのポリシーをキューにアタッチする必要があります。詳細については、「デッドレターキューへのアクセス許可の付与」を参照してください。

17. (オプション) [Add another target] (別のターゲットを追加) を選択して、このルールに別のターゲットを追加します。
18. 次へ をクリックします。
19. (オプション) ルールに 1 つ以上のタグを入力します。詳細については、「Amazon EventBridge タグ」を参照してください。
20. 次へ をクリックします。
21. ルールの詳細を確認し、ルールの作成 を選択します。

AWS Lambda 関数 URLsを使用したイベントの受信

> **Note**
>
> パートナーがインバウンド Webhook にアクセスできるようにするには、サードパーティーパートナーから送信された認証署名を検証することで、Lambda アプリケーションレベルで保護される Open Lambda を AWS アカウントに作成します。この設定をセキュリティチームと確認してください。詳細については、「Security and auth model for Lambda function URLs」(Lambda 関数 URL におけるセキュリティと認証モデル) を参照してください。

Amazon EventBridge イベントバスは、 テンプレートによって作成されたAWS Lambda 関数 URL を使用して、 AWS CloudFormation サポートされている SaaS プロバイダーからイベントを受信できます。関数 URL を使用すると、イベントデータは Lambda 関数に送信されます。次に、関数はこのデータをイベントに変換し、 によって取り込まれ EventBridge 、処理のためにイベントバスに送信できます。イベントがイベントバスに入ると、ルールを使用してイベントをフィルタリングし、設定済みの入力変換を適用して、正しいターゲットにルーティングできます。

> **Note**
>
> Lambda 関数 URL を作成すると、月額のコストが増加します。詳細については、「AWS Lambda 料金表」を参照してください。

への接続を設定するには EventBridge、まず接続を設定する SaaS プロバイダーを選択します。次に、そのプロバイダーで作成した署名シークレットを指定し、 EventBridge イベントを送信するイベントバスを選択します。最後に、 AWS CloudFormation テンプレートを使用して、接続を完了するために必要なリソースを作成します。

現在、以下の SaaS プロバイダーは、Lambda 関数 URLs EventBridge を使用して で使用できます。

- GitHub
- Twilio

トピック

- GitHub への接続をセットアップする
- ステップ 1: AWS CloudFormation スタックを作成する

- ステップ 3: GitHub ウェブフックを作成する
- Twilio への接続をセットアップする
- ウェブフックまたは Auth トークンを更新する
- Lambda 関数を更新する
- 利用可能なイベントタイプ
- クォータ、エラーコード、配信の再試行

GitHub への接続をセットアップする

ステップ 1: AWS CloudFormation スタックを作成する

まず、Amazon EventBridge コンソールを使用して CloudFormation スタックを作成します。

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで、[Quick starts] (クイックスタート) を選択します。
3. [Inbound webhooks using Lambda fURLs] (Lambda fURL を使用したインバウンドウェブフック) で、[Get started] (使用を開始する) を選択します。
4. [GitHub] で、[Set up] (設定) を選択します。
5. [Step 1: Select an event bus] (ステップ 1: イベントバスを選択する) で、ドロップダウンリストからイベントバスを選択します。このイベントバスは、GitHub に指定した Lambda 関数 URL からデータを受け取ります。[New event bus] (新しいイベントバス) を選択して、イベントバスを作成することもできます。
6. ステップ 2: を使用して を設定するで CloudFormation、新しいGitHubウェブフック を選択します。
7. [I acknowledge that the Inbound Webhook I create will be publicly accessible] (作成したインバウンドウェブフックが一般公開されることを承認します) を選択し、[Confirm] (確認) を選択します。
8. スタックの名前を入力します。
9. パラメータの下に正しいイベントバスが表示されていることを確認し、GitHubWebhookSecret の安全なトークンを指定します。安全なトークンの作成の詳細については、GitHub ドキュメントの「シークレットトークンを設定する」を参照してください。
10. [Capabilities and transforms] (機能と変換) で、以下のそれぞれを選択します。
    - が IAM リソースを作成する AWS CloudFormation 可能性があることを確認します。

- がカスタム名で IAM リソースを作成する AWS CloudFormation 可能性があることを確認します。
- 次の機能が必要になる AWS CloudFormation 場合があることを了承します。**CAPABILITY_AUTO_EXPAND**

11. [スタックの作成] を選択します。

ステップ 3: GitHub ウェブフックを作成する

次に、GitHub でウェブフックを作成します。このステップを完了するには、ステップ 2 で作成した安全なトークンと Lambda 関数 URL の両方が必要です。詳細については、GitHub ドキュメントの「ウェブフックの作成」を参照してください。

Twilio への接続をセットアップする

ステップ 1: Twilio Auth トークンを見つける

Twilio と 間の接続を設定するには EventBridge、まずTwilioアカウントの認証トークンまたはシークレットTwilioを使用して への接続を設定します。詳細については、Twilio ドキュメントで「Auth トークンとその変更方法」を参照してください。

ステップ 2: AWS CloudFormation スタックを作成する

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで、[Quick starts] (クイックスタート) を選択します。
3. [Inbound webhooks using Lambda fURLs] (Lambda fURL を使用したインバウンドウェブフック) で、[Get started] (使用を開始する) を選択します。
4. [Twilio] で、[Set up] (設定) を選択します。
5. [Step 1: Select an event bus] (ステップ 1: イベントバスを選択する) で、ドロップダウンリストからイベントバスを選択します。このイベントバスは、Twilio に指定した Lambda 関数 URL からデータを受け取ります。[New event bus] (新しいイベントバス) を選択して、イベントバスを作成することもできます。
6. ステップ 2: を使用してセットアップするで CloudFormation、新しいTwilioウェブフック を選択します。
7. [I acknowledge that the Inbound Webhook I create will be publicly accessible] (作成したインバウンドウェブフックが一般公開されることを承認します) を選択し、[Confirm] (確認) を選択します。

8. スタックの名前を入力します。
9. パラメータの下に正しいイベントバスが表示されていることを確認し、ステップ 1 で作成した TwilioWebhookSecret を入力します。
10. [Capabilities and transforms] (機能と変換) で、以下のそれぞれを選択します。
    - が IAM リソースを作成する AWS CloudFormation 可能性があることを確認します。
    - がカスタム名で IAM リソースを作成する AWS CloudFormation 可能性があることを確認します。
    - CAPABILITY_AUTO_EXPAND の機能が必要になる AWS CloudFormation 場合があることを了承します。
11. [スタックの作成] を選択します。

ステップ 3: Twilio ウェブフックを作成する

Lambda 関数 URL を設定したら、それを Twilio に渡してイベントデータを送信できるようにする必要があります。詳細については、Twilio ドキュメントで「Twilio でパブリック URL を設定する」を参照してください。

ウェブフックまたは Auth トークンを更新する

GitHub シークレットを更新する

> **Note**
>
> GitHub では、2 つのシークレットを同時に持つことをサポートしていません。 AWS CloudFormation スタック内のGitHubシークレットとシークレットが同期していない間は、リソースのダウンタイムが発生する可能性があります。シークレットが同期していない間に送信されるGitHubメッセージは、署名が正しくないために失敗します。シー CloudFormation クレットGitHubと シークレットが同期するまで待ってから、もう一度試してください。

1. 新しい GitHub シークレットを作成する 詳細については、GitHub ドキュメントの「暗号化されたシークレット」を参照してください。
2. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソールを開きます。
3. ナビゲーションペインで [Stacks] (スタック) を選択します。
4. 更新するシークレットが含まれているウェブフックでスタックを選択します。

5. [更新] を選択します。
6. [Use current template] (現在のテンプレートの使用) が選択されていることを確認し、[Next] (次へ) を選択します。
7. でGitHubWebhookSecret、「既存の値を使用する」 をクリアし、ステップ 1 で作成した新しいGitHubシークレットを入力し、「次へ」を選択します。
8. [次へ] をクリックします。
9. [Update stack] (スタックの更新) を選択します。

シークレットが伝播されるまでに最大で 1 時間かかる場合があります。このダウンタイムを減らすには、Lambda 実行コンテキストを更新できます。

Twilio シークレットを更新する

> **Note**
>
> Twilio では、2 つのシークレットを同時に持つことをサポートしていません。 AWS CloudFormation スタック内のTwilioシークレットとシークレットが同期していない間は、リソースのダウンタイムが発生する可能性があります。シークレットが同期していない間に送信されたTwilioメッセージは、署名が正しくないために失敗します。シー CloudFormation クレットTwilioと シークレットが同期するまで待ってから、もう一度試してください。

1. 新しい Twilio シークレットを作成する 詳細については、Twilio ドキュメントで「Auth トークンとその変更方法」を参照してください。
2. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソールを開きます。
3. ナビゲーションペインで [Stacks] (スタック) を選択します。
4. 更新するシークレットが含まれているウェブフックでスタックを選択します。
5. [更新] を選択します。
6. [Use current template] (現在のテンプレートの使用) が選択されていることを確認し、[Next] (次へ) を選択します。
7. でTwilioWebhookSecret、「既存の値を使用する」 をクリアし、ステップ 1 で作成した新しいTwilioシークレットを入力し、「次へ」を選択します。
8. [次へ] をクリックします。
9. [Update stack] (スタックの更新) を選択します。

シークレットが伝播されるまでに最大で 1 時間かかる場合があります。このダウンタイムを減らすには、Lambda 実行コンテキストを更新します。

Lambda 関数を更新する

CloudFormation スタックによって作成される Lambda 関数は、基本的なウェブフックを作成します。ログ記録のカスタマイズなど、特定のユースケースに合わせて Lambda 関数をカスタマイズする場合は、 CloudFormation コンソールを使用して関数にアクセスし、Lambda コンソールを使用して Lambda 関数コードを更新します。

Lambda 関数にアクセスする

1. https://console.aws.amazon.com/cloudformation で AWS CloudFormation コンソールを開きます。
2. ナビゲーションペインで [Stacks] (スタック) を選択します。
3. 更新する Lambda 関数が含まれているウェブフックでスタックを選択します。
4. [Resources] (リソース) タブを選択します。
5. Lambda コンソールで Lambda 関数を開くには、[Physical ID] (物理 ID) で、Lambda 関数の ID を選択します。

Lambda 関数にアクセスしたら、Lambda コンソールを使用して関数コードを更新します。

Lambda 関数コードを更新する

1. [Actions] (アクション) で、[Export function] (関数のエクスポート) を選択します。
2. [Download deployment package] (デプロイパッケージのダウンロード) を選択し、ファイルをコンピュータに保存します。
3. デプロイパッケージの .zip ファイルを解凍して、`app.py` ファイルを更新し、更新したデプロイパッケージを圧縮します。元の .zip ファイル内のすべてのファイルが含まれていることを確認してください。
4. Lambda コンソールで、[Code] (コード) タブを選択します。
5. [Code source (コードソース)] で、[Upload from (アップロード元)] を選択します。
6. [.zip file (.zip ファイル)]、[Upload (アップロード)] の順に選択します。
   - ファイルの選択画面で、更新するファイルを選択し、[Open] (開く)、[Save] (保存) の順に選択します。

7. [Actions] (アクション) メニューで、[Publish new version] (新しいバージョンを発行) を選択します。

利用可能なイベントタイプ

現在、次のイベントタイプは CloudFormation イベントバスでサポートされています。

- GitHub – すべてのイベントタイプがサポートされています。
- Twilio - イベント後のウェブフックをサポートしています。

クォータ、エラーコード、配信の再試行

クォータ

ウェブフックへの受信リクエストの数は、基盤となる AWS サービスによって制限されます。次の表に関連するクォータを示します。

サービス	クォータ
AWS Lambda	デフォルト: 10 件の同時実行 クォータの引き上げのリクエストを含む、クォータの詳細については、「Lambda のクォータ」を参照してください。
AWS Secrets Manager	デフォルト: 1 秒あたり 5,000 個のリクエスト クォータの引き上げのリクエストを含む、クォータの詳細については、「AWS Secrets Manager のクォータ」を参照してください。 **Note** 1 秒あたりのリクエスト数は、AWS Secrets Manager Python キャッシュクライアントを使用して最小化されます。

サービス	クォータ
Amazon EventBridge	PutEvents アクションの最大エントリサイズは 256KB です。 EventBridge は、リージョンベースのレートクォータを適用します。詳細については、「???」を参照してください。

エラーコード

エラーが発生すると、各 AWS サービスは特定のエラーコードを返します。次の表に関連するエラーコードを示します。

サービス	エラーコード	説明
AWS Lambda	429TooManyRequests Exption「」	同時実行のクォータの制限を超えています。
AWS Secrets Manager	500 「Internal Server Error」	1 秒あたりのリクエスト数がクォータを超えています。
Amazon EventBridge	500 「Internal Server Error」	リージョンのレートクォータを超えています。

イベント再配信

エラーが発生した場合は、該当するイベントの配信を再試行できます。SaaS プロバイダーごとに再試行手順が異なります。

GitHub

GitHub ウェブフック API を使用すると、ウェブフック呼び出しの配信ステータスをチェックし、必要に応じてイベントを再配信できます。詳細については、次の GitHubドキュメントを参照してください。

- 組織 - 組織ウェブフックの配信を再配信する
- リポジトリ - リポジトリウェブフックの配信を再配信する

- アプリケーション - アプリウェブフックの配信を再配信

Twilio

Twilio ユーザーは、接続の上書きを使用してイベント再試行オプションをカスタマイズできます。詳細については、Twilio ドキュメントの「Webhooks (HTTP callbacks): Connection Overrides」 (ウェブフック (HTTP コールバック): 接続の上書き) を参照してください。

Salesforce からのイベントの受信

Amazon を使用して EventBridge 、Salesforce次の方法で からイベントを受信できます。

- Salesforce's イベントバスリレー機能を使用して、 EventBridge パートナーイベントバスでイベントを直接受信します。
- をデータソースSalesforceとして使用するフローを Amazon AppFlow で設定します。 AppFlow 次に、Amazon はパートナーSalesforceイベントバス EventBridge を使用して にイベントを送信します。 ???

API 送信先を使用して Salesforce にイベント情報を送信できます。イベントは、Salesforce に送信されると、フローまたは Apex トリガーで処理できます。Salesforce API 送信先のセットアップの詳細については、「???」を参照してください。

トピック

- イベントバスリレーを使用して Salesforce からイベントを受信する
- Amazon Salesforceを使用して からイベントを受信する AppFlow

イベントバスリレーを使用して Salesforce からイベントを受信する

ステップ 1: Salesforceイベントバスリレーと EventBridge パートナーイベントソースを設定する

でイベントリレー設定を作成するとSalesforce、 は保留状態の EventBridge にパートナーイベントソースSalesforceを作成します。

Salesforce イベントバスリレーを設定するには

1. REST API ツールをセットアップします
2. (オプション) プラットフォームイベントを定義します
3. カスタムプラットフォームイベントのチャンネルを作成します
4. カスタムプラットフォームイベントを関連付けるチャンネルメンバーを作成します
5. 名前付きの認証情報を作成します
6. イベントリレー設定を作成します

ステップ 2: EventBridge コンソールでSalesforceパートナーイベントソースをアクティブ化し、イベントリレーを開始する

1. EventBridge コンソールで パートナーイベントソースページを開きます。
2. ステップ 1 で作成した Salesforce パートナーイベントソースを選択します。
3. [Associate with event bus] (イベントバスと関連付ける) を選択します。
4. パートナーイベントバスの名前を検証します。
5. [Associate] (関連付ける) を選択します。
6. イベントリレーを開始します

イベントバスリレーを設定して開始し、パートナーイベントソースを設定したら、EventBridge イベントに反応してデータをフィルタリングしてターゲット に送信するためのルールを作成できます。???

Amazon Salesforceを使用して からイベントを受信する AppFlow

Amazon は、 からのイベントを EventBridge イベントエンベロープSalesforceに AppFlow カプセル化します。次の例は、 EventBridge パートナーSalesforceイベントバスが受信したイベントを示しています。

```
{
    "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"
    }
}
```

ステップ 1: パートナーイベントソース AppFlow Salesforceとして使用するように Amazon を設定する

にイベントを送信するには EventBridge、まずパートナーイベントソース AppFlow Salesforceとして使用するように Amazon を設定する必要があります。

1. Amazon AppFlow コンソール で、フローの作成 を選択します。
2. [Flow details] (フローの詳細) セクションで、[Flow name] (フロー名) にフローの名前を入力します。
3. (オプション) フローの説明を入力して、[Next] (次へ) を選択します。
4. [Source details] (ソースの詳細) で [Source name] ドロップダウンから Salesforce を選択し、[Connect] (接続) を選択して新しい接続を作成します。
5. [Connect to Salesforce] (への接続) ダイアログボックスで、Salesforce 環境として [Production] (本番稼働用) または [Sandbox] (サンドボックス) を選択します。
6. [Connection name] (接続名) フィールドに接続の一意の名前を入力し、[Continue] (続行) を選択します。
7. [Salesforce] ダイアログボックスで、以下の操作を実行します。
   a. Salesforce サインイン認証情報を入力して、Salesforce にログインします。
   b. Amazon が処理 AppFlow するデータのタイプのSalesforceイベントを選択します。
8. Salesforce イベントの選択ドロップダウンで、 に送信するイベントのタイプを選択します EventBridge。
9. 送信先については、Amazon EventBridgeを選択します。

10. [Create new partner event source] (パートナーイベントソースの作成) を選択します。
11. (オプション) パートナーイベントソースの一意のサフィックスを指定します。
12. [Generate partner event source] (パートナーイベントソースの生成) を選択します。
13. 256 KB を超えるイベントペイロードファイルを保存する Amazon S3 バケットを選択します。
14. [Flow trigger] (フロートリガー) セクションで、[Run flow on event] (イベントでフローを実行) が選択されていることを確認します。この設定により、新しい Salesforce イベントが発生するとフローが実行されるようになります。
15. [次へ] をクリックします。
16. フィールドマッピングとして、[Map all fields directly] (すべてのフィールドを直接マッピング) を選択します。あるいは、[Source field name] (ソースフィールド名) リストから対象となるフィールドを選択することもできます。

    フィールドマッピングの詳細については、「データフィールドのマッピング」を参照してください。
17. [次へ] をクリックします。
18. (オプション) Amazon のデータフィールドのフィルターを設定します AppFlow。
19. [次へ] をクリックします。
20. 設定を確認し、[Create] (作成) を選択します。

フローが設定されると、Amazon AppFlow は新しいパートナーイベントソースを作成し、アカウント内のパートナーイベントバスに関連付ける必要があります。

ステップ 2: Salesforceイベントを受信する EventBridge ように を設定する

このセクションの手順に従う前に、送信先 EventBridge として のSalesforceイベントからトリガーされる Amazon AppFlow フローが設定されていることを確認します。

Salesforce イベントを受信する EventBridge ように を設定するには

1. EventBridge コンソールで パートナーイベントソースページを開きます。
2. ステップ 1 で作成した Salesforce パートナーイベントソースを選択します。
3. [Associate with event bus] (イベントバスと関連付ける) を選択します。
4. パートナーイベントバスの名前を検証します。
5. [Associate] (関連付ける) を選択します。

6. Amazon AppFlow コンソールで、作成したフローを開き、フローのアクティブ化 を選択します。
7. EventBridge コンソールでルールページを開きます。
8. [Create rule] を選択します。
9. ルールの一意の名前を入力します。
10. [Define pattern] (パターンの定義) セクションで [Event pattern] (イベントパターン) を選択します。
11. [Event matching pattern] (イベント照合パターン) で、[Pre-defined pattern by service] (サービスごとの事前定義パターン) を選択します。
12. [Service provider section] (サービスプロバイダー) で [All Events] (すべてのイベント) を選択します。
13. [Select event bus] (イベントバスの選択) で [Custom or partner event bus] (カスタムまたはパートナーイベントバス) を選択します。
14. Amazon AppFlow パートナーイベントソースに関連付けたイベントバスを選択します。
15. ターゲットの選択 で、ルールの実行時に実行する AWS サービスを選択します。1 つのルールに、最大 5 つのターゲットを設定できます。
16. [作成] を選択します。

ターゲットサービスは、アカウントに設定されたすべての Salesforce イベントを受信します。イベントをフィルタリングする、または一部のイベントを別のターゲットに送信するには、イベントパターンによるコンテンツベースのフィルタリングを使用できます。

> **Note**
>
> 256KB、Amazon AppFlow は完全なイベントを に送信しません EventBridge。代わりに、Amazon はイベントをアカウントの S3 バケット AppFlow に配置し、Amazon S3 バケットへのポインタ EventBridge ーを使用して にイベントを送信します。このポインタを使用して、バケットから完全なイベントを取得することができます。

イベント配信のデバッグ

イベント配信の問題は特定が難しい場合があり、イベント配信の失敗をデバッグして復旧するいくつかの方法 EventBridge を提供します。

がイベントの配信を EventBridge 再試行する方法

ルールで指定されたターゲットにイベントが正常に配信されないことがあります。これは、次のような場合に発生します。

- ターゲットリソースが使用できない場合
- ネットワーク条件による

再試行可能なエラーが原因でイベントがターゲットに正常に配信されなかった場合、 はイベントの送信を EventBridge 再試行します。ターゲットの再試行ポリシー設定で、再試行する時間の長さと再試行回数を設定します。デフォルトでは、 はエクスポネンシャルバックオフとジッター、またはランダム化された遅延を使用して、イベントの送信を 24 時間、最大 185 回 EventBridge 再試行します。

すべての再試行回数が経過してもイベントが配信されない場合、イベントは削除され、 EventBridge 処理は続行されません。

デッドレターキューを使用した未配信イベントの処理

ターゲットへの配信に失敗した後でイベントが失われるのを避けるために、デッドレターキュー (DLQ) を設定し、失敗したすべてのイベントをそこに送って後で処理することができます。

EventBridge DLQsは、ターゲットに正常に配信できなかったイベントを保存するために EventBridge が使用する標準の Amazon SQS キューです。DLQ を使用するかどうかは、ルールを作成してターゲットを追加するときに選択できます。DLQ を設定すると、正常に配信されなかったイベントを保持できます。そのため、失敗したイベント配信の原因となった問題を解決し、後でイベントを処理できるようになります。

ルールのターゲットに DLQ を設定すると、 は失敗した呼び出しのイベントを選択した Amazon SQS キュー EventBridge に送信します。

イベントエラーには、さまざまな処理方法があります。一部のイベントは、再試行せずに削除されるか、DLQ に送信されます。例えば、ターゲットへのアクセス許可がない、またはターゲットリソースが存在しなくなったために エラーが発生する場合、根本的な問題を解決するアクションが実行されるまで、再試行はすべて失敗します。再試行するのではなく、DLQ がある場合は、 EventBridge はこれらのイベントを DLQ に直接送信します。

イベント配信が失敗すると、 はターゲットが`invocation`失敗したことを示すイベントを Amazon CloudWatch メトリクスに EventBridge 発行します。DLQ を使用す

る場合、 InvocationsSentToDLQと を含む追加のメトリクスが に送信されます CloudWatchInvocationsFailedToBeSentToDLQ。

を使用して保管中のイベントを暗号化する場合は、イベントバス AWS KMS カスタマーマネージドキー の DLQs を指定することもできます。詳細については、「???」を参照してください。

DLQ の各メッセージには、次のカスタム属性が含まれます。

- RULE_ARN
- TARGET_ARN
- ERROR_CODE

  以下は、DLQ が返す可能性のあるエラーコードのサンプルです。

  - 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

  次の条件が返されることがあります。

  - MaximumRetryAttempts
  - MaximumEventAgeInSeconds
- RETRY_ATTEMPTS

次のビデオでは、DLQ の設定について説明します: デッドレターキュー (DLQ) の使用

トピック

- デッドレターキューを使用する際の考慮事項
- デッドレターキューへのアクセス許可の付与
- デッドレターキューからイベントを再送信する方法

デッドレターキューを使用する際の考慮事項

の DLQ を設定するときは、次の点を考慮してください EventBridge。

- サポートされるのは標準キューのみです。の DLQ に FIFO キューを使用することはできません EventBridge。
- EventBridge には、エラーコード、エラーメッセージ、期限切れの再試行条件、ルール ARN、再試行回数、ターゲット ARN など、イベントメタデータとメッセージ属性がメッセージに含まれます。これらの値を使用して、イベントと障害の原因を特定します。
- 同じアカウントの DLQ に対するアクセス許可。
  - コンソールを使用してルールにターゲットを追加し、同じアカウントで Amazon SQS キューを選択すると、キュー EventBridge へのアクセスを許可するリソースベースのポリシーがキューにアタッチされます。
  - EventBridge API の PutTargetsオペレーションを使用してルールのターゲットを追加または更新し、同じアカウントで Amazon SQS キューを選択する場合は、選択したキューにアクセス許

可を手動で付与する必要があります。詳細については、「デッドレターキューへのアクセス許可の付与」を参照してください。

- 別の AWS アカウントから Amazon SQS キューを使用するためのアクセス許可。
  - コンソールからルールを作成する場合、他のアカウントのキューは表示されないため選択できません。他のアカウントのキューへのアクセス許可を付与するには、そのキューの ARN を指定し、リソースベースのポリシーを手動でアタッチする必要があります。詳細については、「デッドレターキューへのアクセス許可の付与」を参照してください。
  - API を使用してルールを作成する場合、デッドレターキューとして使用される別のアカウントの SQS キューに、リソースベースのポリシーを手動でアタッチする必要があります。詳細については、「デッドレターキューへのアクセス許可の付与」を参照してください。
- 使用する Amazon SQS キューは、ルールを作成するリージョンと同じリージョンに存在する必要があります。

デッドレターキューへのアクセス許可の付与

キューにイベントを正常に配信するには、そのアクセス許可 EventBridge が必要です。 EventBridge コンソールを使用して DLQ を指定すると、アクセス許可が自動的に追加されます。これには、以下が含まれます。

- ルールのターゲットに DLQ を設定する場合。
- を使用して保管中のイベントを暗号化するように指定したイベントバスの DLQ EventBridge AWS KMS カスタマー管理キー を設定する場合。

  詳細については、「???」を参照してください。

API を使用して DLQ を指定する場合、または別の AWS アカウントにあるキューを使用する場合は、必要なアクセス許可を付与するリソースベースのポリシーを手動で作成し、キューにアタッチする必要があります。

ターゲットデッドレターキューのアクセス許可の例

次のリソースベースのポリシーは、 が Amazon SQS キュー EventBridge にイベントメッセージを送信するために必要なアクセス許可を付与する方法を示しています。ポリシーの例では、`SendMessage`オペレーションを使用してMyEvent「DLQ」という名前のキューにメッセージを送信するアクセス許可を EventBridge サービスに付与します。キューは、 AWS アカウント 123456789012 の us-west-2 リージョンにある必要があります。`Condition` ステートメントは、

AWS アカウント 123456789012 の us-west-2 リージョンで作成されたMyTestRule「」という名前のルールからのリクエストのみを許可します。

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

イベントバスのデッドレターキューのアクセス許可の例

次のリソースベースのポリシーは、イベントバスの DLQ を指定するときに必要なアクセス許可を付与する方法を示しています。この場合、 は DLQ にイベントを送信するイベントバスの ARN aws:SourceArnを指定します。この例では、キューはイベントバスと同じリージョンにある必要があります。

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

ポリシーをキューにアタッチするには、Amazon SQS コンソールを使用してキューを開き、[Access policy] (アクセスポリシー) を選択してポリシーを編集します。また、 AWS CLIを使用することもできます 詳細については、「Amazon SQS のアクセス許可」を参照してください。

デッドレターキューからイベントを再送信する方法

メッセージを DLQ から移動するには、次の 2 つの方法があります。

- Amazon SQS コンシューマーロジックの作成を避ける - DLQ をイベントソースとして Lambda 関数に設定し、DLQ を吸い出します。
- Amazon SQS コンシューマーロジックの書き込み – Amazon SQS API、 AWS SDK、または AWS CLI を使用して、DLQ 内のメッセージをポーリング、処理、削除するためのカスタムコンシューマーロジックを書き込みます。

Amazon EventBridge イベントパターン

イベントパターンは、一致するイベントと同じ構造をしています。ルールでは、イベントパターンを使用してイベントを選択し、ターゲットに送信します。イベントパターンは、イベントに一致するか、一致しないかのいずれかになります。

> **Important**
>
> では EventBridge、 higher-than-expected 料金やスロットリングにつながる可能性のあるルールを作成できます。例えば、ルールが無限に再帰的に実行される無限ループに陥るようなルールを誤って作成してしまうことがあります。Amazon S3 バケットで ACL が変更されたことを検出し、ソフトウェアをトリガーして目的の状態に変更するルールを作成したとします。このルールが慎重に記述されていない場合は、その後 ACL を変更するとルールが再び開始され、無限ループが作成されます。
> このような予期しない結果を最小限に抑えるための正確なルールやイベントパターンを記述する方法に関するガイダンスについては、??? および ??? を参照してください。

次のビデオでは、イベントパターンの基本について説明します。イベントをフィルタリングするには

トピック

- イベントパターンの作成
- イベントとイベントパターンの例
- Amazon EventBridge イベントパターンでの null 値と空の文字列の一致
- Amazon EventBridge イベントパターンの配列
- Amazon EventBridge イベントパターンでのコンテンツフィルタリング
- EventBridge サンドボックスを使用したイベントパターンのテスト
- Amazon EventBridge イベントパターンを定義する際のベストプラクティス

次のイベントは、Amazon EC2 からのシンプルな AWS イベントを示しています。

```
{
  "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"
  }
}
```

次のイベントパターンは、Amazon EC2 の `instance-termination` イベントすべてを処理します。

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

イベントパターンの作成

イベントパターンを作成するには、イベントパターンにマッチングするイベントのフィールドを指定します。照合に使用するフィールドのみを指定します。前のイベントパターンの例では、最上位フィールド`"source"`と の 3 つのフィールドの値のみを提供し`"detail-type"`、`"detail"`オブジェクトフィールド内の `"state"` フィールドの値のみを提供します。 は、ルールの適用時にイベント内の他のすべてのフィールド EventBridge を無視します。

イベントパターンがイベントに一致するには、イベントパターンに指定されているすべてのフィールド名がイベントに含まれている必要があります。フィールド名は、同じネスト構造になったイベントにも含まれていなければなりません。

イベントと一致するパターンを記述するときは、`TestEventPattern` API または `test-event-pattern` CLI コマンドを使用して、パターンが正しいイベントと一致することをテストできます。詳細については、「」を参照してくださいTestEventPattern。

イベント値の照合

イベントパターンでマッチングする値は JSON 配列形式で、角かっこ ([と]) で囲むと複数の値を指定できます。例えば、Amazon EC2 または からのイベントを照合するには AWS Fargate、次のパターンを使用できます。このパターンは、 `"source"` フィールドの値が `"aws.ec2"`または のイベントを照合します`"aws.fargate"`。

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

イベントパターンを作成する際の考慮事項

イベントパターンを作成する際に考慮すべき点は次のとおりです。

- EventBridge は、イベントパターンに含まれていないイベントのフィールドを無視します。実際には、`"*": "*"` とワイルドカードを指定すると、イベントパターンに含まれないフィールドにも一致します。
- イベントパターンが一致する値は、JSON のルールに従います。二重引用符 (") で囲んだ文字列、数値、およびキーワード `true`、`false`、および `null` を含めることができます。
- 文字列の場合、大文字と小文字の折りたたみやその他の文字列の正規化なしで完全 character-by-character 一致 EventBridge を使用します。
- 数値の場合、文字列表現 EventBridge を使用します。たとえば、300、300.0、3.0e2 は等しいとはみなされません。
- 同じ JSON フィールドに複数のパターンが指定されている場合、 は最後のパターン EventBridge のみを使用します。
- が使用するためにイベントパターンを EventBridge コンパイルするときは、結合文字としてドット (.) を使用することに注意してください。

  つまり EventBridge 、 は次のイベントパターンを同じものとして扱います。

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

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

  また、どちらのイベントパターンも次の 2 つのイベントと一致することになります。

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

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

> **Note**
>
> これは現在の EventBridge 動作を説明するものであり、変更しないことに頼るべきではありません。

- 重複するフィールドを含むイベントパターンは無効です。パターンに重複するフィールドが含まれている場合、 は最後のフィールド値 EventBridge のみを考慮します。

  例えば、以下のイベントパターンは同じイベントと一致します。

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

  また、 は、次の 2 つのイベントを同じものとして EventBridge 扱います。

```
## 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**
>
> これは現在の EventBridge 動作を説明するものであり、変更しないことに頼るべきではありません。

イベントパターンで使用する比較オペレーション

で使用可能なすべての比較演算子の概要の下 EventBridge。

比較オペレーションは、`$or` と `anything-but` を除いてリーフノードでのみ機能します。

比較	例	ルール構文
And	Location が「New York」、および Day が「Monday」	`"Location": [ "New York" ], "Day": ["Monday"]`
Anything-but	状態は、「初期化中」以外の値です。	`"state": [ { "anything-but": "initializing" } ]`

比較	例	ルール構文
Anything-but (最初は)	リージョンが米国にありません。	`"Region": [ { "anything-but": {"prefix": "us-" } } ]`
Anything-but (末尾は)	FileName は .png 拡張子で終わりません。	`"FileName": [ { "anything-but": { "suffix": ".png" } } ]`
Anything-but (大文字と小文字は無視)	状態は、「初期化中」以外の値、または「初期化中」などのその他の大文字と小文字のバリエーションです。	`"state": : [{ "anything-but": { "equals-ignore-case": "initializing" }}]}`
ワイルドカードを使用するもの	FileName は、 を含むファイルパスではありません/lib/。	`"FilePath" : [{ "anything-but": { "wildcard": "*/lib/*" }}]`
で始まる	リージョンは米国にあります。	`"Region": [ {"prefix": "us-" } ]`
で始まる (大文字と小文字は無視)	サービス名は、大文字と小文字に関係なく「eventb」で始まります。	`{"service" : [{ "prefix": { "equals-ignore-case": "eventb" }}]}`
空	LastName は空です。	`"LastName": [""]`
等しい	Name が「Alice」	`"Name": [ "Alice" ]`
等しい (大文字と小文字を区別しない)	Name が「Alice」	`"Name": [ { "equals-ignore-case": "alice" } ]`
で終わる	FileName .png 拡張子で終わる	`"FileName": [ { "suffix": ".png" } ]`

比較	例	ルール構文
で終わる (大文字と小文字は無視)	サービス名は、「tbridge」という文字、または「TDGE」などのその他の大文字と小文字のバリエーションで終わります。	`{"service" : [{ "suffix": { "equals-ignore-case": "tBridge" }}]}`
存在する	ProductName が存在する	`"ProductName": [ { "exists": true } ]`
存在しない	ProductName は存在しません	`"ProductName": [ { "exists": false } ]`
以外	Weather が「Raining」以外	`"Weather": [ { "anything-but": [ "Raining" ] } ]`
Null	UserId が Null	`"UserID": [ null ]`
数値 (等しい)	Price が 100	`"Price": [ { "numeric": [ "=", 100 ] } ]`
数値 (範囲)	Price が 10 より大きく 20 以下	`"Price": [ { "numeric": [ ">", 10, "<=", 20 ] } ]`
または	PaymentType が「クレジット」または「デビット」の場合	`"PaymentType": [ "Credit", "Debit"]`
Or (複数フィールド)	Location が「New York」、または Day が「Monday」	`"$or": [ { "Location": [ "New York" ] }, { "Day": [ "Monday" ] } ]`
ワイルドカード	「dir」フォルダ内の .png 拡張子の付いたすべてのファイル	`"FileName": [ { "wildcard": "dir/*.png" } ]`

イベントとイベントパターンの例

イベントのマッチングには、JSON のデータ型と値をすべて使用できます。以下の例は、イベントとそれに一致するイベントパターンを示しています。

フィールドのマッチング

フィールドの値でマッチングすることができます。次の Amazon EC2 Auto Scaling イベントについて考えてみましょう。

```
{
  "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
  }
}
```

上のイベントでは、マッチングに "responseElements" フィールドを使用できます。

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

値の一致

以下は、Amazon Macie イベントの一部です。これについて考えてみましょう。

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

次のイベントパターンは、重要度スコアが 1 で、カウントが 2 のイベントと一致します。

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

Amazon EventBridge イベントパターンでの null 値と空の文字列の一致

> **Important**
>
> では EventBridge、 higher-than-expected 料金やスロットリングにつながる可能性のあるルールを作成できます。例えば、ルールが無限に再帰的に実行される無限ループに陥るようなルールを誤って作成してしまうことがあります。Amazon S3 バケットで ACL が変更されたことを検出し、ソフトウェアをトリガーして目的の状態に変更するルールを作成したとします。このルールが慎重に記述されていない場合は、その後 ACL を変更するとルールが再び開始され、無限ループが作成されます。
> このような予期しない結果を最小限に抑えるための正確なルールやイベントパターンを記述する方法に関するガイダンスについては、??? および ??? を参照してください。

イベントのうち、Null 値や空の文字列を持つイベントフィールドと一致するイベントパターンを作成できます。次のイベントの例を考えます。

予想よりも高い料金やスロットリングを避けるためのベストプラクティスをご覧ください

```
{
  "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 の値が空の文字列であるイベントとマッチングするには、上のイベント例と一致する次のイベントパターンを使用します。

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

`responseElements` の値が Null であるイベントとマッチングするには、上のイベント例と一致する次のイベントパターンを使用します。

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

> **Note**
>
> Null 値と空の文字列は、パターンマッチングで交換可能ではありません。空の文字列に一致するイベントパターンは、null の値に一致しません。

Amazon EventBridge イベントパターンの配列

イベントパターンの各フィールドの値は、1 つ以上の値を含む配列です。イベントパターンは、配列の値のいずれかがイベントの値と一致すれば、イベントに一致します。イベントの値が配列の場合、イベントパターン配列とイベント配列の交差部分が空でなければイベントパターンが一致したとみなされます。

> **Important**
>
> では EventBridge、 higher-than-expected 料金やスロットリングにつながる可能性のあるルールを作成できます。例えば、ルールが無限に再帰的に実行される無限ループに陥るようなルールを誤って作成してしまうことがあります。Amazon S3 バケットで ACL が変更されたことを検出し、ソフトウェアをトリガーして目的の状態に変更するルールを作成したとします。このルールが慎重に記述されていない場合は、その後 ACL を変更するとルールが再び開始され、無限ループが作成されます。
> このような予期しない結果を最小限に抑えるための正確なルールやイベントパターンを記述する方法に関するガイダンスについては、??? および ??? を参照してください。

例えば、次のフィールドを含むイベントパターンを考えてみましょう

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

上のイベントパターンの例は、次のフィールドが含まれているイベントに一致します。イベントパターン配列の最初の項目が、イベント配列の 2 番目の項目と一致するからです。

```
"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 イベントパターンでのコンテンツフィルタリング

Amazon は、イベントパターン を使用した宣言型コンテンツフィルタリング EventBridge をサポートしています。コンテンツのフィルタリングを使用すると、非常に限定的な条件下でのみイベントに一致する複雑なイベントパターンを作成できます。例えば、次の場合にイベントと一致するイベントパターンを作成できます。

- イベントのフィールドが特定の数値範囲内にある場合。
- イベントは特定の IP アドレスから発生する場合。
- イベント JSON に特定のフィールドが存在しない場合。

> **Important**
>
> では EventBridge、 higher-than-expected 料金やスロットリングにつながる可能性のあるルールを作成できます。例えば、ルールが無限に再帰的に実行される無限ループに陥るようなルールを誤って作成してしまうことがあります。Amazon S3 バケットで ACL が変更されたことを検出し、ソフトウェアをトリガーして目的の状態に変更するルールを作成したとします。このルールが慎重に記述されていない場合は、その後 ACL を変更するとルールが再び開始され、無限ループが作成されます。
> このような予期しない結果を最小限に抑えるための正確なルールやイベントパターンを記述する方法に関するガイダンスについては、??? および ??? を参照してください。

フィルターのタイプ

- プレフィックスマッチング
- サフィックスマッチング
- 「以外」のマッチング
- 数値マッチング
- IP アドレスマッチング
- 存在マッチング
- E quals-ignore-caseマッチング
- ワイルドカードを使用したマッチング

- 複数のマッチングを含む複雑な例
- 複数の $or マッチングを含む複雑な例

プレフィックスマッチング

イベントソース内の値のプレフィックスに応じてイベントをマッチングすることができます。文字列値にはプレフィックスマッチングを使用できます。

例えば、次のイベントパターンは、"time": "2017-10-02T18:43:48Z" のように "time" フィールドが "2017-10-02" で始まるすべてのイベントに一致します。

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

ケースを無視する際のプレフィックスマッチング

また、 と を併用して、値が始まる文字の大文字と小文字に関係なく、プレフィックス値を一致equals-ignore-caseさせることもできます。 prefix.

例えば、次のイベントパターンは、 serviceフィールドが文字列 で始まるイベントだけでなくEventB、EVENTB、eventb、またはそれらの文字の他の大文字と小文字でも一致します。

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

サフィックスマッチング

イベントソース内の値のサフィックスに応じてイベントをマッチングすることができます。文字列値にはサフィックスマッチングを使用できます。

例えば、次のイベントパターンは、"FileName" フィールドが .png ファイル拡張子で終わるすべてのイベントに一致します。

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

ケースを無視する際のサフィックスマッチング

また、 と組み合わせて を使用して、値が終わる文字の大文字と小文字に関係なく、サフィックス値を一致equals-ignore-caseさせることもできます。 suffix.

例えば、次のイベントパターンは、 FileNameフィールドが文字列 で終わるイベントだけでなく.png、それらの文字の.PNG他の大文字と小文字でも一致します。

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

「以外」のマッチング

Anything-but マッチングは、ルールで指定されているもの以外のものと一致します。

文字列のみを含むリスト、または数字のみを含むリストを含む、文字列および数値で「以外」のマッチングを使用できます。

次のイベントパターンは、文字列と数字を使った「以外」のマッチングを示しています。

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

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

次のイベントパターンは、文字列のリストを使った「以外」のマッチングを示しています。

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

次のイベントパターンは、数値のリストを使った「以外」のマッチングを示しています。

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

大文字と小文字の区別なしの一致

equals-ignore-case を と組み合わせて使用してanything-but、文字の大文字と小文字に関係なく文字列値を一致させることもできます。

次のイベントパターンは、文字列「初期化中」、「INITIALIZING」、「初期化中」、またはそれらの文字のその他の大文字を含まないstateフィールドに一致します。

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

を equals-ignore-caseと組み合わせて使用anything-butして、値のリストと照合することもできます。

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

プレフィックスでのモノと一致

を prefixと組み合わせて使用するとanything-but、指定した値で始まらない文字列値を一致させることができます。これには、単一の値、または値のリストが含まれます。

次のイベントパターンは、 "init""state"フィールドに プレフィックスがないイベントに一致する、何でも一致しないマッチングを示しています。

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

次のイベントパターンは、プレフィックス値のリストで使用されるモノ以外の一致を示しています。このイベントパターンは、 "state" フィールドに "init"または のプレフィックスがないイベントと一致し"stop"ます。

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

サフィックスでのモノ以外一致

を suffixと組み合わせて使用するとanything-but、指定した値で終了しない文字列値を一致させることができます。これには、単一の値、または値のリストが含まれます。

次のイベントパターンは、 で終わることのないFileNameフィールドの値と一致します.txt。

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

次のイベントパターンは、サフィックス値のリストで使用されるモノ以外の一致を示しています。このイベントパターンは、 .txtまたは で終わることのないFileNameフィールドの値と一致します.rtf。

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

ワイルドカードを使用したモノとマッチング

anything-but マッチングに指定した値内でワイルドカード文字 (*) を使用できます。これには、単一の値、または値のリストが含まれます。

次のイベントパターンは、 を含まないFileNameフィールドの値と一致します/lib/。

```
{
```

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

次のイベントパターンは、ワイルドカードを含む値のリストで使用される、何でも一致しないマッチングを示しています。このイベントパターンは、 /lib/または を含まないFileNameフィールドの値と一致します/bin/。

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

詳細については、「???」を参照してください。

数値マッチング

数値マッチングは、JSON 数値である値で動作します。精度が 15 桁 (小数点の右に 6 桁) の -5.0e9 から +5.0e9 までの値に制限されています。

以下は、すべてのフィールドが真であるイベントにのみ一致するイベントパターンの数値マッチングを示しています。

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

IP アドレスマッチング

IP アドレスマッチングは、IPv4 アドレスと IPv6 アドレスに使用できます。次のイベントパターンは、10.0.0 で始まり、0 ~ 255 の数値で終わる IP アドレスに一致する IP アドレスを示しています。

```
{
```

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

存在マッチング

存在マッチングは、イベントの JSON 内のフィールドの有無に対して機能します。

存在マッチングは、リーフノードでのみ機能します。中間ノードでは機能しません。

次のイベントパターンは、detail.state フィールドを持つイベントと一致します。

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

前のイベントパターンは、次のイベントと一致します。

```
{
  "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"
  }
}
```

前のイベントパターンは、次のイベントとは一致しません。detail.state フィールドがないからです。

```
{
  "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マッチング

E quals-ignore-caseマッチングは、大文字と小文字に関係なく文字列値に対して機能します。

次のイベントパターンは、大文字小文字に関係なく、指定された文字列と一致する `detail-type` フィールドを持つイベントと一致します。

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

前のイベントパターンは、次のイベントと一致します。

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

ワイルドカードを使用したマッチング

ワイルドカード文字 (*) を使用してイベントパターン内の文字列値と一致させることができます。

> **Note**
>
> 現在、ワイルドカード文字はイベントバスルールでのみサポートされています。

イベントパターンでワイルドカードを使用する場合の考慮事項:

- 1 つの文字列値にはワイルドカード文字をいくつでも指定できますが、ワイルドカード文字を連続して使用することはサポートされていません。
- EventBridge では、バックスラッシュ文字 (\) を使用して、ワイルドカードフィルターでリテラル * および \ 文字を指定できます。
  - 文字列 * はリテラル * 文字を表します。
  - 文字列 \\ はリテラル \ 文字を表します。

  バックスラッシュを使用して他の文字を エスケープすることはサポートされていません。

ワイルドカードとイベントパターンの複雑さ

ワイルドカードを使用するルールの複雑さには制限があります。ルールが複雑すぎる場合、 はルールの作成`InvalidEventPatternException`時に EventBridge を返します。ルールでこのようなエラーが発生した場合は、以下のガイダンスを参考にしてイベントパターンの複雑さを軽減することを検討してください。

- 使用するワイルドカード文字の数を減らす

  ワイルドカード文字は、複数の可能な値と一致させることが本当に必要な場合にのみ使用します。例えば、次のイベントパターンで、同じリージョン内のイベントバスと一致させる場合を考えてみます。

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

  上記の場合、ARN のセクションの多くは、イベントバスが存在するリージョンに直接基づいています。したがって、`us-east-1` リージョンを使用している場合は、次の例のように、複雑さを減らしたパターンでも必要な値と一致します。

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

- ワイルドカード文字の後に繰り返される文字シーケンスを減らす

  ワイルドカードを使用した後に同じ文字シーケンスが複数回出現すると、イベントパターンの処理が複雑になります。イベントパターンを再キャストして、繰り返されるシーケンスを最小限に抑え

ます。次の例で、任意のユーザーのファイルのファイル名 doc.txt と一致させる場合を考えてみます。

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

doc.txt ファイルが、指定したパスにのみ存在することがわかっている場合は、繰り返される文字シーケンスを次の方法で減らすことができます。

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

複数のマッチングを含む複雑な例

複数のマッチングルールを組み合わせて、より複雑なイベントパターンにすることができます。例えば、次のイベントパターンでは、anything-but と numeric を組み合わせています。

```
{
  "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**
>
> イベントパターンを構築するときに、キーを複数回含めると、最後のリファレンスがイベントの評価に使用されるものになります。たとえば、次のパターンの場合：
>
> ```
> {
>   "detail": {
>     "location": [ { "prefix": "us-" } ],
>     "location": [ { "anything-but": "us-east" } ]
>   }
> ```

```
}
```

`{ "anything-but": "us-east" }` のみが `location` を評価する際に考慮されます。

複数の $or マッチングを含む複雑な例

また、複数のフィールドにわたって任意のフィールド値が一致するかどうかを確認する複雑なイベントパターンを作成することもできます。`$or` を使用して、複数のフィールドの値のいずれかが一致した場合に一致するイベントパターンを作成します。

`$or` コンストラクト内の個々のフィールドのパターンマッチングには、数値マッチングや配列など、他のフィルタータイプを含めることができますので注意してください。

次のいずれかの条件が満たされる場合、次のイベントパターンと一致します。

- `c-count` フィールドが 0 より大きい、または 5 以下。
- `d-count` フィールドが 10 未満。
- `x-limit` フィールドが 3.018e2 に等しい。

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

> **Note**
>
> イベントパターン (`PutRule`、`CreateArchive`、`UpdateArchive`、および `TestEventPattern`) を受け入れる API では、`$or` を使用した結果、1000 を超えるルールの組み合わせが生成された場合、`InvalidEventPatternException` をスローします。イベントパターン内のルールの組み合わせの数を決定するには、イベントパターン内の各 `$or` 配列の引数の合計数を掛けます。たとえば、上記のパターンには 3 つの引数を持つ単一

の $or 配列が含まれているため、ルールの組み合わせの総数も 3 つになります。2 つの引数を持つ別の $or 配列を追加した場合、ルールの組み合わせの合計は 6 つになります。

EventBridge サンドボックスを使用したイベントパターンのテスト

ルールでは、イベントパターンを使用してイベントを選択し、ターゲットに送信します。イベントパターンは、一致するイベントと同じ構造をしています。イベントパターンは、イベントに一致するか、一致しないかのいずれかになります。

イベントパターンの定義は、通常、新しいルールの作成や既存のルールの編集という、より大きなプロセスの一環として行います。ただし EventBridge、 のサンドボックスを使用すると、ルールを作成または編集しなくても、イベントパターンをすばやく定義し、サンプルイベントを使用して、パターンが目的のイベントと一致することを確認できます。イベントパターンをテストしたら、サンドボックスから直接そのイベントパターンを使用して新しいルールを作成する EventBridge オプションを提供します。

イベントパターンの詳細については、「???」を参照してください。

> **Important**
>
> では EventBridge、 higher-than-expected 料金やスロットリングにつながる可能性のあるルールを作成できます。例えば、ルールが無限に再帰的に実行される無限ループに陥るようなルールを誤って作成してしまうことがあります。Amazon S3 バケットで ACL が変更されたことを検出し、ソフトウェアをトリガーして目的の状態に変更するルールを作成したとします。このルールが慎重に記述されていない場合は、その後 ACL を変更するとルールが再び開始され、無限ループが作成されます。
> このような予期しない結果を最小限に抑えるための正確なルールやイベントパターンを記述する方法に関するガイダンスについては、??? および ??? を参照してください。

EventBridge サンドボックスを使用してイベントパターンをテストするには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで [デベロッパーリソース]、[サンドボックス] の順に選択し、[サンドボックス] ページで [イベントパターン] タブを選択します。
3. イベントソース で、AWS イベント または EventBridgeパートナーイベント を選択します。

4. [サンプルイベント] セクションで、イベントパターンをテストする対象の [サンプルイベントタイプ] を選択します。

   次のサンプルイベントタイプを使用できます。

   - AWS events – サポートされている から出力されるイベントから選択します AWS のサービス。
   - EventBridge パートナーイベント – Salesforce EventBridgeなどの をサポートするサードパーティーサービスから出力されるイベントから選択します。
   - [自分のイベントを入力] — 自分のイベントを JSON テキストで入力します。

     AWS または パートナーイベントを独自のカスタムイベントを作成するための開始点として使用することもできます。

     1. AWS イベントまたはEventBridge パートナーイベント を選択します。
     2. [サンプルイベント] ドロップダウンを使用して、カスタムイベントの開始点として使用するイベントを選択します。

        EventBridge にサンプルイベントが表示されます。
     3. [コピー] を選択します。
     4. [イベントタイプ] で [自分のイベントを入力] を選択します。
     5. JSON 編集ペインのサンプルイベント構造を削除し、代わりに AWS または パートナーイベントを貼り付けます。
     6. イベント JSON を編集して独自のサンプルイベントを作成します。
5. [Creation method] (作成方法) を選択します。 EventBridge スキーマまたはテンプレートからイベントパターンを作成することも、カスタムイベントパターンを作成することもできます。

   Existing schema

   既存の EventBridge スキーマを使用してイベントパターンを作成するには、次の手順を実行します。

   1. [Creation method] (作成方法) セクションの [Method] (メソッド) で、[Use schema] (スキーマを使用) を選択します。
   2. [Event pattern] (イベントパターン) セクションの [Schema type] (スキーマタイプ) で、[Select schema from Schema registry] (スキーマレジストリからスキーマを選択) を選択します。

3. [Schema registry] (スキーマレジストリ) では、ドロップダウンボックスを選択して、`aws.events` などのスキーマレジストリの名前を入力します。表示されるドロップダウンリストからオプションを選択することもできます。
4. [Schema] (スキーマ) では、ドロップダウンボックスを選択して、使用するスキーマの名前を入力します。例えば `aws.s3@ObjectDeleted` です。表示されるドロップダウンリストからオプションを選択することもできます。
5. [Models] (モデル) セクションでは、任意の属性の横にある [Edit] (編集) ボタンを選択してプロパティを開きます。必要に応じて [Relationship] (関係) フィールドと [Value] (値) フィールドを設定し、次に [Set] (設定) を選択して属性を保存します。

   > **Note**
   >
   > 属性の定義については、属性名の横にある[Info] (情報) アイコンを選択してください。イベントで属性のプロパティを設定する方法については、属性のプロパティダイアログボックスの [Note] (注意) セクションを開いてください。
   > 属性のプロパティを削除するには、その属性の [Edit] (編集) ボタンを選択し、[Clear] (クリア) を選択します。

6. [Generate event pattern in JSON] (JSON でイベントパターンを生成) を選択し、イベントパターンを JSON テキストとして生成して検証します。
7. テストパターンに対してサンプルイベントをテストするには、[テストパターン] を選択します。

   EventBridge は、サンプルイベントがイベントパターンと一致するかどうかを示すメッセージボックスを表示します。

   次のオプションのいずれかを選択することもできます。

   - [Copy] (コピー) — イベントパターンをデバイスのクリップボードにコピーします。
   - [Prettify] (整文) — 改行、タブ、スペースを追加して JSON テキストを読みやすくします。

Custom schema

カスタムスキーマを記述し、それをイベントパターンに変換して、以下の操作を行います。

1. [Creation method] (作成方法) セクションの [Method] (メソッド) で、[Use schema] (スキーマを使用) を選択します。

2. [Event pattern] (イベントパターン) セクションで、[Schema type] (スキーマの種類) で、[Enter schema] (スキーマを入力) を選択します。
3. テキストボックスにスキーマを入力します。スキーマは有効な JSON テキストとしてフォーマットする必要があります。
4. [Models] (モデル) セクションで、任意の属性の横にある[Edit] (編集) ボタンを選択してプロパティを開きます。必要に応じて [Relationship] (関係) フィールドと [Value] (値) フィールドを設定し、次に [Set] (設定) を選択して属性を保存します。

   > **Note**
   >
   > 属性の定義については、属性名の横にある[Info] (情報) アイコンを選択してください。イベントで属性のプロパティを設定する方法については、属性のプロパティダイアログボックスの [Note] (注意) セクションを開いてください。
   > 属性のプロパティを削除するには、その属性の [Edit] (編集) ボタンを選択し、[Clear] (クリア) を選択します。

5. [Generate event pattern in JSON] (JSON でイベントパターンを生成) を選択し、イベントパターンを JSON テキストとして生成して検証します。
6. テストパターンに対してサンプルイベントをテストするには、[テストパターン] を選択します。

   EventBridge は、サンプルイベントがイベントパターンと一致するかどうかを示すメッセージボックスを表示します。

   次のオプションのいずれかを選択することもできます。

   - [Copy] (コピー) — イベントパターンをデバイスのクリップボードにコピーします。
   - [Prettify] (整文) — 改行、タブ、スペースを追加して JSON テキストを読みやすくします。

Event pattern

カスタムイベントパターンを JSON 形式で記述するには、以下を実行します。

1. [Creation method] (作成方法) セクションの [Method] (メソッド) で、[Custom pattern (JSON editor)] (カスタムパターン (JSON エディタ)) を選択します。
2. [Event pattern] (イベントパターン) には、カスタムイベントパターンを JSON 形式のテキストで入力します。

3. テストパターンに対してサンプルイベントをテストするには、[テストパターン] を選択します。

   EventBridge は、サンプルイベントがイベントパターンと一致するかどうかを示すメッセージボックスを表示します。

   次のオプションのいずれかを選択することもできます。

   - [Copy] (コピー) — イベントパターンをデバイスのクリップボードにコピーします。
   - [Prettify] (整文) — 改行、タブ、スペースを追加して JSON テキストを読みやすくします。
   - [Event pattern form] (イベントパターンフォーム) — パターンビルダーでイベントパターンを開きます。Pattern Builder でパターンをそのままレンダリングできない場合、 は Pattern Builder を開く前に EventBridge 警告します。

6. (オプション) このイベントパターンを使用してルールを作成し、そのルールを特定のイベントバスに割り当てるには、[パターンを持つルールを作成] を選択します。

   EventBridge では、ルールの作成 のステップ 1 に進みます。これを使用してルールを作成し、選択したイベントバスに割り当てることができます。

   [ステップ 2 - イベントパターンを構築] には、既に指定したイベントパターン情報が表示されていることに注意してください。これを受け入れるか、更新することができます。

   ルールを作成する方法の詳細については、「???」を参照してください。

Amazon EventBridge イベントパターンを定義する際のベストプラクティス

イベントバスルールでイベントパターンを定義する際に考慮すべきいくつかのベストプラクティスを以下に示します。

無限ループを記述することは避ける

では EventBridge、ルールが繰り返し実行される無限ループにつながるルールを作成できます。たとえば、S3 バケットで ACL が変更されたことを検出し、ソフトウェアをトリガーして ACL を目的の状態に変更するルールがあるとします。このルールが慎重に記述されていない場合は、その後 ACL を変更するとルールが再び開始され、無限ループが作成されます。

このような問題を防ぐには、ルールのイベントパターンをできるだけ正確に記述して、実際にターゲットに送信したいイベントのみと一致するようにします。上の例では、トリガーされたアクションが同じルールを再実行しないように、イベントと一致するイベントパターンを作成します。例えば、何らかの変更の後ではなく、ACL が不良状態であることが判明した場合にのみイベントと一致するイベントパターンをルールに作成します。詳細については、「???」および「???」を参照してください。

無限ループにより、予想よりも高い料金がすぐに発生する可能性があります。また、スロットリングやイベント配信の遅延にもつながる可能性があります。呼び出し率の上限を監視して、予期しないボリュームの急上昇について警告を受けることができます。

予算の設定を使用すると、料金が指定の限度を超えたときにアラートが通知されます。詳細については、「予算によるコストの管理」を参照してください。

イベントパターンをできるだけ正確にする

イベントパターンが正確であればあるほど、実際に望むイベントのみに一致する可能性が高くなり、新しいイベントがイベントソースに追加されたり、既存のイベントが更新されて新しいプロパティを含むようになったりしても、予期せぬ一致が回避されます。

イベントパターンには、次の条件に一致するフィルターを含めることができます。

- イベントに関するイベントメタデータ (例: `source`、`detail-type`。`account`、または `region`)。
- これはイベントデータで、`detail` オブジェクト内のフィールドです。
- イベントの内容、または `detail` オブジェクト内のフィールドの実際の値。

ほとんどのパターンは、`source` や `detail-type` フィルターのみの指定など、単純です。ただし、 EventBridge パターンにはイベントの任意のキーまたは値でフィルタリングする柔軟性が含まれています。さらに、`prefix` などのコンテンツフィルターや `suffix` フィルターを適用して、パターンの精度を向上させることができます。詳細については、「???」を参照してください。

イベントソースと詳細タイプをフィルターとして指定する

`source` および `detail-type` メタデータフィールドを使用してイベントパターンをより正確にすることで、無限ループの生成や望ましくないイベントの一致を減らすことができます。

2 つ以上のフィールド内の特定の値を一致させる必要がある場合は、値の 1 つの配列に考えられるすべての値を一覧表示するのではなく、`$or` 比較演算子を使用してください。

を介して配信されるイベントの場合は AWS CloudTrail、 `eventName`フィールドをフィルターとして使用することをお勧めします。

次のイベントパターンの例では、Amazon Simple Queue Service サービス`SetQueueAttributes`からの `CreateQueue`または 、`CreateKey`または AWS Key Management Service サービスからの`DisableKeyRotation`イベントに一致します。

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

アカウントと地域をフィルターとして指定する

イベントパターンに `account` と `region` フィールドを含めると、クロスアカウントまたはクロスリージョンのイベント照合を制限できます。

コンテンツフィルターを指定する

コンテンツベースのフィルタリングは、イベントパターンの長さを最小限に抑えながら、イベントパターンの精度を向上させるのに役立ちます。例えば、考えられるすべての数値を一覧表示する代わりに、数値範囲に基づく照合が役立つ場合があります。

詳細については、「???」を参照してください。

イベントソースの更新を考慮してイベントパターンの範囲を設定する

イベントパターンを作成するときは、イベントスキーマとイベントドメインが時間の経過とともに進化し拡大する可能性があることを考慮する必要があります。繰り返しになりますが、イベントパターンをできるだけ正確にすることで、イベントソースが変更または拡大された場合に予期しない一致が発生するのを防ぐのに役立ちます。

例えば、支払い関連のイベントを公開する新しいマイクロサービスのイベントと照合するとします。最初に、サービスはドメイン `acme.payments` を使用し、1 つのイベント `Payment accepted` を公開します。

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

この時点で、支払い承認イベントと一致する簡単なイベントパターンを作成できます。

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

ただし、支払い拒否に対して後でサービスが次の新しいイベントを導入したとします。

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

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

この場合、作成した単純なイベントパターンが、Payment accepted と Payment rejected 両方のイベントと一致するようになります。 EventBridge は、両方のタイプのイベントを指定されたターゲットにルーティングして処理するため、処理の失敗や追加処理コストが発生する可能性があります。

Payment accepted イベントのみにイベントパターンの範囲を限定するには、source と detail-type の両方を最小に指定する必要があります。

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

イベントパターンにアカウントとリージョンを指定して、クロスアカウントまたはクロスリージョンのイベントがこのルールに一致するタイミングをさらに制限することもできます。

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

イベントパターンを検証する

ルールが目的のイベントと一致することを確認するために、イベントパターンを検証することを強くお勧めします。イベントパターンは、 EventBridge コンソールまたは API を使用して検証できます。

- EventBridge コンソールでは、ルール の作成の一環として、またはサンドボックス を使用して個別にイベントパターンを作成およびテストできます。
- TestEventPattern アクションを使用して、イベントパターンをプログラムでテストできます。

Amazon EventBridge ルール

各イベントバスに配信されるイベント EventBridge の動作を指定します。これを行うには、ルールを作成します。ルールは、処理するターゲットに送信するイベントを指定します。1 つのルールで複数のターゲットにイベントを送信し、並行して実行することができます。

2 種類のルールを作成できます。

- イベントデータに一致するルール

  イベントデータ基準 (イベントパターン と呼ばれる) に基づいて、受信イベントと照合するルールを作成できます。イベントパターンでは、ルールでマッチングするイベント構造とフィールドを定義します。イベントがイベントパターンで定義された基準に一致する場合、 は指定したターゲット (1 つ以上) にイベント EventBridge を送信します。

  詳細については、「???」を参照してください。
- スケジュールに従って実行されるルール

  指定した間隔で指定したターゲットにイベントを送信するルールを作成することもできます。例えば、 Lambda 関数を定期的に実行するには、スケジュールに従って実行するルールを作成します。

> **Note**
>
> EventBridge は、1 つの中央マネージドサービスからタスクを作成、実行、管理できるサーバーレス EventBridge ケジューラを提供します。 EventBridge スケジューラは高度にカスタマイズ可能で、より広範なターゲット API オペレーションと AWS サービスセットにより、スケジュールされたルールよりも EventBridgeスケーラビリティが向上します。
> スケジュールに従ってターゲットを呼び出すには、ス EventBridge ケジューラを使用することをお勧めします。詳細については、「???」を参照してください。

次のビデオでは、ルールの基本について説明します：ルールとは

Amazon EventBridge マネージドルール

作成したルールに加えて、 AWS サービスは、それらのサービスの特定の機能に必要な EventBridge ルールを AWS アカウントに作成および管理できます。これらは、マネージドルールと呼ばれます。

サービスがマネージドルールを作成するときに、そのサービスにルールを作成するアクセス許可を付与する IAM ポリシーを作成することもできます。この方法で作成された IAM ポリシーは、リソースレベルのアクセス許可で範囲を絞り込み、必要なルールの作成のみを許可します。

マネージドルールは、強制削除オプションを使って削除できますが、他のサービスでそのルールが不要になったことが確実な場合にのみ削除してください。それ以外の場合、マネージドルールを削除すると、このルールに依存する機能が動作しなくなります。

イベントに反応する Amazon EventBridge ルールの作成

Amazon EventBridge が受信したイベントに対してアクションを実行するには、ルールを作成できます。イベントがルールで定義したイベントパターンに一致すると、EventBridge は指定されたターゲットにイベントを送信し、ルールで定義されたアクションをトリガーします。

次のビデオでは、さまざまな種類のルールの作成とそのテスト方法について説明します: ルールについて学ぶ。

次の手順に従って、イベントに応答する Amazon EventBridge ルールを作成します。

イベントに反応するルールを作成する

次の手順では、EventBridge で指定先のイベントバスに送信するイベントのマッチングに使用するルールの作成方法を示します。

ステップ

- ルールを定義する
- イベントパターンを作成する
- ターゲットを選択する
- タグとレビュールールを設定する

ルールを定義する

まず、ルールを識別するため、ルールの名前と説明を入力します。また、ルールがイベントパターンに一致するイベントを検索するイベントバスも定義する必要があります。

ルールの詳細を定義するには

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで [Rules] (ルール) を選択します。
3. [Create rule] (ルールの作成) を選択します。
4. ルールの名前を入力し、オプションで説明を入力します。

   ルールには、同じ AWS リージョン 内および同じイベントバス上の別のルールと同じ名前を付けることはできません。

5. [Event bus] (イベントバス) で、このルールに関連付けるイベントバスを選択します。このルールをアカウントからのイベントと一致させるには、AWS のデフォルトのイベントバスを選択します。アカウントの AWS のサービスで発生したイベントは、常にアカウントのデフォルトのイベントバスに移動します。
6. [Rule type] (ルールタイプ) では、[Rule with an event pattern] (イベントパターンを持つルール) を選択します。
7. [Next] (次へ) をクリックします。

イベントパターンを作成する

次に、イベントパターンを作成します。そのためには、イベントソースを指定し、イベントパターンのベースを選択し、照合する属性と値を定義します。また、JSON でイベントパターンを生成し、サンプルイベントに対してテストすることもできます。

イベントパターンを作成するには

1. [Event source] (イベントソース) で、[AWS events or EventBridge partner events] (イベントまたは EventBridge パートナーイベント) を選択します。
2. (オプション) [サンプルイベント] セクションで、イベントパターンを照らし合わせてテストする [サンプルイベントタイプ] を選択します。

   次のサンプルイベントタイプを使用できます。

   - [AWS イベント] — サポートされている AWS のサービスから発行されたイベントを選択します。
   - [EventBridge partner events] (EventBridge パートナーイベント) — Salesforce など、EventBridge をサポートしているサードパーティのサービスから送信されるイベントから選択できます。
   - [自分のイベントを入力] — 自分のイベントを JSON テキストで入力します。

     独自のカスタムイベントを作成するための開始点として AWS イベントまたはパートナーイベントを使用することもできます。

     1. [AWS イベント] または [EventBridge パートナーイベント] を選択します。
     2. [サンプルイベント] ドロップダウンを使用して、カスタムイベントの開始点として使用するイベントを選択します。

        EventBridge にサンプルイベントが表示されます。

3. [コピー] を選択します。
4. [イベントタイプ] で [自分のイベントを入力] を選択します。
5. JSON 編集ペインのサンプルイベント構造を削除し、その場所に AWS イベントまたはパートナーイベントを貼り付けます。
6. イベント JSON を編集して独自のサンプルイベントを作成します。

3. [Creation method] (作成方法) を選択します。イベントパターンは、EventBridge のスキーマまたはテンプレートから定義するか、またはカスタムイベントパターンを作成できます。

   Existing schema

   既存の EventBridge スキーマを使用してイベントパターンを作成するには、以下を実行します。

   1. [Creation method] (作成方法) セクションの [Method] (メソッド) で、[Use schema] (スキーマを使用) を選択します。
   2. [Event pattern] (イベントパターン) セクションの [Schema type] (スキータイプ) で、[Select schema from Schema registry] (スキーマレジストリからスキーマを選択) を選択します。
   3. [Schema registry] (スキーマレジストリ) では、ドロップダウンボックスを選択して、`aws.events` などのスキーマレジストリの名前を入力します。表示されるドロップダウンリストからオプションを選択することもできます。
   4. [Schema] (スキーマ) では、ドロップダウンボックスを選択して、使用するスキーマの名前を入力します。例えば、`aws.s3@ObjectDeleted` です。表示されるドロップダウンリストからオプションを選択することもできます。
   5. [Models] (モデル) セクションでは、任意の属性の横にある [Edit] (編集) ボタンを選択してプロパティを開きます。必要に応じて [Relationship] (関係) フィールドと [Value] (値) フィールドを設定し、次に [Set] (設定) を選択して属性を保存します。

      > **Note**
      >
      > 属性の定義については、属性名の横にある[Info] (情報) アイコンを選択してください。イベントで属性のプロパティを設定する方法については、属性のプロパティダイアログボックスの [Note] (注意) セクションを開いてください。
      > 属性のプロパティを削除するには、その属性の [Edit] (編集) ボタンを選択し、[Clear] (クリア) を選択します。

6. [Generate event pattern in JSON] (JSON でイベントパターンを生成) を選択し、イベントパターンを JSON テキストとして生成して検証します。
7. (オプション) サンプルイベントをテストパターンと照らし合わせてテストするには、[テストパターン] を選択します。

   サンプルイベントがイベントパターンと一致するかどうかを示すメッセージボックスが表示されます。

   次のオプションのいずれかを選択することもできます。

   - [Copy] (コピー) — イベントパターンをデバイスのクリップボードにコピーします。
   - [Prettify] (整文) — 改行、タブ、スペースを追加して JSON テキストを読みやすくします。

Custom schema

カスタムスキーマを記述し、それをイベントパターンに変換して、以下の操作を行います。

1. [Creation method] (作成方法) セクションの [Method] (メソッド) で、[Use schema] (スキーマを使用) を選択します。
2. [Event pattern] (イベントパターン) セクションで、[Schema type] (スキーマの種類) で、[Enter schema] (スキーマを入力) を選択します。
3. テキストボックスにスキーマを入力します。スキーマは有効な JSON テキストとしてフォーマットする必要があります。
4. [Models] (モデル) セクションで、任意の属性の横にある[Edit] (編集) ボタンを選択してプロパティを開きます。必要に応じて [Relationship] (関係) フィールドと [Value] (値) フィールドを設定し、次に [Set] (設定) を選択して属性を保存します。

   > **Note**
   >
   > 属性の定義については、属性名の横にある[Info] (情報) アイコンを選択してください。イベントで属性のプロパティを設定する方法については、属性のプロパティダイアログボックスの [Note] (注意) セクションを開いてください。
   > 属性のプロパティを削除するには、その属性の [Edit] (編集) ボタンを選択し、[Clear] (クリア) を選択します。

5. [Generate event pattern in JSON] (JSON でイベントパターンを生成) を選択し、イベントパターンを JSON テキストとして生成して検証します。

6. (オプション) サンプルイベントをテストパターンと照らし合わせてテストするには、[テストパターン] を選択します。

   サンプルイベントがイベントパターンと一致するかどうかを示すメッセージボックスが表示されます。

   次のオプションのいずれかを選択することもできます。

   - [Copy] (コピー) — イベントパターンをデバイスのクリップボードにコピーします。
   - [Prettify] (整文) — 改行、タブ、スペースを追加して JSON テキストを読みやすくします。

Event pattern

カスタムイベントパターンを JSON 形式で記述するには、以下を実行します。

1. [Creation method] (作成方法) セクションの [Method] (メソッド) で、[Custom pattern (JSON editor)] (カスタムパターン (JSON エディター)) を選択します。
2. [Event pattern] (イベントパターン) には、カスタムイベントパターンを JSON 形式のテキストで入力します。
3. (オプション) サンプルイベントをテストパターンと照らし合わせてテストするには、[テストパターン] を選択します。

   サンプルイベントがイベントパターンと一致するかどうかを示すメッセージボックスが表示されます。

   次のオプションのいずれかを選択することもできます。

   - [Copy] (コピー) — イベントパターンをデバイスのクリップボードにコピーします。
   - [Prettify] (整文) — 改行、タブ、スペースを追加して JSON テキストを読みやすくします。
   - [Event pattern form] (イベントパターンフォーム) — パターンビルダーでイベントパターンを開きます。パターンをそのままパターンビルダーでレンダリングできない場合、EventBridge はパターンビルダーを開く前に警告を表示します。

4. [Next] (次へ) をクリックします。

ターゲットを選択する

指定されたパターンに一致するイベントを受信するターゲットを 1 つまたは複数選択します。ターゲットには、EventBridge のイベントバス、Salesforce などの SaaS パートナーを含む EventBridge API 送信先、または別の AWS のサービス を含めることができます。

ターゲットを選択するには

1. [ターゲットタイプ] で、以下のいずれかのターゲットタイプを選択します。

   Event bus

   EventBridge イベントバスを選択するには、[EventBridge event bus] (EventBridge イベントバス) を選択してから、次の操作を行います。

   - このルールと同じ AWS リージョンでイベントバスを使用するには:
     1. [同じアカウントとリージョン内のイベントバス] を選択します。
     2. [ターゲットのイベントバス] でドロップダウンボックスを選択し、イベントバスの名前を入力します。ドロップダウンリストからイベントバスを選択することもできます。

        詳細については、「???」を参照してください。
   - 別の AWS リージョン またはアカウント内のイベントバスをこのルールとして使用するには:
     1. [別のアカウントまたはリージョン内のイベントバス] を選択します。
     2. [ターゲットとしてのイベントバス] に、使用するイベントバスの ARN を入力します。

        詳細については、以下を参照してください。

        - ???
        - ???

   API destination

   EventBridge API 送信先を使用するには、[EventBridge API の宛先] を選択し、次のいずれかの操作を行います。

   - 既存の API 送信先を使用するには、[Use an existing API destination] (既存の API 送信先を使用する) を選択します。次に、ドロップダウンリストから API 送信先を選択します。

- 新しい API 送信先を作成するには、[Create a new API destination] (新しい API 送信先を作成する) を選択します。次に、送信先について次の詳細を指定します。
  - [Name] (名前) に、イベント送信先の名前を入力します。

    ロール名は AWS アカウント 内で一意である必要があります。文字列は 64 文字まで設定できます。有効な文字は、A-Z、a-z、0-9、および ._- (ハイフン) です。
  - (オプション) [Description (説明) に、アクションの簡潔な説明を入力します。

    説明の長さは、最大 512 文字です。
  - [API destination endpoint] (API 送信先エンドポイント) — ターゲットの URL エンドポイント。

    エンドポイント URL は **`https`** で始まる必要があります。**`*`** をパスパラメータワイルドカードとして含めることができます。ターゲットの `HttpParameters` 属性からパスパラメータを設定できます。
  - [HTTP method] (HTTP メソッド) — エンドポイントを呼び出すときに使用する HTTP メソッドを選択します。
  - (オプション) [Invocation rate limit per second] (1 秒あたりの呼び出しレート制限) — この送信先で 1 秒あたりに受け入れられる呼び出しの最大数を入力します。

    この値はゼロより大きくなければなりません。デフォルトでは、この値は 300 に設定されます。
  - [接続] — 新しい接続を使用するか既存の接続を使用するかを選択します。
    - 既存の接続を使用するには、[既存の接続を使用] を選択し、ドロップダウンリストから接続を選択します。
    - この送信先に新しい接続を作成するには、[Create a new connection] (新しい接続を作成する) を選択し、その接続の [Name] (名前)、[Destination type] (送信先タイプ)、および[Authorization type] (認証タイプ) を定義します。この接続にオプションの [Description] (説明) を追加することもできます。

詳細については、「???」を参照してください。

AWS のサービス

AWS のサービス を使用するには、AWS のサービス を選択し、次の操作を行います。

1. [Select a target] (ターゲットの選択) で、ターゲットとして使用する AWS のサービス を選択します。選択したサービスにリクエストされた情報を入力します。

> **Note**
>
> 表示されるフィールドは、選択したサービスによって異なります。利用できるターゲットの詳細については、「EventBridge コンソールで利用可能なターゲット」を参照してください。

2. 多くのターゲットタイプでは、EventBridge はターゲットにイベントを送信するためのアクセス許可が必要です。これらの場合、EventBridge は、イベントの実行に必要な IAM ロールを作成できます。

   [Execution role] (実行ロール) では、次のいずれかを実行します。

   - このルールの新しい実行ロールを作成するには:
     a. [この特定のリソースについて新しいロールを作成] を選択します。
     b. この実行ロールの名前を入力するか、EventBridge によって生成された名前を使用します。
   - このルールに既存の実行ロールを使用するには:
     a. [既存のロールを使用] を選択します。
     b. 使用する実行ロールの名前を入力するか、ドロップダウンリストから選択します。

3. (オプション) [追加設定] には、ターゲットタイプで使用できるオプション設定のいずれかを指定します。

   Event bus

   (オプション) [デッドレターキュー] で、標準 Amazon SQS キューをデッドレターキューとして使用するかどうかを選択します。EventBridge は、このルールに一致するイベントがターゲットに正常に配信されなかった場合に、そのイベントをデッドレターキューに送信します。以下のいずれかを実行します。

   - デッドレターキューを使用しない場合は、[None] (なし) を選択します。
   - Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue(デッドレターキューとして使用する現在の アカウントの Amazon SQS キューを選択) を選択し、ドロップダウンリストから使用するキューを選択します。

- Select an Amazon SQS queue in an other AWS account as a dead-letter queue(他の アカウントの Amazon SQS キューをデッドレターキューとして選択) を選択し、使用するキューの ARN を入力します。キューにメッセージを送信するための EventBridge 許可を付与するリソースベースのポリシーをそのキューにアタッチする必要があります。

  詳細については、「デッドレターキューへのアクセス許可の付与」を参照してください。

API destination

1. (オプション) [ターゲット入力を設定] で、一致するイベントに対してターゲットに送信されるテキストをカスタマイズする方法を選択します。次のいずれかを選択します。
   - 一致したイベント — EventBridge は元のソースイベント全体をターゲットに送信します。これがデフォルトです。
   - 一致したイベントの一部 — EventBridge は、元のソースイベントの指定された部分のみをターゲットに送信します。

     [一致したイベントの一部を指定] で、EventBridge からターゲットに送信するイベント部分を定義する JSON パスを指定します。
   - 定数 (JSON テキスト) — EventBridge は指定された JSON テキストのみをターゲットに送信します。元のソースイベントは、いずれの部分も送信されません。

     [JSON で定数を指定] で、EventBridge がイベントの代わりにターゲットに送信する JSON テキストを指定します。
   - 入力トランスフォーマー — EventBridge からターゲットに送信するテキストをカスタマイズするように入力トランスフォーマーを設定します。詳細については、「???」を参照してください。

     a. [入力トランスフォーマーを設定] を選択します。

     b. 「???」の手順に従って入力トランスフォーマーを設定します。
2. (オプション) [再試行ポリシー] で、エラーが発生した後に EventBridge がターゲットへのイベント送信を再試行する方法を指定します。
   - イベントの最大経過時間 — EventBridge が未処理のイベントを保持する最大時間 (時間、分、秒単位) を入力します。デフォルトは 24 時間です。
   - 再試行回数 — エラーが発生した後、EventBridge がターゲットへのイベント送信を再試行する最大回数を入力します。デフォルトは 185 回です。

3. (オプション) [デッドレターキュー] で、標準 Amazon SQS キューをデッドレターキューとして使用するかどうかを選択します。EventBridge は、このルールに一致するイベントがターゲットに正常に配信されなかった場合に、そのイベントをデッドレターキューに送信します。以下のいずれかを実行します。
   - デッドレターキューを使用しない場合は、[None] (なし) を選択します。
   - Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue(デッドレターキューとして使用する現在の アカウントの Amazon SQS キューを選択) を選択し、ドロップダウンリストから使用するキューを選択します。
   - Select an Amazon SQS queue in an other AWS account as a dead-letter queue(他の アカウントの Amazon SQS キューをデッドレターキューとして選択) を選択し、使用するキューの ARN を入力します。キューにメッセージを送信するための EventBridge 許可を付与するリソースベースのポリシーをそのキューにアタッチする必要があります。

     詳細については、「デッドレターキューへのアクセス許可の付与」を参照してください。

AWS service

特定の AWS のサービスに対しては、以下のフィールドがすべて表示されない場合があることに注意してください。

1. (オプション) [ターゲット入力を設定] で、一致するイベントに対してターゲットに送信されるテキストをカスタマイズする方法を選択します。次のいずれかを選択します。
   - 一致したイベント — EventBridge は元のソースイベント全体をターゲットに送信します。これがデフォルトです。
   - 一致したイベントの一部 — EventBridge は、元のソースイベントの指定された部分のみをターゲットに送信します。

     [一致したイベントの一部を指定] で、EventBridge からターゲットに送信するイベント部分を定義する JSON パスを指定します。
   - 定数 (JSON テキスト) — EventBridge は指定された JSON テキストのみをターゲットに送信します。元のソースイベントは、いずれの部分も送信されません。

     [JSON で定数を指定] で、EventBridge がイベントの代わりにターゲットに送信する JSON テキストを指定します。

- 入力トランスフォーマー — EventBridge からターゲットに送信するテキストをカスタマイズするように入力トランスフォーマーを設定します。詳細については、「???」を参照してください。

  a. [入力トランスフォーマーを設定] を選択します。

  b. 「???」の手順に従って入力トランスフォーマーを設定します。

2. (オプション) [再試行ポリシー] で、エラーが発生した後に EventBridge がターゲットへのイベント送信を再試行する方法を指定します。
   - イベントの最大経過時間 — EventBridge が未処理のイベントを保持する最大時間 (時間、分、秒単位) を入力します。デフォルトは 24 時間です。
   - 再試行回数 — エラーが発生した後、EventBridge がターゲットへのイベント送信を再試行する最大回数を入力します。デフォルトは 185 回です。
3. (オプション) [デッドレターキュー] で、標準 Amazon SQS キューをデッドレターキューとして使用するかどうかを選択します。EventBridge は、このルールに一致するイベントがターゲットに正常に配信されなかった場合に、そのイベントをデッドレターキューに送信します。以下のいずれかを実行します。
   - デッドレターキューを使用しない場合は、[None] (なし) を選択します。
   - Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue(デッドレターキューとして使用する現在の アカウントの Amazon SQS キューを選択) を選択し、ドロップダウンリストから使用するキューを選択します。
   - Select an Amazon SQS queue in an other AWS account as a dead-letter queue(他の アカウントの Amazon SQS キューをデッドレターキューとして選択) を選択し、使用するキューの ARN を入力します。キューにメッセージを送信するための EventBridge 許可を付与するリソースベースのポリシーをそのキューにアタッチする必要があります。

     詳細については、「デッドレターキューへのアクセス許可の付与」を参照してください。
4. (オプション) [Add another target] (別のターゲットを追加) を選択して、このルールに別のターゲットを追加します。
5. [Next] (次へ) をクリックします。

特定の AWS のサービスに対しては、以下のフィールドがすべて表示されない場合があることに注意してください。

タグとレビュールールを設定する

最後に、ルールに一致する任意のタグを入力し、ルールを確認して作成します。

タグを設定し、ルールを確認して作成するには

1. (オプション) ルールに 1 つ以上のタグを入力します。詳細については、「Amazon EventBridge タグ」を参照してください。
2. [Next] (次へ) を選択します。
3. 新しいルールの詳細を確認します。セクションを変更するには、そのセクションの横にある [Edit] (編集) ボタンを選択します。

   ルールの詳細を確認したら、[Create rule] (ルールを作成) を選択します。

Amazon EventBridge での Amazon EventBridge スケジューラの使用

Amazon EventBridge スケジューラはサーバーレススケジューラで、一元化されたマネージドサービスからタスクを作成、実行、管理できます。EventBridge スケジューラでは、繰り返しのパターンに cron やレート式を使ってスケジュールを作成したり、1回限りの呼び出しを設定したりできます。配信時間枠の柔軟な設定、再試行制限の定義、失敗した API 呼び出しの最大保持時間の設定を行うことができます。

EventBridge スケジューラは高度にカスタマイズ可能で、EventBridge のスケジュールされたルールよりもスケーラビリティが高く、より幅広いターゲット API オペレーションと AWS のサービスを使用できます。スケジュールに従ってターゲットを呼び出すには、EventBridge スケジューラを使用することをお勧めします。

トピック

実行ロールを設定する

新しいスケジュールを作成する場合、EventBridge スケジューラにはユーザーに代わってターゲット API オペレーションを呼び出すアクセス許可が必要です。実行ロールを使用して、これらのアクセス許可を EventBridge スケジューラに付与します。スケジュールの実行ロールにアタッチするアクセス許可ポリシーによって、必要なアクセス許可が定義されます。これらのアクセス許可は、EventBridge スケジューラが呼び出すターゲット API によって異なります。

次の手順のように EventBridge スケジューラコンソールを使用してスケジュールを作成すると、EventBridge スケジューラは選択したターゲットに基づき実行ロールを自動的に設定します。EventBridge スケジューラ SDK、AWS CLI、または AWS CloudFormation のいずれかを使用してスケジュールを作成する場合、EventBridge スケジューラがターゲットを呼び出すために必要なアクセス許可を付与する既存の実行ロールが必要です。スケジュールに合わせて実行ロールを手動で設定する方法についての詳細は、「EventBridge スケジューラユーザーガイド」の「実行ロールを設定する」を参照してください。

新しいスケジュールを作成する

コンソールを使用してスケジュールを作成するには

1. Amazon EventBridge スケジューラコンソール (https://console.aws.amazon.com/scheduler/home) を開きます。
2. [スケジュール] ページで、[スケジュールを作成] を選択します。
3. [スケジュールの詳細を指定] ページの [スケジュールの名前と説明] セクションで、次を実行します。

   a. [スケジュール名] で、スケジュールの名前を入力します。例えば、**MyTestSchedule** です。

   b. (オプション) [説明] で、スケジュールの説明を入力します。例えば、**My first schedule** です。

   c. [スケジュールグループ] で、ドロップダウンリストからスケジュールグループを選択します。グループがない場合は、[デフォルト] を選択します。スケジュールグループを作成するには、[独自のスケジュールを作成] を選択します。

   スケジュールグループを使用して、スケジュールのグループにタグを追加します。

4. • スケジュールオプションを選択します。

頻度	手順	
[1 回限りのスケジュール] 1 回限りのスケジュールは、指定した日時に 1 回だけターゲットを呼び出します。	[日付と時刻] で、次を実行します。 • 有効な日付を YYYY/MM/DD 形式で入力します。 • タイムスタンプを 24 時間 (hh:mm) 形式で入力します。 • [タイムゾーン] で、タイムゾーンを選択します。	
[繰り返しのスケジュール]	a. [スケジュールの種類] では、次のいずれかを実行します。	

頻度	手順	
繰り返しのスケジュールは、cron 式またはレート式を使用して指定したレートでターゲットを呼び出します。	• Cron 式を使用してスケジュールを定義するには、[cron ベースのスケジュール] を選択して Cron 式を入力します。 • Rate 式を使用してスケジュールを定義するには、[rate ベースのスケジュール] を選択して Rate 式を入力します。 cron およびレート式の詳細については、「Amazon EventBridge スケジューラ ユーザーガイド」の「EventBridge スケジューラのスケジュールタイプ」を参照してください。 b. [フレックスタイムウィンドウ] で、[オフ] を選択してオプションをオフにするか、事前定義された時間枠のいずれかを選択します。例えば、[15 分] を選択し、1 時間に 1 回ターゲットを呼び出す繰り返しのスケジュールを設定した場合、スケジュールは毎時の開始後	

頻度	手順	
	15 分以内に実行されます。	

5. (オプション) 前のステップで [定期的なスケジュール] を選択した場合は、[時間枠] セクションで次を実行します。

   a. [タイムゾーン] で、タイムゾーンを選択します。

   b. [開始日時] で、有効な日付を YYYY/MM/DD 形式で入力してから、タイムスタンプを 24 時間 (hh:mm) 形式で指定します。

   c. [終了日時] で、有効な日付を YYYY/MM/DD 形式で入力してから、タイムスタンプを 24 時間 (hh:mm) 形式で指定します。

6. [Next] (次へ) をクリックします。

7. [ターゲットを選択] ページで、EventBridge スケジューラが呼び出す AWS API オペレーションを選択します。

   a. [ターゲット API] で、[テンプレート化されたターゲット] を選択します。

   b. [Amazon EventBridge PutEvents] を選択します。

   c. [PutEvents] で、以下を指定します。

      - [EventBridge イベントバス] で、ドロップダウンリストからイベントバスを選択します。例えば、**`default`** です。

        EventBridge コンソールで [新しいイベントバスを作成] を選択して、新しいイベントバスを作成することもできます。

      - [詳細タイプ] に、一致させるイベントの詳細タイプを入力します。例えば、**`Object Created`** です。

      - [ソース] に、イベントのソースとなるサービスの名前を入力します。

        AWS サービスイベントの場合は、ソースとしてサービスのプレフィックスを指定します。`aws.` プレフィックスは含めないでください。例えば、Amazon S3 イベントの場合は、「**`s3`**」と入力します。

        サービスのプレフィックスを確認するには、「サービス認可リファレンス」の「条件キーテーブル」を参照してください。ソースと詳細タイプのイベント値について詳しくは、「???」を参照してください。

- (オプション): [詳細] に、EventBridge スケジューラから EventBridge に送信するイベントをさらにフィルタリングするためのイベントパターンを入力します。

  詳細については、「???」を参照してください。

8. [Next] (次へ) を選択します。
9. [Settings] (設定) ページで、以下の操作を行います。

   a. スケジュールをオンにするには、[スケジュールの状態] で [スケジュールを有効にする] をオンに切り替えます。

   b. スケジュールの再試行ポリシーを設定するには、[再試行ポリシーとデッドレターキュー (DLQ)] で次を実行します。

   - [再試行] を切り替えてオンにします。
   - [イベントの最大有効期間] で、EventBridge スケジューラが未処理のイベントを保持しなければならない最大の [時間] と [分] を入力します。
   - 最大 24 時間です。
   - [最大再試行回数] で、ターゲットがエラーを返した場合に EventBridge スケジューラがスケジュールを再試行する最大回数を入力します。

     再試行の最大値は 185 です。

   再試行ポリシーを使用すると、スケジュールがそのターゲットの呼び出しに失敗した場合、EventBridge スケジューラはスケジュールを再実行します。設定されている場合は、スケジュールの最大保持時間と再試行を設定する必要があります。

   c. EventBridge スケジューラが未配信のイベントを保存する場所を選択します。

[デッドレターキュー (DLQ)] オプション	手順
保存しない	[None] を選択します。
スケジュールを作成しようとしている同じ AWS アカウント にイベントを保存する	a. [自分の AWS アカウント の Amazon SQS キューを DLQ として選択] を選択します。

[デッドレターキュー (DLQ)] オプション	手順	
	b. Amazon SQS キューの Amazon リソースネーム (ARN) を選択します。	
スケジュールを作成しようとしているのとは別の AWS アカウント にイベントを保存する	a. [他の AWS アカウント の Amazon SQS キューを DLQ として指定] を選択します。 b. Amazon SQS キューの Amazon リソースネーム (ARN) を入力します。	

d. カスタマーマネージドキーを使用してターゲットの入力を暗号化するには、[暗号化] で [暗号化設定をカスタマイズする (高度)] を選択します。

このオプションを選択した場合は、既存の KMS キー ARN を入力するか、[AWS KMS key を作成] を選択して AWS KMS コンソールに移動します。EventBridge スケジューラが保管中のデータを暗号化する方法の詳細については、「Amazon EventBridge スケジューラユーザーガイド」の「保管中の暗号化」を参照してください。

e. EventBridge スケジューラに新しい実行ロールを作成させるには、[このスケジュールの新しいロールを作成] を選択します。その後、[ロール名] で名前を入力します。このオプションを選択すると、EventBridge スケジューラは、テンプレート化されたターゲットに必要な許可をロールにアタッチします。

10. [Next] (次へ) をクリックします。
11. [スケジュールの確認と作成] ページで、スケジュールの詳細を確認します。各セクションで、そのステップに戻って詳細を編集するには、[編集] を選択します。
12. [スケジュールを作成] を選択します。

[スケジュール] ページで、新規および既存のスケジュールのリストを表示できます。[ステータス] 列で、新しいスケジュールが [有効] になっていることを確認します。

関連リソース

EventBridge スケジューラに関する詳細については、次を参照してください。

- EventBridge スケジューラユーザーガイド
- EventBridge スキーマ API リファレンス
- EventBridge Scheduler Pricing

スケジュールに従って実行する Amazon EventBridge ルールの作成

ルールは、イベントに応じて実行したり、一定の時間間隔で実行したりすることができます。例えば、AWS Lambda 関数を定期的に実行するには、スケジュールに従って実行するルールを作成します。

> **Note**
>
> EventBridge は、Amazon EventBridge スケジューラを提供しています。これはサーバーレススケジューラで、一元化されたマネージドサービスからタスクを作成、実行、管理できます。EventBridge スケジューラは高度にカスタマイズ可能で、EventBridge のスケジュールルールよりもスケーラビリティが高く、ターゲット API 操作と AWS サービスの範囲が広がります。
> スケジュールに従ってターゲットを呼び出すには、EventBridge スケジューラを使用することをお勧めします。詳細については、「???」を参照してください。

EventBridge では、次の 2 種類のスケジュールされたルールを作成できます。

- 一定の間隔で実行するルール

  EventBridge は、これらのルールを定期的に (20 分間隔など) に実行します。

  スケジュールされたルールのレートを指定するには、rate 式を定義します。
- 特定の時間に実行するルール

  EventBridge は、これらのルールを特定の日時 (毎月第一月曜日の 太平洋標準時午前 8:00 時など)

  スケジュールされたルールを実行する時刻と日付を指定するには、cron 式を定義します。

rate 式は定義がより簡単であり、cron 式はスケジュールを詳細に制御します。例えば、cron 式を使用すると、毎週、または毎月の特定の日の指定した時刻に実行されるルールを定義できます。それに対して、rate 式では、1 時間に 1 回または 1 日 1 回など、一定の間隔でルールを実行します。

スケジュールされたイベントはすべて UTC+0 のタイムゾーンを使用し、スケジュールの最小精度は 1 分です。

> **Note**
>
> Eventbridge のスケジュール式は、秒レベルの精度ではありません。cron 式を使用した最小の粒度は 1 分です。EventBridge とターゲットサービスは分散しているため、スケジュールされたルールがトリガーされてからターゲットサービスがターゲットリソースを実行するまでの間に数秒の遅延が発生することもあります。

次のビデオでは、タスクのスケジューリングの概要を示します：EventBridge でスケジュールされたタスクを作成する

トピック

- 定期的に実行するルールの作成
- cron 式のリファレンス
- rate 式のリファレンス

定期的に実行するルールの作成

次の手順では、定期的に実行する Eventbridge ルールを作成する方法を示します。

> **Note**
>
> スケジュールされたルールは、デフォルトのイベントバスを使用してのみ作成できます。

ステップ

- ルールを定義する
- スケジュールを定義する
- ターゲットを選択する
- タグとレビュールールを設定する

ルールを定義する

まず、ルールを識別するため、ルールの名前と説明を入力します。

ルールの詳細を定義するには

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで [Rules] (ルール) を選択します。
3. [Create rule] (ルールの作成) を選択します。
4. ルールの名前を入力し、オプションで説明を入力します。

   ルールには、同じ AWS リージョン 内および同じイベントバス上の別のルールと同じ名前を付けることはできません。
5. [イベントバス] で、デフォルトのイベントバスを選択します。スケジュールされたルールは、デフォルトのイベントバスを使用してのみ作成できます。
6. ルールを作成してすぐに有効にするには、[選択したイベントバスでルールを有効にする] オプションをオンにします。
7. [Rule type] (ルールタイプ) では、[Schedule] (スケジュール) を選択します。

   この時点で、定期的に実行するルールの作成を続けるか、Amazon EventBridge スケジューラを使用するかを選択できます。
8. 続行する方法を選択します。

   - EventBridge スケジューラを使用してスケジュールを作成する

> **Note**
>
> EventBridge スケジューラはサーバーレススケジューラで、一元化されたマネージドサービスからタスクを作成、実行、管理できます。イベントバスやルールに依存しない、1 回限りの定期的なスケジューリング機能を提供します。EventBridge スケジューラは高度にカスタマイズ可能で、EventBridge のスケジュールルールよりもスケーラビリティが高く、ターゲット API 操作と AWS サービスの範囲が広がります。スケジュールに従ってターゲットを呼び出すには、EventBridge スケジューラを使用することをお勧めします。詳細については、「Amazon EventBridge スケジューラユーザーガイド」の「Amazon EventBridge スケジューラとは」を参照してください。

1. [EventBridge スケジューラで続行] を選択します

   EventBridge は、EventBridge スケジューラコンソールを開き、[スケジュールを作成] ページを表示します。

2. EventBridge スケジューラコンソールでスケジュールを作成します。

- 引き続き EventBridge を使用してデフォルトイベントバスのスケジュールされたルールを作成する

  1. [続行してルールを作成する] を選択します。

スケジュールを定義する

次に、スケジュールパターンを定義します。

スケジュールパターンを定義するには

1. [スケジュールパターン] で、スケジュールを特定の時間に実行するか、一定の間隔で実行するかを選択します。

   Specific time

   1. 毎月第一月曜日の太平洋標準日午前 8:00 など、特定の時間に実行するきめ細かいスケジュールを選択します。
   2. [cron 式] で、EventBridge がこのスケジュールされたルールをいつ実行するかを決めるために使用する cron 式を定義するフィールドを指定します。

      すべてのフィールドを指定すると、EventBridge はこのスケジュールされたルールを実行する以降の 10 個の日付を表示します。これらの日付を UTC で表示するか、ローカルタイムゾーンで表示するかを選択できます。

      cron 式の作成の詳細については、「???」を参照してください。

   Regular rate

   1. 一定の間隔 (10 分ごとなど) で実行するスケジュールを選択します。
   2. [rate 式] で、[値] フィールドと [単位] フィールドを指定し、EventBridge がこのスケジュールされたルールを実行するレートを定義します。

rate 式の作成の詳細については、「???」を参照してください。

2. [Next] (次へ) をクリックします。

ターゲットを選択する

指定されたパターンに一致するイベントを受信するターゲットを 1 つまたは複数選択します。ターゲットには、EventBridge のイベントバス、Salesforce などの SaaS パートナーを含む EventBridge API 送信先、または別の AWS のサービス を含めることができます。

ターゲットを選択するには

1. [ターゲットタイプ] で、以下のいずれかのターゲットタイプを選択します。

   Event bus

   EventBridge イベントバスを選択するには、[EventBridge event bus] (EventBridge イベントバス) を選択してから、次の操作を行います。

   - このルールと同じ AWS リージョンでイベントバスを使用するには:
     1. [同じアカウントとリージョン内のイベントバス] を選択します。
     2. [ターゲットのイベントバス] でドロップダウンボックスを選択し、イベントバスの名前を入力します。ドロップダウンリストからイベントバスを選択することもできます。

        詳細については、「???」を参照してください。
   - 別の AWS リージョン またはアカウント内のイベントバスをこのルールとして使用するには:
     1. [別のアカウントまたはリージョン内のイベントバス] を選択します。
     2. [ターゲットとしてのイベントバス] に、使用するイベントバスの ARN を入力します。

        詳細については、以下を参照してください。

        - ???
        - ???

   API destination

   EventBridge API 送信先を使用するには、[EventBridge API の宛先] を選択し、次のいずれかの操作を行います。

- 既存の API 送信先を使用するには、[Use an existing API destination] (既存の API 送信先を使用する) を選択します。次に、ドロップダウンリストから API 送信先を選択します。
- 新しい API 送信先を作成するには、[Create a new API destination] (新しい API 送信先を作成する) を選択します。次に、送信先について次の詳細を指定します。
  - [Name] (名前) に、イベント送信先の名前を入力します。

    ロール名は AWS アカウント 内で一意である必要があります。文字列は 64 文字まで設定できます。有効な文字は、A-Z、a-z、0-9、および ._- (ハイフン) です。
  - (オプション) [Description (説明) に、アクションの簡潔な説明を入力します。

    説明の長さは、最大 512 文字です。
  - [API destination endpoint] (API 送信先エンドポイント) — ターゲットの URL エンドポイント。

    エンドポイント URL は **`https`** で始まる必要があります。**`*`** をパスパラメータワイルドカードとして含めることができます。ターゲットの `HttpParameters` 属性からパスパラメータを設定できます。
  - [HTTP method] (HTTP メソッド) — エンドポイントを呼び出すときに使用する HTTP メソッドを選択します。
  - (オプション) [Invocation rate limit per second] (1 秒あたりの呼び出しレート制限) — この送信先で 1 秒あたりに受け入れられる呼び出しの最大数を入力します。

    この値はゼロより大きくなければなりません。デフォルトでは、この値は 300 に設定されます。
  - [接続] — 新しい接続を使用するか既存の接続を使用するかを選択します。
    - 既存の接続を使用するには、[既存の接続を使用] を選択し、ドロップダウンリストから接続を選択します。
    - この送信先に新しい接続を作成するには、[Create a new connection] (新しい接続を作成する) を選択し、その接続の [Name] (名前)、[Destination type] (送信先タイプ)、および[Authorization type] (認証タイプ) を定義します。この接続にオプションの [Description] (説明) を追加することもできます。

詳細については、「???」を参照してください。

AWS のサービス

AWS のサービス を使用するには、AWS のサービス を選択し、次の操作を行います。

1. [Select a target] (ターゲットの選択) で、ターゲットとして使用する AWS のサービス を選択します。選択したサービスにリクエストされた情報を入力します。

> **Note**
>
> 表示されるフィールドは、選択したサービスによって異なります。利用できるターゲットの詳細については、「EventBridge コンソールで利用可能なターゲット」を参照してください。

2. 多くのターゲットタイプでは、EventBridge はターゲットにイベントを送信するためのアクセス許可が必要です。これらの場合、EventBridge は、イベントの実行に必要な IAM ロールを作成できます。

   [Execution role] (実行ロール) では、次のいずれかを実行します。

   - このルールの新しい実行ロールを作成するには:
     a. [この特定のリソースについて新しいロールを作成] を選択します。
     b. この実行ロールの名前を入力するか、EventBridge によって生成された名前を使用します。
   - このルールに既存の実行ロールを使用するには:
     a. [既存のロールを使用] を選択します。
     b. 使用する実行ロールの名前を入力するか、ドロップダウンリストから選択します。

3. (オプション) [追加設定] には、ターゲットタイプで使用できるオプション設定のいずれかを指定します。

   Event bus

   (オプション) [デッドレターキュー] で、標準 Amazon SQS キューをデッドレターキューとして使用するかどうかを選択します。EventBridge は、このルールに一致するイベントがターゲットに正常に配信されなかった場合に、そのイベントをデッドレターキューに送信します。以下のいずれかを実行します。

   - デッドレターキューを使用しない場合は、[None] (なし) を選択します。

- Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue(デッドレターキューとして使用する現在の アカウントの Amazon SQS キューを選択) を選択し、ドロップダウンリストから使用するキューを選択します。
- Select an Amazon SQS queue in an other AWS account as a dead-letter queue(他の アカウントの Amazon SQS キューをデッドレターキューとして選択) を選択し、使用するキューの ARN を入力します。キューにメッセージを送信するための EventBridge 許可を付与するリソースベースのポリシーをそのキューにアタッチする必要があります。

  詳細については、「デッドレターキューへのアクセス許可の付与」を参照してください。

API destination

1. (オプション) [ターゲット入力を設定] で、一致するイベントに対してターゲットに送信されるテキストをカスタマイズする方法を選択します。次のいずれかを選択します。
   - 一致したイベント — EventBridge は元のソースイベント全体をターゲットに送信します。これがデフォルトです。
   - 一致したイベントの一部 — EventBridge は、元のソースイベントの指定された部分のみをターゲットに送信します。

     [一致したイベントの一部を指定] で、EventBridge からターゲットに送信するイベント部分を定義する JSON パスを指定します。
   - 定数 (JSON テキスト) — EventBridge は指定された JSON テキストのみをターゲットに送信します。元のソースイベントは、いずれの部分も送信されません。

     [JSON で定数を指定] で、EventBridge がイベントの代わりにターゲットに送信する JSON テキストを指定します。
   - 入力トランスフォーマー — EventBridge からターゲットに送信するテキストをカスタマイズするように入力トランスフォーマーを設定します。詳細については、「???」を参照してください。

     a. [入力トランスフォーマーを設定] を選択します。

     b. 「???」の手順に従って入力トランスフォーマーを設定します。
2. (オプション) [再試行ポリシー] で、エラーが発生した後に EventBridge がターゲットへのイベント送信を再試行する方法を指定します。
   - イベントの最大経過時間 — EventBridge が未処理のイベントを保持する最大時間 (時間、分、秒単位) を入力します。デフォルトは 24 時間です。

- 再試行回数 — エラーが発生した後、EventBridge がターゲットへのイベント送信を再試行する最大回数を入力します。デフォルトは 185 回です。

3. (オプション) [デッドレターキュー] で、標準 Amazon SQS キューをデッドレターキューとして使用するかどうかを選択します。EventBridge は、このルールに一致するイベントがターゲットに正常に配信されなかった場合に、そのイベントをデッドレターキューに送信します。以下のいずれかを実行します。

   - デッドレターキューを使用しない場合は、[None] (なし) を選択します。
   - Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue(デッドレターキューとして使用する現在の アカウントの Amazon SQS キューを選択) を選択し、ドロップダウンリストから使用するキューを選択します。
   - Select an Amazon SQS queue in an other AWS account as a dead-letter queue(他の アカウントの Amazon SQS キューをデッドレターキューとして選択) を選択し、使用するキューの ARN を入力します。キューにメッセージを送信するための EventBridge 許可を付与するリソースベースのポリシーをそのキューにアタッチする必要があります。

     詳細については、「デッドレターキューへのアクセス許可の付与」を参照してください。

AWS service

特定の AWS のサービスに対しては、以下のフィールドがすべて表示されない場合があることに注意してください。

1. (オプション) [ターゲット入力を設定] で、一致するイベントに対してターゲットに送信されるテキストをカスタマイズする方法を選択します。次のいずれかを選択します。

   - 一致したイベント — EventBridge は元のソースイベント全体をターゲットに送信します。これがデフォルトです。
   - 一致したイベントの一部 — EventBridge は、元のソースイベントの指定された部分のみをターゲットに送信します。

     [一致したイベントの一部を指定] で、EventBridge からターゲットに送信するイベント部分を定義する JSON パスを指定します。
   - 定数 (JSON テキスト) — EventBridge は指定された JSON テキストのみをターゲットに送信します。元のソースイベントは、いずれの部分も送信されません。

[JSON で定数を指定] で、EventBridge がイベントの代わりにターゲットに送信する JSON テキストを指定します。

- 入力トランスフォーマー — EventBridge からターゲットに送信するテキストをカスタマイズするように入力トランスフォーマーを設定します。詳細については、「???」を参照してください。

  a. [入力トランスフォーマーを設定] を選択します。

  b. 「???」の手順に従って入力トランスフォーマーを設定します。

2. (オプション) [再試行ポリシー] で、エラーが発生した後に EventBridge がターゲットへのイベント送信を再試行する方法を指定します。
   - イベントの最大経過時間 — EventBridge が未処理のイベントを保持する最大時間 (時間、分、秒単位) を入力します。デフォルトは 24 時間です。
   - 再試行回数 — エラーが発生した後、EventBridge がターゲットへのイベント送信を再試行する最大回数を入力します。デフォルトは 185 回です。
3. (オプション) [デッドレターキュー] で、標準 Amazon SQS キューをデッドレターキューとして使用するかどうかを選択します。EventBridge は、このルールに一致するイベントがターゲットに正常に配信されなかった場合に、そのイベントをデッドレターキューに送信します。以下のいずれかを実行します。
   - デッドレターキューを使用しない場合は、[None] (なし) を選択します。
   - Select an Amazon SQS queue in the current AWS account to use as the dead-letter queue(デッドレターキューとして使用する現在の アカウントの Amazon SQS キューを選択) を選択し、ドロップダウンリストから使用するキューを選択します。
   - Select an Amazon SQS queue in an other AWS account as a dead-letter queue(他の アカウントの Amazon SQS キューをデッドレターキューとして選択) を選択し、使用するキューの ARN を入力します。キューにメッセージを送信するための EventBridge 許可を付与するリソースベースのポリシーをそのキューにアタッチする必要があります。

     詳細については、「デッドレターキューへのアクセス許可の付与」を参照してください。
4. (オプション) [Add another target] (別のターゲットを追加) を選択して、このルールに別のターゲットを追加します。
5. [Next] (次へ) をクリックします。

タグとレビュールールを設定する

最後に、ルールに一致する任意のタグを入力し、ルールを確認して作成します。

タグを設定し、ルールを確認して作成するには

1. (オプション) ルールに 1 つ以上のタグを入力します。詳細については、「Amazon EventBridge タグ」を参照してください。
2. [Next] (次へ) を選択します。
3. 新しいルールの詳細を確認します。セクションを変更するには、そのセクションの横にある [Edit] (編集) ボタンを選択します。

   ルールの詳細を確認したら、[Create rule] (ルールを作成) を選択します。

cron 式のリファレンス

cron 式には 6 つの必須フィールドがあり、それらは空白で区切られます。

[Syntax] (構文)

```
cron(fields)
```

フィールド	値	ワイルドカード
分	0-59	, - * /
時間	0-23	, - * /
日	1-31	, - * ? / L W
月	1-12 または JAN-DEC	, - * /
曜日	1-7 または SUN-SAT	, - * ? L #
年	1970-2199	, - * /

ワイルドカード

- [,] (カンマ) のワイルドカードには、追加の値が含まれます。月フィールドの、「JAN,FEB,MAR」は、1 月、2 月、3 月を含みます。
- [-] (ダッシュ) のワイルドカードは、範囲を指定します。日フィールドの「1-15」は、指定した月の 1 日から 15 日を含みます。
- [*] (アスタリスク) のワイルドカードには、フィールドのすべての値が含まれます。[時間] フィールドの * には すべての時間が含まれます。[*] を日および曜日フィールドの両方に使用することはできません。一方に使用する場合は、もう一方に [?] を使用する必要があります。
- [/] (スラッシュ) ワイルドカードで増分を指定します。分フィールドで、「1/10」と入力して、その時間の最初の分から始めて、10 分毎を指定できます (11 分、21 分、31 分など)。
- ? (疑問符) ワイルドカードは任意を意味します。[日] フィールドに 7 と入力し、何曜日であってもかまわない場合、[曜日] フィールドに ? を入力できます。
- Day-of-month フィールドまたは Day-of-week フィールドの、ワイルドカード L は月または週の最終日を指定します。
- Day-of-month フィールドのワイルドカード **W** は、平日を指定します。Day-of-month フィールドで、**3W** は月の 3 日目に最も近い平日を指定します。
- Day-of-week フィールドの # ワイルドカードは、月の指定された曜日の特定のインスタンスを指定します。たとえば、3#2 は、月の第 2 火曜日を示します。3 は週の 3 番目の日 (火曜日) を示し、2 は月のそのタイプの 2 番目の日を示します。

> **Note**
>
> 「#」文字を使用する場合、曜日フィールドには 1 つの式しか定義できません。例えば、"3#1,6#3" は 2 つの式として解釈されるため、無効です。

制限事項

- cron 式の日フィールドと曜日フィールドを同時に指定することはできません。一方のフィールドに値または * (アスタリスク) を指定する場合、もう一方のフィールドで ? (疑問符) を使用する必要があります。
- 1 分より短い間隔を導き出す cron 式はサポートされていません。

例

スケジュールに基づいたルールを作成するときは、以下のサンプルの cron 文字列を使用できます。

分	時間	日	月	曜日	年	意味
0	10	*	*	?	*	毎日午前 10:00 (UTC+0) に実行
15	12	*	*	?	*	毎日午後 12:15 (UTC+0) に実行
0	18	?	*	MON-FRI	*	毎週月曜日から金曜日まで午後 6:00 (UTC+0) に実行
0	8	1	*	?	*	毎月 1 日の午前 8:00 (UTC+0) に実行
0/15	*	*	*	?	*	15 分ごとに実行
0/10	*	?	*	MON-FRI	*	月曜日から金曜日まで 10 分ごとに実行
0/5	8-17	?	*	MON-FRI	*	月曜日から金曜日まで午前 8:00 から午後 5:55 (UTC

分	時間	日	月	曜日	年	意味
						+0) の間に 5 分ごとに実行
0/15	1～20	?	*	MON-FRI	*	月曜日から金曜日まで開始日の午後10時から翌日の午前2時 (UTC) の間、30分間隔で実行 月曜日の午前 12 時から午前 2 時 (UTC) まで実行します。

次の例では、毎日午後 12:00 (UTC+0) に実行されるルールを作成します。

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

次の例では、毎日午前 2:05 と 午後2:35 (UTC+0) に実行されるルールを作成します。

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

次の例では、2019～2022 年の毎月最後の金曜日の午前 10:15 (UTC++0) に実行されるルールを作成します。

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

rate 式のリファレンス

rate 式は、予定されたイベントルールを作成すると開始され、その定義済みのスケジュールに基づいて実行されます。

rate 式には 2 つの必須フィールドがあり、空白で区切られます。

[Syntax] (構文)

```
rate(value unit)
```

値

正数。

単位

時刻の単位。値 1 には、`minute` などさまざまな単位が必要です。また、1 を超える値には `minutes` などの単位が必要です。

有効な値: minute | minutes | hour | hours | day | days

制限事項

値が 1 に等しい場合、単位は単数形であることが必要です。値が 1 より大きい場合、単位は複数であることが必要です。たとえば、rate(1 hours) と rate(5 hour) は有効ではありませんが、rate(1 hour) と rate(5 hours) は有効です。

例

次の例に示しているのは、AWS CLI `put-rule` コマンドで rate 式を使用する方法です。最初の例では、ルールを毎分起動し、次の例は 5 分ごとにルールを起動し、3 番目の例は 1 時間に 1 回ルールを起動し、最後の例は 1 日に 1 回ルールを起動します。

```
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 ルールの無効化または削除

ルールによってイベントが処理されたり、スケジュールに従って実行されたりしないようにするには、ルールを削除または無効化します。次の手順では、EventBridge ルールを削除または無効化する方法について説明します。

ルールを削除または無効化するには

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで [ルール] を選択します。

   Event bus (イベントバス)] で、ルールに関連付けられているイベントバスを選択します。
3. 次のいずれかを行ってください。

   a. ルールを削除するには、ルールの横にあるボタンを選択し、[Actions]、[Delete]、[Delete] の順に選択します。

      ルールがマネージドルールである場合は、ルールの名前を入力してそれがマネージドルールであること、およびルールを削除するとルールの作成元のサービスが機能しなくなる場合があることを了承します。続行するには、ルール名を入力し、[Force 削除 (強制削除)] を選択します。

   b. ルールを一時的に無効化するには、ルールの横にあるボタンを選択し、[Disable (無効化)]、[Disable (無効化)] の順に選択します。

      マネージドルールを無効化することはできません。

Amazon EventBridge のルールを定義する際のベストプラクティス

イベントバスのルールを作成する際に考慮すべきいくつかのベストプラクティスを以下に示します。

ルールごとに 1 つのターゲットを設定する

1 つのルールに最大 5 つのターゲットを指定できますが、ルールごとに 1 つのターゲットを指定する方がルールの管理が容易になります。複数のターゲットが同じイベントセットを受信する必要がある場合は、ルールを複製して同じイベントを異なるターゲットに配信することをお勧めします。このカプセル化により、ルールの管理が簡単になります。イベントターゲットのニーズが時間の経過とともにそれていく場合は、各ルールとそのイベントパターンを他のルールとは独立して更新できます。

ルールのアクセス許可を設定する

イベントを消費するアプリケーションコンポーネントまたはサービスが、独自のルールを管理できるようにすることができます。お客様が採用する一般的なアーキテクチャアプローチは、これらのアプリケーションコンポーネントまたはサービスを個別の AWS アカウントを使用して分離することです。あるアカウントから別のアカウントへのイベントのフローを有効にするには、別のアカウントのイベントバスにイベントをルーティングするルールを 1 つのイベントバスに作成する必要があります。イベントを消費するチームまたはサービスが、独自のルールを管理できるようにすることができます。そのためには、リソースポリシーを通じてアカウントへの適切なアクセス許可を指定します。これはアカウントとリージョン間で機能します。

詳細については、「???」を参照してください。

リソースポリシーの例については、GitHub の「Multi-account design patterns with Amazon EventBridge」(Amazon EventBridge によるマルチアカウントデザインパターン) を参照してください。

ルールのパフォーマンスを監視する

ルールを監視して、想定どおりに動作していることを確認します。

- データポイントの欠落や異常がないか、`TriggeredRules` メトリクスを監視すると、重大な変更を加えたパブリッシャーの差異を検出しやすくなります。詳細については、「???」を参照してください。
- 異常発生時のアラームや最大想定回数の設定も、ルールが新しいイベントと合致しているかどうかを検出するのに役立ちます。これは、AWS サービスや SaaS パートナーを含むイベントパブリッシャーが、新しいユースケースや機能を導入する際に新しいイベントを導入した場合に発生する可能性があります。これらの新しいイベントが予期せぬものであり、ダウンストリームターゲットの処理速度よりも処理量が多くなる場合、イベントバックログにつながる可能性があります。

  このような予期しないイベントの処理は、不要な料金につながる可能性もあります。

  また、アカウントが 1 秒あたりのサービスクォータの総ターゲット呼び出し数の上限を超えると、ルールのスロットリングをトリガーする可能性もあります。EventBridge は引き続き、スロットリングされたルールに一致するイベントの配信を試み、最大 24 時間まで、またはターゲットのカスタム再試行ポリシーに記載されているとおりに再試行します。`ThrottledRules` メトリクスを使用してスロットリングされたルールを検出し、警告することができます。
- 低レイテンシーのユースケースでは、`IngestionToInvocationStartLatency` を使用してレイテンシーを監視することもでき、これにより、イベントバスの状態を確認できます。30 秒を超

える長時間のレイテンシーは、サービスの中断またはルールのスロットリングを示している可能性があります。

Amazon EventBridge と AWS Serverless Application Model テンプレートの使用

EventBridge コンソールでルールの構築とテストを手動で行うことができ、イベントパターンを改良する時の開発プロセスに役立ちます。ただし、アプリケーションをデプロイする準備ができたら、すべてのサーバーレスリソースを一貫して起動できる AWS SAM のようなフレームワークを使用する方が簡単です。

このサンプルアプリケーションを使用して、EventBridge リソースを構築するための AWS SAM テンプレートの使用方法について検討します。この例の template.yaml ファイルは、4 つの AWS Lambda 関数を定義する AWS SAM テンプレートで、Lambda 関数を EventBridge に統合する 2 つの異なる方法を示します。

このサンプルアプリケーションのチュートリアルについては、??? を参照してください。

EventBridge と AWS SAM テンプレートを使用する方法は 2 つあります。1 つのルールによって 1 つの Lambda 関数が呼び出される単純な統合では、[Combined template] (組み合わせテンプレート) アプローチをお勧めします。複雑なルーティングロジックがある場合、または AWS SAM テンプレートの外部のリソースに接続している場合には、[Separated template] (分離テンプレート) アプローチはより良い選択です。

アプローチ：

- 組み合わせテンプレート
- 分離テンプレート

組み合わせテンプレート

最初のアプローチでは、`Events` プロパティを使用して、EventBridge ルールを設定します。次のサンプルコードでは、Lambda 関数を呼び出すイベントを定義します。

> **Note**
>
> この例では、全ての AWS アカウントに存在するデフォルトのイベントバスに関するルールを自動的に作成します。ルールをカスタムイベントバスに関連付けるには、`EventBusName` をテンプレートに追加します。

```
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"
```

この YAML コードは、EventBridge コンソールのイベントパターンと同等です。YAML では、イベントパターンを定義するだけでよく、AWS SAM は、必要なアクセス許可を持つ IAM ロールを自動的に作成します。

分離テンプレート

AWS SAM で EventBridge 設定を定義する 2 番目のアプローチでは、テンプレート内でリソースがより明確に分離されます。

1. まず、Lambda 関数を定義します。

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

2. 次に、`AWS::Events::Rule` リソースを使用してルールを定義します。プロパティはイベントパターンを定義し、ターゲットを指定することもできます。複数のターゲットを明示的に定義できます。

```
EventRuleCase1:
  Type: AWS::Events::Rule
  Properties:
    Description: "Approved transactions"
    EventPattern:
      source:
        - "custom.myATMapp"
      detail-type:
        - transaction
      detail:
        result:
          - "approved"
    State: "ENABLED"
    Targets:
      -
        Arn:
          Fn::GetAtt:
            - "atmConsumerCase1Fn"
            - "Arn"
        Id: "atmConsumerTarget1"
```

3. 最後に、EventBridge にターゲットを呼び出すアクセス許可を付与する AWS::Lambda::Permission リソースを定義します。

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

Amazon EventBridge ルールから AWS CloudFormation テンプレートを生成する

AWS CloudFormation では、インフラストラクチャをコードとして扱うことで、複数のアカウントやリージョンの AWS リソースを一元的かつ繰り返し可能な方法で設定および管理できます。CloudFormation では、これを実現するために、プロビジョニングおよび管理するリソースを定義するテンプレートを作成できます。

EventBridge では、CloudFormation テンプレートの開発をすぐに開始するための補助として、アカウント内の既存のイベントバスからテンプレートを生成できます。テンプレートに含めるルールは 1 つ、または複数選択することができます。次に、これらのテンプレートに基づいて CloudFormation で管理するリソースのスタックを作成できます。

CloudFormation の詳細については、「AWS CloudFormation ユーザーガイド」を参照してください。

> **Note**
>
> EventBridge は、生成されたテンプレートにマネージドルールを含みません。

イベントバスに含まれるルールを含め、既存のイベントバスからテンプレートを生成することもできます。

1 つ以上のルールから AWS CloudFormation テンプレートを生成するには

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで [Rules (ルール)] を選択します。
3. [Select event bus] (イベントバスの選択) で、テンプレートに含めるルールを含むイベントバスを選択します。
4. [Rules] (ルール) で、生成された AWS CloudFormation テンプレートに含めるルールを選択します。

   ルールが 1 つの場合は、ルールの詳細ページを表示するためのルール名を選択することもできます。
5. [CloudFormation Template] (CloudFormation テンプレート) を選択し、EventBridge でテンプレートを生成したいフォーマット (JSON または YAML) を選択します。

EventBridge には、選択した形式で生成されたテンプレートが表示されます。

6. EventBridge では、テンプレートファイルをダウンロードするか、テンプレートをクリップボードにコピーするかを選択できます。

   - テンプレートファイルをダウンロードするには、[Download] (ダウンロード) を選択します。
   - テンプレートをクリップボードにコピーするには、[Copy] (コピー) を選択します。

7. テンプレートを終了するには、[Cancel] (キャンセル) を選択します。

ユースケースに合わせて AWS CloudFormation テンプレートをカスタマイズしたら、それを使用して AWS CloudFormation でスタックを作成できます。

Amazon EventBridge から生成した CloudFormation テンプレートを使用する際の考慮事項

EventBridge から生成した CloudFormation テンプレートを使用するときは、次の点を考慮してください。

- EventBridge は、生成テンプレートにパスワードを含みません。

  テンプレートを編集してテンプレートパラメータを含めると、ユーザーがテンプレートを使用して CloudFormation スタックを作成または更新するときに、パスワードやその他の機密情報を指定できるようになります。

  さらに、ユーザーは Secrets Manager を使用して目的のリージョンにシークレットを作成し、生成されたテンプレートを編集して動的パラメーターを使用できます。
- 生成されたテンプレートのターゲットは、元のイベントバスで指定されていたものとまったく同じままです。テンプレートを使用して他のリージョンにスタックを作成する前に、テンプレートを適切に編集しないと、リージョン間の問題が発生する可能性があります。

  また、生成したテンプレートでは、下流のターゲットが自動的には作成されません。

Amazon EventBridge ターゲット

ターゲットは、イベントがルール に定義されたイベントパターンと一致するときにイベントを EventBridge に送信するリソースまたはエンドポイントです。ルールはイベントデータを処理し、関連情報をターゲットに送信します。イベントデータをターゲットに配信するには、ターゲットリソースにアクセスするためのアクセス許可 EventBridge が必要です。ルールごとに最大 5 つのターゲットを定義できます。

ルールにターゲットを追加し、その直後にルールが実行されると、新しいターゲットまたは更新されたターゲットがすぐに呼び出されない場合があります。変更が有効になるまで、しばらくお待ちください。

次のビデオでは、ターゲットの基本について説明します：ターゲットとは

EventBridge コンソールで利用可能なターゲット

EventBridge コンソールでイベントに対して次のターゲットを設定できます。

- API 送信先
- API Gateway
- AWS AppSync;
- バッチジョブのキュー
- CloudWatch ロググループ
- CodeBuild プロジェクト
- CodePipeline
- Amazon EBS `CreateSnapshot` API コール
- EC2 Image Builder
- EC2 `RebootInstances` API コール
- EC2 `StopInstances` API コール
- EC2 `TerminateInstances` API コール
- ECS タスク
- 別のアカウントまたはリージョンのイベントバス
- 同じアカウントとリージョンのイベントバス

- Firehose 配信ストリーム
- Glue ワークフロー
- Incident Manager レスポンスプラン
- Inspector 評価テンプレート
- Kinesis ストリーミング
- Lambda 関数 (ASYNC)
- Amazon Redshift クラスターデータ API クエリ
- Amazon Redshift Serverless ワークグループデータ API クエリ
- SageMaker パイプライン
- Amazon SNS トピック

  EventBridge は、Amazon SNS FIFO (先入れ先出し) トピック をサポートしていません。
- Amazon SQS キュー
- Step Functions ステートマシン (ASYNC)
- Systems Manager Automation
- Systems Manager OpsItem
- Systems Manager Run Command

ターゲットパラメータ

一部のターゲットは、イベントペイロード内の情報をターゲットに送信せず、代わりにイベントを特定の API を呼び出すトリガーとして扱います。 はターゲットパラメータ EventBridge を使用して、そのターゲットで何が起こるかを判断します。これには以下が含まれます。

- API 送信先 (API 送信先に送信されるデータは API の構造と一致する必要があります。`InputTransformer` オブジェクトを使用して、データが正しく構造化されていることを確認する必要があります。元のイベントペイロードを含める場合は、`InputTransformer` でそれを参照してください)。
- API Gateway (API Gateway に送信されるデータは API の構造と一致する必要があります。`InputTransformer` オブジェクトを使用して、データが正しく構造化されていることを確認してください。 元のイベントペイロードを含める場合は、`InputTransformer` でそれを参照してください)。
- Amazon EC2 Image Builder

- RedshiftDataParameters (Amazon Redshift データ API クラスター)
- SageMakerPipelineParameters (Amazon SageMaker Runtime Model Building Pipelines)

> **Note**
>
> EventBridge は、すべての JSON パス構文をサポートしているわけではなく、実行時に評価されます。サポートされている構文には以下が含まれます。
>
> - ドット表記 ($.detail など)
> - ダッシュ
> - 下線
> - アルファベットの文字
> - 配列インデックス
> - ワイルドカード (*)

動的パスパラメータ

一部のターゲットパラメータでは、オプションの動的 JSON パス構文がサポートされています。この構文では、静的値の代わりに JSON パスを指定できます (例えば、$.detail.state)。値の一部だけではなく全体を JSON パスにする必要があります。例えば、RedshiftParameters.Sql は $.detail.state とすることができますが、"SELECT * FROM $.detail.state" とすることはできません。このようなパスは、実行時に、指定されたパスにあるイベントペイロード自体のデータで動的に置き換えられます。動的パスパラメータは、入力変換の結果として生じる新しい値または変換された値を参照できません。動的パラメータの JSON パスでサポートされている構文は、入力を変換する場合と同じです。詳細については、「???」を参照してください。

動的構文は、これらのパラメータのすべての文字列の非列挙型フィールドで使用できます。

- EcsParameters
- HttpParameters (HeaderParameters キーを除く)
- RedshiftDataParameters
- SageMakerPipelineParameters

アクセス許可

所有しているリソースで API コールを行うには、 に適切なアクセス許可 EventBridge が必要です。AWS Lambda および Amazon SNS リソースの場合、リソースベースのポリシー EventBridge を使用します。EC2 インスタンス、Kinesis データストリーム、Step Functions ステートマシンの場合、 の RoleARNパラメータで指定した IAM ロール EventBridge を使用しますPutTargets。構成された IAM 認可で API Gateway エンドポイントを呼び出すことができますが、認可を構成していない場合、ロールはオプションです。詳細については、「Amazon EventBridge と AWS Identity and Access Management」を参照してください。

別のアカウントが同じリージョンにあって、許可を付与されている場合は、そのアカウントにイベントを送信できます。詳細については、「AWS アカウント間での Amazon EventBridge イベントの送受信」を参照してください。

ターゲットが暗号化されている場合は、KMS キーポリシーに次のセクションを含める必要があります。

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

EventBridge ターゲットの詳細

AWS Batch ジョブキュー

への AWS Batch submitJob特定のパラメータは、 を介して設定できますBatchParameters。

その他はイベントペイロードで指定できます。イベントペイロード (または を経由InputTransformers) に次のキーが含まれている場合、それらはsubmitJobリクエストパラメータにマッピングされます。

- `ContainerOverrides: containerOverrides`

> **Note**
>
> これには、コマンド、環境、メモリ、および VCPU のみが含まれます

- `DependsOn: dependsOn`

> **Note**
>
> これには jobId のみが含まれます

- `Parameters: parameters`

CloudWatch ロググループ

CloudWatch Logs ターゲットInputTransformerで を使用しない場合、イベントペイロードがログメッセージとして使用され、イベントのソースがタイムスタンプとして使用されます。を使用する場合 InputTransformer、テンプレートは次の条件を満たす必要があります。

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

EventBridge は、ログストリームに送信されたエントリをバッチ処理します。したがって、トラフィックに応じて、1 つまたは複数のイベントをログストリームに配信 EventBridge できます。

CodeBuild プロジェクト

InputTransformers を使用して入力イベントを 構造と一致する CodeBuild StartBuildRequestようにターゲットにシェープする場合、パラメータは 1 対 1 でマッピングされ、 に渡されますcodeBuild.StartBuild。

Amazon ECS タスク

InputTransformers を使用して入力イベントをターゲットに形成し、Amazon ECS RunTask TaskOverride構造と一致する場合、パラメータは 1 対 1 でマッピングされ、 に渡されますecs.RunTask。

Incident Manager レスポンスプラン

一致したイベントが CloudWatch アラームから発生した場合、アラーム状態の変更の詳細は、Incident Manager への StartIncidentRequest 呼び出しのトリガーの詳細に入力されます。

ターゲットの設定

EventBridge ターゲットの設定方法を説明します。

ターゲット：

- API 送信先
- Amazon API Gateway の Amazon EventBridge ターゲット
- AWS AppSync Amazon の ターゲット EventBridge
- HTTP エンドポイントターゲットの接続
- AWS アカウント間での Amazon EventBridge イベントの送受信
- AWS リージョン間での Amazon EventBridge イベントの送受信
- 同じアカウントとリージョンの EventBridge イベントバス間で Amazon イベントを送受信する

API 送信先

Amazon EventBridge API の送信先は、 AWS サービスまたはリソースをターゲットとして呼び出す方法と同様に、ルール のターゲットとして呼び出すことができる HTTP エンドポイントです。API 送信先を使用すると、API コールを使用して、 AWS サービス、統合された Software as a Service (SaaS) アプリケーション、および の外部アプリケーション間でイベント AWS をルーティングできます。API 送信先をルールのターゲットとして指定すると、 はルールで指定されたイベントパターンに一致するイベントの HTTP エンドポイントを EventBridge 呼び出し、リクエストとともにイベント情報を配信します。では EventBridge、リクエストに CONNECT と TRACE 以外の任意の HTTP メソッドを使用できます。最もよく使用する HTTP メソッドは PUT と POST です。また、入力トランスフォーマーを使用して、イベントを特定の HTTP エンドポイントのパラメータにカスタマイズすることもできます。詳細については、「Amazon EventBridge 入力変換」を参照してください。

> **Note**
>
> API 送信先は、インターフェイス VPC エンドポイントなどのプライベート送信先をサポートしていません。これには、プライベートネットワーク、Application Load Balancer、インターフェイス VPC エンドポイントを使用する仮想プライベートクラウド (VPC) のプライベート HTTPS APIs が含まれます。
> 詳細については、「???」を参照してください。

> **Important**
>
> EventBridge API 送信先エンドポイントへの リクエストの最大クライアント実行タイムアウトは 5 秒である必要があります。ターゲットエンドポイントの応答に 5 秒以上かかる場合、はリクエストを EventBridge タイムアウトします。 EventBridge は、再試行ポリシーで設定された最大値までリクエストをタイムアウトします。デフォルトでは、最大値は 24 時間と 185 回です。再試行の最大回数を超えると、デッドレターキューがあればイベントはそこに送られ、なければイベントはドロップされます。

次のビデオでは、API 送信先の使用を示しています:API 送信先を使用する

このトピックの内容

API 送信先の作成

API 送信先ごとに接続が必要です。接続 には、API 送信先エンドポイントでの認証に使用する認証タイプと認証情報を指定します。既存の接続を選択することも、API 送信先の作成と同時に接続を作成することもできます。詳細については、「???」を参照してください。

EventBridge コンソールを使用して API 送信先を作成するには

1. コンソール を管理 EventBridge および開くアクセス許可を持つアカウント AWS を使用して、にログインします。 EventBridge
2. 左のナビゲーションペインで、[API destinations] (API 送信先) を選択します。
3. [API destinations] (API 送信先) テーブルまで下にスクロールし、[Create API destination] (API 送信先の作成) を選択します。
4. [Create API destination] (API 送信先の作成) ページで、API 送信先の [Name] (名前) を入力します。大文字または小文字のアルファベット、数字、ドット (.)、ダッシュ (-)、アンダースコア (_) を最大 64 文字まで使用できます。

   この名前は、現在のリージョンのアカウントで一意であることが必要です。
5. API 送信先の [Description] (説明) を入力します。
6. API 送信先の [API destination endpoint] (API 送信先エンドポイント) を入力します。-API 送信先エンドポイントは、イベントの HTTP 呼び出しエンドポイントのターゲットです。この API 送信先に使用される接続に含める認証情報は、このエンドポイントに対する認証に使用されます。URL には HTTPS を使用してください。
7. API 送信先エンドポイントへの接続に使用する HTTP メソッドを入力します。
8. (オプション) [Invocation rate limit per second] (1 秒あたりの呼び出しレート制限) フィールドに、API 送信先エンドポイントに送信する 1 秒あたりの最大呼び出し回数を入力します。

設定したレート制限は、 がイベントを EventBridge 配信する方法に影響を与える可能性があります。詳細については、「呼び出しレートがイベント配信に与える影響」を参照してください。

9. [Connection] (接続) で、次のいずれかを実行します。
   - [Use an existing connection] (既存の接続を使用) を選択して、この API 送信先に使用する接続を選択します。
   - [Create a new connection] (新しい接続を作成) を選択して、作成する接続の詳細を入力します。詳細については、「接続」を参照してください。
10. [作成] を選択します。

API 送信先にイベントを送信するルールの作成

API 送信先を作成したら、それをルールのターゲットとして選択できます。API 送信先をターゲットとして使用するには、適切なアクセス許可を持つ IAM ロールを指定する必要があります。詳細については、「???」を参照してください。

API 送信先をターゲットとして選択することは、ルール作成の一部です。

コンソールを使用して、API 送信先にイベントを送信するルールを作成するには

1. 「???」のステップに従います。
2. ??? ステップでは、ターゲットタイプとして API 送信先を選択するように求められた場合：
   a. EventBridge API 送信先 を選択します。
   b. 次のいずれかを行います。
      - 既存の API 送信先を使用を選択し、既存の API 送信先を選択します。
      - 新しい API 送信先の作成を選択し、新しい API 送信先を定義するために必要な設定を指定します。

        必要な設定の指定の詳細については、「」を参照してください???。
   c. （オプション): イベントのヘッダーパラメータを指定するには、ヘッダーパラメータ で ヘッダーパラメータ を追加 を選択します。

      次に、 ヘッダーパラメータのキーと値を指定します。
   d. （オプション): イベントのクエリ文字列パラメータを指定するには、「クエリ文字列パラメータ」で「クエリ文字列パラメータを追加」を選択します。

次に、クエリ文字列パラメータのキーと値を指定します。

3. 手順のステップ に従ってルールの作成を完了します。

API 送信先のサービスにリンクされたロール

API 送信先の接続を作成すると、 という名前のサービスにリンクされたロールAWS ServiceRoleForAmazonEventBridgeApiDestinationsがアカウントに追加されます。 は、サービスにリンクされたロール EventBridge を使用してシークレットを作成し、Secrets Manager に保存します。サービスにリンクされたロールに必要なアクセス許可を付与するために、 EventBridge はAmazonEventBridgeApiDestinationsServiceRolePolicyポリシーをロールにアタッチします。このポリシーは、付与されるアクセス許可を、ロールが接続用のシークレットと対話するために必要なものだけに制限します。他のアクセス許可は含まれず、ロールはシークレットを管理するためにアカウント内の接続のみと対話できます。

以下のポリシーは AmazonEventBridgeApiDestinationsServiceRolePolicy です。

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

サービスにリンクされたロールの詳細については、IAM ドキュメントの「サービスにリンクされたロールの使用」を参照してください。

AmazonEventBridgeApiDestinationsServiceRolePolicy サービスにリンクされたロールは、次の AWS リージョンでサポートされています。

- 米国東部 (バージニア北部)
- 米国東部 (オハイオ)
- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- アフリカ (ケープタウン)
- アジアパシフィック (香港)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (大阪)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)
- カナダ (中部)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (ミラノ)
- ヨーロッパ (パリ)
- 欧州 (ストックホルム)
- 南米 (サンパウロ)
- 中国 (寧夏)
- 中国 (北京)

API 送信先へのリクエストのヘッダー

次のセクションでは、 が API 送信先へのリクエストで HTTP ヘッダー EventBridge を処理する方法について詳しく説明します。

API 送信先へのリクエストに含まれるヘッダー

API 送信先に使用される接続に定義されている認証ヘッダーに加えて、 は各リクエストに次のヘッダー EventBridge を含めます。

ヘッダーキー	ヘッダー値
ユーザーエージェント	Amazon/EventBridge/ApiDestinations
Content-Type	カスタム Content-Type 値を指定しない場合、 には Content-Type として次のデフォルト値 EventBridge が含まれます。 application/json; charset=utf-8
Range	bytes=0-1048575
Accept-Encoding	gzip、deflate
Connection	CLOSE
Content-Length	受信者に送信されるエンティティボディのサイズをバイト単位で示すエンティティヘッダー。
ホスト	リクエストを送信するサーバーのホストおよびポート番号を指定するリクエストヘッダー。

API 送信先へのリクエストの上書きできないヘッダー

EventBridge では、次のヘッダーを上書きすることはできません。

- ユーザーエージェント
- [Range] (範囲)

ヘッダーは API 送信先へのリクエストから EventBridge を削除します

EventBridge は、すべての API 送信先リクエストの次のヘッダーを削除します。

- A-IM
- Accept-Charset
- Accept-Datetime
- Accept-Encoding
- Cache-Control

- Connection
- Content-Encoding
- Content-Length
- Content−MD5
- 日付
- Expect
- Forwarded
- From
- ホスト
- HTTP2-Settings
- If-Match
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Max-Forwards
- オリジン
- Pragma
- Proxy-Authorization
- [Range] (範囲)
- リファラー
- TE
- Trailer
- Transfer-Encoding
- ユーザーエージェント
- Upgrade
- Via
- 警告

API 送信先のエラーコード

が API 送信先にイベントを配信 EventBridge しようとしてエラーが発生すると、 EventBridge は以下を実行します。

- エラーコード 409、429 および 5xx に関連するイベントが再試行されます。
- エラーコード 1xx、2xx、3xx、4xx (429 を除く) に関連するイベントは再試行されません。

EventBridge API 送信先は、標準の HTTP レスポンスヘッダーを読み取り`Retry-After`、フォローアップリクエストを実行する前に待機する時間を調べます。 は、定義された再試行ポリシーと `Retry-After` ヘッダーの間のより保守的な値 EventBridge を選択します。`Retry-After` 値が負の場合、 はそのイベントの配信の再試行を EventBridge 停止します。

呼び出しレートがイベント配信に与える影響

1 秒あたりの呼び出しレートを、生成された呼び出しの数より大幅に低い値に設定した場合、イベントの 24 時間の再試行時間内にはイベントが配信されないことがあります。例えば、呼び出しレートを 1 秒あたり 10 回に設定しても、1 秒あたり数千のイベントが生成されると、24 時間を超えるイベントのバックログがすぐに作成されてしまいます。イベントが失われないようにするには、デッドレターキューを設定して、呼び出しが失敗したイベントを送信し、後で処理できるようにします。詳細については、「デッドレターキューを使用した未配信イベントの処理」を参照してください。

API 送信先への CloudEvents イベントの送信

CloudEvents はイベントフォーマットのベンダーに依存しない仕様であり、サービス、プラットフォーム、システム間の相互運用性を提供することを目標としています。を使用して EventBridge 、API 送信先などのターゲットに送信される CloudEvents 前に、 AWS サービスイベントを に変換できます。

> **Note**
>
> 次の手順では、ソースイベントを構造化モード CloudEventsに変換する方法について説明します。 CloudEvents 仕様では、構造化モードメッセージは、イベント全体 (属性とデータ) がイベントのペイロードにエンコードされるメッセージです。

CloudEvents 仕様の詳細については、cloudevents.io を参照してください。

コンソールを使用して AWS イベントを CloudEvents 形式に変換するには

イベントをターゲットへの配信前の CloudEvents 形式に変換するには、まずイベントバスルールを作成します。ルールの定義の一環として、指定したターゲットに送信する前に、入力トランスフォーマーを使用して EventBridge 変換イベントを設定します。

1. 「???」のステップに従います。
2. ??? ステップでは、ターゲットタイプとして API 送信先を選択するように求められた場合：

   a. EventBridge API 送信先 を選択します。

   b. 次のいずれかを行います。

      - 既存の API 送信先を使用 を選択し、既存の API 送信先を選択します。
      - 新しい API 送信先の作成を選択し、新しい API 送信先を定義するために必要な設定を指定します。

        必要な設定の指定の詳細については、「」を参照してください???。

   c. CloudEvents イベントに必要な Content-Type ヘッダーパラメータを指定します。

      - 「ヘッダーパラメータ」で「ヘッダーパラメータを追加」を選択します。
      - キー には、 を指定しますContent-Type。

        値 には、 を指定しますapplication/cloudevents+json; charset=UTF-8。
3. ターゲットの実行ロールを指定します。
4. ソースイベントデータを CloudEvents 形式に変換する入力トランスフォーマーを定義します。

   a. 「追加設定」の「ターゲット入力の設定」で、「入力トランスフォーマー」を選択します。

   次に、入力トランスフォーマーの設定 を選択します。

   b. ターゲット入力トランスフォーマー で、入力パス を指定します。

   以下の入力パスでは、region 属性は CloudEvents 形式のカスタム拡張属性です。そのため、 CloudEvents 仕様に準拠する必要はありません。

   CloudEvents では、コア仕様で定義されていない拡張属性を使用および作成できます。既知の拡張属性のリストを含む詳細については、「」のCloudEvents 仕様ドキュメントのCloudEvents 「拡張属性」を参照してください GitHub。

```
{
```

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

c. テンプレート には、ソースイベントデータを CloudEvents 形式に変換するテンプレートを入力します。

以下のテンプレートでは、入力パスの region 属性regionは CloudEvents 仕様の拡張属性であるため、 は厳密に必須ではありません。

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

5. 手順のステップ に従ってルールの作成を完了します。

API 送信先パートナー

以下の AWS パートナーから提供された情報を使用して、サービスまたはアプリケーションの API 送信先と接続を設定します。

Cisco クラウドオブザーバビリティ

API 送信先の呼び出しエンドポイント URL:

```
https://tenantName.observe.appdynamics.com/rest/awsevents/aws-eventbridge-integration/endpoint
```

サポートされている認証タイプ:

OAuth クライアント認証情報

401 または 407 レスポンスが返されると OAuth トークンが更新されます

必要な追加の認証パラメータ:

Cisco AppDynamics クライアント ID とクライアントシークレット

OAuth エンドポイント：

`https://tenantName.observe.appdynamics.com/auth/tenantId/default/oauth2/token`

次の OAuth キーと値のペアのパラメータ：

タイプ	キー	値
本文フィールド	grant_type	client_credentials
[Header] (ヘッダー)	Content-Type	application/x-www-form-urlencoded; charset=utf-8

Cisco AppDynamics のドキュメント：

AWS イベントの取り込み

一般的に使用される API オペレーション:

該当しない

追加情報:

パートナーの送信先ドロップダウンメニューから Cisco AppDynamics を選択すると、API コールに必要なヘッダーと本文のキーと値のペアなど、必要な OAuth 情報が事前に入力されます。

詳細については、Cisco ドキュメントAWS の「イベントの取り込み AppDynamics」を参照してください。

コンフルエント

API 送信先の呼び出しエンドポイント URL:

通常、次の形式です。

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

詳細については、Confluent ドキュメントの「REST エンドポイントアドレスとクラスター ID を検索する」を参照してください。

サポートされている認証タイプ:

Basic (ベーシック)

必要な追加の認証パラメータ:

該当しない

Confluent のドキュメント :

レコードの生成

Apache Kafka 用 Confluent REST Proxy

一般的に使用される API オペレーション:

POST

追加情報:

イベントデータをエンドポイントが処理できるメッセージに変換するには、ターゲット入力トランスフォーマー を作成します。

- Kafka パーティショニングキーを指定せずにレコードを生成するには、入力トランスフォーマーに次のテンプレートを使用します。入力パスは必要ありません。

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

- Kafka パーティショニングキーとしてイベントデータフィールドを使用してレコードを生成するには、以下の入力パスとテンプレートの例に従います。この例では、 orderIdフィールドの入力パスを定義し、そのフィールドをパーティションキーとして指定します。

  まず、イベントデータフィールドの入力パスを定義します。

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

次に、入力トランスフォーマーテンプレートを使用して、データフィールドをパーティションキーとして指定します。

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

Coralogix

API 送信先呼び出しエンドポイント URL

エンドポイントの完全な一覧については、Coralogix API リファレンス を参照してください。

サポートされている認証タイプ

API キー

必要な追加の認証パラメータ

ヘッダー "x-amz-event-bridge-access-key"、値は Coralogix API キー

「Coralogix ドキュメント」

Amazon EventBridge 認証

一般的に使用される API オペレーション

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

シンガポール: https://ingress.coralogixsg.com/aws/event-bridge

アイルランド: https://ingress.coralogix.com/aws/event-bridge

アイルランド: https://ingress.coralogix.com/aws/event-bridge

インド: https://ingress.coralogix.in/aws/event-bridge

追加情報

イベントは、`applicationName=[AWS Account]` と `subsystemName=[event.source]` を使用してログエントリとして保存されます。

Datadog

API 送信先呼び出しエンドポイント URL

エンドポイントの完全な一覧については、Datadog API リファレンス を参照してください。

サポートされている認証タイプ

API キー

必要な追加の認証パラメータ

なし

「Datadog ドキュメント」

認証

一般的に使用される API オペレーション

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

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

追加情報

エンドポイント URL は、Datadog 組織の場所によって異なります。組織の正しい URL については、ドキュメント を参照してください。

Freshworks

API 送信先呼び出しエンドポイント URL

エンドポイントのリストについては、https://developers.freshworks.com/documentation/ を参照してください。

サポートされている認証タイプ

Basic、API キー

必要な追加の認証パラメータ

該当しない

「Freshworks ドキュメント」

認証

一般的に使用される API オペレーション

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

追加情報

なし

MongoDB

API 送信先呼び出しエンドポイント URL

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

サポートされている認証タイプ

API キー

E メール/パスワード

カスタム JWT 認証

必要な追加の認証パラメータ

なし

「MongoDB ドキュメント」

Atlas Data API

エンドポイント

カスタム HTTP エンドポイント

認証

一般的に使用される API オペレーション

なし

追加情報

なし

New Relic

API 送信先呼び出しエンドポイント URL

詳細については、「EU および US リージョンのデータセンター」をご覧ください。

イベント

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

メトリクス

US– https://metric-api.newrelic.com/metric/v1

EU– https://metric-api.eu.newrelic.com/metric/v1

ログ

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

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

トレース

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

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

サポートされている認証タイプ

API キー

「New Relic ドキュメント」

メトリクス API

- イベント API
- ログ API
- トレース API

一般的に使用される API オペレーション

- メトリクス API
- イベント API
- ログ API
- トレース API

追加情報

- メトリクス API の制限
- イベント API の制限
- ログ API の制限
- トレース API の制限

Operata

API 送信先の呼び出しエンドポイント URL:

- https://api.operata.io/v2/aws/events/contact-record

サポートされている認証タイプ:

- Basic (ベーシック)

必要な追加の認証パラメータ:

- なし

Operata のドキュメント:

- API トークンを作成、表示、変更、取り消すにはどうすればよいですか?
- Amazon EventBridge Scheduler Pipes を使用した Operata AWS の統合

一般的に使用される API オペレーション:

```
POST https://api.operata.io/v2/aws/events/contact-record
```

追加情報:

username は Operata グループ ID で、パスワードは API トークンです。

Salesforce

API 送信先呼び出しエンドポイント URL

Sobject – https:// *myDomainName*.my.salesforce.com/services/data/*versionNumber*/sobjects /*SobjectEndpoint*/*

カスタムプラットフォームイベント – https://*myDomainName*.my.salesforce.com/services/data /*versionNumber*/sobjects/*customPlatformEndpoint*/*

エンドポイントの完全な一覧については、Salesforce API リファレンス を参照してください。

サポートされている認証タイプ

OAuth クライアント認証情報

OAUTH トークンは、401 または 407 の応答が返されたときに更新されます。

必要な追加の認証パラメータ

Salesforce Connected App は、クライアント ID とクライアントシークレットを提供します。

次の認可エンドポイントの 1 つです。

- 本番 – https://*MyDomainName*.my.salesforce.com./services/oauth2/token
- 拡張ドメインのないサンドボックス – https://*MyDomainName*-- *SandboxName*.my.salesforce.com/services /oauth2/token
- 拡張ドメインを持つサンドボックス – https://*MyDomainName*--*SandboxName*.sandbox.my.salesforce.com/services/oauth2/token

次のキーと値のペア :

キー	値
grant_type	client_credentials

「Salesforce ドキュメント」

REST API 開発者ガイド

一般的に使用される API オペレーション

オブジェクトメタデータの使用

レコードを使用する

追加情報

EventBridge コンソールを使用して への接続を作成する方法Salesforce、API 送信先、および に情報をルーティングするルールを説明するチュートリアルについてはSalesforce、「」を参照してください???。

Slack

API 送信先呼び出しエンドポイント URL

エンドポイントおよびその他のリソースのリストについては、「Slack ウェブ API を使う」を参照してください。

サポートされている認証タイプ

OAuth 2.0

OAUTH トークンは、401 または 407 の応答が返されたときに更新されます。

Slack アプリケーションを作成してワークスペースにインストールすると、OAuth ベアラトークンがユーザーに代わって作成され、API 宛先接続による呼び出しの認証に使用されます。

必要な追加の認証パラメータ

該当しない

「Slack ドキュメント」

基本的なアプリのセットアップ

OAuth でインストールする

メッセージを取得する

メッセージの送信

着信 Webhook を使用したメッセージの送信

一般的に使用される API オペレーション

https://slack.com/api/chat.postMessage

追加情報

EventBridge ルールを設定する場合、強調表示する設定が 2 つあります。

- コンテンツタイプを「application/json; charset=utf-8」として定義するヘッダーパラメータを含めます。
- 入力トランスフォーマーを使用して、入力イベントを Slack API の期待される出力にマッピングします。つまり、Slack API に送信されるペイロードに「チャネル」と「テキスト」のキー/値のペアがあることを確認します。

Shopify

API 送信先呼び出しエンドポイント URL

エンドポイントおよびその他のリソースとメソッドのリストについては、「エンドポイントとリクエスト」を参照してください。

サポートされている認証タイプ

OAuth、API キー

> **Note**
>
> OAUTH トークンは、401 または 407 の応答が返されたときに更新されます。

必要な追加の認証パラメータ

該当しない

「Shopify ドキュメント」

認証と認可の概要

一般的に使用される API オペレーション

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

追加情報

アプリケーションの作成

Amazon EventBridge Webhook 配信

Shopify 管理者のカスタムアプリケーションのアクセストークン

Product

Shopify 管理者 API

Splunk

API 送信先呼び出しエンドポイント URL

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

サポートされている認証タイプ

Basic、API キー

必要な追加の認証パラメータ

なし

「Splunk ドキュメント」

どちらの認証タイプでも、HEC トークン ID が必要です。詳細については、「Set up and use HTTP Event Collector in Splunk Web」を参照してください。

一般的に使用される API オペレーション

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

追加情報

API キー – のエンドポイントを設定する場合 EventBridge、API キー名は「承認」で、値は Splunk HEC トークン ID です。

基本 (ユーザー名/パスワード) – のエンドポイントを設定する場合 EventBridge、ユーザー名は「Splunk」で、パスワードは Splunk HEC トークン ID です。

Sumo Logic

API 送信先呼び出しエンドポイント URL

HTTP ログとメトリクスソースのエンドポイント URL は、ユーザーごとに異なります。詳細については、「HTTP ログおよびメトリクスのソース」を参照してください。

サポートされている認証タイプ

一意のキーが URL にベイクされているため、Sumo Logic は HTTP ソースでの認証は必要ありません。そのため、この URL は機密情報として扱う必要があります。

EventBridge API 送信先を設定する場合、認証タイプが必要です。この要件を満たすには、[API Key] を選択し、キー名を「dummy-key」、キー値を「dummy-value」にします。

必要な追加の認証パラメータ

該当しない

「Sumo Logic ドキュメント」

Sumo Logic は、多くの AWS のサービスからログとメトリクスを収集するためにホストされたソースをすでに構築しており、ウェブサイトの情報を使用してそれらのソースを操作できます。詳細については、「Amazon Web Services」を参照してください。

アプリケーションからカスタムイベントを生成し、ログまたはメトリクスSumo Logicとして に送信する場合は、 EventBridge API 送信先と Sumo Logic HTTP ログおよびメトリクスソースエンドポイントを使用します。

- サインアップして無料の Sumo Logic インスタンスを作成するには、「今すぐ無料トライアルを始めよう」を参照してください。
- Sumo Logic の使用の詳細については、「 HTTP ログおよびメトリクスソース」を参照してください。

一般的に使用される API オペレーション

POST https://endpoint4.collection.us2.sumologic.com/receiver/v1/http/*UNIQUE_ID_PER_COLLECTOR*

追加情報

なし

TriggerMesh

API 送信先呼び出しエンドポイント URL

「HTTP のイベントソース」のトピックの情報を使用して、エンドポイント URL を策定します。エンドポイント URL には、イベントソース名とユーザー名前空間が次の形式で含まれます。

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

エンドポイントへのリクエストには、Basic 認証パラメータを含めます。

サポートされている認証タイプ

Basic

必要な追加の認証パラメータ

なし

「TriggerMesh ドキュメント」

HTTP のイベントソース

一般的に使用される API オペレーション

該当しない

追加情報

なし

Zendesk

API 送信先呼び出しエンドポイント URL

https://developer.zendesk.com/rest_api/docs/support/tickets

サポートされている認証タイプ

Basic、API キー

必要な追加の認証パラメータ

なし

「Zendesk ドキュメント」

セキュリティと認証

一般的に使用される API オペレーション

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

追加情報

API リクエスト EventBridge は、Zendesk API の制限に対してカウントされます。現在のプランの Zendesk 制限については、「使用制限」を参照してください。

アカウントやデータをより安全に保護するには、基本的なサインイン認証情報による認証ではなく、API キーを使用することをお勧めします。

Amazon API Gateway の Amazon EventBridge ターゲット

Amazon API Gateway を使用すると、API の作成、発行、管理、モニタリング、保護が可能です。Amazon EventBridge は、API Gateway エンドポイントへのイベントの送信をサポートしています。ターゲットとして API Gateway エンドポイントを指定すると、ターゲットに送信される各イベントは、エンドポイントに送信されたリクエストにマッピングされます。

> **Important**
>
> EventBridge は、API Gateway エッジ最適化エンドポイントとリージョンエンドポイントをターゲットとして使用します。プライベートエンドポイントは現在サポートされていません。エンドポイントの詳細については、「https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-api-endpoint-types.html」を参照してください。

API Gateway ターゲットは、次のユースケースで使用できます。

- AWS またはサードパーティーのイベントに基づいて、API Gateway でホストされている顧客指定の API を呼び出すには。
- スケジュールに基づいて定期的にエンドポイントを呼び出す。

EventBridge JSON イベント情報は、HTTP リクエストの本文としてエンドポイントに送信されます。ターゲットの `HttpParameters` フィールドでは、次のように他のリクエスト属性を指定できます。

- `PathParameterValues` は、エンドポイント ARN の任意のパス変数に順次対応する値を一覧で示します。例えば、`"arn:aws:execute-api:us-east-1:112233445566:myapi/dev/POST/pets/*/*"` などです。
- `QueryStringParameters` は、呼び出されたエンドポイントに が EventBridge 追加するクエリ文字列パラメータを表します。
- `HeaderParameters` は、リクエストに追加する HTTP ヘッダーを定義します。

> **Note**
>
> セキュリティを考慮して、以下の HTTP ヘッダーキーは許可されていません。
>
> - `X-Amz` または `X-Amzn` のプレフィックスが付いているキー
> - `Authorization`
> - `Connection`
> - `Content-Encoding`
> - `Content-Length`
> - `Host`
> - `Max-Forwards`
> - `TE`
> - `Transfer-Encoding`
> - `Trailer`
> - `Upgrade`
> - `Via`
> - `WWW-Authenticate`
> - `X-Forwarded-For`

動的パラメータ

API Gateway ターゲットを呼び出すとき、ターゲットに送信されるイベントにデータを動的に追加することができます。詳細については、「the section called “ターゲットパラメータ”」を参照してください。

呼び出しの再試行

すべてのターゲットと同様に、 は失敗した呼び出しを EventBridge 再試行します。API Gateway の場合、 は 5xx または 429 HTTP ステータスコードで送信されたレスポンスを最大 24 時間、エクスポネンシャルバックオフとジッター で EventBridge 再試行します。その後、 は Amazon . CloudWatch EventBridge doesn't retry other 4xx HTTP errors の`FailedInvocations`メトリクスを EventBridge 公開します。

タイムアウト

EventBridge ルール API Gateway リクエストの最大クライアント実行タイムアウトは 5 秒である必要があります。API Gateway の応答に 5 秒以上かかる場合、 はリクエストを EventBridge タイムアウトしてから再試行します。

EventBridge Pipes API Gateway リクエストの最大タイムアウトは 29 秒で、API Gateway の最大タイムアウトです。

AWS AppSync Amazon の ターゲット EventBridge

AWS AppSync を使用すると、開発者は、安全でサーバーレスで高性能な GraphQL および Pub/Sub APIs を使用して、アプリケーションとサービスをデータやイベントに接続できます。を使用すると AWS AppSync、GraphQL ミューテーションを使用してアプリケーションにリアルタイムのデータ更新を発行できます。 EventBridge は、一致したイベントに対して有効な GraphQL ミューテーションオペレーションの呼び出しをサポートします。 AWS AppSync API ミューテーションをターゲットとして指定すると、 はミューテーションオペレーションを介してイベント AWS AppSync を処理し、ミューテーションにリンクされたサブスクリプションをトリガーできます。

> **Note**
>
> EventBridge は AWS AppSync パブリック GraphQL APIsをサポートしています。
> EventBridge は現在 AWS AppSync 、プライベート APIsをサポートしていません。

AWS AppSync GraphQL API ターゲットは、次のユースケースに使用できます。

- 設定したデータソースへのイベントデータのプッシュ、変換、保存
- 接続しているアプリケーションクライアントへの通知のリアルタイム送信

> **Note**
>
> AWS AppSync ターゲットは、認証タイプ を使用した AWS AppSync GraphQL APIs の呼び出しのみをサポートします。 AWS_IAM

AWS AppSync GraphQL APIsAWS AppSync デベロッパーガイド」のGraphQL と AWS AppSync アーキテクチャ」を参照してください。

コンソールを使用して EventBridge ルールの AWS AppSync ターゲットを指定するには

1. ルールを作成または編集します。
2. [ターゲット] で、[AWS のサービス]、[AWS AppSync] の順に選択してターゲットを指定します。
3. 解析して実行するミューテーション操作を、選択セットと共に指定します。
   - AWS AppSync API を選択し、次に呼び出す GraphQL API ミューテーションを選択します。
   - [パラメータと選択セットの設定] で、選択セットの作成にキーと値のマッピングを使用するか、入力トランスフォーマーを使用するかを選択します。

     Key-value mapping

     キーと値のマッピングを使用して選択セットを作成するには:

     - API パラメータの変数を指定します。各変数は、静的な値でも、イベントペイロードへの動的な JSON パス式でもかまいません。
     - [選択セット] で、レスポンスに含める変数を選択します。

     Input transformer

     入力トランスフォーマーを使用して選択セットを作成するには:

     - 使用する変数を定義する入力パスを指定します。
     - ターゲットに渡す情報を定義およびフォーマットする入力テンプレートを指定します。

     詳細については、「???」を参照してください。
4. [実行ロール] で、新しいロールを作成するか、既存のロールを使用するかを選択します。
5. ルールの作成または編集を完了します。

例: Amazon の AWS AppSync ターゲット EventBridge

次の例では、 EventBridge ルールの AWS AppSync ターゲットを指定する方法について説明します。これには、配信するイベントをフォーマットするための入力変換の定義が含まれます。

次のスキーマで定義された AWS AppSync GraphQL APIEc2EventAPI、 があるとします。

```
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"])
}
```

この API を使用するアプリケーションクライアントは、pushEvent ミューテーションによってトリガーされる subscribeToEvent サブスクリプションにサブスクライブできます。

pushEvent ミューテーションを介して AppSync API にイベントを送信するターゲットを持つ EventBridge ルールを作成できます。ミューテーションを呼び出すと、サブスクライブしているすべてのクライアントがイベントを受信します。

この API を EventBridge ルールのターゲットとして指定するには、以下を実行します。

1. ルールターゲットの Amazon リソースネーム (ARN) を Ec2EventAPI API の GraphQL エンドポイント ARN に設定します。
2. GraphQL ミューテーション操作をターゲットパラメータとして指定します。

```
mutation CreatePushEvent($id: ID!, $statusCode: String, $instanceId: String) {
  pushEvent(id: $input, statusCode: $statusCode, instanceId: $instanceId) {
    id
```

```
    statusCode
    instanceId
  }
}
```

ミューテーション選択セットには、GraphQL サブスクリプションでサブスクライブするすべてのフィールドを含める必要があります。

3. 入力トランスフォーマーを設定して、一致したイベントのデータをどのように操作で使用するかを指定します。

次の “EC2 Instance Launch Successful” サンプルイベントを選択したとします。

```
{
  "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."
  }
}
```

ターゲット入力トランスフォーマーの入力パスを使用して、テンプレートで使用する以下の変数を定義できます。

```
{
  "id": "$.id",
  "statusCode": "$.detail.StatusCode",
  "EC2InstanceId": "$.detail.EC2InstanceId"
}
```

入力トランスフォーマーテンプレートを作成して、 AWS AppSync ミューテーションオペレーションに EventBridge 渡す変数を定義します。テンプレートは JSON として評価される必要があります。入力パスを指定すると、次のテンプレートを作成できます。

```
{
  "id": <id>,
  "statusCode": <statusCode>,
  "instanceId": <EC2InstanceId>
}
```

HTTP エンドポイントターゲットの接続

接続は、特定の HTTP エンドポイントへの接続 EventBridge に使用する の認証方法と認証情報を定義します。認証設定を構成して接続を作成すると、 にシークレットが作成され AWS Secrets Manager 、認証情報が安全に保存されます。また、HTTP エンドポイントターゲットに応じて、接続に含めるパラメータを追加することもできます。

次の接続を使用します。

- API 送信先

  API 送信先を作成するときは、その送信先に使用する接続を指定します。アカウントから既存の接続を選択するか、API 送信先の作成時に接続を作成できます。

接続の承認方法

EventBridge 接続は、次の認証方法をサポートしています。

- Basic (ベーシック)

- API キー

  基本認証と API キー認証の場合、 は必要な認証ヘッダー EventBridge を入力します。

- OAuth

  OAuth 認証の場合、 EventBridge はクライアント ID とシークレットをアクセストークンと交換し、安全に管理します。

  OAUTH トークンは、401 または 407 の応答が返されたときに更新されます。

接続を作成するとき、エンドポイントでの認証に必要なヘッダー、本文、およびクエリパラメータを含めることもできます。エンドポイントの認証が同じであれば、複数の HTTP エンドポイントに同じ接続を使用できます。

接続を作成して承認パラメータを追加すると、 は にシークレット EventBridge を作成します AWS Secrets Manager。Secrets Manager のシークレットの保存およびアクセスの両方に要する費用は、API 送信先を使用する料金に含まれています。API 送信先でシークレットを使用するためのベストプラクティスの詳細については、AWS::Events::ApiDestinationCloudFormation 「 ユーザーガイド」の「」を参照してください。

> **Note**
>
> 接続を正常に作成または更新するには、Secrets Manager を使用するアクセス許可を持つアカウントを使用する必要があります。必要なアクセス許可は、AmazonEventBridgeFullAccess ポリシー ポリシーに含まれています。同じ許可が、接続用のアカウントで作成された、サービスにリンクされたロールに付与されます。

HTTP エンドポイントターゲットの接続の作成

EventBridge コンソールを使用して HTTP エンドポイントで使用する接続を作成するには

1. コンソール を管理 EventBridge および開くアクセス許可を持つアカウント AWS を使用して、にログインします。 EventBridge
2. 左のナビゲーションペインで、[API destinations] (API 送信先) を選択します。
3. [API destinations] (API 送信先) テーブルまで下にスクロールし、[Connections] (接続) タブを選択します。
4. [Create connection] (接続の作成) を選択します。

5. [Create connection] (接続の作成) ページで、接続の [Connection name] (接続名) を入力します。
6. 接続の説明を入力します。
7. [Authorization type] (認証タイプ) で、この接続を使用する API 送信先に指定された HTTP エンドポイントへの接続を許可するために使用する認証のタイプを選択します。次のいずれかを行います。
   - [Basic (Username/Password)] (ベーシック (ユーザー名/パスワード)) を選択し、HTTP エンドポイントでの認証に使用する [Username] (ユーザー名) と [Password] (パスワード) を入力します。
   - [OAuth Client Credentials] (OAuth クライアント認証) を選択し、エンドポイントでの認証に使用する [Authorization endpoint] (認証エンドポイント) 、[HTTP method] (HTTP メソッド) 、[Client ID] (クライアント ID)、[Client secret] (クライアントシークレット) を入力します。

     [OAuth Http Parameters] (OAuth HTTP パラメータ) で、認証エンドポイントでの認証に必要なパラメータがあれば追加します。ドロップダウンリストからパラメータを選択し、[Key] (キー) と [Value] (値) を入力します。追加のパラメータを含めるには、[Add parameter] (パラメータの追加) を選択します。

     [Invocation Http Parameters] (呼び出し HTTP パラメータ) で、認可リクエストに含めるパラメータを追加します。パラメータを追加するには、ドロップダウンリストからパラメータを選択し、[Key] (キー) と [Value] (値) を入力します。追加のパラメータを含めるには、[Add parameter] (パラメータの追加) を選択します。
   - API キーを選択し、API キー認証に使用する API キー名とそれに対応する値を入力します。

     [Invocation Http Parameters] (呼び出し HTTP パラメータ) で、認可リクエストに含めるパラメータを追加します。パラメータを追加するには、ドロップダウンリストからパラメータを選択し、[Key] (キー) と [Value] (値) を入力します。追加のパラメータを含めるには、[Add parameter] (パラメータの追加) を選択します。
8. [作成] を選択します。

EventBridge コンソールを使用した接続の編集

既存の接続を編集できます。

EventBridge コンソールを使用して接続を編集するには

1. コンソール を管理 EventBridge および開くアクセス許可を持つアカウント AWS を使用して、にログインします。 EventBridge
2. 左のナビゲーションペインで、[API destinations] (API 送信先) を選択します。
3. [API destinations] (API 送信先) テーブルまで下にスクロールし、[Connections] (接続) タブを選択します。
4. [Connections] (接続) テーブルで、編集する接続を選択します。
5. [Connection details] (接続の詳細) ページで、[Edit] (編集) を選択します。
6. 接続の値を更新し、[Update] (更新) を選択します。

EventBridge コンソールを使用した接続の認証解除

接続の認証を解除すると、すべての認証パラメータが削除されます。認証パラメータを削除すると、接続からシークレットが削除されるため、新しい接続を作成せずに再利用できます。

> **Note**
>
> 認証解除された接続を使用する HTTP エンドポイントを更新して、別の接続を使用して HTTP エンドポイントにリクエストを正常に送信する必要があります。

接続の認証を解除するには

1. コンソール を管理 EventBridge および開くアクセス許可を持つアカウント AWS を使用して、にログインします。 EventBridge
2. 左のナビゲーションペインで、[API destinations] (API 送信先) を選択します。
3. [API destinations] (API 送信先) テーブルまで下にスクロールし、[Connections] (接続) タブを選択します。
4. [Connections] (接続) テーブルで、接続を選択します。
5. [Connection details] (接続の詳細) ページで、[De-authorize] (認証解除) を選択します。
6. [Deauthorize connection?] (接続認証を解除しますか?) ダイアログボックスで、接続の名前を入力して [De-authorize] (認証解除) を選択します。

プロセスが完了するまで、接続のステータスは [De-authorizing] (認証解除中) になります。その後、ステータスは [De-authorized] (認証解除) に変わります。これで、接続を編集して新しい認証パラメータを追加することができます。

AWS アカウント間での Amazon EventBridge イベントの送受信

AWS アカウント内のイベントバス間でイベントを送受信 EventBridge するように を設定できます。アカウント間でイベントを送受信 EventBridge するように を設定する場合、 AWS アカウント内のイベントバスとの間でイベントを送受信できるアカウントを指定できます。また、イベントバスに関連する特定のルールからのイベントや、特定のソースからのイベントを許可または拒否することもできます。詳細については、「Amazon EventBridge リソースポリシーによるクロスアカウントアクセスの簡素化」を参照してください。

> **Note**
>
> を使用する場合は AWS Organizations、組織を指定し、その組織内のすべてのアカウントへのアクセスを許可できます。さらに、別のアカウントにイベントを送信する場合、送信イベントバスには IAM ロールがアタッチされている必要があります。詳細については、AWS Organizationsユーザーガイドの AWS Organizations とは を参照してください。

> **Note**
>
> Incident Manager の対応プランをターゲットとして使用している場合、アカウントで共有されているすべての対応プランがデフォルトで利用できます。

送信先リージョンがサポートされているクロスリージョン送信先リージョンである限り、すべてのリージョンの同じリージョン内の AWS アカウントと異なるリージョンのアカウントの間でイベントバスを送受信できます。

別のアカウントのイベントバスとの間でイベント EventBridge を送受信するように を設定する手順は次のとおりです。

- レシーバーアカウントで、イベントバスのアクセス許可を編集して、指定された AWS アカウント、組織、またはすべての AWS アカウントがレシーバーアカウントにイベントを送信できるようにします。

- 送信側アカウントで、受信側アカウントのイベントバスをターゲットとする 1 つ以上のルールを設定します。

  送信者アカウントが AWS Organization からイベントを送信するアクセス許可を継承する場合、送信者アカウントには、受信側アカウントにイベントを送信できるようにするポリシーを持つ IAM ロールも必要です。を使用してレシーバーアカウントのイベントバスをターゲットとするルール AWS Management Console を作成すると、ロールが自動的に作成されます。を使用する場合は AWS CLI、ロールを手動で作成する必要があります。
- 受信側アカウントで、送信側アカウントからのイベントに一致する 1 つ以上のルールを設定します。

1 つのアカウントから別のアカウントに送信されたイベントは、カスタムイベントとして送信側アカウントに課金されます。受信側アカウントには課金されません。詳細については、「Amazon の EventBridge 料金」を参照してください。

受信側アカウントで、送信側アカウントから受信したイベントを第三のアカウントに送信するルールが設定されていても、それらのイベントは第三のアカウントには送信されません。

同じアカウントに 3 つのイベントバスがあり、1 番目のイベントバスにルールを設定して 2 番目のイベントバスから 3 番目のイベントバスにイベントを転送する場合、それらのイベントは 3 番目のイベントバスに送信されません。

次の動画では、アカウント間のイベントのルーティングについて説明します。他の AWS アカウントのバスへのイベントのルーティング

他の AWS アカウントからのイベントを許可するアクセス許可を付与する

他のアカウントや組織からイベントを受信するには、まず、イベントを受信するイベントバスに対する許可を編集する必要があります。デフォルトのイベントバスは、 AWS サービス、他の認可された AWS アカウント、および `PutEvents` 呼び出しからのイベントを受け入れます。イベントバスに対するアクセス許可は、イベントバスにアタッチされたリソースベースのポリシーを使用して付与または拒否されます。ポリシーでは、アカウント ID を使用して他の AWS アカウントにアクセス許可を付与することも、 AWS 組織 ID を使用して組織にアクセス許可を付与することもできます。ポリシーの例など、イベントバスに対するアクセス許可についての詳細は、Amazon EventBridge イベントバスのアクセス許可 を参照してください。

> **Note**
>
> EventBridge では、すべての新しいクロスアカウントイベントバスターゲットが IAM ロールを追加する必要があるようになりました。これは、2023 年 3 月 2 日以降に作成されたイベントバスターゲットにのみ適用されます。この日より前に IAM ロールなしで作成されたアプリケーションは影響を受けません。ただし、IAM ロールを追加して、ユーザーに別のアカウントのリソースへのアクセスを許可することをお勧めします。これにより、サービスコントロールポリシー (SCP) を使用する組織の境界が適用され、組織内のアカウントからイベントを送受信できるユーザーを特定できるようになります。

> **Important**
>
> すべての AWS アカウントからイベントを受信する場合は、他のユーザーから受信するイベントのみに一致するルールを作成するように注意してください。より安全なルールを作成するには、各ルールのイベントパターンに、Account フィールドと、イベントの受信元の 1 つ以上のアカウントのアカウント ID が必ず含まれるようにします。アカウントフィールドを含むイベントパターンを持つルールは、Account フィールドにリストされていないアカウントから送信されたイベントと一致しません。詳細については、「Amazon EventBridge イベント」を参照してください。

AWS アカウント間のイベントに関するルール

アカウントが他のアカウントのイベントバスからイベントを受信するように設定されている場合は AWS 、それらのイベントに一致するルールを記述できます。他のアカウントのイベントバスから受信しているイベントと一致するように、ルールのイベントパターンを設定します。

ルールのイベントパターンで account を指定した場合を除き、アカウントのルール (新規と既存の両方) のうち、他のアカウントのイベントバスから受信しているイベントに一致するすべてのルールがトリガーされます。別のアカウントのイベントバスからイベントを受信し、自分のアカウントから生成されたときにそのイベントパターンのみでルールがトリガーされるようにするには、account を追加し、自分のアカウント ID をルールのイベントパターンに指定する必要があります。

すべての AWS アカウントのイベントバスからのイベントを受け入れるように AWS アカウントを設定する場合は、アカウント内のすべての EventBridge ルールaccountに を追加することを強くお勧めします。これにより、アカウントのルールが不明な AWS アカウントからのイベントでトリガーさ

れるのを防ぐことができます。ルールで account フィールドを指定するときは、1 つ以上の AWS アカウントのアカウント ID をフィールドで指定できます。

アクセス許可を付与した AWS アカウントのイベントバスから一致するイベントに対してルールをトリガーするには、ルールの account フィールドに * を指定しないでください。これにより、* はイベントの account フィールドに表示されないため、どのイベントとも一致しません。代わりに、ルールから account フィールドを省略します。

AWS アカウント間でイベントを送信するルールの作成

別のアカウント内のイベントバスをターゲットとして指定することは、ルール作成の一部です。

コンソールを使用して別の AWS アカウントにイベントを送信するルールを作成するには

1. 「???」のステップに従います。
2. 「???」ステップで、ターゲットタイプを選択するように求めるプロンプトが表示された場合:
    a. EventBridge イベントバス を選択します。
    b. [別のアカウントまたはリージョン内のイベントバス] を選択します。
    c. [ターゲットとしてのイベントバス] に、使用するイベントバスの ARN を入力します。
3. ステップに従ってルールの作成を完了します。

AWS リージョン間での Amazon EventBridge イベントの送受信

AWS リージョン間でイベントを送受信 EventBridge するように を設定できます。また、特定のリージョンからのイベント、イベントバスに関連付けられた特定のルール、または特定のソースからのイベントを許可または拒否することもできます。詳細については、「Amazon でのクロスリージョンイベントルーティングの紹介 EventBridge」を参照してください。

以下のリージョンは、送信先リージョンとしてサポートされています。

- 米国東部 (バージニア北部)
- 米国東部 (オハイオ)
- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- アフリカ (ケープタウン)
- アジアパシフィック (香港)

- アジアパシフィック (東京)
- アジアパシフィック (ソウル)
- アジアパシフィック (大阪)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (ハイデラバード)
- アジアパシフィック (シンガポール)
- アジアパシフィック (ジャカルタ)
- アジアパシフィック (シドニー)
- アジアパシフィック (メルボルン)
- カナダ (中部)
- カナダ西部 (カルガリー)
- 欧州 (フランクフルト)
- 欧州 (スペイン)
- 欧州 (チューリッヒ)
- 欧州 (ストックホルム)
- 欧州 (ミラノ)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (パリ)
- イスラエル (テルアビブ)
- 中東 (アラブ首長国連邦)
- 中東 (バーレーン)
- 南米 (サンパウロ)

次のビデオでは AWS CloudFormation、https://console.aws.amazon.com/events/、および を使用してリージョン間でイベントをルーティングする AWS Serverless Application Model: クロスリージョンイベントルーティングについて説明します。

別の AWS リージョンにイベントを送信するルールの作成

別の AWS リージョンのイベントバスをターゲットとして指定することは、ルールの作成の一部です。

コンソールを使用して別の AWS アカウントにイベントを送信するルールを作成するには

1. 「???」のステップに従います。
2. 「???」ステップで、ターゲットタイプを選択するように求めるプロンプトが表示された場合:
    a. EventBridge イベントバス を選択します。
    b. [別のアカウントまたはリージョン内のイベントバス] を選択します。
    c. [ターゲットとしてのイベントバス] に、使用するイベントバスの ARN を入力します。
3. ステップに従ってルールの作成を完了します。

同じアカウントとリージョンの EventBridge イベントバス間で Amazon イベントを送受信する

同じ AWS アカウントとリージョンのイベントバス間でイベントを送受信 EventBridge するようにを設定できます。

イベントバス間でイベントを送受信 EventBridge するように を設定する場合、送信者イベントバスの IAM ロールを使用して、送信者イベントバスにイベントをレシーバーイベントバスに送信するアクセス許可を付与します。送信側イベントバスからイベントを受信する許可を受信側イベントバスに付与するには、受信側イベントバスでリソースベースのポリシーを使用します。また、特定のイベントバス、イベントバスに関連する特定のルール、または特定のソースからのイベントを許可または拒否することもできます。ポリシーの例など、イベントバスに対するアクセス許可についての詳細は、「Amazon EventBridge イベントバスのアクセス許可」を参照してください。

アカウント内のイベントバス間でイベントを EventBridge 送受信するように を設定する手順は次のとおりです。

- 既存の IAM ロールを使用するには、送信側イベントバスのアクセス許可を受信側イベントバスに付与するか、受信側イベントバスのアクセス許可を送信側イベントバスに付与する必要があります。
- 送信側イベントバスで、受信側イベントバスをターゲットとする 1 つ以上のルールを設定し、IAM ロールを作成します。ロールにアタッチする必要があるポリシーの例については、「???」を参照してください。

- 受信側イベントバスで、他のイベントバスからのイベントの受け渡しを許可するアクセス許可を編集します。
- 受信側イベントで、送信側イベントからのイベントに一致する 1 つ以上のルールを設定します。

> **Note**
>
> EventBridge は、送信者イベントバスから受信したイベントを 3 番目のイベントバスにルーティングできません。

1 つのイベントバスから別のイベントバスに送信されたイベントは、カスタムイベントとして請求されます。詳細については、「Amazon EventBridge 料金表」を参照してください。

同じ AWS アカウントとリージョン内の別のイベントバスにイベントを送信するルールの作成

イベントを別のイベントバスに送信するには、イベントバスをターゲットとしてルールを作成します。ターゲットと同じ AWS アカウントとリージョンでイベントバスを指定することは、ルールの作成の一部です。

コンソールを使用して、同じ AWS アカウントとリージョン内の別のイベントバスにイベントを送信するルールを作成するには

1. 「???」のステップに従います。
2. 「???」ステップで、ターゲットタイプを選択するように求めるプロンプトが表示された場合:
    a. EventBridge イベントバス を選択します。
    b. 同じ AWS アカウントとリージョン のイベントバスを選択します。
    c. ターゲットとしてのイベントバスでは、ドロップダウンリストからイベントタイプを選択します。
3. ステップに従ってルールの作成を完了します。

Amazon EventBridge 入力変換

がルール のターゲットに情報を渡す前に EventBridge、イベントからのテキストをカスタマイズできます。コンソールまたは API の入力トランスフォーマーを使用すると、JSON パスを使用する変数を定義して、元のイベントソースの値を参照することができます。変換されたイベントは、元のイベントではなくターゲットに送信されます。ただし、動的パスパラメータは、変換されたイベントではなく、元のイベントを参照する必要があります。最大 100 の変数を定義して、それぞれに入力から値を割り当てることができます。その後、こうした変数は、<*variable-name*> 形式の入力テンプレートで使用できます。

入力トランスフォーマーの使い方については、「???」を参照してください。

> **Note**
>
> EventBridge は、すべての JSON パス構文をサポートしているわけではなく、実行時に評価されます。サポートされている構文には以下が含まれます。
>
> - ドット表記 (`$.detail` など)
> - ダッシュ
> - 下線
> - アルファベットの文字
> - 配列インデックス
> - ワイルドカード (*)

このトピックの内容

定義済みの変数

JSON パスを定義せずに使用できる、事前定義された変数があります。これらの変数は予約されており、これらの名前で変数を作成することはできません。

- `aws.events.rule-arn` — EventBridge ルールの Amazon リソースネーム (ARN)。
- `aws.events.rule-name` — EventBridge ルールの名前。
- `aws.events.event.ingestion-time` — イベントが によって受信された時刻 EventBridge。これは ISO 8601 タイムスタンプです。この変数は によって生成 EventBridge され、上書きすることはできません。
- `aws.events.event` — JSON としての元のイベントペイロード (`detail` フィールドなし)。内容はエスケープされないため、JSON フィールドの値としてのみ使用できます。
- `aws.events.event.json` — JSON としての完全な元のイベントペイロード (`detail` フィールドあり)。内容はエスケープされないため、JSON フィールドの値としてのみ使用できます。

入力変換の例

Amazon EC2 イベントの例を次に示します。

```
{
  "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"
  }
}
```

コンソールでルールを定義する際には、[Configure input] (入力の設定) の [Input Transformer] (入力トランスフォーマー) オプションを選択します。このオプションでは、2 つのテキストボックスが表示されます。1 つは入力パス用で、もう 1 つは入力テンプレート用です。

[Input Path] (入力パス) は、変数を定義するために使用されます。JSON パスを使用してイベント内の項目を参照し、それらの値を変数に格納します。たとえば、最初のテキストボックスに次のように入力すると、イベント例の値を参照する入力パスを作成できます。角かっことインデックスを使用して、配列から項目を取得することもできます。

> **Note**
>
> EventBridge は、実行時に入力トランスフォーマーを置き換えて、有効な JSON 出力を確保します。このため、JSON パスパラメータを参照する変数は、引用符で囲んでください。JSON オブジェクトまたは配列を参照する変数は、引用符で囲まないでください。

```
{
  "timestamp" : "$.time",
  "instance" : "$.detail.instance-id",
  "state" : "$.detail.state",
  "resource" : "$.resources[0]"
}
```

これにより、<timestamp>、<instance>、<state>、および <resource> の 4 つの変数が定義されます。入力テンプレートの作成時にこれらの変数を参照できます。

入力テンプレートは、ターゲットに渡す情報のテンプレートです。文字列または JSON をターゲットに渡すテンプレートを作成できます。上のイベントと入力パスを使用した以下の入力テンプレートの例では、イベントを出力例に変換してからターゲットにルーティングしています。

説明	テンプレート	出力
単純な文字列	`"instance <instance> is in <state>"`	`"instance i-0123456789 is in RUNNING"`
エスケープされた引用符を含む文字列	`"instance \"<instance>\" is in <state>"`	`"instance \"i-0123456789\" is in RUNNING"`

説明	テンプレート	出力
		これは EventBridge コンソールの動作であることに注意してください。 AWS CLI はスラッシュ文字をエスケープし、結果は `"instance \"i-0123456789\" is in RUNNING"` です。
単純な JSON	`{ "instance" : <instance>, "state": <state> }`	`{ "instance" : "i-0123456789", "state": "RUNNING" }`
文字列と変数を含む JSON	`{ "instance" : <instance>, "state": "<state>", "instanceStatus": "instance \"<instance>\" is in <state>" }`	`{ "instance" : "i-0123456789", "state": "RUNNING", "instanceStatus": "instance \"i-0123456789\" is in RUNNING" }`
変数と静的情報が混在した JSON	`{ "instance" : <instance>, "state": [ 9, <state>, true ], "Transformed" : "Yes" }`	`{ "instance" : "i-0123456789", "state": [ 9, "RUNNING", true ], "Transformed" : "Yes" }`

説明	テンプレート	出力

JSON に予約変数を含める	<pre>{
	  "instance" :
	 <instance>,
	  "state": <state>,
	  "ruleArn" : <aws.even
	ts.rule-arn>,
	  "ruleName" :
	 <aws.events.rule-n
	ame>,
	  "originalEvent" :
	 <aws.events.event.
	json>
	}</pre>
		<pre>{
		  "instance" :
		 "i-0123456789",
		  "state": "RUNNING",
		  "ruleArn" : "arn:aws:
		events:us-east-2:1
		23456789012:rule/e
		xample",
		  "ruleName" :
		 "example",
		  "originalEvent" : {
		  ... // commented for
		 brevity
		  }
		}</pre>

文字列に予約変数を含める	<pre>"<aws.events.rule-
	name> triggered"</pre>
		<pre>"example triggered"</pre>

Amazon CloudWatch ロググループ	<pre>{
	  "timestamp" :
	 <timestamp>,
	  "message": "instance
	 \"<instance>\" is in
	 <state>"
	}</pre>
		<pre>{
		  "timestamp" :
		 2015-11-11T21:29:54Z,
		  "message": "instance
		 "i-0123456789" is in
		 RUNNING
		}</pre>

API を使用した入力の EventBridge変換

EventBridge API を使用して入力を変換する方法については、「Input Transformer を使用してイベントからデータを抽出し、そのデータをターゲット に入力する」を参照してください。

を使用した入力の変換 AWS CloudFormation

を使用して入力 AWS CloudFormation を変換する方法については、「」を参照してくださいAWS::Events::Rule InputTransformer。

入力変換に関する一般的な問題

で入力を変換する際の一般的な問題は次のとおりです EventBridge。

- 文字列の場合は、引用符が必要です。
- テンプレートの JSON パスを作成する場合、検証は行われません。
- 指定した変数と一致する JSON パスがイベントに存在しない場合、その変数は作成されず、出力にも表示されません。
- `aws.events.event.json` のような JSON プロパティは JSON フィールドの値としてのみ使用でき、他の文字列に埋め込んで使用することはできません。
- EventBridge は、ターゲットの入力テンプレートを入力するときに、入力パス によって抽出された値をエスケープしません。
- JSON パスが JSON オブジェクトまたは配列を参照しているが、変数が文字列で参照されている場合、 は内部引用符を削除して有効な文字列を確保 EventBridge します。例えば、 を指す変数の場合`$.detail`、`<detail>`「Detail is <detail>」はオブジェクトから引用符 EventBridge を削除します。

  したがって、単一の JSON パス変数に基づいて JSON オブジェクトを出力する場合、それをキーとして配置する必要があります。この例では、`{"detail": <detail>}` です。
- 文字列を表す変数に引用符は必要ありません。これらは許可されますが、変換中に文字列変数値に引用符 EventBridge を自動的に追加して、変換出力が有効な JSON であることを確認します。JSON オブジェクトまたは配列を表す変数に引用符を追加 EventBridge しません。JSON オブジェクトや配列を表す変数に引用符を追加しないでください。

  たとえば、次の入力テンプレートには、文字列と JSON オブジェクトの両方を表す変数が含まれています。

```
{
  "ruleArn" : <aws.events.rule-arn>,
  "ruleName" : <aws.events.rule-name>,
  "originalEvent" : <aws.events.event.json>
}
```

  正しい引用符で囲まれた有効な JSON が生成されます。

```
{
  "ruleArn" : "arn:aws:events:us-east-2:123456789012:rule/example",
  "ruleName" : "example",
```

```
  "originalEvent" : {
    ... // commented for brevity
  }
}
```

- (JSON 以外の) テキスト出力を複数行文字列として使用する場合、入力テンプレートの各行を二重引用符で囲みます。

  例えば、Amazon Inspector 検出結果イベントを次のイベントパターンと照合する場合です。

```
{
  "detail": {
    "severity": ["HIGH"],
    "status": ["ACTIVE"]
  },
  "detail-type": ["Inspector2 Finding"],
  "source": ["inspector2"]
}
```

  また、次の入力パスを使用します。

```
{
  "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"
}
```

  以下の入力テンプレートを使用して、複数行の文字列出力を生成できます。

```
"<severity> severity finding <title>"
"Description: <description>"
"ARN: \"<arn>\""
"Type: <type>"
"AWS Account: <account>"
```

```
"Region: <region>"
"EC2 Instance: <instance>"
"Platform: <platform>"
"AMI: <ami>"
```

ルール作成の一環としての入力トランスフォーマーの設定

ルールの作成の一環として、 の入力トランスフォーマー EventBridge を指定して、それらのイベントを指定されたターゲットに送信する前に、一致するイベントを処理するために を使用できます。AWS サービスまたは API 送信先であるターゲットの入力トランスフォーマーを設定できます。

ルールの一部としてターゲットの入力トランスフォーマーを作成するには

1. ??? に詳述しているルール作成の手順に従います。
2. [ステップ 3 - ターゲットを選択] で、[追加設定] を展開します。
3. [ターゲット入力を設定] で、ドロップダウンリストから [入力トランスフォーマー] を選択します。

   [入力トランスフォーマーを設定] をクリックします。

   EventBridge は、入力トランスフォーマーの設定ダイアログボックスを表示します。
4. [サンプルイベント] セクションで、イベントパターンをテストする対象の [サンプルイベントタイプ] を選択します。 AWS イベント、パートナーイベントを選択するか、独自のカスタムイベントを入力できます。

   AWS events

   サポートされている AWS のサービスから発行されたイベントを選択します。

   1. [AWS イベント] を選択します。
   2. サンプルイベント で、目的の AWS イベントを選択します。イベントは AWS サービスごとに整理されます。

      イベントを選択すると、 によってサンプルイベント EventBridge が入力されます。

      例えば、S3 オブジェクト作成 を選択した場合、 はサンプル S3 オブジェクト作成イベント EventBridge を表示します。

3. (オプション) [コピー] を選択して、サンプルイベントをデバイスのクリップボードにコピーすることもできます

Partner events

Salesforce など EventBridge、 をサポートするサードパーティーサービスから出力されるイベントから選択します。

1. EventBridge パートナーイベント を選択します。
2. [サンプルイベント] で、目的のパートナーイベントを選択します。イベントはパートナーごとに整理されています。

   イベントを選択すると、 によってサンプルイベント EventBridge が入力されます。
3. (オプション) [コピー] を選択して、サンプルイベントをデバイスのクリップボードにコピーすることもできます

Enter your own

自分のイベントを JSON テキストで入力します。

1. [独自のイベントを入力] を選択します。
2. EventBridge は、必要なイベント属性のテンプレートをサンプルイベントに入力します。
3. 必要に応じてサンプルイベントを編集および追加します。サンプルイベントは有効な JSON である必要があります。
4. (オプション) 以下のいずれかのオプションを選択することもできます。
   - [コピー] — サンプルイベントをデバイスのクリップボードにコピーします
   - [Prettify] (整文) — 改行、タブ、スペースを追加して JSON テキストを読みやすくします。

5. (オプション) [入力パス、テンプレート、および出力の例] セクションを展開して、以下の例を表示します。

   - JSON パスを使用してイベントデータを表す変数を定義する方法
   - これらの変数を入力トランスフォーマーテンプレートで使用する方法
   - がターゲット EventBridge に送信する結果の出力

入力変換の詳細な例については、「???」を参照してください。

6. [ターゲット入力トランスフォーマー] セクションで、入力テンプレートで使用する変数を定義します。

   変数は、JSON パスを使用して元のイベントソースの値を参照します。その後、これらの変数を入力テンプレートで参照して、 EventBridge ターゲットに渡される変換されたイベントに元のソースイベントのデータを含めることができます。最大 100 の変数を定義できます。入力トランスフォーマーは有効な JSON である必要があります。

   例えば、この入力トランスフォーマーのサンプル AWS イベントとしてイベント S3 オブジェクト作成 を選択した場合を考えます。この場合、以下の変数を定義してテンプレートで使用できます。

   ```
   {
     "requester": "$.detail.requester",
     "key": "$.detail.object.key",
     "bucket": "$.detail.bucket.name"
   }
   ```

   (オプション) [コピー] を選択して、入力トランスフォーマーをデバイスのクリップボードにコピーすることもできます。

7. テンプレートセクションで、ターゲットに EventBridge 渡すテンプレートを作成します。

   JSON、文字列、静的情報、定義した変数、予約変数を使用できます。入力変換の詳細な例については、「???」を参照してください。

   例えば、前の例で変数を定義したとします。次に、これらの変数、予約変数、静的情報を参照する次のテンプレートを作成できます。

   ```
   {
       "message": "<requester> has created the object \"<key>\" in the bucket
    \"<bucket>\"",
       "RuleName": <aws.events.rule-name>,
       "ruleArn" : <aws.events.rule-arn>,
       "Transformed": "Yes"
   }
   ```

(オプション) [コピー] を選択して、テンプレートをデバイスのクリップボードにコピーすることもできます。

8. テンプレートをテストするには、[出力を生成] を選択します。

   EventBridge は入力テンプレートに基づいてサンプルイベントを処理し、出力 で生成された変換された出力を表示します。これは、元のソースイベントの代わりにターゲットに渡す情報 EventBridge です。

   上で説明したサンプル入力テンプレートで生成される出力は、次のようになります。

```
{
    "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"
}
```

   (オプション) [コピー] を選択して、生成された出力をデバイスのクリップボードにコピーすることもできます。

9. [確認] を選択します。

10. 「???」に詳述しているルールの作成に関する残りのステップに従います。

EventBridge サンドボックスを使用したターゲット入力トランスフォーマーのテスト

入力トランスフォーマーを使用して、 がルール のターゲットに情報を渡す前に EventBridge、イベントからのテキストをカスタマイズできます。

入力トランスフォーマーの設定は、通常、新しいルールの作成や既存のルールの編集に伴ってターゲットを指定するという、より大きなプロセスの一環として行います。ただし EventBridge、 でサンドボックスを使用すると、ルールを作成または編集しなくても、入力トランスフォーマーをすばやく設定し、サンプルイベントを使用して必要な出力を取得していることを確認できます。

入力変換の詳細については、「???」を参照してください。

ターゲット入力トランスフォーマーをテストするには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. [デベロッパーリソース] で [サンドボックス] を選択し、[サンドボックス] ページで [ターゲット入力トランスフォーマー] タブを選択します。
3. [サンプルイベント] セクションで、イベントパターンをテストする対象の [サンプルイベントタイプ] を選択します。 AWS イベント、パートナーイベントを選択するか、独自のカスタムイベントを入力できます。

   AWS events

   サポートされている AWS のサービスから発行されたイベントを選択します。

   1. [AWS イベント] を選択します。
   2. サンプルイベント で、目的の AWS イベントを選択します。イベントは AWS サービスごとに整理されます。

      イベントを選択すると、 によってサンプルイベント EventBridge が入力されます。

      例えば、S3 オブジェクト作成 を選択した場合、 はサンプル S3 オブジェクト作成イベント EventBridge を表示します。
   3. (オプション) [コピー] を選択して、サンプルイベントをデバイスのクリップボードにコピーすることもできます

   Partner events

   Salesforce など EventBridge、 をサポートするサードパーティーサービスから出力されるイベントから選択します。

   1. EventBridge パートナーイベント を選択します。
   2. [サンプルイベント] で、目的のパートナーイベントを選択します。イベントはパートナーごとに整理されています。

      イベントを選択すると、 によってサンプルイベント EventBridge が入力されます。
   3. (オプション) [コピー] を選択して、サンプルイベントをデバイスのクリップボードにコピーすることもできます

Enter your own

自分のイベントを JSON テキストで入力します。

1. [独自のイベントを入力] を選択します。
2. EventBridge は、必要なイベント属性のテンプレートをサンプルイベントに入力します。
3. 必要に応じてサンプルイベントを編集および追加します。サンプルイベントは有効な JSON である必要があります。
4. (オプション) 以下のいずれかのオプションを選択することもできます。
   - [コピー] — サンプルイベントをデバイスのクリップボードにコピーします
   - [Prettify] (整文) — 改行、タブ、スペースを追加して JSON テキストを読みやすくします。

4. (オプション) [入力パス、テンプレート、および出力の例] セクションを展開して、以下の例を表示します。

- JSON パスを使用してイベントデータを表す変数を定義する方法
- これらの変数を入力トランスフォーマーテンプレートで使用する方法
- がターゲット EventBridge に送信する結果の出力

入力変換の詳細な例については、「???」を参照してください。

5. [ターゲット入力トランスフォーマー] セクションで、入力テンプレートで使用する変数を定義します。

変数は、JSON パスを使用して元のイベントソースの値を参照します。その後、これらの変数を入力テンプレートで参照して、 EventBridge ターゲットに渡される変換されたイベントに元のソースイベントのデータを含めることができます。最大 100 の変数を定義できます。入力トランスフォーマーは有効な JSON である必要があります。

例えば、この入力トランスフォーマーのサンプル AWS イベントとしてイベント S3 オブジェクト作成 を選択した場合を考えます。この場合、以下の変数を定義してテンプレートで使用できます。

```
{
  "requester": "$.detail.requester",
  "key": "$.detail.object.key",
```

```
  "bucket": "$.detail.bucket.name"
}
```

(オプション) [コピー] を選択して、入力トランスフォーマーをデバイスのクリップボードにコピーすることもできます。

6. テンプレートセクションで、ターゲットに EventBridge 渡すテンプレートを作成します。

   JSON、文字列、静的情報、定義した変数、予約変数を使用できます。入力変換の詳細な例については、「???」を参照してください。

   例えば、前の例で変数を定義したとします。次に、これらの変数、予約変数、静的情報を参照する次のテンプレートを作成できます。

```
{
    "message": "<requester> has created the object \"<key>\" in the bucket
 \"<bucket>\"",
    "RuleName": <aws.events.rule-name>,
    "ruleArn" : <aws.events.rule-arn>,
    "Transformed": "Yes"
}
```

   (オプション) [コピー] を選択して、テンプレートをデバイスのクリップボードにコピーすることもできます。

7. テンプレートをテストするには、[出力を生成] を選択します。

   EventBridge は入力テンプレートに基づいてサンプルイベントを処理し、出力 で生成された変換された出力を表示します。これは、元のソースイベントの代わりにターゲットに渡す情報 EventBridge です。

   上で説明したサンプル入力テンプレートで生成される出力は、次のようになります。

```
{
    "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"
}
```

(オプション) [コピー] を選択して、生成された出力をデバイスのクリップボードにコピーすることもできます。

Amazon EventBridge のアーカイブと再生

EventBridge では、後で簡単に再生できるようにイベントのアーカイブを作成することができます。例えば、イベントを再生して、エラーから回復したり、アプリケーションの新機能を検証したりする場合があります。

> **Note**
>
> イベントがイベントバスに公開されてからアーカイブに到着するまでの間に遅延が生じる場合があります。すべてのイベントが確実に再生されるよう、アーカイブしたイベントの再生を 10 分間遅らせることをお勧めします。

次の動画では、アーカイブと再生の使用方法を示します: アーカイブと再生の作成

トピック

- Amazon EventBridge イベントのアーカイブ
- アーカイブされた Amazon EventBridge イベントの再生

Amazon EventBridge イベントのアーカイブ

でアーカイブを作成するときに EventBridge、イベントパターン を指定することで、アーカイブに送信されるイベントを判断できます。 は、イベントパターンに一致するイベントをアーカイブ EventBridge に送信します。また、保存期間を設定して、イベントが破棄される前にアーカイブに保存します。

デフォルトでは、 は 所有の AWS CMK で 256 ビットの Advanced Encryption Standard (AES-256) を使用してアーカイブ内のイベントデータを EventBridge 暗号化します。これにより、不正アクセスからデータを保護できます。

> **Note**
>
> `DescribeArchive` オペレーションの `EventCount`と `SizeBytes`の値は、調整期間が 24 時間です。したがって、最近期限切れになったイベントや新しくアーカイブされたイベントは、これらの値にすぐに反映されない場合があります。

すべてのイベントのアーカイブを作成するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. 左側のナビゲーションペインで [Archives] (アーカイブ) を選択します。
3. [Create archive] (アーカイブの作成) を選択します。
4. [Archive detail] (アーカイブの詳細) で、アーカイブの [Name] (名前) を入力します。この名前は、選択されたリージョンのアカウントで一意である必要があります。

   アーカイブの作成後は、名前を変更できません。
5. (オプション) アーカイブの [Description] (説明) を入力します。
6. [Source] (ソース) には、アーカイブに送信するイベントを発するイベントバスを選択します。
7. [Retention period] (保持期間) で次のいずれかの操作を行います。
   - [Indefinite] (不定期) を選択して、アーカイブ内のイベントを保持し、削除しないようにする。
   - イベントを保持する日数を入力する。指定された日数が経過すると、 はアーカイブからイベント EventBridge を削除します。
8. [次へ] をクリックします。

9. [Event pattern] (イベントパターン) で、[No event filtering] (イベントフィルタリングなし) を選択します。
10. [Create archive] (アーカイブの作成) を選択します。

イベントパターンでアーカイブを作成するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. 左側のナビゲーションペインで [Archives] (アーカイブ) を選択します。
3. [Create archive] (アーカイブの作成) を選択します。
4. [Archive detail] (アーカイブの詳細) で、アーカイブの [Name] (名前) を入力します。この名前は、選択されたリージョンのアカウントで一意である必要があります。

   アーカイブの作成後は、名前を変更できません。
5. (オプション) アーカイブの [Description] (説明) を入力します。
6. [Source] (ソース) には、アーカイブに送信するイベントを発するイベントバスを選択します。
7. [Retention period] (保持期間) で次のいずれかの操作を行います。
   - [Indefinite] (不定期)* を選択して、アーカイブ内のイベントを保持し、削除しないようにする。
   - イベントを保持する日数を入力する。指定された日数が経過すると、 はアーカイブからイベント EventBridge を削除します。
8. [次へ] をクリックします。
9. [Event pattern] (イベントパターン) で、[Filtering events by event pattern matching] (イベントパターンマッチングによるイベントのフィルタリング) を選択します。
10. 次のいずれかを行います。
    - [Pattern builder] (パターンビルダー) を選択し、[Service provider] (サービスプロバイダー) を選択します。を選択した場合はAWS、パターンで使用するAWS サービス名とイベントタイプも選択します。
    - [JSON editor] (JSON エディタ) を選択して、パターンを手動で作成します。ルールからパターンをコピーし、JSON エディタに貼り付けることもできます。
11. [Create archive] (アーカイブの作成) を選択します。

イベントがアーカイブに正常に送信されたことを確認するには、 EventBridge API の DescribeArchiveオペレーションを使用して、 がアーカイブ内のイベント数EventCountを反映しているかどうかを確認します。0 の場合、アーカイブにはイベントはありません。

アーカイブされた Amazon EventBridge イベントの再生

アーカイブを作成すると、アーカイブからイベントを再生することができます。例えば、アプリケーションを追加機能で更新する場合、履歴イベントを再生して、イベントの再処理がアプリケーションの一貫性を維持するようにすることができます。アーカイブを利用して、新しい機能に対するイベントを再生することもできます。イベントを再生する場合、再生するアーカイブ、再生するイベントの開始時間と終了時間、イベントバスまたは、1 つ以上のルールを指定することができます。

イベントは、必ずしもアーカイブに追加された順序通りに再生されるわけではありません。再生では、イベント内の時間に基づいて再生するイベントを処理し、1 分間隔で再生します。イベントの開始時刻と終了時刻を 20 分単位で指定すると、イベントは、その 20 分単位の最初の 1 分目から再生されます。その後、2 分目以降のイベントが再生されます。EventBridge API の `DescribeReplay` 操作を使用して、再生の進行状況を確認することができます。`EventLastReplayedTime` は、最後に再生されたイベントのタイムスタンプを返します。

イベントは、AWS アカウントの 1 秒あたりの `PutEvents` トランザクション制限に基づいて再生されますが、これとは区別されます。PutEvents の制限の引き上げをリクエストすることができます。詳細については、Amazon EventBridge クォータを参照してください。

> **Note**
>
> アカウントあたり、AWS リージョンあたり最大 10 個のアクティブな同時再生を設定できます。

イベント再生を開始するには

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. 左のナビゲーションペインの [Replays] (再生) を選択します。
3. [Start new replay] (新規再生を開始) を選択します。
4. 再生の [Name] (名前) を入力し、必要に応じて [Description] (説明) を入力します。
5. [Source] (ソース) で、イベントを再生するアーカイブを選択します。
6. 送信先に対して、イベントを出したものと同じイベントバスにだけ、イベントを再生できます
7. [Specify rules] (ルールの指定) で、次のいずれかの操作を行います。
   - [All rules] (すべてのルール) を選択して、すべてのルールにイベントを再生します。
   - [Specify rules] (ルールの指定) を選択して、イベントを再生するルールを選択します。

8. [Replay time frame] (再生時間枠) で、[Date] (日付)、[Time] (時間)、[Time zone] (タイムゾーン) の [Start time] (開始時間)と [End time] (終了時間) を指定します。[Start time (開始時間)] と [End time (終了時間)] の間に発生したイベントのみが再生されます。
9. [Start replay] (再生を開始) を選択します。

保存されたイベントが再生されると、再生のステータスは、[Completed] (完了) となります。

再生を開始した後に中断したい場合、ステータスが [Starting] (開始中) または [Running] (実行中) であれば、キャンセルすることができます。

再生をキャンセルするには

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. 左のナビゲーションペインの [Replays] (再生) を選択します。
3. キャンセルする再生を選択します。
4. [Cancel] (キャンセル) を選択します。

Amazon EventBridge Pipes

Amazon EventBridge Pipes はソースをターゲットに接続します。パイプは、サポートされているソースとターゲット の統合を目的として point-to-pointおり、高度な変換とエンリッチメント をサポートしています。イベント駆動型アーキテクチャを開発する際に専門知識や統合コードが不要になり、会社のアプリケーション全体で一貫性が保持されます。パイプをセットアップするには、ソースを選択し、オプションのフィルタリングを追加し、オプションのエンリッチメントを定義し、イベントデータのターゲットを選択します。

> **Note**
>
> イベントバスを使用してイベントをルーティングすることもできます。イベントバスは、イベント駆動型サービス間のイベントのルーティングに適しています many-to-many。詳細については、「???」を参照してください。

EventBridge Pipes の仕組み

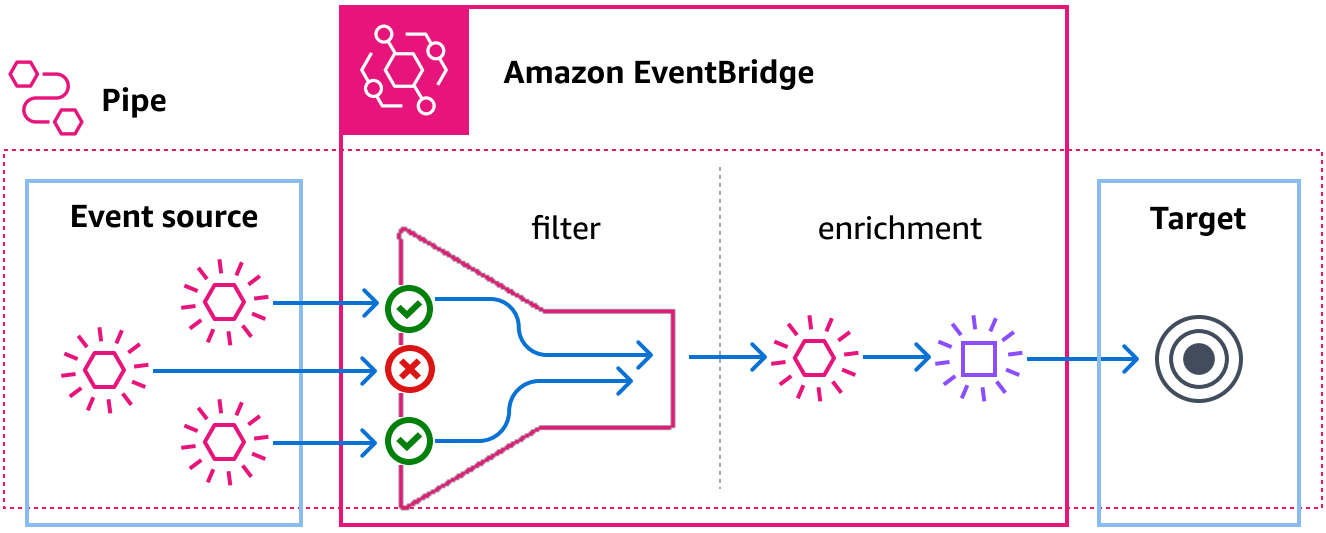
大まかに言うと、 EventBridge パイプの仕組みは次のとおりです。

1. アカウントにパイプを作成します。これには、以下が含まれます。
   - サポートされているイベントソースのうち、パイプにイベントを受信させたいもののいずれかを指定します。
   - オプションで、ソースから受信したイベントのサブセットだけをパイプが処理するようにフィルターを設定します。
   - オプションで、ターゲットに送信する前にイベントデータを拡張するエンリッチメントステップを設定します。
   - サポートされているターゲットのうち、パイプにイベントを送信させたいもののいずれかを指定します。
2. イベントソースはパイプへのイベントの送信を開始し、パイプはイベントを処理してからターゲットに送信します。
   - フィルターを設定した場合、パイプはイベントを評価し、そのフィルターに一致する場合にのみターゲットに送信します。

     フィルターに一致するイベントに対してのみ課金されます。

- エンリッチメントを設定した場合、パイプはターゲットに送信する前にイベントに対してエンリッチメントを実行します。

  イベントがバッチ処理される場合、エンリッチメントはバッチ内のイベントの順序を維持します。



例えば、パイプを使って e コマースシステムを作ることができます。配送先住所などの顧客情報を含む API があると想定します。

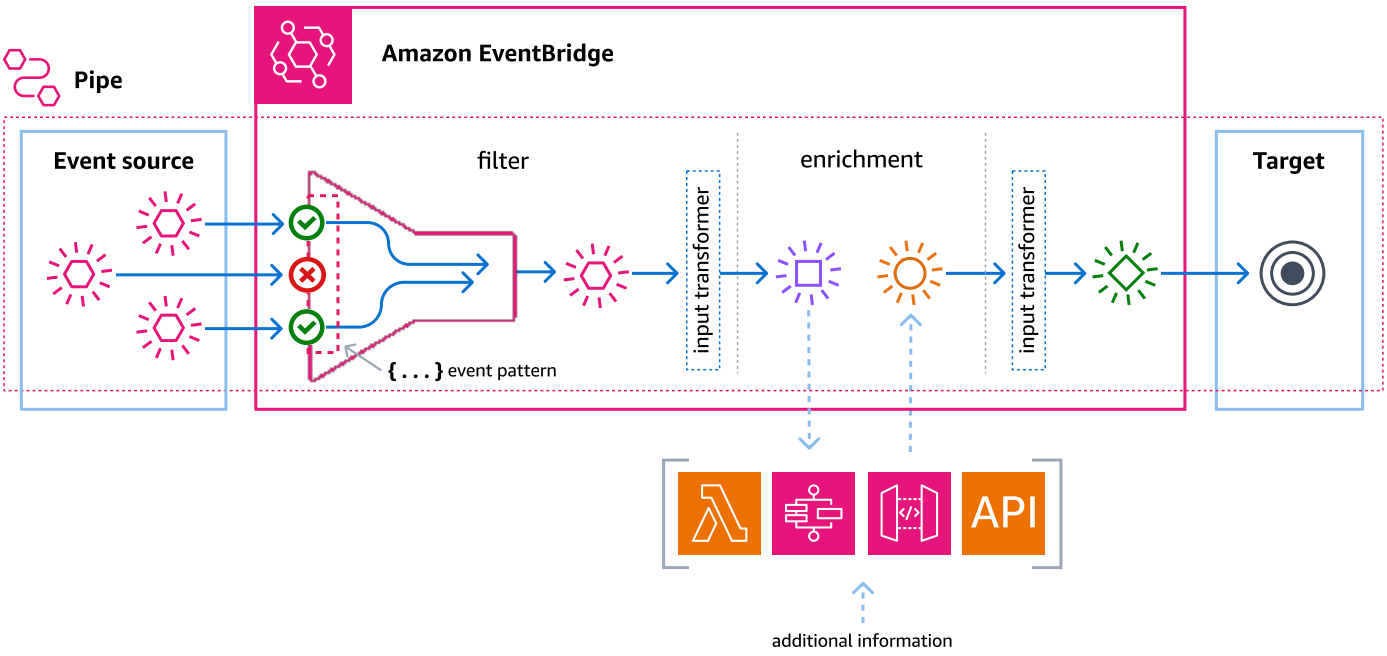
1. パイプは以下を使用して作成できます。
   - Amazon SQS 注文が、イベントソースとしてメッセージキューを受信しました。
   - エンリッチメントとしての EventBridge API 送信先
   - ターゲットとしての AWS Step Functions ステートマシン
2. その後、Amazon SQS の注文受領メッセージがキューに表示されると、そのメッセージがパイプに送信されます。
3. 次に、パイプはそのデータを EventBridge API 送信先エンリッチメントに送信し、API 送信先エンリッチメントはその注文の顧客情報を返します。
4. 最後に、パイプはエンリッチされたデータを AWS Step Functions ステートマシンに送信し、ステートマシンは注文を処理します。

EventBridge Pipes の概念

EventBridge Pipes の基本コンポーネントを詳しく見ていきます。

パイプ

パイプは 1 つのソースから単一のターゲットにイベントをルーティングします。このパイプには、特定のイベントをフィルタリングする機能や、ターゲットに送信される前にイベントデータを強化する機能も含まれています。



ソース

EventBridge Pipes はさまざまなソースからイベントデータを受け取り、そのデータにオプションのフィルターとエンリッチメントを適用して、ターゲットに送信します。ソースがパイプに送信されるイベントに順序を強制する場合、その順序はターゲットまでのプロセス全体を通して維持されます。

sources の詳細については、「???」を参照してください。

フィルター

パイプは、特定のソースのイベントをフィルタリングして、それらのイベントのサブセットのみを処理できます。パイプのフィルタリングを設定するには、パイプがターゲットに送信するイベントを決定するのに使用するイベントパターンを定義します。

フィルターに一致するイベントに対してのみ課金されます。

詳細については、「???」を参照してください。

エンリッチメント

EventBridge Pipes のエンリッチメントステップを使用すると、ターゲットに送信する前にソースからのデータを強化できます。たとえば、チケットデータがすべて含まれていないチケット作成イベントを受け取る場合があります。エンリッチメントを使用すると、Lambda 関数で `get-ticket` API を呼び出して、チケットの詳細をすべて確認できます。その後、パイプはその情報をターゲットに送信できます。

イベントデータの強化の詳細については、「???」を参照してください。

Target

イベントデータをフィルタリングして強化したら、パイプが Amazon Kinesis ストリームや Amazon CloudWatch ロググループなどの特定のターゲットに送信することを指定できます。使用可能なターゲットのリストについては、「???」を参照してください。

データは、拡張してからパイプがターゲットに送信する前に変換できます。詳細については、「???」を参照してください。

ソースが異なる複数のパイプが同じターゲットにイベントを送信できます。

パイプとイベントバスを一緒に使用して、複数のターゲットにイベントを送信することもできます。一般的なユースケースは、イベントバスをターゲットとするパイプを作成することです。パイプはイベントをイベントバスに送信し、イベントバスはそれらのイベントを複数のターゲットに送信します。たとえば、ソースに DynamoDB ストリームを使用し、ターゲットとしてイベントバスを含むパイプを作成できます。パイプは DynamoDB ストリームからイベントを受け取り、イベントバスに送信します。イベントバスは、イベントバスで指定したルールに従ってイベントを複数のターゲットに送信します。

Amazon EventBridge Pipes のアクセス許可

パイプをセットアップするときは、既存の実行ロールを使用するか、必要なアクセス許可により、実行ロールを EventBridge に作成させることができます。EventBridge Pipes に必要なアクセス許可は、以下に示すようにソースタイプによって異なります。独自の実行ロールを設定する場合は、これらのアクセス許可を自分で追加する必要があります。

> **Note**
>
> ソースへのアクセスに必要なアクセス許可が正確にわからない場合は、EventBridge Pipes コンソールを使用して新しいロールを作成し、ポリシーに記載されているアクションを確認してください。

トピック

DynamoDB 実行ロールのアクセス許可

DynamoDB Streams の場合、EventBridge Pipes は、DynamoDB データストリームに関連するリソースを管理するための以下のアクセス許可が必要です。

- `dynamodb:DescribeStream`
- `dynamodb:GetRecords`
- `dynamodb:GetShardIterator`
- `dynamodb:ListStreams`

失敗したバッチのレコードをパイプのデッドレターキューに送信するには、パイプ実行ロールに次のアクセス許可が必要です。

- `sqs:SendMessage`

Kinesis 実行ロールのアクセス許可

Kinesis の場合、EventBridge Pipes は、Kinesis データストリームに関連するリソースを管理するための以下のアクセス許可が必要です。

- kinesis:DescribeStream
- kinesis:DescribeStreamSummary
- kinesis:GetRecords
- kinesis:GetShardIterator
- kinesis:ListShards
- kinesis:ListStreams
- kinesis:SubscribeToShard

失敗したバッチのレコードをパイプデッドレターキューに送信するには、パイプ実行ロールに次のアクセス許可が必要です。

- sqs:SendMessage

Amazon MQ 実行ロールのアクセス許可

Amazon MQ の場合、EventBridge では Amazon MQ メッセージブローカーに関連するリソースを管理するために以下のアクセス許可が必要です。

- mq:DescribeBroker
- secretsmanager:GetSecretValue
- ec2:CreateNetworkInterface
- ec2:DeleteNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:DescribeSecurityGroups
- ec2:DescribeSubnets
- ec2:DescribeVpcs
- logs:CreateLogGroup
- logs:CreateLogStream

- logs:PutLogEvents

Amazon MSK 実行ロールのアクセス許可

Amazon MSK の場合、EventBridge では Amazon MSK トピックに関連するリソースを管理するために以下のアクセス許可が必要です。

> **Note**
>
> IAM ロールベースの認証を使用している場合、実行ロールには以下のアクセス許可に加えて ??? でリストされているアクセス許可が必要です。

- kafka:DescribeClusterV2
- kafka:GetBootstrapBrokers
- ec2:CreateNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:DescribeVpcs
- ec2:DeleteNetworkInterface
- ec2:DescribeSubnets
- ec2:DescribeSecurityGroups
- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents

セルフマネージド型の Apache Kafka 実行ロールのアクセス許可

セルフマネージド Apache Kafka の場合、EventBridge はセルフマネージド Apache Kafka ストリームに関連するリソースを管理するために以下のアクセス許可を必要とします。

必要なアクセス権限

Amazon CloudWatch Logs のロググループでログを作成して保存するには、パイプの実行ロールに以下の許可が必要です。

- logs:CreateLogGroup
- logs:CreateLogStream
- logs:PutLogEvents

オプションのアクセス許可

パイプには、以下を実行する許可も必要になる場合があります。

- Secrets Manager シークレットを記述する。
- AWS Key Management Service (AWS KMS) カスタマー管理のキーにアクセスする。
- Amazon VPC にアクセスする。

Secrets Manager と AWS KMS 許可

Apache Kafka ブローカーに設定しているアクセスコントロールのタイプに応じて、パイプには Secrets Manager シークレットにアクセスするための許可、または AWS KMS カスタマーマネージドキーを復号化するための許可が必要になる場合があります。それらのリソースにアクセスするには、関数の実行ロールに次のアクセス許可が必要です。

- secretsmanager:GetSecretValue
- kms:Decrypt

VPC アクセス許可

セルフマネージド Apache Kafka クラスターにアクセスできるのが VPC 内のユーザーのみである場合、パイプには Amazon VPC リソースにアクセスするための許可が必要です。これらのリソースには、VPC、サブネット、セキュリティグループ、ネットワークインターフェイスが含まれます。それらのリソースにアクセスするには、パイプの実行ロールに次のアクセス許可が必要です。

- ec2:CreateNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:DescribeVpcs
- ec2:DeleteNetworkInterface
- ec2:DescribeSubnets

- ec2:DescribeSecurityGroups

Amazon SQS 実行ロールのアクセス許可

Amazon SQS の場合、EventBridge では Amazon SQS キューに関連するリソースを管理するために以下のアクセス許可が必要です。

- sqs:ReceiveMessage
- sqs:DeleteMessage
- sqs:GetQueueAttributes

エンリッチメントとターゲットのアクセス許可

所有するリソースで API をコールするには、EventBridge Pipes に適切な許可が必要です。EventBridge Pipes は、IAM プリンシパル `pipes.amazonaws.com` を使用して、パイプで指定した IAM ロールをエンリッチメントとターゲットコールに使用します。

Amazon EventBridge パイプの作成

EventBridge Pipes を使用すると、高度なイベント変換やエンリッチメントなど、 point-to-point ソースとターゲットの間の統合を作成できます。 EventBridge パイプを作成するには、以下の手順を実行します。

1. ???
2. ???
3. ???
4. ???
5. ???

CLI を使用してパイプを作成する方法については、 AWS AWS CLI コマンドリファレンスの create-pipe を参照してください。

ソースの指定

まず、パイプでイベントを受信するソースを指定します。

コンソールを使用してパイプソースを指定するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで、[パイプ] を選択します。
3. [Create pipe] (パイプの作成) を選択します。
4. パイプの名前を入力します。
5. (オプション) パイプの説明を追加します。
6. [パイプを構築] タブの [ソース] で、このパイプに指定するソースのタイプを選択し、ソースを設定します。

   設定のプロパティは、選択するソースの種類によって異なります。

   Confluent

   Confluent Cloud ストリームをソースとして設定するには、コンソールを使用して Confluent Cloud ストリームをソースとして設定します。

   1. [ソース] には Confluent クラウドを選択します。
   2. [Bootstrap servers] (ブートストラップサーバー) には、ブローカーの `host:port` ペアアドレスを入力します。
   3. [Topic name] (トピック名) には、読み取り元のトピック名を入力します。
   4. (オプション) [VPC] では、使用する VPC を選択します。次に、[VPC subnets] (VPC サブネット) に、目的のサブネットを選択します。[VPCSecurity groups] (VPC セキュリティグループ) で、セキュリティグループを選択します。
   5. [認証-オプション] で [認証を使用] をオンにし、次の操作を行います。
      a. [Authentication method] (認証方法) で、認証タイプを選択します。
      b. [Secret key] (シークレットキー) で、シークレットキーを選択します。

      詳細については、Confluent ドキュメントの「Confluent Cloud リソースへの認証」を参照してください。
   6. (オプション) [Additional settings - optional] (追加設定 - オプション) では、以下を実行します。
      a. [Starting position] (開始位置) で、次のいずれかを選択します。
         - Latest (最新) — シャード内の最新のレコードでストリームの読み取りを開始します。

- Trim horizon (水平トリム) — シャード内の最後のトリミングされていないレコードからストリームの読み取りを開始します。これはシャード内の最も古いレコードです。

b. [Batch size - optional] (バッチサイズ - オプション) に、各バッチの最大レコード数を入力します。デフォルト値は 100 です。

c. [Batch window - optional] (バッチウィンドウ - オプション) の場合、次に進む前にレコードを収集する最大秒数を入力します。

DynamoDB

1. [ソース] で、[DynamoDB] を選択します。
2. DynamoDB ストリームの場合は、ソースとして使用するストリームを選択します。
3. [Starting position] (開始位置) で、次のいずれかを選択します。
   - Latest (最新) — シャード内の最新のレコードでストリームの読み取りを開始します。
   - Trim horizon (水平トリム) — シャード内の最後のトリミングされていないレコードからストリームの読み取りを開始します。これはシャード内の最も古いレコードです。
4. (オプション) [Additional settings - optional] (追加設定 - オプション) では、以下を実行します。
   a. [Batch size - optional] (バッチサイズ - オプション) に、各バッチの最大レコード数を入力します。デフォルト値は 100 です。
   b. [Batch window - optional] (バッチウィンドウ - オプション) の場合、次に進む前にレコードを収集する最大秒数を入力します。
   c. [Concurrent batches per shard - optional] (シャードあたりの同時バッチ数 - オプション) の場合、同じシャードから同時に読み取ることができるバッチの数を入力します。
   d. [On partial batch item failure] (バッチ項目の一部に障害が発生した場合) では、次の項目を選択します。
      - AUTOMATIC_BISECT — すべてのレコードが処理されるか、バッチに 1 つの失敗メッセージが残るまで、各バッチを半分ずつ再試行します。

> **Note**
>
> AUTOMATIC_BISECT を選択しない場合は、特定の失敗したレコードを返すことができ、そのレコードだけが再試行されます。

Kinesis

コンソールを使用して Kinesis ソースを設定するには

1. [ソース] で、[Kinesis] を選択します。
2. [Kinesis stream] (Kinesis ストリーム) には、ソースとして使用するストリームを選択します。
3. [Starting position] (開始位置) で、次のいずれかを選択します。
   - Latest (最新) — シャード内の最新のレコードでストリームの読み取りを開始します。
   - Trim horizon (水平トリム) — シャード内の最後のトリミングされていないレコードからストリームの読み取りを開始します。これはシャード内の最も古いレコードです。
   - [At timestamp] (タイムスタンプ時) — 指定した時刻からストリームの読み取りを開始します。[Timestamp] (タイムスタンプ) に、YYYY/MM/DD と hh:mm:ss 形式でデータと時刻を入力します。
4. (オプション) [Additional settings - optional] (追加設定 - オプション) では、以下を実行します。
   a. [Batch size - optional] (バッチサイズ - オプション) に、各バッチの最大レコード数を入力します。デフォルト値は 100 です。
   b. (オプション) [Batch window - optional] (バッチウィンドウ - オプション) の場合、次に進む前にレコードを収集する最大秒数を入力します。
   c. [Concurrent batches per shard - optional] (シャードあたりの同時バッチ数 - オプション) の場合、同じシャードから同時に読み取ることができるバッチの数を入力します。
   d. [On partial batch item failure] (バッチ項目の一部に障害が発生した場合) では、次の項目を選択します。
      - AUTOMATIC_BISECT — すべてのレコードが処理されるか、バッチに 1 つの失敗メッセージが残るまで、各バッチを半分ずつ再試行します。

> **Note**
>
> AUTOMATIC_BISECT を選択しない場合は、特定の失敗したレコードを返すことができ、そのレコードだけが再試行されます。

Amazon MQ

コンソールを使用して Amazon MQ ソースを設定するには

1. [ソース] で、[Amazon MQ] を選択します。
2. Amazon MQ ブローカーの場合は、ソースとして使用するストリームを選択します。
3. [Queue name] (キュー名) には、読み取り元のトピック名を入力します。
4. [Authentication Method] (認証方法) には、[BASIC_AUTH] を選択します。
5. [Secret key] (シークレットキー) で、シークレットキーを選択します。
6. (オプション) [Additional settings - optional] (追加設定 - オプション) では、以下を実行します。
   a. [Batch size - optional] (バッチサイズ - オプション) に、各バッチの最大メッセージ数を入力します。デフォルト値は 100 です。
   b. [Batch window - optional] (バッチウィンドウ - オプション) の場合、次に進む前にレコードを収集する最大秒数を入力します。

Amazon MSK

コンソールを使用して Amazon MSK ソースを設定するには

1. [ソース] で、[Amazon MSK] を選択します。
2. [Amazon MSK cluster] (Amazon MSK クラスター) で、開くクラスターを選択します。
3. [Topic name] (トピック名) には、読み取り元のトピック名を入力します。
4. (オプション) [Consumer Group ID - optional] (コンシューマーグループ ID) で、パイプが参加するコンシューマーグループの ID を入力します。
5. (オプション) [Authentication - optional] (認証 - オプション) で、[Use Authentication] (認証を使用) をオンにして次の操作を行います。
   a. [Authentication method] (認証方法) で、認証タイプを選択します。
   b. [Secret key] (シークレットキー) で、シークレットキーを選択します。
6. (オプション) [Additional settings - optional] (追加設定 - オプション) では、以下を実行します。
   a. [Batch size - optional] (バッチサイズ - オプション) に、各バッチの最大レコード数を入力します。デフォルト値は 100 です。

b. [Batch window - optional] (バッチウィンドウ - オプション) の場合、次に進む前にレコードを収集する最大秒数を入力します。

c. [Starting position] (開始位置) で、次のいずれかを選択します。

- Latest (最新) — シャード内の最新のレコードでトピックの読み取りを開始します。
- Trim horizon (水平トリム) — シャード内の最後のトリミングされていないレコードからトピックの読み取りを開始します。これはシャード内の最も古いレコードです。

> **Note**
>
> Trim horizon (水平トリム) はApache Kafka の[Earliest] (最も早い時間) と同じです。

Self managed Apache Kafka

コンソールを使用してセルフマネージド型の Apache Kafka ソースを設定するには

1. [ソース] で、[セルフマネージド型の Apache Kafka] を選択します。
2. [Bootstrap servers] (ブートストラップサーバー) には、ブローカーの `host:port` ペアアドレスを入力します。
3. [Topic name] (トピック名) には、読み取り元のトピック名を入力します。
4. (オプション) [VPC] では、使用する VPC を選択します。次に、[VPC subnets] (VPC サブネット) に、目的のサブネットを選択します。[VPCSecurity groups] (VPC セキュリティグループ) で、セキュリティグループを選択します。
5. (オプション) [Authentication - optional] (認証 - オプション) で、[Use Authentication] (認証を使用) をオンにして次の操作を行います。
   a. [Authentication method] (認証方法) で、認証タイプを選択します。
   b. [Secret key] (シークレットキー) で、シークレットキーを選択します。
6. (オプション) [Additional settings - optional] (追加設定 - オプション) では、以下を実行します。
   a. [Starting position] (開始位置) で、次のいずれかを選択します。
      - Latest (最新) — シャード内の最新のレコードでストリームの読み取りを開始します。
      - Trim horizon (水平トリム) — シャード内の最後のトリミングされていないレコードからストリームの読み取りを開始します。これはシャード内の最も古いレコードです。

b. [Batch size - optional] (バッチサイズ - オプション) に、各バッチの最大レコード数を入力します。デフォルト値は 100 です。

c. [Batch window - optional] (バッチウィンドウ - オプション) の場合、次に進む前にレコードを収集する最大秒数を入力します。

Amazon SQS

コンソールを使用して Amazon SQS ソースを設定するには

1. [ソース] で [SQS] を選択します。
2. [SQS Queue] (SQS キュー) には、使用するキューを選択します。
3. (オプション) [Additional settings - optional] (追加設定 - オプション) では、以下を実行します。

   a. [Batch size - optional] (バッチサイズ - オプション) に、各バッチの最大レコード数を入力します。デフォルト値は 100 です。

   b. [Batch window - optional] (バッチウィンドウ - オプション) の場合、次に進む前にレコードを収集する最大秒数を入力します。

イベントフィルタリングの設定 (オプション)

パイプにフィルタリングを追加して、ソースからイベントのサブセットのみをターゲットに送信できます。

コンソールを使用してフィルタリングを設定するには

1. [Filtering] (フィルタリング) を選択します。
2. [サンプルイベント - オプション] には、イベントパターンの作成に使用できるサンプルイベントが表示されます。また、[独自のイベントを入力] を選択して独自のイベントを入力することもできます。
3. [イベントパターン] に、イベントのフィルタリングに使用するイベントパターンを入力します。フィルターの作成に関する詳細は、を参照してください。???

   以下は、[City] (都市) フィールドの値が [Seattle] のイベントのみを送信するイベントパターンの例です。

```
{
```

```
  "data": {
    "City": ["Seattle"]
  }
}
```

イベントがフィルタリングされたので、オプションのエンリッチメントとパイプのターゲットを追加できます。

イベントのエンリッチメントの定義 (オプション)

強化するイベントデータは、Lambda 関数、 AWS Step Functions ステートマシン、Amazon API ゲートウェイ、または API 送信先に送信できます。

エンリッチメントを選択するには

1. [Enrichment] (エンリッチメント) を選択します。
2. [Details] (詳細) の [Service] (サービス) で、エンリッチメントに使用したいサービスと関連設定を選択します。

データを送信する前にデータを変換して拡張することもできます。

(オプション) Input Transformer を定義するには

1. [Enrichment Input Transformer - optional] (エンリッチメント Input Transformer - オプション) を選択してください。
2. [Sample events/Event Payload] (サンプルイベント/イベントペイロード)では、サンプルイベントタイプを選択します。
3. [Transformer] (トランスフォーマー) には、サンプルイベントからフィールドへ参照する`"Event happened at <$.detail.field>."`場所`<$.detail.field>` など、トランスフォーマー構文を入力します。また、サンプルイベントのフィールドをダブルクリックしてトランスフォーマーに追加することもできます。
4. [Output] (出力) で、出力が希望どおりであることを確認します。

これでデータのフィルタリングと拡張が完了したので、次はイベントデータを送信するターゲットを定義する必要があります。

ターゲットの設定

ターゲットを設定するには

1. [Target] を選択します。
2. [Details] (詳細) の [Target service] (ターゲットサービス) で、ターゲットを選択します。表示されるフィールドは、選択したターゲットによって異なります。必要に応じて、このターゲットタイプに固有の情報を入力します。

ターゲットに送信する前にデータを変換することもできます。

(オプション) Input Transformer を定義するには

1. [Target Input Transformer - optional] (ターゲットインプットトランスフォーマー - オプション) を選択します。
2. [Sample events/Event Payload] (サンプルイベント/イベントペイロード)では、サンプルイベントタイプを選択します。
3. [Transformer] (トランスフォーマー) には、サンプルイベントからフィールドへ参照する`"Event happened at <$.detail.field>."`場所`<$.detail.field>` など、トランスフォーマー構文を入力します。また、サンプルイベントのフィールドをダブルクリックしてトランスフォーマーに追加することもできます。
4. [Output] (出力) で、出力が希望どおりであることを確認します。

これでパイプが設定されたので、その設定が正しく設定されていることを確認します。

パイプ設定の指定

パイプはデフォルトで有効ですが、無効にすることもできます。パイプのアクセス許可の指定、パイプのロギングの設定、タグの追加を行うこともできます。

パイプ設定を行うには

1. [Pipe settings] (パイプ設定) タブを選択します。
2. デフォルトでは、新しく作成されたパイプは作成されるとすぐに有効になります。無効なパイプを作成する場合は、[Activation] (アクティベーション) の [Activate pipe]] (パイプを有効化) で [Active] (有効) をオフにします。
3. [Permissions] (アクセス許可) の [Execution role] (実行ロール) で、次のいずれかを実行します。

a. EventBridge このパイプに新しい実行ロールを作成するには、[この特定のリソースの新しいロールを作成] を選択します。[Role name] (ロール名) では、ロール名をオプションで編集できます。

b. 既存の実行ロールを使用するには、[Use an existing role] (既存のロールを使用する) を選択します。[Logging role] (ログ記録ロール) でロールを選択します。

4. (オプション) Kinesis DynamoDB パイプソースとしてまたはストリームを指定した場合は、再試行ポリシーとデッドレターキュー (DLQ) を設定できます。

   [再試行ポリシーとデッドレターキュー – オプション] で、次の操作を行います。

   [Retry policy] (再試行ポリシー) で、以下の作業を行います。

   a. 再試行ポリシーを有効にする場合は、[Retry] (再試行) をオンにします。デフォルトでは、新しく作成されたパイプは、再試行ポリシーが有効になっていません。

   b. Maximum age of event (最大イベント有効期間) に、1 分 (00:01) から 24 時間 (24:00) の間の値を入力します。

   c. 再試行 で、0 ~ 185 の数値を入力します。

   d. デッドレターキュー (DLQ) を使用する場合は、[デッドレターキュー] を有効にして、希望する方法を選択し、使用するキューまたはトピックを選択します。デフォルトでは、新規作成したパイプは DLQ を使用しません。

5. (オプション) [ログ - オプション] で、 EventBridge Pipe から、サポートされているサービスにログ情報を送信する方法 (ログの設定方法を含む) を設定できます。

   パイプレコードのログ記録の詳細については、「???」を参照してください。

   CloudWatch デフォルトでは、ログレベルと同様に、ログがログの送信先として選択されます。ERROR そのため、デフォルトでは、 EventBridge Pipes CloudWatch ERROR は詳細レベルを含むログレコードを送信する新しいロググループを作成します。

   EventBridge Pipes がサポートされているログ宛先のいずれかにログレコードを送信するように設定するには、次の操作を行います。

   a. [ログ - オプション] で、ログレコードの配信先を選択します。

   b. [ログレベル] では、 EventBridge ログレコードに含める情報のレベルを選択します。ERROR ログレベルはデフォルトで選択されています。

      詳細については、「???」を参照してください。

c. EventBridge イベントペイロード情報やサービスリクエスト/レスポンス情報をログレコードに含める場合は、「実行データを含める」を選択します。

詳細については、「???」を参照してください。

d. 選択したログの送信先をそれぞれ設定します。

CloudWatch Logs ログの場合は、[CloudWatch ログ] で次の操作を行います。

- CloudWatch ロググループでは、 EventBridge 新しいロググループを作成するか、既存のロググループを選択するか、既存のロググループの ARN を指定するかを選択します。
- 新しいロググループを作成する場合は、必要に応じてロググループ名を編集します。

CloudWatch logs はデフォルトで選択されます。

Firehose ストリームログの場合は、[Firehose ストリームログ] Firehose でストリームを選択します。

Amazon S3 ログの場合は、[S3 ログ] で次の操作を行います。

- ログの送信先として使用するバケットの名前を入力します。
- AWS バケット所有者のアカウント ID を入力します。
- EventBridge で S3 オブジェクトの作成時に使用するプレフィックステキストを入力します。

  詳細については、「Amazon Simple Storage Service ユーザーガイド」の「プレフィックスを使用してオブジェクトを整理する」を参照してください。
- S3 EventBridge ログレコードのフォーマット方法を選択します。
  - `json`: JSON
  - `plain`: プレーンテキスト
  - `w3c`: W3C 拡張ログファイル形式

6. (オプション) [Tags - optional] (タグ-オプション) で、[Add new tag] (新しいタグを追加する) を選択し、ルールに対する 1 つまたは複数のタグを入力します。詳細については、「???」を参照してください。
7. [Create pipe] (パイプの作成) を選択します。

構成パラメータの検証

パイプが作成されたら、 EventBridge 以下の設定パラメータを検証します。

- IAM ロール — パイプのソースはパイプの作成後に変更できないため、指定された IAM EventBridge ロールがソースにアクセスできることを確認します。

> **Note**
>
> EventBridge エンリッチメントやターゲットはパイプの作成後に更新できるため、同じ検証は実行されません。

- バッチング — EventBridge ソースのバッチサイズがターゲットの最大バッチサイズを超えていないことを検証します。その場合は、 EventBridge バッチサイズを小さくする必要があります。さらに、ターゲットがバッチ処理をサポートしていない場合、ソースのバッチ処理を設定することはできません。 EventBridge
- エンリッチメント — サポートされるバッチサイズは 1 のみであるため、API Gateway と API デスティネーションエンリッチメントのバッチサイズが 1 EventBridge であることを検証します。

パイプの起動または停止

デフォルトでは、パイプは `Running` 作成時にイベントとして処理されます。

Amazon SQS、Kinesis、または DynamoDB ソースを使用してパイプを作成する場合、パイプの作成には通常 1 ~ 2 分かかることがあります。

Amazon MSK、セルフマネージド Apache Kafka、または Amazon MQ ソースを使用してパイプを作成する場合、パイプの作成には最大 10 分かかることがあります。

コンソールを使用してイベントを処理せずにパイプを作成するには

- [パイプをアクティブ化する] 設定をオフにします。

イベントをプログラムで処理せずにパイプを作成するには

- API コールで、`DesiredState` を `Stopped` に設定します。

コンソールを使用して既存のパイプを起動または停止するには

- [パイプ設定] タブの [アクティブ化] で、[パイプをアクティブ化する] の [アクティブ] をオンまたはオフにします。

既存のパイプをプログラムで起動または停止するには

- API コールで、`DesiredState` パラメータを `RUNNING` または `STOPPED` に設定します。

パイプが `STOPPED` のときからイベントを処理しなくなるまでに遅延が生じることがあります。

- Amazon SQS およびストリームソースの場合、この遅延は通常 2 分未満です。
- Amazon MQ および Apache Kafka ソースの場合、この遅延は最大 15 分となる場合があります。

Amazon EventBridge パイプソース

EventBridge Pipes は、さまざまなソースからイベントデータを受け取り、そのデータにオプションのフィルタとエンリッチメントを適用して、宛先に送信します。

EventBridge 送信元がパイプに送信されるイベントに順序を強制する場合、その順序は宛先へのプロセス全体を通して維持されます。

EventBridge Pipes AWS のソースとして以下のサービスを指定できます。

- Amazon DynamoDB ストリーム
- Amazon Kinesis Streams
- Amazon MQ ブローカー
- Amazon MSK ストリーム
- Amazon SQS キュー
- Apache カフカストリーム

  Apache Kafka ストリームをパイプソースとして指定する場合、自分で管理する Apache Kafka ストリームを指定することも、次のようなサードパーティプロバイダーが管理する Apache Kafka ストリームを指定することもできます。

  - Confluent Cloud
  - CloudKarafka

- Redpanda

ソースとしての Amazon DynamoDB Streams

EventBridge Pipes を使用して DynamoDB Streams のレコードを受け取ることができます。その後、オプションでこれらのレコードをフィルタリングまたは拡張してから、ターゲットに送信して処理できます。Amazon DynamoDB Streams に固有の設定があり、パイプをセットアップするときに選択できます。EventBridge Pipes は、データを送信先に送信するときに、データストリームのレコードの順序を維持します。

> **Important**
>
> パイプのソースである DynamoDB ストリームを無効にすると、ストリームを再度有効にしても、そのパイプは使用できなくなります。この理由は、以下のとおりです。
>
> - ソースが無効になっているパイプを停止、開始、または更新することはできません。
> - 作成後のパイプを新しいソースで更新することはできません。DynamoDB ストリームを再度有効にすると、そのストリームには新しい Amazon リソースネーム (ARN) が割り当てられ、パイプとの関連付けはなくなります。
>
> DynamoDB ストリームを再度有効にする場合は、ストリームの新しい ARN を使用して新しいパイプを作成する必要があります。

イベントの例

次のサンプルイベントは、パイプが受信した情報を示しています。このイベントを使用して、イベントパターンを作成およびフィルタリングしたり、入力変換を定義したりできます。すべてのフィールドをフィルタリングできるわけではありません。フィルターできるフィールドの詳細については、「???」を参照してください。

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

ポーリングストリームとバッチストリーム

EventBridge は、レコードの DynamoDB Streams にあるシャードを 1 秒あたり 4 回の基本レートでポーリングします。レコードが利用可能になると、EventBridge はイベントを処理して結果を待機します。処理が成功すると、EventBridge は、さらに多くのレコードを受け取るまでポーリングを再開します。

デフォルトで、EventBridge はレコードが使用可能になると同時にパイプを呼び出します。EventBridge がソースから読み取るバッチにレコードが 1 つしかない場合、1 つのイベントだけを処理します。少数のレコードを処理しないようにするには、バッチ処理ウィンドウを設定して、最大 5 分間レコードをバッファリングするようにパイプに指示できます。イベントを処理する前に、EventBridge は、完全なバッチを収集する、バッチ処理ウィンドウの期限が切れる、またはバッチが 6 MB のペイロード制限に到達するまでソースからのレコードの読み取りを継続します。

また、各シャードから複数のバッチを並行して処理することで、並行性を高めることもできます。EventBridge は、各シャードで最大 10 個のバッチを同時に処理できます。シャードあたりの同時バッチの数を増やしても、EventBridge はパーティションキーレベルでの順序立った処理を確実に行います。

`ParallelizationFactor` 設定を使用することで、複数のパイプの同時実行により、Kinesis または DynamoDB データストリームの 1 つのシャードを処理します。EventBridge がシャードからポーリングする同時バッチの数は、1 (デフォルト) ~ 10 の並列化係数で指定できます。例えば、`ParallelizationFactor` を 2 に設定すると、最大 200 個の EventBridge Pipe の同時実行により、100 個の Kinesis データシャードを処理できます。これにより、データボリュームが揮発性で `IteratorAge` が高いときに処理のスループットをスケールアップすることができます。Kinesis 集約を使用している場合、並列化係数は機能しません。

ポーリングとストリームの開始位置

パイプの作成時と更新時のストリームソースポーリングは、最終的に一貫性があることに注意してください。

- パイプ作成中、ストリームからのイベントのポーリングが開始されるまでに数分かかること場合があります。
- ソースのポーリング構成をパイプで更新している間、ストリームのポーリングイベントを停止して再開するまでに数分かかることがあります。

つまり、LATEST をストリームの開始位置として指定すると、パイプ作成または更新中に送信されるイベントをパイプが見逃す可能性があります。イベントを見逃さないようにするには、ストリームの開始位置を TRIM_HORIZON として指定します。

バッチ項目の失敗の報告

EventBridge がソースからストリーミングデータを使用および処理する場合、デフォルトでは、バッチが完全に成功した場合にのみ、バッチの最大シーケンス番号に チェックポイントが設定されます。正常に処理されたメッセージが失敗したバッチで再処理されないようにするには、成功したメッセージと失敗したメッセージを示すオブジェクトを返すようにエンリッチメントまたはターゲットを設定できます。これを部分的なバッチレスポンスと呼びます。

詳細については、「???」を参照してください。

成功条件と失敗の条件

次のいずれかを返すと、EventBridge は完全な成功として処理します:

- 空の `batchItemFailure` リスト
- null の `batchItemFailure` リスト
- 空の `EventResponse`
- null の `EventResponse`

次のいずれかを返すと、EventBridge は完全な失敗として処理します:

- 空の文字列 `itemIdentifier`
- ヌル `itemIdentifier`
- 不正なキー名を持つ `itemIdentifier`

EventBridge は、再試行戦略に基づいて失敗を再試行します。

Amazon Kinesis ストリームをソースとする場合

EventBridge Pipes を使用して Kinesis データストリームでレコードを受信できます。その後、オプションでこれらのレコードをフィルタリングまたは拡張してから、処理可能な送信先のいずれかに送信できます。パイプをセットアップするときに選択できる Kinesis 固有の設定があります。EventBridge Pipes は、データを送信先に送信するときに、データストリームのレコードの順序を維持します。

Kinesis データストリームは、シャードのセットです。各シャードには、一連のデータレコードが含まれます。コンシューマーは、Kinesis データストリームからのデータを処理するアプリケーションです。EventBridge Pipe を共有スループットコンシューマー (標準イテレーター) にマップすることも、拡張ファンアウトを使用する専用スループットコンシューマーにマップすることもできます。

標準イテレーターの場合、EventBridge は HTTP プロトコルを使用して、Kinesis ストリームの各シャードにレコードがあるかどうかをポーリングします。このパイプでは、シャードの他のコンシューマーと読み取りスループットを共有します。

レイテンシーを最小限に抑え、読み取りスループットを最大化するために、拡張ファンアウトを使用するデータストリームコンシューマーを作成できます。ストリームコンシューマーは、ストリームから読み取る他のアプリケーションに影響を及ぼさないように、専用の接続を各シャードに割り当てます。専用のスループットは、多数のアプリケーションで同じデータを読み取っている場合や、大きなレコードでストリームを再処理する場合に役立ちます。Kinesis は、HTTP/2 経由で EventBridge にレコードをプッシュします。Kinesis Data Streams の詳細については、「Amazon Kinesis Data Streams からのデータの読み取り」を参照してください。

イベントの例

次のサンプルイベントは、パイプが受信した情報を示しています。このイベントを使用して、イベントパターンを作成およびフィルタリングしたり、入力変換を定義したりできます。すべてのフィールドをフィルタリングできるわけではありません。フィルターできるフィールドの詳細については、「???」を参照してください。

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

ポーリングストリームとバッチストリーム

EventBridge は、レコードの Kinesis Streams にあるシャードを 1 秒あたり 4 回の基本レートでポーリングします。レコードが利用可能になると、EventBridge はイベントを処理して結果を待機します。処理が成功すると、EventBridge は、さらに多くのレコードを受け取るまでポーリングを再開します。

デフォルトで、EventBridge はレコードが使用可能になると同時にパイプを呼び出します。EventBridge がソースから読み取るバッチにレコードが 1 つしかない場合、1 つのイベントだけを処理します。少数のレコードを処理しないようにするには、バッチ処理ウィンドウを設定して、最大 5 分間レコードをバッファリングするようにパイプに指示できます。イベントを処理する前に、EventBridge は、完全なバッチを収集する、バッチ処理ウィンドウの期限が切れる、またはバッチが 6 MB のペイロード制限に到達するまでソースからのレコードの読み取りを継続します。

また、各シャードから複数のバッチを並行して処理することで、並行性を高めることもできます。EventBridge は、各シャードで最大 10 個のバッチを同時に処理できます。シャードあたりの同

時バッチの数を増やしても、EventBridge はパーティションキーレベルでの順序立った処理を確実に行います。

ParallelizationFactor 設定を使用することで、複数のパイプの同時実行により、Kinesis または DynamoDB データストリームの 1 つのシャードを処理します。EventBridge がシャードからポーリングする同時バッチの数は、1 (デフォルト) ~ 10 の並列化係数で指定できます。例えば、ParallelizationFactor を 2 に設定すると、最大 200 個の EventBridge Pipe の同時実行により、100 個の Kinesis データシャードを処理できます。これにより、データボリュームが揮発性で IteratorAge が高いときに処理のスループットをスケールアップすることができます。Kinesis 集約を使用している場合、並列化係数は機能しません。

ポーリングとストリームの開始位置

パイプの作成時と更新時のストリームソースポーリングは、最終的に一貫性があることに注意してください。

- パイプ作成中、ストリームからのイベントのポーリングが開始されるまでに数分かかること場合があります。
- ソースのポーリング構成をパイプで更新している間、ストリームのポーリングイベントを停止して再開するまでに数分かかることがあります。

つまり、LATEST をストリームの開始位置として指定すると、パイプ作成または更新中に送信されるイベントをパイプが見逃す可能性があります。イベントを見逃さないようにするには、ストリームの開始位置を TRIM_HORIZON または AT_TIMESTAMP として指定します。

バッチ項目の失敗の報告

EventBridge がソースからストリーミングデータを使用および処理する場合、デフォルトでは、バッチが完全に成功した場合にのみ、バッチの最大シーケンス番号に チェックポイントが設定されます。正常に処理されたメッセージが失敗したバッチで再処理されないようにするには、成功したメッセージと失敗したメッセージを示すオブジェクトを返すようにエンリッチメントまたはターゲットを設定できます。これを部分的なバッチレスポンスと呼びます。

詳細については、「???」を参照してください。

成功条件と失敗の条件

次のいずれかを返すと、EventBridge は完全な成功として処理します:

- 空の batchItemFailure リスト

- null の batchItemFailure リスト
- 空の EventResponse
- null の EventResponse

次のいずれかを返すと、EventBridge は完全な失敗として処理します:

- 空の文字列 itemIdentifier
- ヌル itemIdentifier
- 不正なキー名を持つ itemIdentifier

EventBridge は、再試行戦略に基づいて失敗を再試行します。

ソースとしての Amazon MQ メッセージブローカー

EventBridge Pipes を使用して、Amazon MQ メッセージブローカーからレコードを受信できます。その後、オプションでこれらのレコードをフィルタリングまたは拡張してから、処理可能な送信先のいずれかに送信できます。パイプを設定するときに選択できる Amazon MQ 固有の設定があります。 EventBridge Pipes は、そのデータを宛先に送信するときに、メッセージブローカーからのレコードの順序を維持します。

Amazon MQ は、Apache ActiveMQ および RabbitMQ 用のマネージドメッセージブローカーサービスです。メッセージブローカーを使用すると、ソフトウェアアプリケーションおよびコンポーネントは、さまざまなプログラミング言語、オペレーティングシステム、および、トピックまたはキューをイベント送信先とする正式なメッセージングプロトコルを使って、通信できるようになります。

また Amazon MQ は、ActiveMQ か RabbitMQ ブローカーをインストールすることにより、ユーザーに代わって Amazon Elastic Compute Cloud (Amazon EC2)インスタンスを管理することもできます。ブローカーをインストールすると、さまざまなネットワークトポロジやその他のインフラストラクチャのニーズがインスタンスに提供されます。

Amazon MQ ソースには、次の設定制限があります。

- クロスアカウント – クロスアカウント処理は EventBridge サポートされていません。を使用して EventBridge 、別の AWS アカウントにある Amazon MQ メッセージブローカーからのレコードを処理することはできません。
- 認証 – ActiveMQ では、ActiveMQ SimpleAuthenticationPluginのみがサポートされています。RabbitMQ の場合、PLAIN 認証メカニズムのみサポートされています。認証情報を管理する

には、 AWS Secrets Managerを使用してください。ActiveMQ 認証の詳細については、「Amazon MQ デベロッパーガイド」の「 Integrating ActiveMQ brokers with LDAP」を参照してください。

- 接続クォータ - ブローカーは、ワイヤレベルプロトコルごとに最大の接続可能数を持っています。このクォータは、ブローカーインスタンスタイプに基づいています。これらの制限の詳細については、「Amazon MQ デベロッパーガイド」の「*Amazon MQ のクォータ」 の「ブローカー」のセクションを参照してください。
- 接続 - ブローカーをパブリックまたはプライベートの Virtual Private Cloud (VPC) に作成できます。プライベート VPC の場合、 パイプが VPC にアクセスしてメッセージを受信する必要があります。
- イベント送信先 - キューの送信先のみがサポートされます。ただし、仮想トピックを使用することができます。仮想トピックは、パイプと対話するとき、内部的にトピックとして、かつ外部的にキューとして動作します。詳細については、Apache ActiveMQ ウェブサイトの Virtual Destinations および RabbitMQ ウェブサイトの Virtual Hostsを参照してください。
- ネットワークトポロジ - ActiveMQ の場合、パイプに対して、 1 つの単一インスタンスまたはスタンバイブローカーがサポートされます。RabbitMQ の場合、パイプごとに、単一インスタンスブローカーまたはクラスターデプロイメントがサポートされます。単一インスタンスブローカーには、フェイルオーバーエンドポイントが必要です。これらのブローカーデプロイメントモードの詳細については、「Amazon MQ デベロッパーガイド」の「Active MQ ブローカーアーキテクチャ」および「Rabbit MQ ブローカーアーキテクチャ」を参照してください。
- プロトコル — サポートされるプロトコルは、使用する Amazon MQ の統合のタイプによって異なります。
  - ActiveMQ 統合の場合、 EventBridge は OpenWire/Java Message Service (JMS) プロトコルを使用してメッセージを消費します。メッセージの使用は、他のプロトコルではサポートされていません。 EventBridge は JMS プロトコル内の TextMessageおよび BytesMessageオペレーションのみをサポートします。 OpenWire プロトコルの詳細については、Apache ActiveMQ ウェブサイトのOpenWire「」を参照してください。
  - RabbitMQ 統合の場合、 EventBridge は AMQP 0-9-1 プロトコルを使用してメッセージを消費します。その他のプロトコルは、メッセージの使用をサポートしていません。RabbitMQ による AMQP 0-9-1 プロトコルの実装の詳細については、RabbitMQ ウェブサイトの AMQP 0-9-1 Complete Reference Guide を参照してください。

EventBridge は、Amazon MQ がサポートする最新バージョンの ActiveMQ と RabbitMQ を自動的にサポートします。 Amazon MQ サポートされている最新バージョンについては、「Amazon MQ デベロッパーガイド」の「Amazon MQ リリースノート」を参照してください。

> **Note**
>
> デフォルトでは、Amazon MQ には毎週、ブローカー用のメンテナンスウィンドウがあります。その期間中、ブローカーは利用できません。スタンバイのないブローカー EventBridge の場合、ウィンドウが終了するまでメッセージを処理しません。

イベントの例

次のサンプルイベントは、パイプが受信した情報を示しています。このイベントを使用して、イベントパターンを作成およびフィルタリングしたり、入力変換を定義したりできます。すべてのフィールドをフィルタリングできるわけではありません。フィルターできるフィールドの詳細については、「???」を参照してください。

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=="
  }
]
```

コンシューマーグループ

Amazon MQ とやり取りするために、 は Amazon MQ ブローカーから読み取ることができるコンシューマーグループ EventBridge を作成します。コンシューマーグループは、パイプ UUID と同じ ID で作成されます。

Amazon MQ ソースの場合、 はレコードをまとめて EventBridge バッチ処理し、単一のペイロードで関数に送信します。動作を制御するには、バッチ処理ウィンドウとバッチサイズを設定できます。EventBridge は、次のいずれかが発生するまでメッセージをプルします。

- 処理されたレコードのペイロードサイズは最大 6 MB に達します。
- バッチウィンドウの有効期限が切れる。
- レコード数がバッチサイズ全体に達します。

EventBridge はバッチを単一のペイロードに変換し、関数を呼び出します。メッセージは永続化も逆シリアル化もされません。その代わりに、コンシューマーグループはそれらをバイトの BLOB として取得します。次に、base64 でエンコードして JSON ペイロードにします。パイプがバッチ内のいずれかのメッセージに対してエラーを返した場合、 は処理が成功するか、メッセージが期限切れになるまで、メッセージのバッチ全体を EventBridge 再試行します。

ネットワーク構成

デフォルトではAmazon MQ、 ブローカーは `PubliclyAccessible` フラグを false に設定して作成されます。ブローカーにパブリック IP アドレスが与えられるのは、`PubliclyAccessible` が true に設定されている場合のみです。パイプでフルアクセスする場合、ブローカーはパブリックエンドポイントを使用するか、VPC へアクセスを提供する必要があります。

Amazon MQ ブローカーがパブリックアクセス可能でない場合は、ブローカーに関連付けられた Amazon Virtual Private Cloud (Amazon VPC) リソースにアクセスできる EventBridge 必要があります。

- Amazon MQ ブローカーの VPC にアクセスするには、ソースのサブネットにアウトバウンドインターネットアクセス EventBridge を使用できます。パブリックサブネットの場合、これはマネージド NAT ゲートウェイである必要があります。プライベートサブネットの場合は NAT ゲートウェイでも、独自の NAT でもかまいません。NAT にパブリック IP アドレスが割り当てられ、インターネットに接続できることを確認します。
- EventBridge Pipes は を介したイベント配信もサポートしているためAWS PrivateLink、 Amazon Virtual Private Cloud （ Amazon VPC) にあるイベントソースから Pipes ターゲットにイベントを送信できます。パブリックインターネットを経由する必要はありません。Pipes を使用すると、インターネットゲートウェイをデプロイしたり、ファイアウォールルールを設定したり、プロキシサーバーを設定したりすることなく、 Amazon Managed Streaming for Apache Kafka （ Amazon MSK ）、セルフマネージド Apache Kafka、およびプライベートサブネットに存在する Amazon MQ ソースからポーリングできます。

  VPC エンドポイントを設定するには、「 ユーザーガイド」の「VPC エンドポイントの作成AWS PrivateLink 」を参照してください。サービス名で、 を選択しますcom.amazonaws.*region*.pipes-data。

Amazon VPC セキュリティグループは、少なくとも以下のルールを使用して設定してください。

- インバウンドルール – ソースに指定されたセキュリティグループの Amazon MQ ブローカーポート上のすべてのトラフィックを許可します。
- アウトバウンドルール – すべての送信先に対して、ポート 443 上のすべてのトラフィックを許可します。ソースに指定されたセキュリティグループの Amazon MQ ブローカーポート上のすべてのトラフィックを許可します。

  ブローカーポートには以下が含まれます。

  - プレーンテキストの場合は 9092

- TLS の場合は 9094
- SASL の場合は 9096
- IAM 用 9098

> **Note**
>
> Amazon VPC の設定は、Amazon MQ API を使用して検出できます。セットアップ中に設定する必要はありません。

Amazon Managed Streaming for Apache Kafka トピックをソースとして使用する場合

EventBridge Pipes を使用して、Amazon Managed Streaming for Apache Kafka (Amazon MSK) トピックからレコードを受信できます。オプションでこれらのレコードをフィルタリングまたは拡張してから、処理可能な送信先のいずれかに送信できます。パイプを設定するときに選択できる Amazon MSK 固有の設定があります。 EventBridge Pipes は、そのデータを宛先に送信するときに、メッセージブローカーからのレコードの順序を維持します。

Amazon MSK は、Apache Kafka をストリーミングデータの処理に使用するアプリケーションを構築および実行できるようにするフルマネージドサービスです。Amazon MSK は、Apache Kafka を実行するクラスターのセットアップ、スケーリング、管理を簡素化します。Amazon MSK では、複数のアベイラビリティーゾーンと AWS Identity and Access Management (IAM) によるセキュリティのためにアプリケーションを設定できます。Amazon MSK は、Kafka の複数のオープンソースバージョンをサポートします。

ソースとしての Amazon MSK は、Amazon Simple Queue Service (Amazon SQS) または Amazon Kinesis の使用と同様に動作します。 EventBridge は、ソースからの新しいメッセージを内部的にポーリングし、ターゲットを同期的に呼び出します。 EventBridge はメッセージをバッチで読み取り、これらをイベントペイロードとして関数に提供します。最大バッチサイズは調整可能です。(デフォルト値は 100 メッセージ)。

Apache Kafka ベースのソースの場合、 はバッチ処理ウィンドウやバッチサイズなどの処理コントロールパラメータ EventBridge をサポートします。

EventBridge は、パーティションごとにメッセージを順番に読み取ります。が各バッチ EventBridge を処理すると、そのバッチ内のメッセージのオフセットがコミットされます。パイプのターゲットが

バッチ内のいずれかのメッセージに対してエラーを返した場合、 は処理が成功するか、メッセージが期限切れになるまで、メッセージのバッチ全体を EventBridge 再試行します。

EventBridge は、ターゲットを呼び出すときに、イベントのメッセージのバッチを送信します。イベントペイロードにはメッセージの配列が含まれています。各配列項目には、Amazon MSK トピックとパーティション識別子の詳細が、タイムスタンプおよび base64 でエンコードされたメッセージとともに含まれています。

イベントの例

次のサンプルイベントは、パイプが受信した情報を示しています。このイベントを使用して、イベントパターンを作成およびフィルタリングしたり、入力変換を定義したりできます。すべてのフィールドをフィルタリングできるわけではありません。フィルターできるフィールドの詳細については、「???」を参照してください。

```
[
  {
    "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
```

```
        ]
      }
    ]
  }
]
```

ポーリングとストリームの開始位置

パイプの作成時と更新時のストリームソースポーリングは、最終的に一貫性があることに注意してください。

- パイプ作成中、ストリームからのイベントのポーリングが開始されるまでに数分かかること場合があります。
- ソースのポーリング構成をパイプで更新している間、ストリームのポーリングイベントを停止して再開するまでに数分かかることがあります。

つまり、LATEST をストリームの開始位置として指定すると、パイプ作成または更新中に送信されるイベントをパイプが見逃す可能性があります。イベントを見逃さないようにするには、ストリームの開始位置を TRIM_HORIZON として指定します。

MSK クラスター認証

EventBridge には、Amazon MSK クラスターへのアクセス、レコードの取得、その他のタスクの実行のためのアクセス許可が必要です。Amazon MSK は、MSK クラスターへのクライアントアクセスを制御するためのいくつかのオプションをサポートしています。どの認証方法がいつ使用されるかについての詳細は、「???」を参照してください。

クラスターアクセスオプション

- 非認証アクセス
- SASL/SCRAM 認証
- IAM ロールベースの認証
- 相互 TLS 認証
- mTLS シークレットの設定
- がブートストラップブローカー EventBridge を選択する方法

非認証アクセス

開発には非認証アクセスのみを使用することをお勧めします。非認証アクセスは、クラスターの IAM ロールベース認証が無効になっている場合にのみ機能します。

SASL/SCRAM 認証

Amazon MSK は、Transport Layer Security (TLS) 暗号化を使用した Simple Authentication and Security Layer/Salted Challenge Response Authentication Mechanism (SASL/SCRAM) 認証をサポートしています。がクラスターに接続する EventBridge には、認証情報 (サインイン認証情報) を AWS Secrets Manager シークレットに保存します。

Secrets Manager の使用に関する詳細については、「Amazon Managed Streaming for Apache Kafka デベロッパーガイド」の「AWS Secrets Managerを使用したユーザーネームとパスワードの認証」を参照してください。

Amazon MSK は SASL/PLAIN 認証をサポートしません。

IAM ロールベースの認証

IAM を使用して、MSK クラスターに接続するクライアントのアイデンティを認証することができます。MSK クラスターで IAM 認証がアクティブで、認証用のシークレットを指定しない場合、EventBridge は自動的にデフォルトで IAM 認証を使用します。IAM ユーザーまたはロールベースのポリシーを作成してデプロイするには、IAM コンソール、または API を使用します。詳細については、「Amazon Managed Streaming for Apache Kafka Developer Guide」(Amazon Managed Streaming for Apache Kafka デベロッパーガイド) の「IAM access control」(IAM アクセスコントロール) を参照してください。

EventBridge が MSK クラスターに接続し、レコードを読み取り、その他の必要なアクションを実行できるようにするには、パイプの実行ロールに次のアクセス許可を追加します。

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

これらの許可は、特定のクラスター、トピック、およびグループにスコープできます。詳細については、「Amazon Managed Streaming for Apache Kafka Developer Guide」(Amazon Managed Streaming for Apache Kafka デベロッパーガイド) の「Amazon MSK Kafka actions」(Amazon MSK Kafka アクション) を参照してください。

相互 TLS 認証

相互 TLS (mTLS) は、クライアントとサーバー間の双方向認証を提供します。クライアントは、サーバーによるクライアントの検証のためにサーバーに証明書を送信し、サーバーは、クライアントによるサーバーの検証のためにクライアントに証明書を送信します。

Amazon MSK の場合、 EventBridge はクライアントとして機能します。MSK クラスター内のブローカー EventBridge で認証するように、クライアント証明書を (Secrets Manager のシークレットとして) 設定します。クライアント証明書は、 サーバーのトラストストア内の認証局 (CA) によって署名される必要があります。MSK クラスターはサーバー証明書を に送信 EventBridge して、 でブローカーを認証します EventBridge。サーバー証明書は、 AWS 信頼ストアにある CA によって署名される必要があります。

Amazon MSK は自己署名サーバー証明書をサポートしていません。Amazon MSK のすべてのブローカーは、デフォルトで が EventBridge 信頼する Amazon Trust Services CAs によって署名されたパブリック証明書を使用するためです。

Amazon MSK のための mTLS に関する詳細については、「Amazon Managed Streaming for Apache Kafka Developer Guide」(Amazon Managed Streaming for Apache Kafka デベロッパーガイド) の「Mutual TLS Authentication」(相互 TLS 認証) を参照してください。

mTLS シークレットの設定

CLIENT_CERTICATE_TLS_AUTH シークレットは、証明書フィールドとプライベートキーフィールドを必要とします。暗号化されたプライベートキーの場合、シークレットはプライベートキーのパスワードを必要とします。証明書とプライベートキーは、どちらも PEM 形式である必要があります。

> **Note**
>
> EventBridge は、PBES1 (PBES2 はサポートしていません) プライベートキー暗号化アルゴリズムをサポートします。

証明書フィールドには、クライアント証明書で始まり、その後に中間証明書が続き、ルート証明書で終わる証明書のリストが含まれている必要があります。各証明書は、以下の構造を使用した新しい行で始める必要があります。

```
-----BEGIN CERTIFICATE-----
        <certificate contents>
-----END CERTIFICATE-----
```

Secrets Manager は最大 65,536 バイトのシークレットをサポートします。これは、長い証明書チェーンにも十分な領域です。

プライベートキーは、以下の構造を使用した PKCS #8 形式にする必要があります。

```
-----BEGIN PRIVATE KEY-----
         <private key contents>
-----END PRIVATE KEY-----
```

暗号化されたプライベートキーには、以下の構造を使用します。

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
          <private key contents>
-----END ENCRYPTED PRIVATE KEY-----
```

以下は、暗号化されたプライベートキーを使用する mTLS 認証のシークレットの内容を示す例です。暗号化されたプライベートキーの場合は、シークレットにプライベートキーのパスワードを含めます。

```
{
 "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-----"
}
```

がブートストラップブローカー EventBridge を選択する方法

EventBridge は、クラスターで使用可能な認証方法と、認証にシークレットを提供するかどうかに基づいて、ブートストラップブローカーを選択します。mTLS または SASL/SCRAM のシークレットを指定すると、 はその認証方法 EventBridge を自動的に選択します。シークレットを指定しない場合、 EventBridge はクラスターでアクティブな最も強力な認証方法を選択します。がブローカー EventBridge を選択する際の優先度の順序を、最も強力な認証から最も弱い認証まで次に示します。

- mTLS (mTLS 用のシークレットを提供)
- SASL/SCRAM (SASL/SCRAM 用のシークレットを提供)
- SASL IAM (シークレットが提供されておらず、IAM 認証がアクティブ)
- 非認証の TLS (シークレットが提供されておらず、IAM 認証も非アクティブ)
- プレーンテキスト (シークレットが提供されておらず、IAM 認証と非認証 TLS の両方が非アクティブ)

> **Note**
>
> が最も安全なブローカータイプに接続 EventBridge できない場合、 は別の (弱い) ブローカータイプに接続しようとしません。より弱いブローカータイプ EventBridge を選択する場合は、クラスターでより強力な認証方法をすべて無効にします。

ネットワーク構成

EventBridge は、Amazon MSK クラスターに関連付けられた Amazon Virtual Private Cloud (Amazon VPC) リソースにアクセスできる必要があります。

- Amazon MSK クラスターの VPC にアクセスするには、ソースのサブネットにアウトバウンドインターネットアクセス EventBridge を使用できます。パブリックサブネットの場合、これはマネージド NAT ゲートウェイである必要があります。プライベートサブネットの場合は NAT ゲートウェイでも、独自の NAT でもかまいません。NAT にパブリック IP アドレスが割り当てられ、インターネットに接続できることを確認します。
- EventBridge Pipes は を介したイベント配信もサポートしているためAWS PrivateLink、 Amazon Virtual Private Cloud （ Amazon VPC) にあるイベントソースから Pipes ターゲットにイベントを送信できます。パブリックインターネットを経由する必要はありません。Pipes を使用すると、インターネットゲートウェイをデプロイしたり、ファイアウォールルールを設定したり、プロキシサーバーを設定したりすることなく、 Amazon Managed Streaming for Apache Kafka （ Amazon MSK ）、セルフマネージド Apache Kafka、およびプライベートサブネットに存在する Amazon MQ ソースからポーリングできます。

  VPC エンドポイントを設定するには、「 ユーザーガイド」の「VPC エンドポイントの作成AWS PrivateLink 」を参照してください。サービス名で、 を選択しますcom.amazonaws.*region*.pipes-data。

Amazon VPC セキュリティグループは、少なくとも以下のルールを使用して設定してください。

- インバウンドルール — ソースに指定されたセキュリティグループの Amazon MSK ブローカーポート上のすべてのトラフィックを許可します。
- アウトバウンドルール – すべての送信先に対して、ポート 443 上のすべてのトラフィックを許可します。ソースに指定されたセキュリティグループの Amazon MSK ブローカーポート上のすべてのトラフィックを許可します。

ブローカーポートには以下が含まれます。

- プレーンテキストの場合は 9092
- TLS の場合は 9094
- SASL の場合は 9096
- IAM 用 9098

> **Note**
>
> Amazon VPC の設定は、Amazon MSK API を使用して検出できます。セットアップ中に設定する必要はありません。

カスタマイズ可能なコンシューマーグループ ID

Apache Kafka をソースとして設定する場合、コンシューマーグループIDを指定できます。このコンシューマーグループ ID は、パイプを結合したい Apache Kafka コンシューマーグループの既存の識別子です。この機能を使用して、進行中の Apache Kafka レコード処理設定を他のコンシューマーから に移行できます EventBridge。

コンシューマーグループ ID を指定し、そのコンシューマーグループ内に他のアクティブなポーラーが存在する場合、Apache Kafka はすべてのコンシューマーにメッセージを配信します。つまり、Apache Kafka EventBridge トピックのすべてのメッセージを受信しません。トピック内のすべてのメッセージ EventBridge を処理する場合は、そのコンシューマーグループ内の他のポーラーをオフにします。

さらに、コンシューマーグループ ID を指定し、Apache Kafka が同じ ID を持つ有効な既存のコンシューマーグループを検索すると、 はパイプの `StartingPosition`パラメータ EventBridge を無視します。代わりに、 はコンシューマーグループのコミットされたオフセットに従ってレコードの処理 EventBridge を開始します。コンシューマーグループ ID を指定し、Apache Kafka が既存のコンシューマーグループを見つけられない場合、 は指定された を使用してソース EventBridge を設定しますStartingPosition。

指定するコンシューマーグループ ID は、すべての Apache Kafka イベントソースの中で一意でなければなりません。コンシューマーグループ ID を指定してパイプを作成した後は、この値を更新することはできません。

Amazon MSK ソースの Auto Scaling

最初に Amazon MSK ソースを作成すると、 は Apache Kafka トピック内のすべてのパーティションを処理するために 1 つのコンシューマーを EventBridge 割り当てます。各コンシューマーには、増加したワークロードを処理するために同時実行される複数のプロセッサがあります。さらに、 は、ワークロードに基づいてコンシューマーの数 EventBridge を自動的にスケールアップまたはスケールダウンします。各パーティションでメッセージの順序を保つため、コンシューマーの最大数は、トピック内のパーティションあたり 1 つとなっています。

1 分間隔で、 はトピック内のすべてのパーティションのコンシューマーオフセットラグ EventBridge を評価します。遅延が高すぎる場合、パーティションはメッセージを処理 EventBridge できるよりも速くメッセージを受信しています。必要に応じて、コンシューマーをトピック EventBridge に追加または削除します。コンシューマーを追加または削除するスケーリングプロセスは、評価から 3 分以内に行われます。

ターゲットが過負荷になっている場合、 はコンシューマーの数 EventBridge を減らします。このアクションにより、コンシューマーが取得しパイプに送信するメッセージの数が減り、パイプへのワークロードが軽減されます。

ソースとしての Apache Kafka ストリーム

Apache Kafka は、データパイプラインやストリーミング分析などのワークロードをサポートする、オープンソースのイベントストリーミングプラットフォームです。Amazon Managed Streaming for Apache Kafka (Amazon MSK) またはセルフマネージド Apache Kafka クラスターを使用できます。用語 AWS では、セルフマネージドクラスターとは、 によってホストされていないすべての Apache Kafka クラスターを指します AWS。これには、自分で管理するクラスターと、、、 Confluent Cloud CloudKarafkaなどのサードパーティープロバイダーによってホストされるクラスターの両方が含まれますRedpanda。

クラスターの他の AWS ホスティングオプションの詳細については、 AWS ビッグデータブログの「で Apache Kafka を実行するためのベストプラクティス AWS」を参照してください。

ソースとしての Apache Kafka は、Amazon Simple Queue Service (Amazon SQS) または Amazon Kinesis の使用と同様に動作します。 EventBridge は、ソースからの新しいメッセージを内部的にポーリングし、ターゲットを同期的に呼び出します。 EventBridge はメッセージをバッチで読み取り、これらをイベントペイロードとして関数に提供します。最大バッチサイズは調整可能です。(デフォルト値は 100 メッセージ)。

Apache Kafka ベースのソースの場合、 はバッチ処理ウィンドウやバッチサイズなどの処理コントロールパラメータ EventBridge をサポートします。

EventBridge は、パイプを呼び出すときにイベントパラメータにメッセージのバッチを送信します。イベントペイロードにはメッセージの配列が含まれています。各配列項目には、Apache Kafka トピックと Apache Kafka パーティション識別子の詳細が、タイムスタンプおよび base64 でエンコードされたメッセージとともに含まれています。

イベントの例

次のサンプルイベントは、パイプが受信した情報を示しています。このイベントを使用して、イベントパターンを作成およびフィルタリングしたり、入力変換を定義したりできます。すべてのフィールドをフィルタリングできるわけではありません。フィルターできるフィールドの詳細については、「???」を参照してください。

```
[
  {
    "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 クラスター認証

EventBridge Pipes は、セルフマネージド Apache Kafka クラスターで認証するためのいくつかの方法をサポートしています。これらのサポートされる認証方法のいずれかを使用するように、Apache Kafka クラスターを設定しておいてください。Apache Kafka セキュリティの詳細については、Apache Kafka ドキュメントの「Security」(セキュリティ) セクションを参照してください。

VPC アクセス

VPC 内の Apache Kafka ユーザーのみが Apache Kafka ブローカーにアクセスできるセルフマネージド Apache Kafka 環境を使用している場合は、Apache Kafka ソースで Amazon Virtual Private Cloud (Amazon VPC) を設定する必要があります。

SASL/SCRAM 認証

EventBridge Pipes は、Transport Layer Security (TLS) 暗号化を使用した Simple Authentication and Security Layer/Salted Challenge Response Authentication Mechanism (SASL/SCRAM) 認証をサポートしています。 EventBridge Pipes は暗号化された認証情報を送信してクラスターで認証します。SASL/SCRAM 認証の詳細については、「RFC 5802」を参照してください。

EventBridge Pipes は、TLS 暗号化による SASL/PLAIN 認証をサポートしています。SASL/PLAIN 認証では、 EventBridge Pipes は認証情報をクリアテキスト (暗号化されていない) としてサーバーに送信します。

SASL 認証の場合は、サインイン認証情報をシークレットとして AWS Secrets Managerに保存します。

相互 TLS 認証

相互 TLS (mTLS) は、クライアントとサーバー間の双方向認証を提供します。クライアントは、サーバーによるクライアントの検証のためにサーバーに証明書を送信し、サーバーは、クライアントによるサーバーの検証のためにクライアントに証明書を送信します。

セルフマネージド Apache Kafka では、 EventBridge Pipes がクライアントとして機能します。Apache Kafka ブローカーで EventBridge Pipes を認証するように、クライアント証明書を (Secrets Manager のシークレットとして) 設定します。クライアント証明書は、 サーバーのトラストストア内の認証局 (CA) によって署名される必要があります。

Apache Kafka クラスターは Pipes にサーバー証明書を送信して、 EventBridge Pipes で Apache Kafka EventBridge ブローカーを認証します。サーバー証明書は、パブリック CA 証明書またはプライベート CA/自己署名証明書にすることができます。パブリック CA 証明書は、 EventBridge Pipes トラストストアにある CA によって署名される必要があります。プライベート CA/自己署名証明書の場合は、サーバーのルート CA 証明書を (Secrets Manager のシークレットとして) 設定します。EventBridge Pipes はルート証明書を使用して Apache Kafka ブローカーを検証します。

mTLS の詳細については、「Introducing mutual TLS authentication for Amazon MSK as an source」(ソースとしての Amazon MSK のための相互 TLS の紹介) を参照してください。

クライアント証明書シークレットの設定

CLIENT_CERTICATE_TLS_AUTH シークレットは、証明書フィールドとプライベートキーフィールドを必要とします。暗号化されたプライベートキーの場合、シークレットはプライベートキーのパスワードを必要とします。証明書とプライベートキーは、どちらも PEM 形式である必要があります。

> **Note**
>
> EventBridge Pipes は、PBES1 (PBES2 はサポートしていません) プライベートキー暗号化アルゴリズムをサポートしています。

証明書フィールドには、クライアント証明書で始まり、その後に中間証明書が続き、ルート証明書で終わる証明書のリストが含まれている必要があります。各証明書は、以下の構造を使用した新しい行で始める必要があります。

```
-----BEGIN CERTIFICATE-----
        <certificate contents>
-----END CERTIFICATE-----
```

Secrets Manager は最大 65,536 バイトのシークレットをサポートします。これは、長い証明書チェーンにも十分な領域です。

プライベートキーは、以下の構造を使用した PKCS #8 形式にする必要があります。

```
-----BEGIN PRIVATE KEY-----
         <private key contents>
-----END PRIVATE KEY-----
```

暗号化されたプライベートキーには、以下の構造を使用します。

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
          <private key contents>
-----END ENCRYPTED PRIVATE KEY-----
```

以下は、暗号化されたプライベートキーを使用する mTLS 認証のシークレットの内容を示す例です。暗号化されたプライベートキーの場合は、シークレットにプライベートキーのパスワードを含めます。

```
{
 "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-----"
}
```

サーバルート CA 証明書シークレットの設定

このシークレットは、Apache Kafka ブローカーがプライベート CA によって署名された証明書で TLS 暗号化を使用する場合に作成します。TLS 暗号化は、VPC、SASL/SCRAM、SASL/PLAIN、または mTLS 認証に使用できます。

サーバールート CA 証明書シークレットには、PEM 形式の Apache Kafka ブローカーのルート CA 証明書が含まれるフィールドが必要です。以下は、このシークレットの構造を示す例です。

```
{
     "certificate": "-----BEGIN CERTIFICATE-----
```

```
MIID7zCCAtegAwIBAgIBADANBgkqhkiG9w0BAQsFADCBmDELMAkGA1UEBhMCVVMx
EDAOBgNVBAgTB0FyaXpvbmExEzARBgNVBAcTClNjb3R0c2RhbGUxJTAjBgNVBAoT
HFN0YXJmaWVsZCBUZWNobm9sb2dpZXMsIEluYy4xOzA5BgNVBAMTMlN0YXJmaWVs
ZCBTZXJ2aWNlcyBSb290IENlcnRpZmljYXRlIEF1dG...
-----END CERTIFICATE-----"
```

ネットワーク構成

プライベート VPC 接続を使用するセルフマネージド Apache Kafka 環境を使用している場合は、Apache Kafka ブローカーに関連付けられた Amazon Virtual Private Cloud (Amazon VPC) リソースにアクセスできる EventBridge 必要があります。

- Apache Kafka クラスターの VPC にアクセスするには、ソースのサブネットにアウトバウンドインターネットアクセス EventBridge を使用できます。パブリックサブネットの場合、これはマネージド NAT ゲートウェイである必要があります。プライベートサブネットの場合は NAT ゲートウェイでも、独自の NAT でもかまいません。NAT にパブリック IP アドレスが割り当てられ、インターネットに接続できることを確認します。
- EventBridge Pipes は を介したイベント配信もサポートしているためAWS PrivateLink、 Amazon Virtual Private Cloud （ Amazon VPC) にあるイベントソースから Pipes ターゲットにイベントを送信できます。パブリックインターネットを経由する必要はありません。Pipes を使用すると、インターネットゲートウェイをデプロイしたり、ファイアウォールルールを設定したり、プロキシサーバーを設定したりすることなく、 Amazon Managed Streaming for Apache Kafka （ Amazon MSK)、セルフマネージド Apache Kafka、およびプライベートサブネットに存在する Amazon MQ ソースからポーリングできます。

  VPC エンドポイントを設定するには、「 ユーザーガイド」の「VPC エンドポイントの作成AWS PrivateLink 」を参照してください。サービス名で、 を選択しますcom.amazonaws.*region*.pipes-data。

Amazon VPC セキュリティグループは、少なくとも以下のルールを使用して設定してください。

- インバウンドルール – ソースに指定されたセキュリティグループの Apache Kafka ブローカーポート上のすべてのトラフィックを許可します。
- アウトバウンドルール – すべての送信先に対して、ポート 443 上のすべてのトラフィックを許可します。ソースに指定されたセキュリティグループの Apache Kafka ブローカーポート上のすべてのトラフィックを許可します。

  ブローカーポートには以下が含まれます。

- プレーンテキストの場合は 9092
- TLS の場合は 9094
- SASL の場合は 9096
- IAM 用 9098

Apache Kafka ソースによるコンシューマーの自動スケーリング

最初に Apache Kafka ソースを作成すると、 は Kafka トピック内のすべてのパーティションを処理するために 1 つのコンシューマーを EventBridge 割り当てます。各コンシューマーには、増加したワークロードを処理するために同時実行される複数のプロセッサがあります。さらに、 は、ワークロードに基づいてコンシューマーの数 EventBridge を自動的にスケールアップまたはスケールダウンします。各パーティションでメッセージの順序を保つため、コンシューマーの最大数は、トピック内のパーティションあたり 1 つとなっています。

1 分間隔で、 はトピック内のすべてのパーティションのコンシューマーオフセットラグ EventBridge を評価します。遅延が高すぎる場合、パーティションはメッセージを処理 EventBridge できるよりも速くメッセージを受信しています。必要に応じて、コンシューマーをトピック EventBridge に追加または削除します。コンシューマーを追加または削除するスケーリングプロセスは、評価から 3 分以内に行われます。

ターゲットが過負荷になっている場合、 はコンシューマーの数 EventBridge を減らします。このアクションにより、コンシューマーが取得し関数に送信するメッセージの数が減り、関数への負荷が軽減されます。

Amazon Simple Queue Service をソースとして使用する場合

EventBridge Pipes を使用して、Amazon SQS キューからレコードを受信できます。その後、必要に応じてこれらのレコードをフィルタリングまたは拡張してから、処理可能な送信先に送信できます。

パイプを使用して、Amazon Simple Queue Service (Amazon SQS) キュー内のメッセージを処理できます。 EventBridge Pipes は、スタンダードキューと先入れ先出し (FIFO) キューをサポートします。Amazon SQS を使用すると、タスクをキューに送信して非同期的に処理することで、アプリケーションの 1 つのコンポーネントからタスクを任せることができます。

EventBridge はキューをポーリングし、キューメッセージを含むイベントでパイプを同期的に呼び出します。 はメッセージをバッチで EventBridge 読み取り、バッチごとにパイプを 1 回呼び出します。パイプがバッチを正常に処理すると、 はキューからそのメッセージ EventBridge を削除します。

デフォルトでは、 はキューに最大 10 個のメッセージを同時に EventBridge ポーリングし、そのバッチをパイプに送信します。少数のレコードでパイプを呼び出さないようにするには、バッチウィンドウを設定して、最大 5 分間レコードをバッファリングするようにイベントソースに指示できます。パイプを呼び出す前に、 は、次のいずれかが発生するまで Amazon SQS 標準キューからのメッセージのポーリング EventBridge を続行します。

- バッチウィンドウの有効期限が切れる。
- 呼び出しペイロードのサイズがクォータに達する。
- 最大バッチサイズに達する。

> **Note**
>
> バッチウィンドウを使用していて、Amazon SQS キューのトラフィックが少ない場合は、パイプを呼び出す前に最大 20 秒間 EventBridge 待機することがあります。これは、バッチウィンドウを 20 秒未満に設定した場合であっても同様です。FIFO キューの場合、レコードには、重複除外と順序付けに関連する追加属性が含まれます。

がバッチ EventBridge を読み取ると、メッセージはキューに留まりますが、キューの可視性タイムアウトの間は非表示になります。パイプがバッチを正常に処理すると、 はキューからメッセージ EventBridge を削除します。デフォルトで、パイプがバッチを処理しているときにエラーが発生すると、そのバッチ内のすべてのメッセージが再びキューに表示されます。このため、パイプコードは、意図しない副次的影響を及ぼすことなく同じメッセージを複数回処理できるようにする必要があります。バッチアイテムの失敗をパイプのレスポンスに含めることで、この再処理動作を変更できます。以下の例では、2 つのメッセージのバッチのイベントを示しています。

イベントの例

次のサンプルイベントは、パイプが受信した情報を示しています。このイベントを使用して、イベントパターンを作成およびフィルタリングしたり、入力変換を定義したりできます。すべてのフィールドをフィルタリングできるわけではありません。フィルターできるフィールドの詳細については、「???」を参照してください。

スタンダードキュー

```
[
  {
    "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 キュー

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

スケーリングと処理

スタンダードキューの場合、キューがアクティブになるまでキューをポーリングするためにロングポーリング EventBridge を使用します。メッセージが利用可能な場合、 は最大 5 つのバッチを EventBridge 読み取り、パイプに送信します。メッセージがまだ利用可能な場合は、バッチを読み取るプロセスの数を EventBridge 1 分あたり最大 300 インスタンス増加させます。パイプによって同時に処理できるバッチの最大数は 1,000 です。

FIFO キューの場合、 EventBridge は受信した順序でパイプにメッセージを送信します。FIFO キューにメッセージを送信する場合、メッセージグループ ID を指定します。Amazon SQS は EventBridge、同じグループ内のメッセージを に順番に配信することを容易にします。 は受信したメッセージをグループに EventBridge ソートし、グループに対して一度に 1 つのバッチのみを送信します。パイプがエラーを返す場合、パイプは同じグループから追加のメッセージ EventBridge を受信する前に、影響を受けるメッセージですべての再試行を試みます。

EventBridge Pipes で使用するキューの設定

Amazon SQS キューを作成して、パイプのソースとして機能できるようにします。次に、パイプがイベントの各バッチを処理し、スケールアップ時にスロットリングエラーに応じて が EventBridge 再試行できるように、キューを設定します。

パイプがレコードの各バッチを処理するために十分な時間を取るため、ソースキューの可視性タイムアウトは、パイプのエンリッチメントとターゲットコンポーネントのランタイムを組み合わせた時間の少なくとも 6 倍に設定してください。追加の時間は EventBridge 、パイプが前のバッチの処理中にスロットリングされた場合に が再試行することを許可します。

パイプがメッセージの処理に何回も失敗する場合、Amazon SQS はこのメッセージをデッドレターキューに送信できます。パイプがエラーを返すと、 はそれをキューに EventBridge 保持します。可

視性タイムアウトが発生すると、 EventBridge はメッセージをもう一度受信します。多数の受信後に 2 番目のキューにメッセージを送信するには、ソースキューにデッドレターキューを設定します。

> **Note**
>
> パイプではなく、ソースキューのデッドレターキューを設定するようにしてください。パイプで設定したデッドレターキューは、ソースキューではなく、パイプの非同期呼び出しキューに使用されます。

パイプからエラーが返された場合や、同時実行数の最大値に達しているために関数を呼び出せない場合は、追加の試行で処理が成功する場合があります。メッセージをデッドレターキューに送信する前にメッセージが処理される確率を高めるには、送信元キューのリドライブポリシーの `maxReceiveCount` を 5 以上に設定します。

バッチ項目の失敗の報告

が ソースからストリーミングデータを EventBridge 消費して処理する場合、デフォルトではバッチの最大シーケンス番号にチェックポイントされますが、バッチが完全に成功した場合に限ります。正常に処理されたメッセージが失敗したバッチで再処理されないようにするには、成功したメッセージと失敗したメッセージを示すオブジェクトを返すようにエンリッチメントまたはターゲットを設定できます。これを部分的なバッチレスポンスと呼びます。

詳細については、「???」を参照してください。

成功条件と失敗の条件

次のいずれかを返すと、 はバッチを完全に成功として EventBridge 処理します。

- 空の `batchItemFailure` リスト
- null の `batchItemFailure` リスト
- 空の `EventResponse`
- null の `EventResponse`

次のいずれかを返すと、 はバッチを完全に失敗として EventBridge 処理します。

- 空の文字列 `itemIdentifier`
- ヌル `itemIdentifier`

- 不正なキー名を持つ itemIdentifier

EventBridge は、再試行戦略に基づいて失敗を再試行します。

Amazon EventBridge Pipes フィルタリング

EventBridge Pipes を使用すると、特定のソースのイベントをフィルタリングし、それらのサブセットのみを処理できます。このフィルタリングは、 EventBridge イベントパターンを使用して、イベントバスまたは Lambda イベントソースマッピングでフィルタリングするのと同じ方法で機能します。イベントパターンの詳細については、「???」を参照してください。

フィルター条件 FilterCriteria オブジェクトは、フィルターのリスト (Filters) で構成される構造です。各フィルターは、フィルタリングパターン (Pattern) を定義する構造です。Pattern は、JSON フィルタールールの文字列表現です。FilterCriteria オブジェクトは、以下の例のようになります。

```
{
  "Filters": [
    {"Pattern": "{ \"Metadata1\": [ rule1 ], \"data\": { \"Data1\": [ rule2 ] }}"
    }
  ]
}
```

以下は、わかりやすくするためにプレーン JSON で展開したフィルターの Pattern の値を記載しています。

```
{
  "Metadata1": [ pattern1 ],
  "data": {"Data1": [ pattern2 ]}
}
```

FilterCriteria オブジェクトには、メタデータプロパティおよびデータプロパティの主要部分で構成されています。

- メタデータプロパティは、イベントオブジェクトのフィールドです。この例では、FilterCriteria.Metadata1 はメタデータプロパティを参照します。
- データプロパティは、イベント本文のフィールドです。この例では、 FilterCriteria.Data1 はデータプロパティを参照します。

たとえば、Kinesis ストリームに次のようなイベントが含まれているとします。

```
{
  "kinesisSchemaVersion": "1.0",
  "partitionKey": "1",
  "sequenceNumber": "49590338271490256608559692538361571095921575989136588898",
  "data": {"City": "Seattle",
    "State": "WA",
    "Temperature": "46",
    "Month": "December"
  },
  "approximateArrivalTimestamp": 1545084650.987
}
```

イベントがパイプを通過すると、data フィールドが base64 でエンコードされた状態で次のようになります。

```
{
  "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 イベントのメタデータプロパティは、partitionKey または sequenceNumber など、data オブジェクト外の任意のフィールドです。

Kinesis イベントのデータプロパティは、City または Temperature などの data オブジェクト内のフィールドです。

このイベントに一致するようにフィルタリングすると、デコードされたフィールドにフィルターを使用できます。たとえば、partitionKey および City でフィルターをオンにするには、次のフィルターを使用します。

```
{
  "partitionKey": [
    "1"
  ],
  "data": {
    "City": [
      "Seattle"
    ]
  }
}
```

イベントフィルターを作成すると、 EventBridge Pipes はイベントコンテンツにアクセスできます。このコンテンツは、Amazon SQS body フィールドのように JSON でエスケープされるか、Kinesis data フィールドのように base64 でエンコードされます。データが有効な JSON であれば、ターゲットパラメータの入力テンプレートまたは JSON パスでコンテンツを直接参照できます。たとえば、Kinesis イベントソースが有効な JSON の場合は、<$.data.someKey> を使用して変数を参照できます。

イベントパターンを作成する場合、ポーリング操作によって追加されたフィールドではなく、ソース API から送信されたフィールドに基づいてフィルタリングできます。次のフィールドはイベントパターンで使用することはできません。

- awsRegion
- eventSource
- eventSourceARN
- eventVersion
- eventID
- eventName
- invokeIdentityArn
- eventSourceKey

メッセージフィールドとデータフィールド

すべての EventBridge パイプソースには、コアメッセージまたはデータを含む フィールドが含まれています。これらをメッセージフィールドまたはデータフィールドと呼びます。これらのフィールドは JSON でエスケープまたは base64 でエンコードされている可能性があるため特殊ですが、有効

な JSON の場合は、本文がエスケープされていないかのように JSON パターンでフィルタリングできます。これらのフィールドの内容は、Input Transformers でもシームレスに使用できます。

フィルタリングが適切な Amazon SQS メッセージ

Amazon SQSメッセージがフィルター条件を満たさない場合、 はキューからメッセージ EventBridge を自動的に削除します。Amazon SQS でこれらのメッセージを手動で削除する必要はありません。

Amazon SQS の場合、メッセージ の body は任意の文字列にすることができますが、body が有効な JSON フォーマットであることを `FilterCriteria` が期待する場合は、これが問題になる可能性があります。逆の場合も同様です。着信メッセージの body が JSON 形式である場合に、body がプレーン文字列であることをフィルター条件が期待していると、意図しない動作が発生する可能性があります。

この問題を回避するには、`FilterCriteria` 内の body の形式が、キューから受け取るメッセージの body に期待される形式と一致することを確認してください。メッセージをフィルタリングする前に、 は受信メッセージの形式bodyと のフィルターパターンの形式 EventBridge を自動的に評価しますbody。一致しない場合、 はメッセージを EventBridge ドロップします。この評価のまとめは、以下の表のとおりです。

着信メッセージの **body** の形式	フィルターパターンの **body** の形式	結果として生じるアクション
プレーン文字列	プレーン文字列	EventBridge は、フィルター条件に基づいてフィルタリングします。
プレーン文字列	データプロパティのフィルターパターンがない	EventBridge は、フィルター条件に基づいて (他のメタデータプロパティでのみ) をフィルタリングします。
プレーン文字列	有効な JSON	EventBridge はメッセージをドロップします。
有効な JSON	プレーン文字列	EventBridge はメッセージをドロップします。

着信メッセージの **body** の形式	フィルターパターンの **body** の形式	結果として生じるアクション
有効な JSON	データプロパティのフィルターパターンがない	EventBridge は、フィルター条件に基づいて (他のメタデータプロパティでのみ) をフィルタリングします。
有効な JSON	有効な JSON	EventBridge は、フィルター条件に基づいてフィルタリングします。

body の一部として を含めない場合は`FilterCriteria`、このチェック EventBridge をスキップします。

フィルタリングが適切な Kinesis メッセージと DynamoDB メッセージ

フィルター条件が Kinesis または DynamoDB レコードを処理すると、ストリームイテレータはこのレコードを通り越して先に進みます。レコードがフィルター条件を満たさない場合に、そのレコードをイベントソースから手動で削除する必要はありません。保持期間が過ぎると、Kinesis と DynamoDB はこれらの古いレコードを自動的に削除します。それより早くレコードを削除したい場合は、「Changing the Data Retention Period」(データ保持期間の変更) を参照してください。

ストリームイベントソースからのイベントを適切にフィルタリングするには、データフィールドとデータフィールドのフィルター条件の両方が有効な JSON 形式である必要があります。(Kinesis のデータフィールドは `data` で、DynamoDB のデータフィールドは `dynamodb` です。) いずれかのフィールドが有効な JSON 形式でない場合、 はメッセージを EventBridge ドロップするか、例外をスローします。以下は、特定の動作を要約した表です。

着信データの形式 (**`data`** または **`dynamodb`**)	データプロパティのフィルターパターンの形式	結果として生じるアクション
有効な JSON	有効な JSON	EventBridge は、フィルター条件に基づいてフィルタリングします。
有効な JSON	データプロパティのフィルターパターンがない	EventBridge は、フィルター条件に基づいて (他のメタデー

着信データの形式 (**data** または **dynamodb**)	データプロパティのフィルターパターンの形式	結果として生じるアクション
		タプロパティでのみ) をフィルタリングします。
有効な JSON	JSON 以外	EventBridge はパイプまたは更新時に例外をスローします。データプロパティのフィルターパターンは、有効な JSON 形式である必要があります。
JSON 以外	有効な JSON	EventBridge はレコードを削除します。
JSON 以外	データプロパティのフィルターパターンがない	EventBridge は、フィルター条件に基づいて (他のメタデータプロパティでのみ) をフィルタリングします。
JSON 以外	JSON 以外	EventBridge は、パイプの作成時または更新時に例外をスローします。データプロパティのフィルターパターンは、有効な JSON 形式である必要があります。

Amazon Managed Streaming for Apache Kafka、セルフマネージド Apache Kafka、および Amazon MQ メッセージの適切なフィルタリング

Amazon MQ ソースの場合、メッセージフィールドは `data` になります。Apache Kafka ソース (Amazon MSK および セルフマネージド Apache Kafka) の場合は、`key` と `value` の 2 つのメッセージフィールドがあります。

EventBridge は、フィルターに含まれるすべてのフィールドと一致しないメッセージを削除します。Apache Kafka の場合、 は関数を正常に呼び出した後、一致したメッセージと一致しないメッセージのオフセットを EventBridge コミットします。Amazon MQ の場合、 EventBridge は関数を正

常に呼び出した後に一致したメッセージを確認し、フィルタリング時に一致しないメッセージを確認します。

Apache Kafka メッセージと Amazon MQ メッセージは UTF-8 でエンコードされた文字列 (プレーン文字列または JSON 形式) である必要があります。これは、フィルター条件を適用する前に Apache Kafka と Amazon MQ のバイト配列を UTF-8 に EventBridge デコードするためです。メッセージが UTF-16 や ASCII などの別のエンコーディングを使用している場合、またはメッセージ形式が`FilterCriteria`形式と一致しない場合、 はメタデータフィルターのみを EventBridge 処理します。以下は、特定の動作を要約した表です。

着信メッセージの形式 (**`data`**、または **`key`** と **`value`**)	メッセージプロパティのフィルターパターン形式	結果として生じるアクション
プレーン文字列	プレーン文字列	EventBridge は、フィルター条件に基づいてフィルタリングします。
プレーン文字列	データプロパティのフィルターパターンがない	EventBridge は、フィルター条件に基づいて (他のメタデータプロパティでのみ) をフィルタリングします。
プレーン文字列	有効な JSON	EventBridge は、フィルター条件に基づいて (他のメタデータプロパティでのみ) をフィルタリングします。
有効な JSON	プレーン文字列	EventBridge は、フィルター条件に基づいて (他のメタデータプロパティでのみ) をフィルタリングします。
有効な JSON	データプロパティのフィルターパターンがない	EventBridge は、フィルター条件に基づいて (他のメタデータプロパティでのみ) をフィルタリングします。

着信メッセージの形式 (**data**、または **key** と **value**)	メッセージプロパティのフィルターパターン形式	結果として生じるアクション
有効な JSON	有効な JSON	EventBridge は、フィルター条件に基づいてフィルタリングします。
UTF-8 以外でエンコードされた文字	JSON、プレーン文字列、またはパターンなし	EventBridge は、フィルター条件に基づいて (他のメタデータプロパティでのみ) をフィルタリングします。

Lambda ESM と EventBridge Pipes の違い

イベントをフィルタリングする場合、Lambda ESM と EventBridge Pipes は一般的に同じように動作します。主な違いは、`eventSourceKey` フィールドが ESM ペイロードに存在しないことです。

Amazon EventBridge Pipes イベントのエンリッチメント

EventBridge Pipes のエンリッチメントステップでは、ソースからのデータをターゲットに送信する前に拡張できます。たとえば、チケットデータがすべて含まれていないチケット作成イベントを受け取る場合があります。エンリッチメントを使用すると、Lambda 関数で `get-ticket` API を呼び出して、チケットの詳細をすべて確認できます。その後、パイプはその情報をターゲットに送信できます。

EventBridge でパイプをセットアップする場合、以下のエンリッチメントを設定できます。

- API 送信先
- Amazon API Gateway
- Lambda 関数
- Step Functions ステートマシン

> **Note**
>
> EventBridge Pipes は、エクスプレスワークフローをエンリッチメントとしてのみサポートしています。

EventBridge は、エンリッチメントからのレスポンスを待ってからターゲットを呼び出す必要があるため、エンリッチメントを同期的に呼び出します。

エンリッチメントのレスポンスは、最大サイズが 6 MB に制限されます。

また、ソースから受け取ったデータを送信する前に変換して拡張することもできます。詳細については、「???」を参照してください。

エンリッチメントによるイベントのフィルタリング

EventBridge Pipes は、エンリッチメントのレスポンスを設定されたターゲットに直接渡します。これには、バッチをサポートするターゲットの配列レスポンスが含まれます。バッチ動作の詳細については、「???」を参照してください。エンリッチメントをフィルターとして使用して、ソースから受信したイベントよりも少ないイベントを渡すこともできます。ターゲットを呼び出したくない場合は、""、{}、または [] などの空のレスポンスを返します。

> **Note**
>
> 空のペイロードでターゲットを呼び出す場合は、空の JSON [{}] を含む配列を返します。

エンリッチメントの呼び出し

EventBridge は、エンリッチメントからのレスポンスを待ってからターゲットを呼び出す必要があるため、エンリッチメントを同期的に (呼び出しタイプを REQUEST_RESPONSE に設定して) 呼び出します。

> **Note**
>
> Step Functions ステートマシンでは、EventBridge はエンリッチメントとして Express ワークフローのみサポートします。このワークフローは同期的に呼び出すことができるためです。

Amazon EventBridge Pipes ターゲット

パイプ内のデータを特定のターゲットに送信できます。でパイプを設定するときに、次のターゲットを設定できます EventBridge。

- API 送信先

- API Gateway
- バッチジョブのキュー
- CloudWatch ロググループ
- ECS タスク
- 同じアカウントとリージョンのイベントバス
- Firehose 配信ストリーム
- Inspector 評価テンプレート
- Kinesis ストリーミング
- Lambda 関数 (同期または非同期)
- Redshift クラスターデータ API クエリ
- SageMaker パイプライン
- Amazon SNS トピック (SNS FIFO トピックはサポートされていません)
- Amazon SQS キュー
- Step Functions ステートマシン
  - Express ワークフロー (SYNC または ASYNC)
  - Standard ワークフロー (ASYNC)
- Timestream LiveAnalytics テーブルの

ターゲットパラメータ

一部のターゲットサービスはイベントペイロードをターゲットに送信せず、代わりにイベントを特定の API を呼び出すトリガーとして扱います。 EventBridge は `PipeTargetParameters`を使用して、その API に送信される情報を指定します。これには以下が含まれます。

- API 送信先 (API 送信先に送信されるデータは API の構造と一致する必要があります。 `InputTemplate` オブジェクトを使用して、データが正しく構造化されていることを確認する必要があります。元のイベントペイロードを含める場合は、`InputTemplate` でそれを参照してください)。
- API ゲートウェイ (API ゲートウェイに送信されるデータは API の構造と一致する必要があります。 `InputTemplate` オブジェクトを使用して、データが正しく構造化されていることを確認する必要があります。元のイベントペイロードを含める場合は、`InputTemplate` でそれを参照してください)。
- `PipeTargetRedshiftDataParameters` (Amazon Redshift データ API クラスター)

- PipeTargetSageMakerPipelineParameters (Amazon SageMaker Runtime Model Building Pipelines)
- PipeTargetBatchJobParameters (AWS Batch)

> **Note**
>
> EventBridge は、すべての JSON パス構文をサポートしているわけではなく、実行時に評価されます。サポートされている構文には以下が含まれます。
>
> - ドット表記 ($.detail など)
> - ダッシュ
> - 下線
> - アルファベットの文字
> - 配列インデックス
> - ワイルドカード (*)

動的パスパラメータ

EventBridge Pipes ターゲットパラメータは、オプションの動的 JSON パス構文をサポートします。この構文を使って、静的値の代わりに JSON パスを指定できます (例えば、`$.detail.state`)。値の一部だけではなく全体を JSON パスにする必要があります。例えば、`RedshiftParameters.Sql` は `$.detail.state` とすることができますが、`"SELECT * FROM $.detail.state"` とすることはできません。このようなパスは、実行時に、指定されたパスにあるイベントペイロード自体のデータで動的に置き換えられます。動的パスパラメータは、入力変換の結果として生じる新しい値または変換された値を参照できません。動的パラメータの JSON パスでサポートされている構文は、入力を変換する場合と同じです。詳細については、「???」を参照してください。

動的構文は、以下を除くすべての EventBridge Pipes エンリッチメントおよびターゲットパラメータのすべての文字列、非列挙型フィールドで使用できます。

- PipeTargetCloudWatchLogsParameters.LogStreamName
- PipeTargetEventBridgeEventBusParameters.EndpointId
- PipeEnrichmentHttpParameters.HeaderParameters
- PipeTargetHttpParameters.HeaderParameters

例えば、パイプ Kinesis ターゲットPartitionKeyの をソースイベントのカスタムキーに設定するには、 を次のように設定しますKinesisTargetParameter。PartitionKey

- "$.data.*someKey*" (Kinesis ソースの場合)
- "$.body.*someKey*" (Amazon SQS ソースの場合)

次に、イベントペイロードが などの有効な JSON 文字列である場合、 は{"*someKey*":"*someValue*"}JSON パスから値を EventBridge 抽出し、それをターゲットパラメータとして使用します。この例では、Kinesis を EventBridge PartitionKey*someValue*」に設定します。

アクセス許可

所有しているリソースで API コールを行うには、 EventBridge Pipes に適切なアクセス許可が必要です。 EventBridge PIpes は、IAM プリンシパル を使用してエンリッチメントとターゲット呼び出しのためにパイプで指定した IAM ロールを使用しますpipes.amazonaws.com。

ターゲットの呼び出し

EventBridge には、ターゲットを呼び出す次の方法があります。

- 同期的 (呼び出しタイプを に設定REQUEST_RESPONSE) — 先に進む前にターゲットからの応答を EventBridge 待ちます。
- 非同期 (呼び出しタイプを に設定FIRE_AND_FORGET) — EventBridge 続行する前にレスポンスを待たないでください。

デフォルトでは、ソースが順序付けられたパイプの場合、 はターゲットを同期的に EventBridge 呼び出します。これは、次のイベントに進む前にターゲットからの応答が必要なためです。

ソースが標準の Amazon SQS キューなどの順序を強制しない場合は、サポートされているターゲットを同期的にまたは非同期的に呼び出す EventBridge ことができます。

Lambda 関数と Step Functions ステートマシンでは、呼び出しタイプを設定できます。

> **Note**
>
> Step Functions ステートマシンの場合、標準ワークフローを非同期で呼び出す必要があります。

EventBridge パイプターゲットの詳細

AWS Batch ジョブキュー

すべての AWS Batch `submitJob`パラメータは で明示的に設定され`BatchParameters`、すべての Pipe パラメータと同様に、受信イベントペイロードへの JSON パスを使用して動的に設定できます。

CloudWatch ロググループ

Input Transformer を使用するかどうかにかかわらず、イベントペイロードはログメッセージとして使用されます。`PipeTarget` では、`CloudWatchLogsParameters` を使用して `Timestamp` (または送信先の明示的な `LogStreamName`) を設定できます。これらのパラメータは、すべてのパイプのパラメータと同様に、受信イベントペイロードへの JSON パスを使用するとき、動的に設定できます。

Amazon ECS タスク

すべての Amazon ECS `runTask` パラメータは、`EcsParameters` を使用して明示的に設定されます。これらのパラメータは、すべてのパイプのパラメータと同様に、受信イベントペイロードへの JSON パスを使用するとき、動的に設定できます。

Lambda 関数とステップ関数のワークフロー

Lambda 関数とステップ関数にバッチ API はありません。パイプソースからのイベントのバッチを処理するには、バッチを JSON 配列に変換し、Lambda または Step Functions ターゲットへの入力として渡します。詳細については、「???」を参照してください。

Timestream LiveAnalytics テーブルの

LiveAnalytics テーブル Timestream の をパイプターゲットとして指定する際の考慮事項は次のとおりです。

- Apache Kafka ストリーム (Amazon MSK またはサードパーティープロバイダーからのストリームを含む) は現在、パイプソースとしてサポートされていません。
- パイプソースとして Kinesis または DynamoDB ストリームを指定している場合は、再試行回数を指定する必要があります。

  詳細については、「???」を参照してください。

Amazon EventBridge Pipes のバッチ処理と同時実行

バッチ処理動作

EventBridge Pipes は、ソースとそれをサポートするターゲットからのバッチ処理をサポートします。また、AWS Lambda および AWS Step Functions では、エンリッチメントのバッチ処理もサポートされています。サービスが異なればサポートされるバッチ処理のレベルも異なるため、ターゲットがサポートするバッチサイズよりも大きいバッチサイズのパイプを構成することはできません。例えば、Amazon Kinesis ストリームソースは、最大バッチサイズ 10,000 件のレコードをサポートしますが、Amazon Simple Queue Service は、バッチあたり最大 10 件のメッセージをターゲットとしてサポートします。そのため、Kinesis ストリームから Amazon SQS キューへのパイプは、ソースの最大バッチサイズを 10 に設定できます。

バッチ処理をサポートしないエンリッチメントまたはターゲットを使用してパイプを設定すると、ソースでバッチ処理を有効にすることはできません。

ソースでバッチ処理が有効化されると、JSON レコードの配列がパイプを通過し、サポートされているエンリッチメントまたはターゲットのバッチ API にマッピングされます。Input transformers は、配列全体ではなく、配列内の個々の JSON レコードに個別に適用されます。これらの配列の例については、??? を参照し、特定のソースを選択してください。パイプは、バッチサイズが 1 の場合でも、サポートされているエンリッチメントまたはターゲットにバッチ API を使用します。エンリッチメントまたはターゲットにバッチ API がないものの、Lambda や Step Functions などの完全な JSON ペイロードを受け取る場合、JSON 配列全体が 1 回のリクエストで送信されます。バッチサイズが 1 の場合でも、リクエストは JSON 配列として送信されます。

パイプがソースでバッチ処理するように設定されていて、ターゲットがバッチ処理をサポートしている場合、エンリッチメントから JSON 項目の配列を返すことができます。この配列は、元のソースよりも短い配列でも長い配列でもかまいません。ただし、配列がターゲットがサポートするバッチサイズよりも大きい場合、パイプはターゲットを呼び出しません。

サポートされているバッチ処理可能なターゲット

Target	最大バッチサイズ
CloudWatch ログ	10,000
EventBridge イベントバス	10

Target	最大バッチサイズ
Firehose ストリーム	500
Kinesis ストリーミング	500
Lambda 関数	顧客による定義
Step Functions ステートマシン	顧客による定義
Amazon SNS トピック	10
Amazon SQS キュー	10

次のエンリッチメントとターゲットは、バッチイベントペイロード全体を受け取って処理しますが、バッチのサイズではなく、イベントの合計ペイロードサイズによって制約されます。

- Step Functions ステートマシン (262144 文字)
- Lambda 関数 (6MB)

部分的なバッチ処理失敗

Kinesis や DynamoDB などの Amazon SQS およびストリームソースの場合、 EventBridge Pipes はターゲット障害の部分的なバッチ障害処理をサポートします。 DynamoDB ターゲットがバッチ処理をサポートしており、バッチの一部のみが成功した場合、 はペイロードの残りの部分のバッチ処理 EventBridge を自動的に再試行します。最も up-to-date 強化されたコンテンツの場合、この再試行は設定されたエンリッチメントの再呼び出しを含め、パイプ全体で行われます。

エンリッチメントの部分的なバッチ障害処理はサポートされていません。

Lambda と Step Functions ターゲットの場合、ターゲットから構造を定義したペイロードを返すことで部分的な障害を指定することもできます。これは再試行が必要なイベントを示しています。

部分的障害ペイロード構造の例

```
{
  "batchItemFailures": [
    {
      "itemIdentifier": "id2"
```

```
    },
    {
      "itemIdentifier": "id4"
    }
]
```

この例では、itemIdentifier が、元のソースからのターゲットにより処理されるイベントの ID と一致しています。Amazon SQS の場合、これは messageId です。Kinesis と DynamoDB の場合、これは eventID です。 EventBridge Pipes がターゲットからの部分的なバッチ障害を適切に処理するには、これらのフィールドをエンリッチメントによって返される配列ペイロードに含める必要があります。

スループットと同時実行動作

パイプが受信したエンリッチメントまたはターゲットに送信されるすべてのイベントまたはイベントのバッチは、パイプの実行と見なされます。STARTED 状態のパイプは、ソースからのイベントを継続的にポーリングし、利用可能なバックログと設定されたバッチ設定に応じてスケールアップとスケールダウンを行います。

パイプの同時実行のクォータ、およびアカウントとリージョンごとのパイプ数については、??? を参照してください。

デフォルトでは、1 つのパイプは、ソースに応じて次の最大同時実行数にスケーリングされます。

- DynamoDB — 同時実行数は、パイプ上で設定されている ParallelizationFactor にストリーム内のシャード数を掛けた数まで増加する可能性があります。
- Apache Kafka — 同時実行数は、トピックのパーティション数 (最高 1000) まで増加する可能性があります。
- Kinesis — 同時実行数は、パイプ上で設定されている ParallelizationFactor にストリーム内のシャード数を掛けた数まで増加する可能性があります。
- Amazon MQ — 5
- Amazon SQS – 1250

最大ポーリングスループットまたは同時実行数の制限をより高くする必要がある場合は、サポートにお問い合わせください。

> **Note**
>
> 実行制限はベストエフォート型の安全制限と見なされます。ポーリングがこれらの値を下回ることはありませんが、パイプやアカウントがこれらの推奨値よりも高くなる可能性があります。

パイプの実行は、エンリッチメント処理とターゲット処理を含めて最大 5 分に制限されています。この制限を引き上げることはできません。

ソースが厳密に順序付けされたパイプ (Amazon SQS FIFO キュー、Kinesis および DynamoDB Streams、Apache Kafka トピックなど) は、FIFO キューのメッセージグループ ID 数や Kinesis キューのシャード数など、ソースの設定によって同時実行性はさらに制限されます。順序付けはこれらの制約の範囲内で厳密に保証されているため、順序付けされたソースを含むパイプは、これらの同時実行制限を超えることはできません。

Amazon EventBridge Pipes の入力変換

Amazon EventBridge Pipes では、データをエンリッチメントとターゲットに渡すときに、オプションで Input Transformer を使用できます。Input Transformer を使用すると、エンリッチメントサービスまたはターゲットサービスのニーズに対応するように JSON イベント入力ペイロードを再構成できます。Amazon API Gateway と API 送信先では、このようにして入力イベントを API の RESTful モデルに合わせて調整します。Input Transformer は `InputTemplate` パラメータとしてモデル化されます。フリーテキスト、イベントペイロードへの JSON パス、またはイベントペイロードへのインライン JSON パスを含む JSON オブジェクトを使用できます。エンリッチメントの場合、イベントペイロードはソースから送信されます。ターゲットの場合、イベントペイロードはエンリッチメントから返されるものです (パイプ上に設定されている場合)。イベントペイロード内のサービス固有のデータに加えて、`InputTemplate` の予約変数を使用してパイプのデータを参照できます。

配列内の項目にアクセスするには、角括弧表記を使用します。

> **Note**
>
> EventBridge ですべての JSON パス構文がサポートされているわけではなく、構文は実行時に評価されます。サポートされている構文には以下が含まれます。
>
> - ドット表記 (`$.detail` など)
> - ダッシュ

- 下線
- アルファベットの文字
- 配列インデックス
- ワイルドカード (*)

Amazon SQS イベントペイロードを参照するサンプル `InputTemplate` パラメータは次のとおりです。

静的文字列

```
InputTemplate: "Hello, sender"
```

JSON パス

```
InputTemplate: <$.attributes.SenderId>
```

動的文字列

```
InputTemplate: "Hello, <$.attributes.SenderId>"
```

静的 JSON

```
InputTemplate: >
{
  "key1": "value1",
  "key2": "value2",
  "key3": "value3",
}
```

動的 JSON

```
InputTemplate: >
{
  "key1": "value1"
  "key2": <$.body.key>,
  "d": <aws.pipes.event.ingestion-time>
}
```

角括弧表記を使用して配列内の項目にアクセスします。

```
InputTemplate: >
{
  "key1": "value1"
  "key2": <$.body.Records[3]>,
  "d": <aws.pipes.event.ingestion-time>
}
```

> **Note**
>
> EventBridge では実行時に入力トランスフォーマーを置き換えることで、有効な JSON 出力を確保します。このため、JSON パスパラメータを参照する変数は、引用符で囲んでください。JSON オブジェクトまたは配列を参照する変数は、引用符で囲まないでください。

予約変数

入力テンプレートは次の予約変数を使用できます。

- `<aws.pipes.pipe-arn>` – パイプの Amazon リソースネーム (ARN)。
- `<aws.pipes.pipe-name>`— パイプの名前。
- `<aws.pipes.source-arn>`— パイプのイベントソースの ARN。
- `<aws.pipes.enrichment-arn>`— パイプのエンリッチメントの ARN。
- `<aws.pipes.target-arn>`— パイプのターゲットの ARN。
- `<aws.pipes.event.ingestion-time>`— Input Transformer によってイベントが受信された時間。これは ISO 8601 タイムスタンプです。この時間は、エンリッチメントがイベントの処理をいつ完了したかによって、エンリッチメント Input Transformer とターゲット Input Transformer で異なります。
- `<aws.pipes.event>` — Input Transformerが受信したイベント。

  エンリッチメント Input Transformer の場合、これはソースからのイベントです。これには、ソースからの元のペイロードに加え、追加のサービス固有のメタデータが含まれます。このサービスに固有の例については、「???」のトピックを参照してください。

  ターゲット Input Transformer の場合、これはエンリッチメントによって返されるイベントです (設定されている場合)。追加のメタデータはありません。そのため、エンリッチメントで返され

るペイロードは JSON 以外の場合があります。パイプにエンリッチメントが設定されていない場合、これはメタデータを含むソースからのイベントです。

- <aws.pipes.event.json> — aws.pipes.event と同じですが、変数に値があるのは、ソースまたはエンリッチメントによって返された元のペイロードが JSON の場合に限られます。パイプに Amazon SQS body フィールドや Kinesis data などのエンコードされたフィールドがある場合、それらのフィールドはデコードされ、有効な JSON に変換されます。エスケープされないため、変数は JSON フィールドの値としてのみ使用できます。詳細については、「???」を参照してください。

入力変換の例

サンプルイベントとして使用できる Amazon EC2 イベントの例を次に示します。

```
{
  "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"
  }
}
```

以下の JSON をTransformer として使用してみましょう。

```
{
  "instance" : <$.detail.instance-id>,
  "state": <$.detail.state>,
  "pipeArn" : <aws.pipes.pipe-arn>,
  "pipeName" : <aws.pipes.pipe-name>,
  "originalEvent" : <aws.pipes.event.json>
```

```
}
```

次はその出力です。

```
{
  "instance" : "i-0123456789",
  "state": "RUNNING",
  "pipeArn" : "arn:aws:pipe:us-east-1:123456789012:pipe/example",
  "pipeName" : "example",
  "originalEvent" : {
    ... // commented for brevity
  }
}
```

暗示的な本文データ解析

受信ペイロードの以下のフィールドは、Amazon SQS body オブジェクトのように JSON エスケープすることも、Kinesis data オブジェクトのように base64 でエンコードすることもできます。フィルタリングと入力変換の両方で、EventBridge はこれらのフィールドを有効な JSON に変換するので、サブ値を直接参照できます。例えば、<$.data.someKey> は Kinesis 場合です。

追加のメタデータなしでターゲットが元のペイロードを受け取れるようにするには、ソース固有のこの本文データを含む Input Transformer を使用します。例えば、Amazon SQS では <$.body>、Kinesis では <$.data> です。元のペイロードが有効な JSON 文字列 (例えば {"key": "value"}) の場合、ソース固有の本文データで Input Transformer を使用すると、元のソースペイロード内の引用符が削除されます。例えば、{"key": "value"} は、ターゲットに配信されたとき "{key: value}" になります。ターゲットに有効な JSON ペイロード (EventBridge Lambda や Step Functions など) が必要な場合、配信が失敗します。無効な JSON を生成せずにターゲットが元のソースデータを受信できるようにするには、ソース本文データ Input Transformer を JSON でラップします。例えば、{"data": <$.data>} です。

また、暗示的な本文解析を使用すると、ほとんどのパイプターゲットまたはエンリッチメントパラメータの値を動的に入力することができます。詳細については、「???」を参照してください。

> **Note**
>
> 元のペイロードが有効な JSON の場合、このフィールドには、エスケープされていない base64 でエンコードされていない JSON が含まれます。ただし、ペイロードが有効な

JSON でない場合、EventBridge は Amazon SQS を除き、以下に示すフィールドを base64 でエンコードします。

- Active MQ —`data`
- Kinesis – `data`
- Amazon MSK – `key` および `value`
- Rabbit MQ – `data`
- セルフマネージド Apache Kafka – `key` および `value`
- Amazon SQS – `body`

入力変換に関する一般的な問題

これらは、EventBridge パイプで入力を変換するときの一般的な問題です。

- 文字列の場合は、引用符が必要です。
- テンプレートの JSON パスを作成する場合、検証は行われません。
- 指定した変数と一致する JSON パスがイベントに存在しない場合、その変数は作成されず、出力にも表示されません。
- `aws.pipes.event.json` のような JSON プロパティは JSON フィールドの値としてのみ使用でき、他の文字列に埋め込んで使用することはできません。
- EventBridge は、ターゲットの入力テンプレートに入力する際に、入力パスによって抽出される値をエスケープしません。
- JSON パスが JSON オブジェクトまたは配列を参照しているにもかかわらず、文字列では変数が参照されている場合、EventBridge は内部の引用符をすべて削除して文字列が有効であることを確認します。例えば、"Body is <$.body>" が作成されると、EventBridge はオブジェクトから引用符を削除します。

  したがって、単一の JSON パス変数に基づいて JSON オブジェクトを出力する場合、それをキーとして配置する必要があります。この例では、`{"body": <$.body>}` です。
- 文字列を表す変数に引用符は必要ありません。使用することはできますが、EventBridge Pipes では、変換出力が有効な JSON になるように、変換中に文字列変数の値に自動的に引用符を追加します。EventBridge Pipes は JSON オブジェクトや配列を表す変数に引用符を追加しません。JSON オブジェクトや配列を表す変数に引用符を追加しないでください。

たとえば、次の入力テンプレートには、文字列と JSON オブジェクトの両方を表す変数が含まれています。

```
{
  "pipeArn" : <aws.pipes.pipe-arn>,
  "pipeName" : <aws.pipes.pipe-name>,
  "originalEvent" : <aws.pipes.event.json>
}
```

正しい引用符で囲まれた有効な JSON が生成されます。

```
{
  "pipeArn" : "arn:aws:events:us-east-2:123456789012:pipe/example",
  "pipeName" : "example",
  "originalEvent" : {
    ... // commented for brevity
  }
}
```

- Lambda または Step Functions エンリッチメントまたはターゲットの場合、バッチサイズが 1 であっても、バッチは JSON 配列としてターゲットに配信されます。ただし、Input Transformers は引き続き JSON 配列の個々のレコードに適用され、配列全体には適用されません。詳細については、「???」を参照してください。

Amazon EventBridge Pipes のログ記録

EventBridge Pipes ログ記録を使用すると、 EventBridge パイプのパフォーマンスを詳細に説明したレコードをサポートされている AWS のサービスに送信できます。ログを使用して得たパイプの実行パフォーマンスへのインサイトを、トラブルシューティングやデバッグに役立てることができます。

EventBridge Pipes がレコードを配信するログ送信先として、次の AWS サービスを選択できます。

- CloudWatch ログ

  EventBridge は、指定された Logs CloudWatch ロググループにログレコードを配信します。

  CloudWatch Logs を使用して、使用するすべてのシステム、アプリケーション、および AWS サービスのログを 1 つの高度にスケーラブルなサービスに一元化します。詳細については、

「Amazon Logs ユーザーガイド」の「ロググループとログストリームの操作」を参照してください。 CloudWatch

- Firehose ストリームログ

  EventBridge は、Firehose 配信ストリームにログレコードを配信します。

  Amazon Data Firehose は、特定のサービス、およびサポートされているサードパーティーサービスプロバイダーが所有する AWS カスタム HTTP エンドポイントや HTTP エンドポイントなどの宛先にリアルタイムのストリーミングデータを配信するためのフルマネージドサービスです。詳細については、「Amazon Data Firehose ユーザーガイド」の「Amazon Data Firehose 配信ストリームの作成」を参照してください。

- Amazon S3 ログ

  EventBridge は、ログレコードを Amazon S3 オブジェクトとして指定されたバケットに配信します。

  Amazon S3 は、業界をリードするスケーラビリティ、データ可用性、セキュリティ、パフォーマンスを提供するオブジェクトストレージサービスです。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「Amazon S3 でのオブジェクトのアップロード、ダウンロード、操作」を参照してください。

Amazon EventBridge Pipes ログ記録の仕組み

パイプの実行は、パイプによって受信され、エンリッチメントやターゲットに送信されるイベントまたはイベントのバッチです。有効にすると、イベントバッチが処理されるときに実行する実行ステップごとにログレコード EventBridge が生成されます。レコード内の情報は、イベントバッチ (単一のイベントから最大 10,000 までのイベント) に適用されます。

パイプのソースとターゲットでイベントバッチのサイズを設定できます。詳細については、「???」を参照してください。

各ログ送信先に送信されるレコードデータは同じです。

Amazon CloudWatch Logs の送信先が設定されている場合、すべての送信先に配信されるログレコードの制限は 256 KB です。フィールドは必要に応じて切り捨てられます。

選択したログ送信先に EventBridge 送信されるレコードは、次の方法でカスタマイズできます。

- ログレベル を指定できます。これにより、 が選択したログの送信先にレコード EventBridge を送信する実行ステップが決まります。詳細については、「???」を参照してください。
- EventBridge Pipes が関連する実行ステップの実行データをレコードに含めるかどうかを指定できます。これには、以下のデータが含まれます。
  - イベントバッチのペイロード
  - AWS エンリッチメントまたはターゲットサービスに送信されたリクエスト
  - エン AWS リッチメントまたはターゲットサービスによって返されるレスポンス

  詳細については、「???」を参照してください。

EventBridge Pipes ログレベルの指定

が選択したログ送信先にレコード EventBridge を送信する実行ステップのタイプを指定できます。

ログレコードに含める詳細レベルを以下から選択します。ログレベルは、パイプに指定したすべてのログ送信先に適用されます。各ログレベルには、以前のログレベルの実行ステップが含まれます。

- OFF – EventBridge 指定されたログ送信先にレコードを送信しません。これはデフォルトの設定です。
- ERROR – パイプの実行中に生成されたエラーに関連するレコードを、指定されたログ送信先 EventBridge に送信します。
- INFO – エラーに関連するレコード EventBridge を送信し、パイプの実行中に実行される他のステップを指定されたログ送信先に送信します。
- TRACE – パイプ実行の任意のステップ中に生成されたレコードを、指定されたログ送信先 EventBridge に送信します。

EventBridge コンソールでは、 CloudWatch ログレベルと同様に、ログがデフォルトでERROR ログの送信先として選択されます。そのため、Pipes EventBridge はデフォルトで新しい CloudWatch ロググループを作成し、そのグループに詳細ERROR レベルを含むログレコードを送信します。ログをプログラムで設定する場合、デフォルトは選択されません。

次の表は、各ログレベルに含まれる実行ステップを示しています。

[ステップ]	TRACE	INFO	ERROR	VOFF
実行の失敗	x	x	x	

[ステップ]	TRACE	INFO	ERROR	VOFF
実行が部分的に失敗	x	x	x	
実行の開始	x	x		
実行の成功	x	x		
実行がスロットリング	x	x	x	
実行タイムアウト	x	x	x	
エンリッチメントの呼び出しが失敗	x	x	x	
エンリッチメント呼び出しをスキップ	x	x		
エンリッチメント呼び出しが開始	x			
エンリッチメント呼び出しが成功	x			
エンリッチメントステージ開始	x	x		
エンリッチメントステージが失敗	x	x	x	
エンリッチメントステージが成功	x	x		
エンリッチメント変換が失敗	x	x	x	
エンリッチメント変換が開始	x			
エンリッチメント変換が成功	x			
ターゲット呼び出しが失敗	x	x	x	

[ステップ]	TRACE	INFO	ERROR	VOFF
ターゲット呼び出しが部分的に失敗	x	x	x	
ターゲット呼び出しをスキップ	x			
ターゲット呼び出しが開始	x			
ターゲット呼び出しが成功	x			
ターゲットステージ開始	x	x		
ターゲットステージが失敗	x	x	x	
ターゲットステージが部分的に失敗	x	x	x	
ターゲットステージをスキップ	x			
ターゲットステージが成功	x	x		
ターゲット変換が失敗	x	x	x	
ターゲット変換が開始	x			
ターゲット変換が成功	x			

EventBridge Pipes ログに実行データを含める

が生成するレコード EventBridge に実行データを含めるように に指定できます。実行データには、イベントバッチペイロード、エンリッチメントとターゲットに送信されたリクエスト、エンリッチメントとターゲットからの応答を表すフィールドが含まれます。

実行データは、トラブルシューティングやデバッグに役立ちます。payload フィールドには、バッチに含まれる各イベントの実際の内容が含まれているため、個々のイベントを特定のパイプ実行に関連付けることができます。

実行データを含めることを選択した場合、実行データはパイプに指定したすべてのログ送信先に含まれます。

> **Important**
>
> これらのフィールドには機密情報が含まれている場合があります。 EventBridge は、ログ記録中にこれらのフィールドの内容を編集しようとしません。

実行データを含めると、 は関連するレコードに次のフィールド EventBridge を追加します。

- **payload**

  パイプで処理しているイベントバッチの内容を表します。

  EventBridge は、イベントバッチの内容が更新された可能性のあるステップで生成されたレコードに payloadフィールドを含めます。これには以下のステップが含まれます。

  - EXECUTION_STARTED
  - ENRICHMENT_TRANSFORMATION_SUCCEEDED
  - ENRICHMENT_STAGE_SUCCEEDED
  - TARGET_TRANSFORMATION_SUCCEEDED
  - TARGET_STAGE_SUCCEEDED

- **awsRequest**

  エンリッチメントまたはターゲットに送信された JSON 文字列のリクエストを表します。API 送信先に送信されるリクエストの場合、これはそのエンドポイントに送信された HTTP リクエストを表します。

  EventBridge は、エンリッチメントとターゲット化の最終ステップで生成されたレコードに awsRequestフィールドを含めます。つまり、 が指定されたエンリッチメントまたはターゲットサービスに対してリクエスト EventBridge を実行または実行しようとした後です。これには以下のステップが含まれます。

  - ENRICHMENT_INVOCATION_FAILED
  - ENRICHMENT_INVOCATION_SUCCEEDED
  - TARGET_INVOCATION_FAILED
  - TARGET_INVOCATION_PARTIALLY_FAILED

- TARGET_INVOCATION_SUCCEEDED

- **awsResponse**

  エンリッチメントまたはターゲットから返された JSON 形式のレスポンスを表します。API 送信先に送信されたリクエストの場合、これはそのエンドポイントから返された HTTP レスポンスを表します。

  と同様にawsRequest、 は、エンリッチメントとターゲット化の最終ステップで生成されたレコードに awsResponseフィールド EventBridge を含めます。つまり、 が指定されたエンリッチメントまたはターゲットサービスに対してリクエスト EventBridge を実行または実行を試み、レスポンスを受信した後です。これには以下のステップが含まれます。

  - ENRICHMENT_INVOCATION_FAILED
  - ENRICHMENT_INVOCATION_SUCCEEDED
  - TARGET_INVOCATION_FAILED
  - TARGET_INVOCATION_PARTIALLY_FAILED
  - TARGET_INVOCATION_SUCCEEDED

パイプ実行ステップの説明については、「???」を参照してください。

Pipes ログレコードの実行データの EventBridge切り捨て

パイプのログレコードに実行データ EventBridge を含めることを選択した場合、レコードが 256 KB のサイズ制限を超える可能性があります。これを防ぐために、 EventBridge は実行データフィールドを自動的に切り捨てます。 は、次のフィールドを切り捨てに進む前に、各フィールドを完全に EventBridge 切り捨てます。 は、データ文字列の末尾から文字を削除するだけでフィールドデータ EventBridge を切り捨てます。データの重要度に基づいて切り捨てようとはせず、切り捨ては JSON 形式を無効にします。

- payload
- awsRequest
- awsResponse

EventBridge がイベントのフィールドを切り捨てる場合、 truncatedFieldsフィールドには切り捨てられたデータフィールドのリストが含まれます。

EventBridge Pipes ログレコードでのエラーレポート

EventBridge には、障害状態を表すパイプ実行ステップに、エラーデータも含まれます。ステップには以下が含まれます。

- `ExecutionThrottled`
- `ExecutionTimeout`
- `ExecutionFailed`
- `ExecutionPartiallyFailed`
- `EnrichmentTransformationFailed`
- `EnrichmentInvocationFailed`
- `EnrichmentStageFailed`
- `TargetTransformationFailed`
- `TargetInvocationFailed`
- `TargetInvocationPartiallyFailed`
- `TargetStageFailed`
- `TargetStagePartiallyFailed`

EventBridge パイプ実行ステップ

パイプ実行ステップのフローを理解しておくと、ログを使用したパイプのパフォーマンスのトラブルシューティングやデバッグに役立ちます。

パイプ実行は、パイプによって受信され、エンリッチメントまたはターゲットに送信されるイベントまたはイベントのバッチです。有効にすると、イベントバッチが処理されるときに実行する実行ステップごとにログレコード EventBridge が生成されます。

簡単に言うと、実行にはエンリッチメントとターゲットという 2 つのステージ (ステップの集合) が含まれます。各ステージは、変換ステップと呼び出しステップで構成されます。

パイプを正常に実行するための主なステップのフローは次のとおりです。

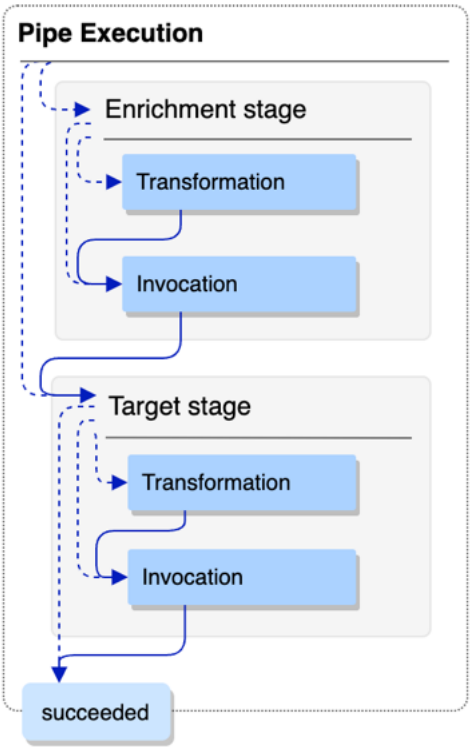
- パイプの実行を開始します。
- イベントのエンリッチメントを指定した場合、実行はエンリッチメントステージに入ります。エンリッチメントを指定していない場合、実行はターゲットステージに進みます。

エンリッチメントステージの場合、パイプは、指定した変換を実行してからエンリッチメントを呼び出します。

- ターゲットステージの場合、パイプは、指定した変換を実行してからターゲットを呼び出します。

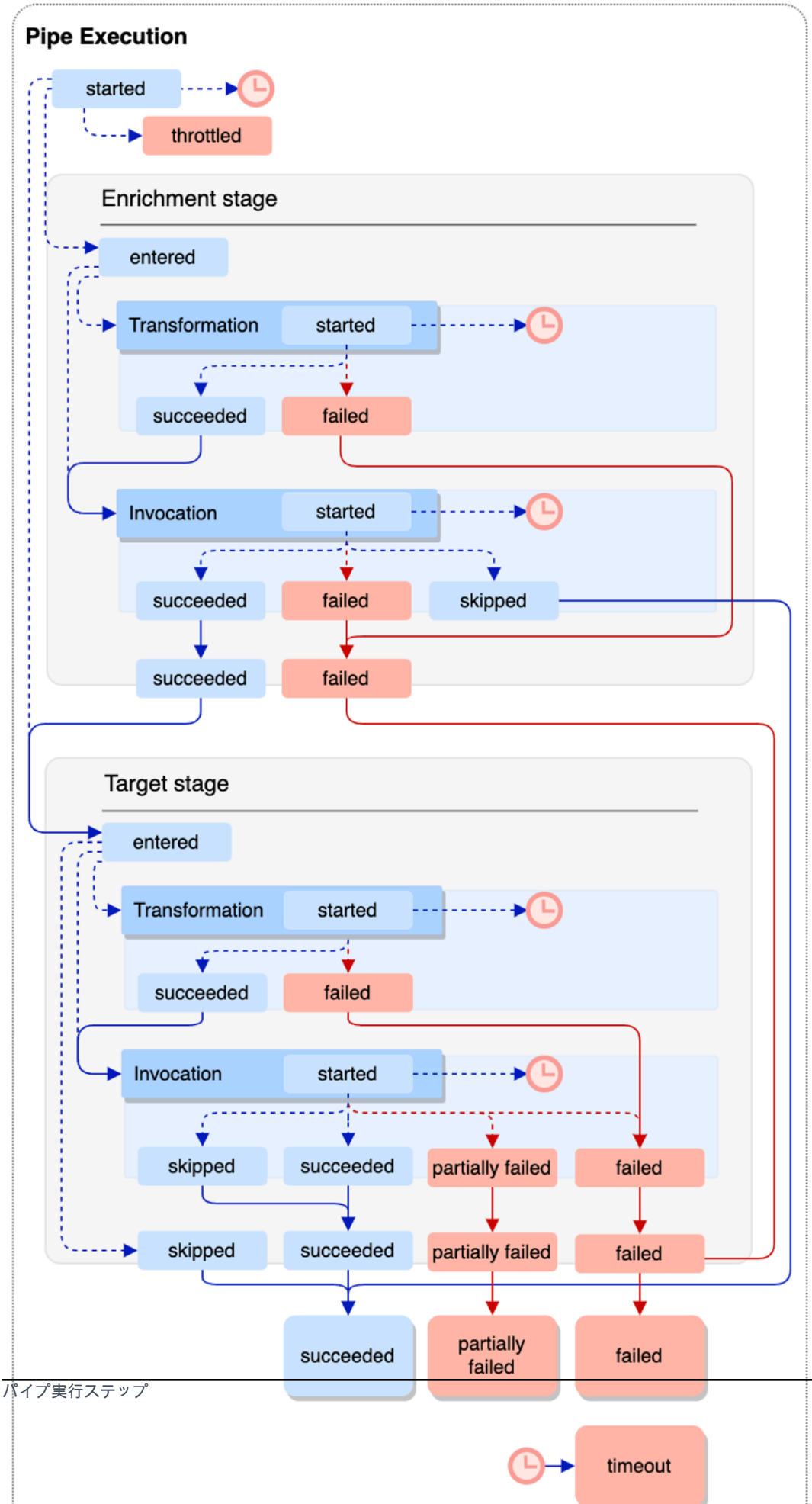
  変換またはターゲットを指定していない場合、実行はターゲットステージをスキップします。
- パイプの実行が正常に完了します。

次の図は、このフローを示しています。分岐経路は点線で示しています。



次は、パイプの実行フローの詳細図であり、考えられるすべての実行ステップを示しています。ここでも、分岐経路は点線で示しています。

パイプ実行ステップの詳細なリストについては、「???」を参照してください。


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

ターゲットの呼び出しにより、バッチが部分的に失敗する可能性があることに注意してください。詳細については、「???」を参照してください。

EventBridge Pipes ログスキーマリファレンス

次のリファレンスでは、Pipes EventBridge ログレコードのスキーマについて詳しく説明します。

各ログレコードはパイプ実行ステップを表します。パイプのソースとターゲットがバッチ処理用に設定されている場合、各ログレコードには最大 10,000 件のイベントが含まれます。

詳細については、「???」を参照してください。

```
{
    "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

パイプ実行の ID。

パイプ実行は、パイプによって受信され、エンリッチメントまたはターゲットに送信されるイベントまたはイベントのバッチです。詳細については、「???」を参照してください。

timestamp

ログイベントが発生した日時。

単位: ミリ秒

messageType

レコードが生成されたパイプ実行ステップ。

パイプ実行ステップの詳細については、「???」を参照してください。

resourceArn

パイプの Amazon リソースネーム (ARN)。

logLevel

パイプログに指定した詳細レベル。

有効な値: `ERROR | INFO | TRACE`

詳細については、「???」を参照してください。

payload

パイプが処理しているイベントバッチの内容。

EventBridge には、このパイプのログに実行データを含めるように を指定している場合にのみ、このフィールドが含まれます。詳細については、「???」を参照してください。

> **Important**
>
> これらのフィールドには機密情報が含まれている場合があります。 EventBridge は、ログ記録中にこれらのフィールドの内容を編集しようとしません。

詳細については、「???」を参照してください。

awsRequest

エンリッチメントまたはターゲットに送信される JSON 形式のリクエスト。API 送信先に送信されるリクエストの場合、これはそのエンドポイントに送信された HTTP リクエストを表します。

EventBridge には、このパイプのログに実行データを含めるように を指定している場合にのみ、このフィールドが含まれます。詳細については、「???」を参照してください。

> **Important**
>
> これらのフィールドには機密情報が含まれている場合があります。 EventBridge は、ログ記録中にこれらのフィールドの内容を編集しようとしません。

詳細については、「???」を参照してください。

awsResponse

エンリッチメントまたはターゲットから返される JSON 形式のレスポンス。API 送信先に送信されたリクエストの場合、これはそのエンドポイントから返された HTTP レスポンスを表します。API 送信先のサービス自体から返されたレスポンスではありません。

EventBridge には、このパイプのログに実行データを含めるように を指定している場合にのみ、このフィールドが含まれます。詳細については、「???」を参照してください。

> **Important**
>
> これらのフィールドには機密情報が含まれている場合があります。 EventBridge は、ログ記録中にこれらのフィールドの内容を編集しようとしません。

詳細については、「???」を参照してください。

truncatedFields

実行データフィールドのリスト EventBridge が切り捨てられ、レコードが 256 KB のサイズ制限を下回っています。

EventBridge が実行データフィールドを切り捨てる必要がない場合、このフィールドは存在しますが、 ですnull。

詳細については、「???」を参照してください。

エラー

このパイプ実行ステップ中に発生したエラーの情報を示します。

このパイプ実行ステップ中にエラーが発生しなかった場合、このフィールドは表示されますが、null となります。

httpStatusCode

呼び出したサービスから返された HTTP ステータスコード。

メッセージ

呼び出したサービスから返されたエラーメッセージ。

details

呼び出したサービスから返された詳細なエラー情報。

awsService

呼び出したサービスの名前。

requestId

呼び出したサービスからの、このリクエストのリクエスト ID。

および Amazon CloudWatch Logs を使用した Amazon EventBridge Pipes のログ記録 AWS CloudTrail とモニタリング

EventBridge Pipes の呼び出しをログに記録し、 CloudWatch CloudTrail メトリクスを使用してパイプの状態をモニタリングできます。

CloudWatch メトリクス

EventBridge Pipes は、パイプの実行がスロットリングされてからターゲットが正常に呼び出されるまで、すべてのメトリクスを 1 分 CloudWatch ごとに Amazon に送信します。

メトリクス	説明	ディメンション	単位
`Concurrency`	パイプの同時実行数。	AwsAccountId	なし
`Duration`	パイプの実行にかかった時間。	PipeName	ミリ秒
`EventCount`	パイプが処理したイベントの数。	PipeName	なし
`EventSize`	パイプを呼び出したイベントのペイロードのサイズ。	PipeName	バイト
`ExecutionThrottled`	パイプの実行がスロットリングされた回数。 **Note** この値は、実行がスロットリングされなかった場合 `0` になります。	AwsAccountId, PipeName	なし

メトリクス	説明	ディメンション	単位
ExecutionTimeout	実行が完了する前にパイプの実行がタイムアウトした回数。 Note この値は、実行がタイムアウトしなかった場合 0 になります。	PipeName	なし
ExecutionFailed	パイプの実行が失敗した回数。 Note この値は、実行が失敗しなかった場合 0 になります。	PipeName	なし
ExecutionPartiallyFailed	パイプの実行が部分的に失敗した回数。 Note この値は、実行が部分的に失敗しなかった場合 0 になります。	PipeName	なし
EnrichmentStageDuration	エンリッチメントステージが完了するまでにかかった時間。	PipeName	ミリ秒
EnrichmentStageFailed	パイプのエンリッチメントステージの実行が失敗した回数。 Note この値は、実行が失敗しなかった場合 0 になります。	PipeName	なし

メトリクス	説明	ディメンション	単位
Invocations	呼び出しの合計数。	AwsAccountId, PipeName	なし
TargetStageDuration	ターゲットステージが完了するまでにかかった時間。	PipeName	ミリ秒
TargetStageFailed	パイプのターゲットステージの実行が失敗した回数。 **Note** この値は、実行が失敗しなかった場合 0 になります。	PipeName	なし
TargetStagePartiallyFailed	パイプのターゲットステージの実行が部分的に失敗した回数。 **Note** この値は、ターゲットステージ実行が部分的に失敗しなかった場合 0 になります。	PipeName	なし
TargetStageSkipped	パイプのターゲットステージの実行がスキップされた回数 (例えば、エンリッチメントによって空のペイロードが返された場合)。	PipeName	カウント

CloudWatch メトリクスのディメンション

CloudWatch メトリクスには、ディメンション またはソート可能な属性があり、以下にリストされています。

ディメンション	説明
AwsAccountId	使用可能なメトリクスをアカウント ID でフィルター処理します。
PipeName	使用可能なメトリクスをパイプ名でフィルター処理します。

Amazon EventBridge Pipes のエラー処理とトラブルシューティング

再試行動作とエラー処理

EventBridge Pipes は、ソースサービス、エンリッチメントまたはターゲットサービス、または で再試行可能な AWS 障害が発生した場合、エンリッチメントとターゲット呼び出しを自動的に再試行します EventBridge。ただし、エンリッチメントやターゲットであるお客様の実装によって障害が返された場合、パイプポーリングのスループットは徐々に低下します。ほぼ連続的な 4xx エラー (IAM の認証問題やリソース不足など) の場合は、StateReason で説明メッセージを表示してパイプを自動的に無効化できます。

パイプ呼び出しエラーと再試行動作

パイプを呼び出すとき、主にパイプの内部エラーとお客様呼び出しエラーの 2 種類のエラーが発生する可能性があります。

パイプの内部エラー

パイプの内部エラーは、 EventBridge Pipes サービスによって管理される呼び出しの側面に起因するエラーです。

この種のエラーは、以下のような問題が原因です。

- カスタマーターゲットサービスを呼び出そうとしたときに HTTP 接続が失敗しました。
- パイプサービス自体の可用性が一時的に低下しました。

一般に、 EventBridge Pipes は内部エラーを無期限に再試行し、ソースでレコードの有効期限が切れたときにのみ停止します。

ストリームソースを持つパイプの場合、 EventBridge Pipes はストリームソースの再試行ポリシーで指定された最大再試行回数に対して内部エラーの再試行をカウントしません。Amazon SQS ソースを持つパイプの場合、 EventBridge Pipes は Amazon SQS ソースの最大受信数に対して内部エラーの再試行をカウントしません。

お客様呼び出しエラー

お客様呼び出しエラーは、ユーザーが管理する設定またはコードに起因するエラーです。

この種のエラーは、以下のような問題が原因です。

- ターゲットを呼び出すにはパイプのアクセス許可が不十分です。
- 同期的に呼び出されたお客様の Lambda、Step Functions、API 送信先、または API Gateway エンドポイントのロジックエラー。

顧客呼び出しエラーの場合、 EventBridge Pipes は以下を実行します。

- ストリームソースを持つパイプの場合、 EventBridge Pipes はパイプ再試行ポリシーで設定された最大再試行時間まで、または最大レコード期間が終了するまでのいずれか早い方まで再試行します。
- Amazon SQS ソースを持つパイプの場合、 EventBridge Pipes はソースキューの最大受信数まで顧客エラーを再試行します。
- Apache Kafka または Amazon MQ ソースを持つパイプの場合、 は内部エラーを EventBridge 再試行するのと同じ方法でカスタマーエラーを再試行します。

コンピューティングターゲットを持つパイプの場合、 EventBridge Pipes が顧客のコンピューティングロジックからスローされたランタイムエラーを認識し、そのようなエラーを再試行するには、パイプを同期的に呼び出す必要があります。Pipes は、Step Functions の Standard ワークフローのロジックからスローされたエラーは再試行できません。このターゲットは非同期で呼び出す必要があるためです。

Kinesis や DynamoDB などの Amazon SQS およびストリームソースの場合、 EventBridge Pipes はターゲット障害の部分的なバッチ障害処理をサポートします。 DynamoDB 詳細については、「部分的なバッチ処理失敗」を参照してください。

パイプの DLQ 動作

パイプはデッドレターキュー (DLQ) の動作をソースから継承します。

- ソース Amazon SQS キューに DLQ が設定されている場合、メッセージは指定された回数の試行後に自動配信されます。
- DynamoDB や Kinesis ストリームなどのストリームソースでは、パイプイベントとルートイベントに DLQ を設定できます。DynamoDB と Kinesis ストリームソースは、Amazon SQS キューと Amazon SNS トピックを DLQ ターゲットとしてサポートしています。

Kinesis または DynamoDB ソースを含むパイプに `DeadLetterConfig` を指定する場合は、パイプの `MaximumRecordAgeInSeconds` プロパティがソースイベントの `MaximumRecordAge` プロパティよりも小さいことを確認してください。 `MaximumRecordAgeInSeconds` は、パイプポーラーがイベントをあきらめて DLQ に配信するタイミングを制御し、`MaximumRecordAge` は、メッセージが削除されるまでのソースストリームに表示される時間を制御します。そのため、`MaximumRecordAgeInSeconds` は、イベントが DLQ に送信されてからソースによって自動的に削除されるまでの間に十分な時間があって、イベントが DLQ に送信された理由を判断できるように、ソース `MaximumRecordAge` よりも小さい値に設定します。

Amazon MQ ソースの場合、DLQ はメッセージブローカー上で直接設定できます。

EventBridge Pipes は、ストリームソースの先入れ先出し (FIFO) DLQs をサポートしていません。

EventBridge Pipes は、Amazon MSK ストリームおよびセルフマネージド Apache Kafka ストリームソースの DLQ をサポートしていません。

パイプの障害状態

パイプの作成、削除、更新は非同期操作であり、障害状態になる可能性があります。同様に、パイプは障害により自動的に無効になる場合があります。いずれの場合も、パイプ `StateReason` は障害のトラブルシューティングに役立つ情報を提供します。

以下は可能性のある `StateReason` 値の例を示しています。

- ストリームが見つかりません。処理を再開するには、パイプを削除して新しいパイプを作成します。
- Pipes には、キューオペレーション (sqs:ReceiveMessage、sqs:DeleteMessage 、sqs:GetQueueAttributes) を実行するために必要なアクセス許可がありません
- 接続エラー。VPC はパイプに接続できる必要があります。Pipes-data に NAT ゲートウェイまたは VPC エンドポイントを設定することで、アクセスを提供できます。pipes-data に NAT ゲートウェ

イまたは VPC エンドポイントを設定する方法については、 AWS ドキュメントを確認してください。

- MSK クラスターはセキュリティグループが関連付けられていません

パイプは、StateReason を更新すると自動的に停止することがあります。考えられる理由は以下のとおりです。

- エンリッチメントとして設定された Step Functions の標準ワークフロー。
- 同期的に呼び出されるターゲットとして設定された Step Functions の標準ワークフロー。

カスタム暗号化の失敗

AWSが管理するキーではなく、 AWS KMS カスタム暗号化キー (CMK) を使用するようにソースを設定する場合は AWS KMS 、パイプの実行ロールの復号化アクセス許可を明示的に付与する必要があります。そのためには、カスタム CMK ポリシーに次の追加のアクセス許可を含めてください。

```
  {
      "Sid": "Allow Pipes access",
      "Effect": "Allow",
      "Principal": {
          "AWS": "arn:aws:iam::01234567890:role/service-role/
Amazon_EventBridge_Pipe_DDBStreamSourcePipe_12345678"
      },
      "Action": "kms:Decrypt",
      "Resource": "*"
  }
```

上記のロールをパイプの実行ロールに置き換えます。

これは、以下を含む AWS KMS CMK を使用するすべてのパイプソースに当てはまります。

- Amazon DynamoDB Streams
- Amazon Kinesis Data Streams
- Amazon MQ
- Amazon MSK
- Amazon SQS

チュートリアル: ソースイベントをフィルタリングする EventBridge パイプを作成する

このチュートリアルでは、DynamoDB ストリームソースを Amazon SQS キューターゲットに接続するパイプを作成します。これには、キューに配信するイベントをフィルタリングするときに使用するパイプのイベントパターンを作成することも含まれます。次に、パイプをテストして、必要なイベントだけが配信されることを確認します。

前提条件: ソースとターゲットを作成する

パイプを作成する前に、パイプを接続する先のソースとターゲットを作成する必要があります。この場合、Amazon DynamoDB データストリームがパイプのソースとなり、Amazon SQS キューがパイプのターゲットとなります。

このステップを簡単にするために、AWS CloudFormation を使用してソースリソースとターゲットリソースをプロビジョニングできます。そのためには、以下のリソースを定義する CloudFormation テンプレートを作成します。

- パイプのソース

  Amazon DynamoDB テーブル内の項目の変更に関する情報の順序付けされたフローを提供するストリームが有効になっている、`pipe-tutorial-source` という名前の DynamoDB テーブル。
- パイプのターゲット

  パイプからイベントの DynamoDB ストリームを受信するための `pipe-tutorial-target` という名前の Amazon SQS キュー。

パイプリソースをプロビジョニングするための CloudFormation テンプレートを作成するには

1. 以下の「???」セクションにある JSON テンプレートテキストをコピーします。
2. テンプレートを JSON ファイル (例: `~/pipe-tutorial-resources.json`) として保存します。

次に、作成したテンプレートファイルを使用して CloudFormation スタックをプロビジョニングします。

> **Note**
>
> CloudFormation スタックを作成すると、そのスタックでプロビジョニングした AWS リソースに対して課金されます。

AWS CLI を使用してチュートリアルの前提条件をプロビジョニングする

- 次の CLI コマンドを実行します。ここで、`--template-body` はテンプレートファイルの場所を指定します。

```
aws cloudformation create-stack --stack-name pipe-tuturial-resources --template-body file://~/pipe-tutorial-resources.json
```

CloudFormation コンソールを使用してチュートリアルの前提条件をプロビジョニングする

1. AWS CloudFormation コンソール (https://console.aws.amazon.com/cloudformation) を開きます。
2. [スタック]、[スタックを作成]、[新しいリソースを使用 (標準)] の順に選択します。

   CloudFormation は、[スタックを作成] ウィザードを表示します。
3. [前提条件 – テンプレートの準備] で、デフォルトの [テンプレートの準備完了] を選択したままにします。
4. [テンプレートを指定] で、[テンプレートファイルをアップロード] を選択し、ファイルを選択して [次へ] を選択します。
5. スタックと、プロビジョニングするリソースを設定します。
   - [スタック名] に「`pipe-tuturial-resources`」と入力します。
   - [パラメータ] で、DynamoDB テーブルと Amazon SQS キューの名前をデフォルトのままにします。
   - [次へ] をクリックします。
6. [次へ] を選択し、[送信] を選択します。

   CloudFormation はスタックを作成し、テンプレートで定義されたリソースをプロビジョニングします。

CloudFormation の詳細については、「AWS CloudFormation ユーザーガイド」の「AWS CloudFormation とは」を参照してください。

ステップ 1: パイプを作成する

パイプのソースとターゲットをプロビジョニングしたら、2 つのサービスを接続するパイプを作成できます。

EventBridge コンソールを使用してパイプを作成する

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで、[パイプ] を選択します。
3. [パイプを作成] を選択します。
4. [名前] で、パイプ名を「`pipe-tutorial`」とします。
5. DynamoDB データストリームソースを指定します。
   a. [詳細] の [ソース] で、[DynamoDB データストリーム] を選択します。

      EventBridge は、DynamoDB 固有のソース構成設定を表示します。
   b. [DynamoDB ストリーム] で、[`pipe-tutorial-source`] を選択します。

      [開始位置] は、デフォルトの `Latest` に設定したままにします。
   c. [次へ] をクリックします。
6. イベントパターンを指定してテストし、イベントをフィルタリングします。

   フィルタリングにより、パイプからエンリッチメントまたはターゲットに送信するイベントを制御できます。パイプは、イベントパターンに一致するイベントのみを、エンリッチメントまたはターゲットに送信します。

   詳細については、「???」を参照してください。

   > **Note**
   >
   > エンリッチメントまたはターゲットに送信されたイベントにのみ課金されます。

   a. [サンプルイベント - オプション] で、[AWS イベント] を選択したままにし、[DynamoDB ストリームサンプルイベント 1] が選択されていることを確認します。

これは、イベントパターンをテストするために使用するサンプルイベントです。

b. [イベントパターン] に、次のイベントパターンを入力します。

```
{
  "eventName": ["INSERT", "MODIFY"]
}
```

c. [パターンをテスト] を選択します。

EventBridge は、サンプルイベントがイベントパターンと一致するというメッセージを表示します。これは、サンプルイベントの `eventName` の値が `INSERT` であるためです。

d. [次へ] をクリックします。

7. [次へ] を選択して、エンリッチメントの指定をスキップします。

この例では、エンリッチメントを選択しません。エンリッチメントを使用すると、ソースからターゲットにデータを送信する前に、データを強化するサービスを選択できます。詳細については、「???」を参照してください。

8. Amazon SQS キューをパイプのターゲットとして指定します。

a. [詳細] の [ターゲットサービス] で、[Amazon SQS キュー] を選択します。

b. [キュー] で、[`pipe-tutorial-target`] を選択します。

c. [ターゲット入力トランスフォーマー] セクションは空白のままにしておきます。

詳細については、「???」を参照してください。

9. [パイプを作成] を選択します。

EventBridge はパイプを作成し、パイプの詳細ページを表示します。パイプのステータスが `Running` に更新されると、パイプの準備は完了です。

ステップ 2: パイプがイベントをフィルタリングすることを確認する

パイプは設定済みでも、まだテーブルからイベントを受信していません。

パイプをテストするには、DynamoDB テーブルのエントリを更新します。更新するたびに、DynamoDB ストリームがパイプに送信するイベントが生成されます。指定したイベントパターンと一致するものもあれば、一致しないものもあります。次に Amazon SQS キューを調べて、パイプがイベントパターンに一致するイベントのみを配信したことを確認できます。

テーブル項目を更新してイベントを生成する

1. DynamoDB コンソール (https://console.aws.amazon.com/dynamodb/) を開きます。
2. 左のナビゲーションで、[テーブル] を選択します。pipe-tutorial-source テーブルを選択します。

   DynamoDB は pipe-tutorial-source のテーブル詳細ページを表示します。
3. [テーブル項目の探索] を選択し、[項目を作成] を選択します。

   DynamoDB は [項目を作成] ページを表示します。
4. [属性] で、新しいテーブル項目を作成します。

   a. [アルバム] に「Album A」と入力します。

   b. [アーティスト] に「Artist A」と入力します。

   c. [項目を作成] を選択します。
5. テーブル項目を更新します。

   a. [返された項目] で [アルバム A] を選択します。

   b. [新しい属性を追加]] を選択し、[文字列] を選択します。

   c. Song の新しい値として「Song A」と入力します。

   d. [変更を保存] を選択します。
6. テーブル項目を削除します。

   a. [返された項目] で、[アルバム A] をオンにします。

   b. [アクション] メニューから、[項目を削除] を選択します。

テーブル項目を 3 回更新しました。これにより、DynamoDB データストリームに次の 3 つのイベントが生成されます。

- 項目を作成したときの INSERT イベント。
- 項目に属性を追加したときの MODIFY イベント。
- 項目を削除したときの REMOVE イベント。

ただし、パイプに指定したイベントパターンにより、INSERT または MODIFY 以外のイベントは除外されるはずです。次に、パイプが予期したイベントをキューに配信したことを確認します。

予期したイベントがキューに配信されたことを確認する

1. Amazon SQS コンソール (https://console.aws.amazon.com/sqs/) を開きます。
2. `pipe-tutorial-target` キューを選択します。

   Amazon SQS は、キューの詳細ページを表示します。
3. [メッセージを送受信] を選択し、[メッセージを受信] で [メッセージをポーリング] を選択します。

   キューはパイプをポーリングし、受信したイベントを一覧表示します。
4. イベント名を選択すると、配信されたイベント JSON が表示されます。

キューには 2 つのイベントがあるはずです。1 つは `eventName` の `INSERT`、もう 1 つは `eventName` の `MODIFY` です。ただし、パイプはテーブル項目を削除するためのイベントは配信しませんでした。このイベントは `eventName` が `REMOVE` であり、パイプで指定したイベントパターンと一致しないためです。

ステップ 3: リソースをクリーンアップする

まず、パイプ自体を削除します。

EventBridge コンソールを使用してパイプを削除する

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで、[パイプ] を選択します。
3. `pipe-tutorial` パイプを選択し、[削除] を選択します。

次に、CloudFormation スタックを削除して、そのスタック内でプロビジョニングされたリソースの継続的な使用に対して請求されないようにします。

AWS CLI を使用してチュートリアルの前提条件を削除する

- 次の CLI コマンドを実行します。ここで、`--stack-name` はスタックの名前を指定します。

```
aws cloudformation delete-stack --stack-name pipe-tuturial-resources
```

AWS CloudFormation コンソールを使用してチュートリアルの前提条件を削除する

1. AWS CloudFormation コンソール (https://console.aws.amazon.com/cloudformation) を開きます。
2. [スタック] ページで、スタックを選択して [削除] を選択します。
3. [削除] を選択して削除を確定します。

前提条件を生成するための AWS CloudFormation テンプレート

次の JSON を使用して、このチュートリアルに必要なソースリソースとターゲットリソースをプロビジョニングするための CloudFormation テンプレートを作成します。

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

AWS CloudFormation EventBridge パイプからテンプレートを生成

AWS CloudFormation インフラストラクチャーをコードとして扱うことで、 AWS 複数のアカウントやリージョンのリソースを一元的かつ繰り返し可能な方法で設定および管理できます。CloudFormation これは、プロビジョニングや管理の対象となるリソースを定義するテンプレートを作成できるようにすることで実現されます。

EventBridge アカウント内の既存のパイプからテンプレートを生成できるため、テンプレートの開発をすぐに開始できます。 CloudFormation 1 つまたは複数のパイプを選択してテンプレートに含めることができます。その後、これらのテンプレートを基礎として使用して、管理対象のリソーススタックを作成できます。 CloudFormation

詳細については CloudFormation、『 AWS CloudFormation ユーザーガイド』を参照してください。

イベントバスでは、 CloudFormation イベントバスとイベントバスルールからテンプレートを生成できます。

EventBridge パイプテンプレートに含まれるリソース

EventBridge CloudFormation テンプレートを生成すると、AWS::Pipes::Pipe選択したパイプごとにリソースが作成されます。さらに、 EventBridge 説明されている条件下では以下のリソースが含まれます。

- AWS::Events::ApiDestination

  パイプに API 宛先がエンリッチメントまたはターゲットとして含まれている場合は、 EventBridge CloudFormation AWS::Events::ApiDestinationそれらをリソースとしてテンプレートに含めます。
- AWS::Events::EventBus

  パイプにターゲットとしてイベントバスが含まれている場合は、 EventBridge CloudFormation AWS::Events::EventBus それをリソースとしてテンプレートに含めます。
- AWS::IAM::Role

  EventBridge パイプを構成したときに新しい実行ロールを作成した場合は、 EventBridge AWS::IAM::Role そのロールをリソースとしてテンプレートに含めるように選択できます。 EventBridge 作成したロールは含まれません。(いずれの場合も、`RoleArn` AWS::Pipes::Pipe リソースのプロパティにはロールの ARN が含まれます)。

Pipes CloudFormation EventBridge から生成されたテンプレートを使用する際の考慮事項

CloudFormation から生成したテンプレートを使用するときは、以下の要素を考慮してください EventBridge。

- EventBridge 生成するテンプレートにはパスワードは一切含まれません。

テンプレートを編集して、CloudFormationユーザーがテンプレートを使用してスタックを作成または更新する際にパスワードやその他の機密情報を指定できるようにするテンプレートパラメータを含めることができます。

さらに、ユーザーは Secrets Manager を使用して目的のリージョンにシークレットを作成し、生成されたテンプレートを編集して動的パラメーターを使用できます。

- 生成したテンプレートのターゲットは、元のパイプで指定していたものと同じです。テンプレートを使用して他のリージョンにスタックを作成する前に、テンプレートを適切に編集しないと、リージョン間の問題が発生する可能性があります。

  また、生成したテンプレートでは、下流のターゲットが自動的には作成されません。

EventBridge Pipes CloudFormation からテンプレートを生成します。

EventBridge コンソールを使用して 1 CloudFormation つまたは複数のパイプからテンプレートを生成するには、次の操作を行います。

1 CloudFormation つまたは複数のパイプからテンプレートを生成するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで、[パイプ] を選択します。
3. 「パイプ」で、 CloudFormation 生成されたテンプレートに含めたいパイプを 1 つ以上選択します。

   パイプが 1 つの場合は、パイプの詳細ページに表示するパイプ名を選択することもできます。
4. [CloudFormation テンプレート] を選択し、 EventBridge テンプレートを生成する形式 (JSON または YAML) を選択します。

   EventBridge 選択した形式で生成されたテンプレートが表示されます。
5. EventBridge 選択したパイプのいずれかに新しい実行ロールを作成し、 EventBridge そのロールをテンプレートに含めたい場合は、「 IAM ユーザーに代わってコンソールによって作成されたロールを含める」を選択します。
6. EventBridge テンプレートファイルをダウンロードするか、テンプレートをクリップボードにコピーするかを選択できます。

   - テンプレートファイルをダウンロードするには、[Download] (ダウンロード) を選択します。
   - テンプレートをクリップボードにコピーするには、[Copy] (コピー) を選択します。

7. テンプレートを終了するには、[Cancel] (キャンセル) を選択します。

グローバルエンドポイントとイベントレプリケーションにより、アプリケーションをリージョンフォールトトレラントにする

Amazon EventBridge グローバルエンドポイントを使用すると、アプリケーションの可用性を向上させることができます。グローバルエンドポイントを使用すると、追加コストなしでアプリケーションをリージョンフォールトトレラントにできます。開始するには、エンドポイントに Amazon Route 53 ヘルスチェックを割り当てます。フェイルオーバーが開始されると、ヘルスチェックは「unhealty」状態を報告します。フェイルオーバーの開始から数分以内に、すべてのカスタムイベントがセカンダリリージョンのイベントバスにルーティングされ、そのイベントバスによって処理されます。ヘルスチェックが「healty」状態を報告すると、イベントはプライマリリージョンのイベントバスによって処理されます。

グローバルエンドポイントを使用する場合、イベントレプリケーションを有効にできます。イベントレプリケーションは、マネージドルールを使用して、すべてのカスタムイベントをプライマリリージョンとセカンダリリージョンのイベントバスに送信します。

> **Note**
>
> カスタムバスを使用している場合、フェイルオーバーが正常に機能するためには、各リージョンに同じ名前と同じアカウントを持つカスタムバスが必要です。

トピック

- 目標復旧時間と目標復旧ポイント
- イベントレプリケーション
- グローバルエンドポイントの作成
- AWS SDK を使用したグローバルエンドポイントの操作
- 利用できるリージョン
- Amazon EventBridge グローバルエンドポイントを使用するためのベストプラクティス
- Route 53 ヘルスチェックを設定するための AWS CloudFormation テンプレート

目標復旧時間と目標復旧ポイント

目標復旧時間 (RTO) は、障害発生後にセカンダリリージョンがイベントの受信を開始するまでの所要時間です。RTO の場合、時間には、Route 53 ヘルスチェックの CloudWatch アラームをトリガーし、ステータスを更新する期間が含まれます。目標復旧ポイント (RPO) は、障害時に未処理のままにされるデータの尺度です。RPO の場合、時間には、セカンダリリージョンにレプリケートされず、サービスまたはリージョンが回復するまでプライマリリージョンで停止するイベントが含まれます。グローバルエンドポイントを使用すると、アラーム設定に関する規範的なガイダンスに従った場合、RTO と RPO は 360 秒、最大 420 秒になると予想されます。

イベントレプリケーション

イベントは、セカンダリリージョンで非同期的に処理されます。つまり、イベントは両方のリージョンで同時に処理されるとは限りません。フェイルオーバーがトリガーされると、イベントはセカンダリリージョンによって処理され、使用可能な場合はプライマリリージョンによって処理されます。イベントレプリケーションを有効にすると、月額のコストが増加します。詳細については、「Amazon EventBridge の料金」を参照してください。

グローバルエンドポイントを設定する場合は、次の理由から、イベントレプリケーションを有効にすることをお勧めします。

- イベントレプリケーションは、グローバルエンドポイントが正しく設定されていることを確認するのに役立ちます。これにより、フェイルオーバーが発生した場合に確実にカバーできます。
- フェイルオーバーイベントから自動的にリカバリするには、イベントレプリケーションが必要です。イベントレプリケーションを有効にしていない場合は、イベントがプライマリリージョンに戻る前に、Route 53 ヘルスチェックを手動で「healty」にリセットする必要があります。

レプリケートされたイベントのペイロード

レプリケートされたイベントのペイロードの例を次に示します。

> **Note**
>
> `region` については、イベントのレプリケーション元のリージョンがリストされます。

```
{
```

```
    "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"
    }
}
```

グローバルエンドポイントの作成

グローバルエンドポイントを設定するには、次の手順を完了します。

1. プライマリリージョンとセカンダリリージョンの両方で、一致するイベントバスとルールがあることを確認します。
2. イベントバスを監視するために、Route 53 ヘルスチェックを作成します。ヘルスチェックの作成を容易にするには、グローバルエンドポイントを作成するときに [New Health Check] (新しいヘルスチェック) を選択します。
3. グローバルエンドポイントを作成します。

Route 53 ヘルスチェックを設定したら、グローバルエンドポイントを作成できます。

コンソールを使用してグローバルエンドポイントを作成するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで、[Global endpoint] (グローバルエンドポイント) を選択します。
3. [Create Endpoint] (エンドポイントの作成) を選択します。
4. エンドポイントの名前と説明を入力します。
5. [Event bus in primary Region] (プライマリリージョンのイベントバス) については、エンドポイントを関連付けるイベントバスを選択します。
6. [Secondary Region] (セカンダリリージョン) については、フェイルオーバー時にイベントを送信するリージョンを選択します。

> **Note**
>
> [Event bus in secondary Region] (セカンダリリージョンのイベントバス) は自動入力され、編集できません。

7. [Route 53 health check for triggering failover and recovery] (フェイルオーバーとリカバリをトリガーする Route 53 ヘルスチェック) については、エンドポイントが監視するヘルスチェックを選択します。まだヘルスチェックがない場合は、新しいヘルスチェックを選択して AWS CloudFormation コンソールを開き、 CloudFormation テンプレートを使用してヘルスチェックを作成します。

> **Note**
>
> データが不足すると、ヘルスチェックは失敗します。イベントを断続的に送信するだけの場合は、より長い を使用するかMinimumEvaluationPeriod、欠落しているデータを「違反」ではなく「欠落」として扱うことを検討してください。

8. (オプション) [Event repolication] (イベントレプリケーション) については、以下を実行します。

   a. [Event replication enabled] (イベントレプリケーションを有効にする) を選択します。

   b. [Execution role] (実行ロール) については、新しい AWS Identity and Access Management ロールを作成するか、既存のロールを使用するか選択します。以下の操作を実行します。

      - [Create a new role for this specific resource] を選択します。オプションで、[Role name] (ロール名) を更新して、新しいロールを作成できます。
      - [Use existing role] (既存のロールを使用する) を選択します。次に、[Execution role] (実行ロール) として、使用する目的のロールを選択します。

9. [作成] を選択します。

API を使用してグローバルエンドポイントを作成するには

EventBridge API を使用してグローバルエンドポイントを作成するには、「Amazon API リファレンスCreateEndpoint」の EventBridge 「」を参照してください。

AWS CloudFormationを使用してグローバルエンドポイントを作成するには

AWS CloudFormation API を使用してグローバルエンドポイントを作成するには、「 ユーザーガイド」のAWS::Events::Endpoints AWS CloudFormation 「」を参照してください。

AWS SDK を使用したグローバルエンドポイントの操作

> **Note**
>
> C++ のサポートは近日公開されます。

AWS SDK を使用してグローバルエンドポイントを操作する場合は、次の点に注意してください。

- 特定の SDK に AWS Common Runtime (CRT) ライブラリをインストールする必要があります。CRT がインストールされていない場合、インストールする必要があるものを示す例外メッセージが表示されます。詳細については、次を参照してください。
  - AWS Common Runtime (CRT) ライブラリ
  - awslabs/aws-crt-java
  - awslabs/aws-crt-nodejs
  - awslabs/aws-crt-python
- グローバルエンドポイントを作成した後、使用する `PutEvents` 呼び出しに `endpointId` と `EventBusName` を追加する必要があります。
- グローバルエンドポイントは、署名バージョン 4A をサポートします。このバージョンの SigV4 では、複数の AWS リージョンについてリクエストに署名できます。これは、いくつかのリージョンのいずれかからデータにアクセスする可能性がある API 操作で便利です。 AWS SDK を使用する場合、認証情報を指定すると、グローバルエンドポイントへのリクエストでは、追加の設定なしで署名バージョン 4A が使用されます。SigV4A の詳細については、AWS 全般リファレンスのAWS API リクエストの署名を参照してください。

  グローバル AWS STS エンドポイント (sts.amazonaws.com) から一時的な認証情報をリクエストすると、 はデフォルトで SigV4A をサポートしていない認証情報 AWS STS を提供します。詳細については、「 ユーザーガイド」の AWS 「 リージョン AWS STS での の管理」を参照してください。 AWS Identity and Access Management

利用できるリージョン

以下のリージョンはグローバルエンドポイントをサポートしています。

- 米国東部 (バージニア北部)
- 米国東部 (オハイオ)
- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- カナダ (中部)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (ミラノ)
- ヨーロッパ (パリ)
- 欧州 (ストックホルム)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (大阪)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)
- 南米 (サンパウロ)

Amazon EventBridge グローバルエンドポイントを使用するためのベストプラクティス

グローバルエンドポイントを設定する場合は、次のベストプラクティスを推奨します。

トピック

- イベントレプリケーションの有効化
- イベントスロットリングの防止
- Amazon Route 53 ヘルスチェックでのサブスクライバーのメトリクスの使用

イベントレプリケーションの有効化

レプリケーションをオンにして、グローバルエンドポイントに割り当てるセカンダリリージョンでイベントを処理することを強くお勧めします。これにより、セカンダリリージョンのアプリケーションが正しく構成されます。また、問題が軽減された後でプライマリリージョンへの自動リカバリを確実にするためにも、レプリケーションを有効にする必要があります。

イベント ID は API 呼び出し間で変更されることがあるため、リージョン間でイベントを関連付けるには、不変の一意の識別子が必要です。消費者も冪等性を念頭に置いて設計する必要があります。そうすれば、イベントを複製したり、アーカイブから再生したりする場合、イベントが両方のリージョンで処理されることによる副作用はありません。

イベントスロットリングの防止

イベントのスロットリングを防止するには、PutEvents ターゲット制限を更新して、リージョン間で一貫性を持たせることをお勧めします。

Amazon Route 53 ヘルスチェックでのサブスクライバーのメトリクスの使用

Amazon Route 53 ヘルスチェックにサブスクライバーのメトリクスを含めないようにします。これらのメトリクスを含めると、プライマリリージョンで他のすべてのサブスクライバーが正常であるにもかかわらず、サブスクライバーが問題に遭遇した場合、パブリッシャーがセカンダリリージョンにフェイルオーバーすることがあります。サブスクライバーの 1 人がプライマリリージョンでイベントを処理できない場合は、レプリケーションをオンにして、セカンダリリージョンのサブスクライバーがイベントを正常に処理できるようにする必要があります。

Route 53 ヘルスチェックを設定するための AWS CloudFormation テンプレート

グローバルエンドポイントを使用する場合は、リージョンのステータスを監視するために Route 53 ヘルスチェックが必要です。次のテンプレートは、Amazon CloudWatch アラームを定義し、それを使用して Route 53 ヘルスチェックを定義します。

トピック

- Route 53 ヘルスチェックを定義するための AWS CloudFormation テンプレート
- CloudWatch アラームテンプレートのプロパティ

- Route 53 ヘルスチェックテンプレートのプロパティ

Route 53 ヘルスチェックを定義するための AWS CloudFormation テンプレート

Route 53 ヘルスチェックを定義するには、次のテンプレートを使用します。

```
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
```

イベント ID は API 呼び出し間で変更されることがあるため、リージョン間でイベントを関連付けるには、不変の一意の識別子が必要です。消費者も冪等性を念頭に置いて設計する必要があります。そうすれば、イベントを複製したり、アーカイブから再生したりする場合、イベントが両方のリージョンで処理されることによる副作用はありません。

CloudWatch アラームテンプレートのプロパティ

> **Note**
>
> すべての **editable** フィールドについて、1 秒あたりのスループットを考慮します。断続的にイベントを送信するだけの場合は、ヘルスチェックを、より長い評価期間を使用するか、欠損データを breaching ではなく missing として扱うように変更することを検討してください。

テンプレートの CloudWatch アラームセクションでは、次のプロパティが使用されます。

メトリクス	説明
AlarmDescription	アラームの説明。 デフォルト: **High Latency in Amazon EventBridge**
MetricName	アラームに関連付けられているメトリクスの名前。これは、メトリクスに基づくアラームの場合に必須です。数式に基づくアラームの場合は、代わりに Metrics を使用し、MetricName を指定することはできません。 デフォルト: IngestionToInvocationStartLatency
Namespace	アラームに関連付けられているメトリクスの名前空間。これは、メトリクスに基づくアラームの場合に必須です。数式に基づくアラームの場合、Namespace を指定することはできず、代わりに Metrics を使用します。 デフォルト: AWS/Events
Statistic	アラームに関連付けられているメトリクスの統計 (パーセンタイル以外)。 デフォルト: Average
Period	統計を適用する時間 (秒)。これは、メトリクスに基づくアラームの場合に必須です。可能な値は 10、30、60、および 60 の倍数です。 デフォルト: **60**
EvaluationPeriods	指定した Threshold の値とデータを比較する時間。アラームをトリガーするために、複数の連続するデータポイントがしきい値を超過することが必要なアラームを設定する場合、この値はその数を指定します。「N 個中 M 個」のアラームを設定する場合、この値は N で、DatapointsToAlarm は M です。 デフォルト: **5**
Threshold	指定された統計と比較する値。

メトリクス	説明
	デフォルト: **30,000**
ComparisonOperator	指定した statistic および threshold の値を比較するときに使用する算術演算。指定した statistic の値が 1 番目のオペランドとして使用されます。 デフォルト: GreaterThanThreshold
TreatMissingData	このアラームが不足しているデータポイントを処理する方法を設定します。 有効な値: breaching 、notBreaching 、ignore、および missing デフォルト: breaching

Route 53 ヘルスチェックテンプレートのプロパティ

> **Note**
>
> すべての **editable** フィールドについて、1 秒あたりのスループットを考慮します。断続的にイベントを送信するだけの場合は、ヘルスチェックを、より長い評価期間を使用するか、欠損データを breaching ではなく missing として扱うように変更することを検討してください。

テンプレートの Route 53 ヘルスチェックセクションでは、次のプロパティが使用されます。

メトリクス	説明
HealthCheckName	ヘルスチェックの名前。 デフォルト: **LatencyFailuresHealthCheck**
InsufficientDataHealthStatus	CloudWatch のメトリクスに関するデータが不十分なためにアラーム状態を判断できないときに、Amazon Route 53 がヘルスチェックに割り当てるステータス。

メトリクス	説明
	有効な値: • Healthy: Route 53 のヘルスチェックが正常であると見なされます。 • Unhealthy : Route 53 のヘルスチェックが不正常であると見なされます。 • LastKnownStatus : Route 53 は、CloudWatch がアラーム状態を判断するのに十分なデータを持っていた最後の時点からのヘルスチェックのステータスを使用します。既知の最新ステータスがない新しいヘルスチェックの場合、ヘルスチェックのデフォルトステータスは "正常" になります。 デフォルト: Unhealthy **Note** このフィールドは、TreatMissingData フィールドへの入力に基づいて更新されます。TreatingMissingData が Missing に設定された場合、LastKnownStatus に更新されます。TreatingMissingData が Breaching に設定された場合、Unhealthy に更新されます。

Amazon EventBridge スキーマ

スキーマは、 に送信されるイベントの構造を定義します EventBridge。 EventBridge は、 AWS サービスによって生成されるすべてのイベントのスキーマを提供します。また、カスタムスキーマを作成またはアップロードしたり、イベントバス上のイベントから直接自動的にスキーマを推測することもできます。イベントのスキーマがあれば、一般的なプログラミング言語のコードバインディングをダウンロードして開発の速度を上げることができます。スキーマのコードバインディングを操作し、EventBridge コンソールから、 API を使用して、または AWS ツールキットを使用して IDE で直接スキーマを管理できます。イベントを使用するサーバーレスアプリケーションを構築するには、 AWS Serverless Application Modelを使用します。

> **Note**
>
> 入力トランスフォーマー機能を使用する場合、元のイベントは、ターゲットに送信される変換されたイベントではなく、スキーマディスカバリによって推測されます。

EventBridge はOpenAPI 3 形式と JSONSchema Draft4 形式の両方をサポートしています。

AWS Toolkit for JetBrains および AWS Toolkit for VS Code では、スキーマを参照または検索し、IDE で直接スキーマのコードバインディングをダウンロードできます。

次のビデオでは、スキーマとスキーマレジストリの概要を示します：Schema Registry の使用

トピック

- スキーマレジストリ API プロパティ値のマスキング
- Amazon EventBridge スキーマの検索
- Amazon EventBridge スキーマレジストリ
- Amazon EventBridge スキーマの作成
- Amazon EventBridge コードバインディング

スキーマレジストリ API プロパティ値のマスキング

スキーマレジストリの作成に使用されるイベントのプロパティ値には、機密性の高い顧客情報が含まれている場合があります。顧客情報を保護するために、値はアスタリスク (*) でマスクされます。こ

れらの値はマスクされているため、 では、次のプロパティまたはその値に明示的に依存するアプリケーションを構築しない EventBridge ことをお勧めします。

- CreateSchema - `requestParameters`本文の`Content`プロパティ
- GetDiscoveredSchema – `requestParameters`本文の `Events`プロパティと`responseElements`本文の `Content`プロパティ
- SearchSchemas – の `keywords`プロパティ `requestParameters`
- UpdateSchema – の `Content`プロパティ `requestParameters`

Amazon EventBridge スキーマの検索

EventBridge には、イベントを生成するすべてのサービスのスキーマが含まれます。 AWS これらのスキーマは EventBridge コンソールで確認できます。または、 API アクション を使用して見つけることができますSearchSchemas。

EventBridge コンソールで AWS サービスのスキーマを検索するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで、[Schemas] (スキーマ) を選択します。
3. [Schemas] (スキーマ) ページで、[AWS event schema registry] (イベントスキーマレジストリ) を選択します。
   <result>

   使用可能なスキーマの最初のページが表示されます。
   </result>
4. スキーマを検索するには、検索 AWS イベントスキーマ で検索語を入力します。

   検索では、使用可能なスキーマの名前と内容の両方の一致が返され、その一致を含むスキーマのバージョンが表示されます。
5. スキーマの名前を選択して、イベントスキーマを開きます。

Amazon EventBridge スキーマレジストリ

スキーマレジストリは、スキーマのコンテナです。スキーマが論理グループに入るようにスキーマを収集して整理します。デフォルトのスキーマレジストリは次のとおりです。

- すべてのスキーマ – AWS イベント、検出、カスタムスキーマレジストリのすべてのスキーマ。
- AWS イベントスキーマレジストリ — 組み込みスキーマ。
- [Discovered schema registry] (検出されたスキーマレジストリ) — スキーマ検出によって検出されたスキーマ。

カスタムレジストリを作成して、作成またはアップロードしたスキーマを整理できます。

カスタムレジストリを作成するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで、[Schemas] (スキーマ)、[Create registry] (レジストリの作成) の順に選択します。
3. [Registry details (レジストリの詳細)] ページで、[Name (名前)] を入力します。
4. (オプション) 新しいレジストリの説明を入力します。
5. [作成] を選択します。

新しいレジストリにカスタムスキーマを作成するには、[Create custom schema]] (カスタムスキーマの作成) を選択します。レジストリにスキーマを追加するには、新しいスキーマを作成するときにそのレジストリを選択します。

API を使用してレジストリを作成するには、`CreateRegistry` を使用します。詳細については、「Amazon EventBridge Schema Registry API Reference」を参照してください。

で EventBridge スキーマレジストリを使用する方法については AWS CloudFormation、「」のEventSchemas 「リソースタイプのリファレンス」を参照してください AWS CloudFormation。

Amazon EventBridge スキーマの作成

スキーマは、OpenAPI 仕様か JSONSchema Draft4 仕様のいずれかの JSON ファイルを使用して作成します。で独自のスキーマを作成またはアップロードするには、テンプレート EventBridge を使用するか、イベント の JSON に基づいてスキーマを生成します。また、イベントバス上のイベントからスキーマを推測することもできます。Schema Registry API を使用して EventBridge スキーマを作成するには、 `CreateSchema` API アクションを使用します。

OpenAPI 3 形式と JSONSchema Draft4 形式を選択する際には、以下の違いを考慮してください。

- JSONSchema 形式は、`$schema`, `additionalItems` など、OpenAPI ではサポートされていない追加のキーワードをサポートしています。
- `type` や `format` など、キーワードの処理方法にも細かな違いがあります。
- OpenAPI は、JSON ドキュメントでの JSONSchema Hyper-Schema ハイパーリンクをサポートしていません。
- OpenAPI 用のツールはビルド時を重視する傾向があり、JSONSchema 用のツールはスキーマ検証用のクライアントツールなど、ランタイム操作を重視する傾向があります。

JSONSchema 形式を使用してクライアント側の検証を実装し、スキーマ EventBridge に準拠するために送信されるイベントを実装することをお勧めします。JSONSchema を使用して、有効な JSON ドキュメントの契約を定義し、関連するイベントを送信する前に JSON スキーマバリデータを使用することができます。

新しいスキーマを作成したら、コードバインディングをダウンロードして、そのスキーマを持つイベントのアプリケーションを作成することができます。

トピック

- テンプレートを使用してスキーマを作成する
- コンソールでスキーマテンプレートを直接編集する
- イベントの JSON からスキーマを作成する
- イベントバス上のイベントからスキーマを作成する

テンプレートを使用してスキーマを作成する

テンプレートからスキーマを作成することも、コンソールで EventBridgeテンプレートを直接編集することもできます。テンプレートを入手するには、コンソールからテンプレートをダウンロードしま

す。スキーマがイベントと一致するようにテンプレートを編集できます。次に、コンソールを使用して新しいテンプレートをアップロードします。

スキーマテンプレートをダウンロードするには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで、[Schema registry] (スキーマレジストリ) を選択します。
3. [Schema template] (スキーマテンプレート) の [Getting started] (開始方法) セクションで、[Download] (ダウンロード) を選択します。

または、次のコード例から JSON テンプレートをコピーすることもできます。

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

スキーマテンプレートをアップロードするには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで、[Schemas] (スキーマ)、[Create schema] (スキーマの作成) の順に選択します。
3. (オプション) スキーマレジストリを選択または作成します。
4. [Schema details (スキーマの詳細)] に、スキーマの名前を入力します。
5. (オプション) スキーマの説明を入力します。
6. [Schema type] (スキーマタイプ) で、[OpenAPI 3.0] または [JSON Schema Draft 4] を選択します。
7. [Create](作成) タブで、スキーマファイルをテキストボックスにドラッグするか、スキーマソースを貼り付けます。
8. [作成] を選択します。

コンソールでスキーマテンプレートを直接編集する

コンソールでスキーマを編集するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで、[Schemas] (スキーマ)、[Create schema] (スキーマの作成) の順に選択します。
3. (オプション) スキーマレジストリを選択または作成します。
4. [Schema details (スキーマの詳細)] に、スキーマの名前を入力します。

5. [Schema type] (スキーマタイプ) で、[OpenAPI 3.0] または [JSON Schema Draft 4] を選択します。
6. (オプション) 作成するスキーマの説明を入力します。
7. [Create] (作成) タブで、[Load template] (テンプレートのロード) を選択します。
8. テキストボックスで、スキーマがイベントと一致するようにテンプレートを編集します。
9. [作成] を選択します。

イベントの JSON からスキーマを作成する

イベントの JSON がある場合は、そのタイプのイベントのスキーマを自動的に作成できます。

イベントの JSON に基づいてスキーマを作成するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで、[Schemas] (スキーマ)、[Create schema] (スキーマの作成) の順に選択します。
3. (オプション) スキーマレジストリを選択または作成します。
4. [Schema details (スキーマの詳細)] に、スキーマの名前を入力します。
5. (オプション) 作成したスキーマの説明を入力します。
6. [Schema type] (スキーマタイプ) で [OpenAPI 3.0] を選択します。

   イベントの JSON からスキーマを作成する場合、JSONSchema は使用できません。
7. [Discover from JSON] (JSON から検出) を選択します。
8. [JSON] の下のテキストボックスで、イベントの JSON ソースを貼り付けるかドラッグします。

   例えば、実行に失敗した場合は、この AWS Step Functions イベントのソースに貼り付けることができます。

```
{
    "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. [スキーマの検出] を選択します。
10. EventBridge は、イベントの OpenAPI スキーマを生成します。例えば、上のステップ関数イベントに対して以下のスキーマが生成されます。

```
{
  "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. スキーマが生成されたら、[Create] (作成) を選択します。

イベントバス上のイベントからスキーマを作成する

EventBridge は、イベントを検出してスキーマを推測できます。スキーマを推測するには、イベントバスでイベント検出をオンにし、クロスアカウントイベント用のスキーマを含む、あらゆる固有のスキーマをスキーマレジストリに追加します。によって EventBridge 検出されたスキーマは、スキーマページの検出されたスキーマレジストリに表示されます。

イベントバス上のイベントの内容が変更されると、 は関連する EventBridge スキーマの新しいバージョン EventBridge を作成します。

> **Note**
>
> イベントバスでイベント検出を有効にすると、コストが発生する場合があります。毎月最初に処理される 500 万件のイベントは無料です。

> **Note**
>
> EventBridge はデフォルトでクロスアカウントイベントからスキーマを推測しますが、`cross-account`プロパティを更新することで無効にできます。詳細については、EventBridge スキーマレジストリ API リファレンスの「Discoverers」を参照してください。

イベントバスでのスキーマ検出を有効にするには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインの [Event Buses] (イベントバス) を選択します。
3. 次のいずれかを行います。
   - [Default event bus] (既定のイベントバス) で検出を有効にするには、[Start discovery] (検出の開始) を選択します。
   - [Custom event bus] (カスタムイベントバス) で検出を有効にするには、カスタムイベントバスのラジオボタンを選択し、[Start discovery] (検出の開始) を選択します。

Amazon EventBridge コードバインディング

Golang、Java、Python、および での開発を高速化するために、イベントスキーマのコードバインディングを生成できます TypeScript。コードバインディングは、 AWS サービスイベント、作成するスキーマ、およびイベントバス上のイベントに基づいて生成するスキーマで使用できます。EventBridge コンソール、Schema EventBridge Registry API、または IDE でツールキットを使用して、スキーマのコードバインディングを生成できます。 AWS

EventBridge スキーマからコードバインディングを生成するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで、[Schemas] (スキーマ) を選択します。
3. スキーマレジストリを参照するか、スキーマを検索して、コードバインディングを使用するスキーマを検索します。
4. スキーマ名を選択します。
5. [Schema details] (スキーマの詳細) ページの [Version] (バージョン) セクションで [Download code bindings] (コードバインディングのダウンロード) を選択します。
6. [Download code bindings] (コードバインディングのダウンロード) ページで、ダウンロードするコードバインディングの言語を選択します。
7. [Download] (ダウンロード) を選択します。

   ダウンロードが開始されるまで数秒かかる場合があります。ダウンロードファイルは、選択した言語のコードバインディングの zip ファイルです。

Amazon EventBridge 関連サービスおよびツール

Amazon EventBridge は、他の AWS のサービスやツールと連携して、イベントを処理したり、ルールのターゲットとしてリソースを呼び出したりします。EventBridge と他の AWS サービスとの統合の詳細については、以下を参照してください。

トピック

- インターフェイス VPC エンドポイントでの Amazon EventBridge の使用
- Amazon EventBridge と AWS X-Ray の統合
- AWS 統合アプリケーションテストキット EventBridge での の使用
- スタックに AWS CloudFormation Amazon EventBridge リソースを含める

インターフェイス VPC エンドポイントでの Amazon EventBridge の使用

Amazon Virtual Private Cloud (Amazon VPC) を使用して AWS リソースをホストする場合、VPC と EventBridge の間のプライベート接続を確立できます。VPC 上のリソースは、この接続を使用して、EventBridge と通信できます。

VPC を使用すると、IP アドレス範囲、サブネット、ルートテーブル、ネットワークゲートウェイなどのネットワーク設定を制御できます。VPC を EventBridge に接続するには、EventBridge のインターフェイス VPC エンドポイントを定義します。このエンドポイントは、インターネットゲートウェイ、ネットワークアドレス変換 (NAT) インスタンス、または VPN 接続を必要とせず、信頼性が高くスケーラブルな EventBridge への接続を提供します。詳細については、Amazon VPC ユーザーガイドの Amazon VPC とはを参照してください。

インターフェイス VPC エンドポイントは AWS PrivateLink を利用しています。これは、Elastic Network Interface とプライベート IP アドレスを使用して AWS のサービス間のプライベート通信を可能にします。詳細については、「AWS PrivateLink および VPC エンドポイント」を参照してください。

プライベートインターフェイス VPC エンドポイントを使用すると、VPC が EventBridge に送信するカスタムイベントがそのエンドポイントを使用します。この場合、EventBridge は設定したルールとターゲットに基づいて、そのイベントを他の AWS サービスに送信します。イベントが他のサービスに送信されると、そのサービスのパブリックエンドポイントまたは VPC エンドポイントのどちらかを通じてイベントを受け取ることができます。例えば、Amazon SQS キューにイベントを送信するルールを作成した場合、Amazon SQS 用のインターフェイス VPC エンドポイントを設定して、パブリックエンドポイントを使用せずに VPC 内のキューからメッセージを受信することができます。

可用性

現在、EventBridge は、次のリージョンで VPC エンドポイントをサポートしています。

- 米国東部 (オハイオ)
- 米国東部 (バージニア北部)
- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- アフリカ (ケープタウン)

- アジアパシフィック (ムンバイ)
- アジアパシフィック (ハイデラバード)
- アジアパシフィック (香港)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (ジャカルタ)
- アジアパシフィック (メルボルン)
- アジアパシフィック (東京)
- アジアパシフィック (ソウル)
- アジアパシフィック (大阪)
- カナダ (中部)
- カナダ西部 (カルガリー)
- 中国 (北京)
- 中国 (寧夏)
- 欧州 (フランクフルト)
- 欧州 (チューリッヒ)
- 欧州 (アイルランド)
- ヨーロッパ (ロンドン)
- 欧州 (ミラノ)
- 欧州 (スペイン)
- 欧州 (パリ)
- 欧州 (ストックホルム)
- 中東 (アラブ首長国連邦)
- 中東 (バーレーン)
- 南米 (サンパウロ)
- イスラエル (テルアビブ)
- AWS GovCloud (米国西部)
- AWS GovCloud (米国東部)

EventBridge 用の VPC エンドポイントの作成

VPC で EventBridge を使用するには、EventBridge 用 のインターフェイス VPC エンドポイントを作成し、サービス名として com.amazonaws.***Region***.events を選択します。詳細については、『Amazon VPC ユーザーガイド』の「インターフェイスエンドポイントの作成」を参照してください。

EventBridge Pipes の詳細

インターフェイス VPC エンドポイントでは EventBridge Pipes のプルサポートは提供されません。EventBridge Pipes を使って VPC 内で以下のソースを使用するには、以下を参照してください。

- Amazon MSK ネットワーク設定
- セルフマネージド Apache Kafka ネットワーク設定
- Amazon MQ ネットワーク設定

Amazon EventBridge と AWS X-Ray の統合

AWS X-Ray を使用すると、EventBridge を通過するイベントをトレースすることができます。EventBridge は元のトレースヘッダーをターゲットに渡し、ターゲットサービスが追跡、分析、およびデバッグできるようにします。

EventBridgeがイベントのトレースヘッダーを渡すことができるのは、そのイベントが、トレースコンテキストを渡した PutEvents リクエストから来ている場合のみです。X-Ray は、サードパーティーパートナーから発生したイベント、スケジュールされたイベント、またはAWS サービスをトレースしないため、これらのイベントソースは X-Ray のサービスマップに表示されません。

X-Ray がトレースヘッダーを検証し、有効でないトレースヘッダーは削除されます。ただし、イベントは引き続き処理されます。

> **Important**
>
> トレースヘッダーは、呼び出しターゲットに配信されるイベントでは使用できません。
>
> - イベントアーカイブがある場合、アーカイブされたイベントでトレースヘッダは使用できません。アーカイブされたイベントを再生する場合、トレースヘッダーは含まれません。
> - デッドレターキュー (DLQ) がある場合、トレースヘッダは DLQ にイベントを送信する SendMessage リクエストに含まれます。ReceiveMessage を使用して DLQ からイベント (メッセージ) を取得する場合、そのイベントに対応するトレースヘッダーは Amazon SQS のメッセージ属性に含まれ、イベントメッセージには含まれません。

EventBridge イベントノードがソースおよびターゲットサービスを接続する方法については、『AWS X-Ray 開発者ガイド』の「X-Ray サービスマップでのソースおよびターゲットの表示」を参照してください。

EventBridge を通じて、次のトレースヘッダー情報を渡すことができます。

- デフォルトの HTTP ヘッダー - X-Ray SDK は、すべての呼び出しターゲットの X-Amzn-Trace-Id HTTP ヘッダーとして自動的にトレースヘッダーを入力します。デフォルトの HTTP ヘッダーの詳細については、『AWS X-Ray 開発者ガイド』の「トレースヘッダー」を参照してください。
- **TraceHeader** システム属性 - TraceHeader は、X-Ray トレースヘッダをターゲットに渡すために EventBridge で予約されている PutEventsRequestEntry 属性です。PutEventsRequestEntry も使用している場合、PutEventsRequestEntry は HTTP トレースヘッダを上書きします。

> **Note**
>
> トレースヘッダーは、PutEventsRequestEntry イベントサイズに含まれません。詳細については、「Amazon EventBridge PutEvents イベントエントリサイズの計算」を参照してください。

次のビデオは、X-Ray と EventBridge を一緒に使用する方法を示しています：AWS X-Rayトレースの使用

AWS 統合アプリケーションテストキット EventBridge での の使用

Lambda EventBridgeや Step Functions などのサーバーレスサービスで構成されるアプリケーションを作成する場合、アーキテクチャコンポーネントの多くをデスクトップにデプロイすることはできませんが、クラウドにのみ存在します AWS 。ローカルにデプロイされたアプリケーションを操作するのとは対照的に、これらのタイプのアプリケーションは、自動テストを実行するためのクラウドベースの戦略からメリットを得られます。 AWS 統合アプリケーションテストキット (AWS IATK) は、これらの戦略の一部をアプリケーションに実装するのに役立ちます。

AWS IATK は、クラウドベースのアプリケーションの自動テストの作成に役立つソフトウェアライブラリです。

EventBridge AWS IATK との統合

AWS IATK で EventBridge イベントとイベントバスを使用して、次のような自動テストを実装できます。

テストハーネスの実装

イベント駆動型アーキテクチャの統合テストを作成するには、アプリケーションをサブシステムに分割して論理的な境界を設定します。サブシステムのテストに役立つ手法の 1 つは、テストハーネス (サブシステムのテスト専用のリソース) を作成することです。

例えば、統合テストでは、入力テストイベントを渡すことでサブシステムプロセスを開始できます。 AWS IATK は、出力イベント EventBridge をリッスンするテストハーネスを作成できます。(内部では、ハーネスは出力イベントを Amazon SQS に転送する EventBridge ルールで構成さ

れます) 。次に、統合テストはテストハーネスにクエリを実行して出力を調べ、テストの合否を判断します。

モックイベントの生成

AWS IATK は、スキーマ EventBridge レジストリに保存されているスキーマからモックイベントを生成する機能を提供します。これにより、モックイベントを生成し、生成したイベントを使用して任意のコンシューマー (Lambda 関数や Step Functions ステートマシンなど) を呼び出すことができます。

詳細については、「」のAWS 「統合アプリケーションテストキットの概要」を参照してください GitHub。

スタックに AWS CloudFormation Amazon EventBridge リソースを含める

AWS CloudFormation では、Infrastructure as Code. CloudFormation を使用して、アカウントとリージョン間で AWS リソースを一元的かつ反復可能な方法で設定および管理できます。これにより、プロビジョニングおよび管理したいリソースを定義するテンプレート を作成できます。これらのリソースには、イベントバスやルール、パイプ、スキーマ、スケジュールなどの EventBridge アーティファクトを含めることができます。これらのリソースを使用して、 を通じてプロビジョニングおよび管理するテクノロジースタックに機能を含め EventBridgeます CloudFormation。

で利用可能な Amazon EventBridge リソース AWS CloudFormation

EventBridge は、次のリソース名前空間の CloudFormation テンプレートで使用するリソースを提供します。

- AWS::Events

  テンプレートの例は以下のとおりです。
  - の API 送信先を作成する PagerDuty
  - Slack の API 送信先を作成する
  - ApiKey 認証パラメータを使用して接続を作成する
  - OAuth 認証パラメータを使用して接続を作成する
  - イベントレプリケーションを使用してグローバルエンドポイントを作成する
  - 複数のプリンシパルとアクションを使用してポリシーを拒否する

  - カスタムイベントバスを使用して組織にアクセス許可を付与する
  - リージョン間ルールを作成する
  - ターゲットにデッドレターキューを含めるルールを作成する
  - Lambda 関数を定期的に呼び出す
  - イベントに応じて Lambda 関数を呼び出す
  - ログエントリに応じてトピックを通知する
- AWS::Eventスキーマ
- AWS::Pipes

  テンプレートの例は以下のとおりです。

  - イベントフィルターを使用してパイプを作成する
- AWS::Scheduler

AWS CloudFormation テンプレートの Amazon EventBridge リソース定義の生成

テンプレートの開発をすぐに開始できるように CloudFormation 、 EventBridge コンソールを使用すると、アカウント内の既存のイベントバス、ルール、パイプから CloudFormation テンプレートを作成できます。

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

デフォルトのイベントバス AWS CloudFormation の管理

はデフォルトのイベントバスを自動的にアカウントに EventBridge プロビジョニングするため、スタックに含める CloudFormation リソースの場合と同様に、 CloudFormation テンプレートを使用して作成することはできません。 CloudFormation スタックにデフォルトのイベントバスを含めるには、まずスタックにインポートする必要があります。デフォルトのイベントバスをスタックにインポートしたら、必要に応じてイベントバスのプロパティを更新できます。

詳細については、「???」を参照してください。

を使用した AWS CloudFormation スタックイベントの管理 EventBridge

CloudFormation スタックに EventBridge リソースを含めるだけでなく、 EventBridge を使用して CloudFormation スタック自体によって生成されたイベントを管理できます。 は、スタックに対して作成、更新、削除、またはドリフト検出オペレーションが実行される EventBridge たびにイベントを CloudFormation に送信します。 CloudFormation また、 は、スタックセットとスタックセットインスタンスのステータス変更 EventBridge のためにイベントを に送信します。 EventBridge ルールを使用して、定義したターゲットにイベントをルーティングできます。

詳細については、「 ユーザーガイド」の「 を使用した CloudFormation イベントの管理 EventBridgeAWS CloudFormation 」を参照してください。

Amazon EventBridge チュートリアル

EventBridge は、多数の AWS のサービスや SaaS パートナーと統合されます。これらのチュートリアルは、EventBridge の基本と、サーバーレスアーキテクチャの一部となる方法を理解するのに役立つように設計されています。

チュートリアル:

- Amazon EventBridge 開始方法のチュートリアル
- 他の AWS のサービスとの統合に関するAmazon EventBridge のチュートリアル
- SaaS プロバイダとの統合に関する Amazon EventBridge のチュートリアル

Amazon EventBridge 開始方法のチュートリアル

EventBridge の機能とその使用方法のチュートリアルを説明します。

チュートリアル:

- Amazon EventBridge イベントのアーカイブと再生
- Amazon EventBridge サンプルアプリケーションを作成する
- チュートリアル: Eventbridge スキーマレジストリを使用してイベントのコードバインディングをダウンロードする
- チュートリアル: Eventbridge がイベントターゲットに渡すものを Input Transformer を使用してカスタマイズする

Amazon EventBridge イベントのアーカイブと再生

EventBridge を使用して、ルールを使用した特定のAWS Lambda関数にイベントをルーティングできます。

このチュートリアルでは、Lambda コンソールを使用して、EventBridge ルールのターゲットとして使用する関数を作成します。次に、アーカイブと EventBridge コンソールを使用して、テストイベントをアーカイブするルールを作成します。そのアーカイブにイベントがある場合は、それらを再生します。

ステップ

- ステップ 1: Lambda 関数を作成する
- ステップ 2: アーカイブの作成
- ステップ 3: ルールを作成する
- ステップ 4: テストイベントの送信
- ステップ 5 : イベントの再生
- ステップ 6: リソースをクリーンアップする

ステップ 1: Lambda 関数を作成する

始めに、Lambda 関数を作成してイベントのログを記録します。

Lambda 関数を作成するには

1. AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. 関数の作成 を選択します。
3. Author from scratch (製作者を最初から) を選択します。
4. Lambda 関数の名前と説明を入力します。例えば、関数名を `LogScheduledEvent` とします。
5. 残りのオプションはデフォルトのまま、[Create function] (関数の作成) を選択します。
6. 関数ページの [Code] (コード) タブで、index.js をダブルクリックします。
7. 既存の JavaScript コードを以下のコードに置き換えます。

```
'use strict';

exports.handler = (event, context, callback) => {
    console.log('LogScheduledEvent');
```

```
    console.log('Received event:', JSON.stringify(event, null, 2));
    callback(null, 'Finished');
};
```

8. [Deploy] (デプロイ) をクリックします。

ステップ 2: アーカイブの作成

次に、すべてのテストイベントを保持するアーカイブを作成します。

ステップ 5: アーカイブを作成するには

1. アーカイブにテストイベントが保存されると、それらを再生できます。
2. ナビゲーションペインで [Archives (アーカイブ)] を選択します。
3. [Create archive] (アーカイブの作成) を選択します。
4. アーカイブの名前と説明を入力します。たとえば、アーカイブ `ArchiveTest` に名前を付けます。
5. 残りのオプションはデフォルトのまま、[Next] (次へ) を選択します。
6. [Create archive] (アーカイブの作成) を選択します。

ステップ 3: ルールを作成する

イベントバスに送信されるイベントをアーカイブするルールを作成します。

ルールを作成するには:

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで [Rules] (ルール) を選択します。
3. [Create rule] (ルールの作成) を選択します。
4. ルールの名前と説明を入力します。たとえば、ルール `ARTestRule` に名前を付けます。

   ルールには、同じリージョン内および同じイベントバス上の別のルールと同じ名前を付けることはできません。
5. [Event bus] (イベントバス) では、このルールに関連付けるイベントバスを選択します。このルールをアカウントからのイベントと一致させるには、[default] (デフォルト) を選択します。アカウントの AWS サービスがイベントを発行すると、常にアカウントのデフォルトのイベントバスに移動します。

6. [Rule type] (ルールタイプ) では、[Rule with an event pattern] (イベントパターンを持つルール) を選択します。
7. [Next] (次へ) をクリックします。
8. [Event source] (イベントソース) では、[Other] (その他) を選択します。
9. [Event pattern] (イベントパターン) では、次のように入力します。

```
{
  "detail-type": [
    "customerCreated"
  ]
}
```

10. [Next] (次へ) をクリックします。
11. [Target types] (ターゲットタイプ) では、AWS[services] (サービス) を選択します。
12. ターゲットの選択では、ドロップダウンリストから [Lambda function] (Lambda 関数) を選択します。
13. [Function] (関数) で、[Step 1: Create a Lambda function] (ステップ 1 : Lambda 関数を作成する) セクションで作成した Lambda 関数を選択します。この例では、`LogScheduledEvent` を選択します。
14. [Next] (次へ) をクリックします。
15. [Next] (次へ) をクリックします。
16. ルールの詳細を確認し、[Create rule] (ルールの作成) を選択します。

ステップ 4: テストイベントの送信

アーカイブとルールを設定したので、アーカイブが正しく動作していることを確認するためにテストイベントを送信します。

> **Note**
>
> イベントがアーカイブに到達するまでに、時間がかかることがあります。

テストイベントを送信するには (コンソール)

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインの [Event Buses] (イベントバス) を選択します。

3. [Default event bus] (デフォルトのイベントバス) タイルで、[Actions] (アクション)、[Send events] (イベントの送信) を選択します。
4. イベントソースを入力します。例えば、`TestEvent` です。
5. [Detail type] (詳細タイプ) を使用する場合、`customerCreated` と入力します。
6. [Event detail] (イベントの詳細) を使用する場合、`{}` と入力します。
7. [Send] (送信) を選択します。

ステップ 5：イベントの再生

テストイベントがアーカイブに保存されたら、それらを再生できます。

アーカイブされたイベントを再生するには (コンソール)

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインの [Replays (再生)] を選択します。
3. [Start new replay (新規再生を開始)] を選択します。
4. 再生の名前と説明を入力します。たとえば、再生 `ReplayTest` に名前を付けます。
5. [Source] (ソース) で、[Step 2: Create archive] (ステップ 2：アーカイブの作成) セクションで作成したアーカイブを選択します。
6. [Replay time frame] (再生時間フレーム) で、次の作業を行います。
   a. [Start time] (開始時間) で、テストイベントを送信した日付と送信する前の時刻を選択します。例えば、`2021/08/11` と `08:00:00` です。
   b. [End time] (終了時間) で、現在の日付と時刻を選択します。例えば、`2021/08/11` と `09:15:00` です。
7. [Start Replay] (再生を開始) を選択します。

ステップ 6: リソースをクリーンアップする

このチュートリアル用に作成したリソースは、保存を希望しない限り、すぐに削除できます。使用しなくなった AWS リソースを削除することで、AWS アカウントに請求される料金が発生しないようにできます。

Lambda 関数を削除するには

1. Lambda コンソールの [Functions] (関数) ページを開きます。

2. 作成した関数を選択します。
3. [Actions] (アクション) で、[Delete] (削除) を選択します。
4. [削除] を選択します。

EventBridge アーカイブを削除するには

1. Eventbridge コンソールの [Archives] (アーカイブ) ページを開きます。
2. 作成したアーカイブを選択します。
3. [Delete] (削除) をクリックします。
4. アーカイブ名を入力し、[Delete] (削除) を選択します。

EventBridge ルールを削除するには

1. Eventbridge コンソールの [Rules] (ルール) ページを開きます。
2. 作成したルールを選択します。
3. [Delete] (削除) をクリックします。
4. [Delete] (削除) を選択します。

Amazon EventBridge サンプルアプリケーションを作成する

EventBridge を使用して、ルールを使用した特定の Lambda 関数にイベントをルーティングできます。

このチュートリアルでは、AWS CLI、Node.js、および GitHub レポ 内のコードを使用して、以下の項目を作成します。

- 銀行 ATM トランザクションのイベントを生成する AWS Lambda 関数。
- EventBridge ルールのターゲットとして使用する 3 つの Lambda 関数。
- また、イベントパターンに基づいて作成したイベントを、正しいダウンストリーム関数にルーティングするルール。

この例では、EventBridge ルールを定義する AWS SAM テンプレートを使用します。EventBridge と AWS SAM テンプレートの使用の詳細については、??? を参照してください。

レポでは、atmProducer サブディレクトリに `handler.js` が含まれ、イベントを生成する ATM サービスを表しています。このコードは Node.js で記述された Lambda ハンドラーで、JavaScript コードのこの行を使用する AWSSDK からイベントを EventBridge に発行します。

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

このディレクトリには `events.js` を含み、Entries 配列にいくつかのテストトランザクションをリストしています。JavaScript では、1 つのイベントは次のように定義されています。

```
{
  // 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
  })
}
```

イベントの詳細セクションでは、トランザクション属性を指定します。これには ATM の口ケーション、金額、パートナー銀行、取引の結果が含まれます。

atmConsumer サブディレクトリの `handler.js` ファイルには、3 つの関数が含まれています。

```
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))
}
```

各関数はトランザクションイベントを受信し、`console.log` ステートメントから Amazon CloudWatch Logs に記録されます。コンシューマー関数はプロデューサーとは独立して動作し、イベントのソースを認識しません。

ルーティングロジックは、アプリケーションの AWS SAM テンプレートによってデプロイされる EventBridge ルールに含まれています。ルールは、イベントの受信ストリームを評価し、一致するイベントをターゲットの Lambda 関数にルーティングします。

ルールでは、一致するイベントと同じ構造の JSON オブジェクトであるイベントパターンを使用します。ルールの 1 つのイベントパターンを次に示します。

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

```
    }]
  }
}
```

ステップ

- 前提条件
- ステップ 1: アプリケーションを作成する
- ステップ 2: アプリケーションを実行する
- ステップ 3: ログを確認し、アプリケーションが動作することを検証する
- ステップ 4: リソースをクリーンアップする

前提条件

このチュートリアルを完了するには、以下のリソースが必要です。

- AWS アカウント。まだ持っていない場合は、AWS アカウントを作成してください。
- AWS CLI インストール済み。AWS CLI をインストールするには、「AWS CLI バージョン 2 のインストール、更新、およびアンインストール」を参照してください。
- Node.js 12.x インストール済み。Node.js をインストールするには、ダウンロードを参照してください。

ステップ 1: アプリケーションを作成する

サンプルアプリケーションをセットアップするには、AWS CLI と Git を使用して、必要な AWS リソースを作成します。

アプリケーションを作成するには

1. AWSにサインインします。
2. ローカルマシンで、Git をインストールし、AWS Serverless Application Model CLI をインストールします。
3. 新しいディレクトリを作成し、ターミナルのそのディレクトリに移動します。
4. コマンドラインで `git clone https://github.com/aws-samples/amazon-eventbridge-producer-consumer-example` と入力します。
5. コマンドラインから、以下のコマンドを実行します。

```
cd ./amazon-eventbridge-producer-consumer-example
sam deploy --guided
```

6. ターミナルで次のように実行します。

   a. **Stack Name** にスタックの名前を入力します。たとえば、スタック Test に名前を付けます。

   b. **AWS Region** にはリージョンと入力します。例えば、us-west-2 です。

   c. [**Confirm changes before deploy**] に「Y」と入力します。

   d. [**Allow SAM CLI IAM role creation**] に「Y」と入力します。

   e. [**Save arguments to configuration file**] に「Y」と入力します。

   f. [**SAM configuration file**] に「samconfig.toml」と入力します。

   g. [**SAM configuration environment**] に「default」と入力します。

ステップ 2: アプリケーションを実行する

これでリソースが設定されたので、コンソールを使用して関数をテストします。

アプリケーションを実行するには

1. AWS SAM アプリケーションをデプロイしたのと同じリージョンで Lambda コンソールを開きます。
2. プレフィックス [atm-demo] が付いた 4 つの Lambda 関数があります。[atmProducerFn] 関数を選択してから、[Actions] (アクション)、[Test] (テスト) を選択します。
3. Name (名前) には、Test と入力します。
4. [Test] (テスト) を選択します。

ステップ 3: ログを確認し、アプリケーションが動作することを検証する

これでアプリケーションが実行されたので、コンソールを使用して CloudWatch Logs を確認します。

ログを確認するには

1. AWS SAM アプリケーションを実行したのと同じリージョンで CloudWatch コンソールを開きます。

2. [Logs] を選択し、ロググループを選択します。
3. [atmConsumerCase1] を含むロググループを選択します。ATM によって承認された 2 つの取引を表す 2 つのストリームが表示されます。アウトプットを表示するログストリームを選択します。
4. ロググループのリストに戻り、[atmConsumerCase2] を含むロググループを選択します。ニューヨーク のロケーションフィルターに一致する、2 つの取引を表す 2 つのストリームが表示されます。
5. ロググループのリストに戻り、atmConsumerCase3 を含むロググループを選択します。ストリームを開いて、拒否された取引を確認します。

ステップ 4: リソースをクリーンアップする

このチュートリアル用に作成したリソースは、保存を希望しない限り、すぐに削除できます。使用しなくなった AWS リソースを削除することで、AWS アカウントに請求される料金が発生しないようにできます。

EventBridge ルールを削除するには

1. Eventbridge コンソールの [Rules] (ルール) ページを開きます。
2. 作成したルールを選択します。
3. [Delete] (削除) をクリックします。
4. [Delete] (削除) をクリックします。

Lambda 関数を削除するには

1. Lambda コンソールの [Functions] (関数) ページを開きます。
2. 作成した関数を選択します。
3. [Actions] (アクション) で、[Delete] (削除) を選択します。
4. [Delete] (削除) をクリックします。

CloudWatch Logs ロググループを削除するには

1. CloudWatch コンソールを開きます。
2. [Logs] (ログ)、[Log groups] (ロググループ) を選択します。
3. このチュートリアルで作成したロググループを選択します。

4. [アクション]、[ロググループの削除] の順にクリックします。
5. [Delete] (削除) をクリックします。

チュートリアル: Eventbridge スキーマレジストリを使用してイベントのコードバインディングをダウンロードする

Golang、Java、Python、および TypeScript での開発を高速化するために、イベントスキーマのコードバインディングを生成できます。既存の AWS のサービス、作成したスキーマ、およびイベントバスのイベントに基づいて生成するスキーマのコードバインディングを取得できます。スキーマのコードバインディングを生成するには、次のいずれかを使用します。

- EventBridge コンソール
- EventBridge スキーマレジストリ API
- IDE と AWS ツールキット

このチュートリアルでは、AWS のサービスのイベントの Eventbridge スキーマからコードバインディングを生成し、ダウンロードします。

Eventbridge スキーマからコードバインディングを生成するには

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで、[Schemas] (スキーマ) を選択します。
3. [AWS event schema registry] (AWS イベントスキーマレジストリ) タブを選択します。
4. スキーマレジストリを参照するか、スキーマを検索して、コードバインディングを使用する AWS のサービスのスキーマを見つけます。
5. スキーマ名を選択します。
6. [Schema details] (スキーマの詳細) ページの [Version] (バージョン) セクションで [Download code bindings] (コードバインディングのダウンロード) を選択します。
7. [Download code bindings] (コードバインディングのダウンロード) ページで、ダウンロードするコードバインディングの言語を選択します。
8. [Download] (ダウンロード) を選択します。

   ダウンロードが開始されるまで数秒かかる場合があります。ダウンロードファイルは、選択した言語のコードバインディングの zip ファイルです。
9. ダウンロードしたファイルを解凍し、プロジェクトに追加します。

   ダウンロードしたパッケージには、さまざまなフレームワークでパッケージの依存関係を設定する方法を説明している README ファイルが含まれています。

これらのコードバインディングを独自のコードで使用すると、この Eventbridge イベントを使用してアプリケーションをすばやく構築できます。

チュートリアル: Eventbridge がイベントターゲットに渡すものを Input Transformer を使用してカスタマイズする

EventBridge の Input Transformer を使用すると、ルールのターゲットに送信する前に、イベントからのテキストをカスタマイズすることができます。

そのためには、イベントからの JSON パスを定義し、その出力をさまざまな変数に割り当てます。その後、こうした変数は、入力テンプレートで使用できるようになります。文字 < および > は工スケープできません。詳細については、「Amazon EventBridge 入力変換」を参照してください。

> **Note**
>
> 指定した変数と一致する JSON パスがイベントに存在しない場合、その変数は作成されず、出力にも表示されません。

このチュートリアルでは、`detail-type: "customerCreated"` のイベントに一致するルールを作成します。インプットトランフォーマーは、type 変数をこのイベントからの $.detail-type の JSON パスにマッピングします。次に、EventBridge は変数を「This event was <type>.」入力テンプレートに入力します。この結果は、次の Amazon SNS メッセージのようになります。

```
"This event was of customerCreated type."
```

ステップ:

- ステップ 1: Amazon SNS トピックを作成する
- ステップ 2: Amazon SNS サブスクリプションを作成する
- ステップ 3: ルールを作成する
- ステップ 4: テストイベントの送信
- ステップ 5: 成功を確認する
- ステップ 6: リソースをクリーンアップする

ステップ 1: Amazon SNS トピックを作成する

EventBridge からイベントを受信するトピックを作成します。

トピックを作成するには

1. https://console.aws.amazon.com/sns/v3/home で Amazon SNS コンソールを開きます。
2. ナビゲーションペインで、[Topics] (トピック) を選択します。
3. [Create topic] (トピックの作成) を選択します。
4. [Type] (タイプ) で、[Standard] (標準) を選択します。
5. テーブルの名前として **`eventbridge-IT-test`** を入力します。
6. [Create topic] (トピックの作成) を選択します。

ステップ 2: Amazon SNS サブスクリプションを作成する

サブスクリプションを作成して、変換された情報を含む E メールを送付します。

サブスクリプションを作成するには

1. Amazon SNS コンソール (https://console.aws.amazon.com/sns/v3/home) を開きます。
2. ナビゲーションペインで [Subscriptions] (サブスクリプション) を選択します。
3. [Create subscription] を選択します。
4. [トピック ARN] で、ステップ 1 で作成したトピックを選択します。このチュートリアルでは、eventbridge-IT-test を選択します。
5. [Protocol] (プロトコル) で [Email] (E メール) を選択します。
6. [エンドポイント] に E メールアドレスを入力します。
7. [Create subscription] (サブスクリプションの作成) を選択します。
8. AWS 通知から受信した E メールで、[サブスクリプションを確認] を選択してサブスクリプションを確認します。

ステップ 3: ルールを作成する

Input Transformer を使用して、ターゲットに送信されるインスタンス状態情報をカスタマイズするルールを作成します。

ルールを作成するには:

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで [Rules] (ルール) を選択します。

3. [Create rule] (ルールの作成) を選択します。
4. ルールの名前と説明を入力します。例えば、ルール ARTestRule に名前を付けます
5. [Event bus] (イベントバス) では、このルールに関連付けるイベントバスを選択します。このルールをアカウントからのイベントと一致させるには、[default] (デフォルト) を選択します。アカウントの AWS サービスがイベントを発行すると、常にアカウントのデフォルトのイベントバスに移動します。
6. [Rule type] (ルールタイプ) では、[Rule with an event pattern] (イベントパターンを持つルール) を選択します。
7. [Next] (次へ) をクリックします。
8. [Event source] (イベントソース) では、[Other] (その他) を選択します。
9. [Event pattern] (イベントパターン) では、次のように入力します。

```
{
  "detail-type": [
    "customerCreated"
  ]
}
```

10. [Next] (次へ) をクリックします。
11. [Target types] (ターゲットタイプ) では、AWS[services] (サービス) を選択します。
12. ターゲットの選択では、ドロップダウンリストから [SNS topic] (SNS トピック) を選択します。
13. トピックでは、ステップ 1 で作成した Amazon SNS トピックを選択します。このチュートリアルでは、eventbridge-IT-test を選択します。
14. [Additional settings] (追加設定) では、以下を実行します。

    a. ターゲット入力の設定では、ドロップダウンリストから[Input transformer] (インプットトランスフォーマー) を選択します。

    b. [Configure input transformer] (インプットトランスフォーマーの設定) を選択します。

    c. [Sample events] (イベント例) では、以下を入力します。

```
{
  "detail-type": "customerCreated"
}
```

    d. [Target input transformer] (ターゲットインプットトランスフォーマー) では、以下を実行します。

i. [Input Path] (入力パス) では、以下を入力します。

```
{"detail-type":"$.detail-type"}
```

ii. [Input Template] (入力テンプレート) では、以下を入力します。

```
"This event was of <detail-type> type."
```

e. [Confirm] (確認) を選択します。

15. [Next] (次へ) をクリックします。
16. [Next] (次へ) をクリックします。
17. ルールの詳細を確認し、[Create rule] (ルールの作成) を選択します。

ステップ 4: テストイベントの送信

SNS トピックとルールを設定したので、ルールが正しく動作していることを確認するためにテストイベントを送信します。

テストイベントを送信するには (コンソール)

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインの [Event Buses] (イベントバス) を選択します。
3. [Default event bus] (デフォルトのイベントバス) タイルで、[Actions] (アクション)、[Send events] (イベントの送信) を選択します。
4. イベントソースを入力します。例えば、`TestEvent` です。
5. [Detail type] (詳細タイプ) を使用する場合、`customerCreated` と入力します。
6. [Event detail] (イベントの詳細) を使用する場合、`{}` と入力します。
7. [Send] (送信) を選択します。

ステップ 5: 成功を確認する

AWS 通知から予想される出力と一致する E メールが届いたら、チュートリアルは正常に終了しています。

ステップ 6: リソースをクリーンアップする

このチュートリアル用に作成したリソースは、保存を希望しない限り、すぐに削除できます。使用しなくなった AWS リソースを削除することで、AWS アカウントに請求される料金が発生しないようにできます。

SNS トピックを削除するには

1. SNS コンソールの [トピック] ページを開きます。
2. 先ほど作成したトピックを選択します。
3. [Delete] (削除) をクリックします。
4. **`delete me`** と入力します。
5. [Delete] (削除) をクリックします。

SNS サブスクリプションを削除するには

1. SNS コンソールの [サブスクリプションページ] を開きます。
2. 作成したサブスクリプションを選択します。
3. [Delete] (削除) をクリックします。
4. [Delete] (削除) をクリックします。

EventBridge ルールを削除するには

1. Eventbridge コンソールの [Rules] (ルール) ページを開きます。
2. 作成したルールを選択します。
3. [Delete] (削除) をクリックします。
4. [Delete] (削除) を選択します。

他の AWS のサービスとの統合に関するAmazon EventBridge のチュートリアル

Amazon EventBridge は、他の AWS サービスと連携して、イベントを処理したり、ルールのターゲットとして AWS リソースを呼び出したりします。以下のチュートリアルでは、EventBridge を他の AWS のサービスと統合する方法を示します。

チュートリアル:

- チュートリアル: Eventbridge を使用して Auto Scaling グループの状態をログに記録する
- チュートリアル: を使用した AWS API コールのログ記録 EventBridge
- チュートリアル: を使用して Amazon EC2 インスタンスの状態をログに記録する EventBridge
- チュートリアル: Eventbridge を使用して Amazon S3 オブジェクトレベル操作のログを記録する
- チュートリアル: EventBridge とaws.eventsスキーマを使用して Amazon Kinesis ストリームにイベントを送信する
- チュートリアル: Eventbridge を使用した、自動化された Amazon EBS スナップショットのスケジュール
- チュートリアル: Amazon S3 オブジェクトが作成されたときに通知を送信する
- チュートリアル: EventBridge を使用した AWS Lambda 関数のスケジュール

チュートリアル: Eventbridge を使用して Auto Scaling グループの状態をログに記録する

Auto Scaling グループが Amazon EC2 インスタンスを起動または終了するたびにイベントをログに記録し、そのイベントが成功したかどうかをログに記録する AWS Lambda 関数を実行できます。

Amazon EC2 Auto Scaling イベントを使用するその他のシナリオについては、Amazon EC2 Auto Scaling ユーザーガイドの「Auto Scaling イベントの処理に EventBridge を使用する」を参照してください。

このチュートリアルでは、Lambda 関数を作成し、Amazon EC2 Auto Scaling グループがインスタンスを起動または終了したときにその関数を呼び出す EventBridge コンソールのルールを作成します。

ステップ:

- 前提条件
- ステップ 1: Lambda 関数を作成する
- ステップ 2: ルールを作成する
- ステップ 3: ルールをテストする
- ステップ 4: 成功を確認する
- ステップ 5: リソースをクリーンアップする

前提条件

このチュートリアルを完了するには、以下のリソースが必要です。

- Auto Scaling グループ。グループ作成の詳細については、Amazon EC2 Auto Scaling ユーザーガイドの起動設定を使用した Auto Scaling グループの作成を参照してください。

ステップ 1: Lambda 関数を作成する

Lambda 関数を作成して、Auto Scaling グループのスケールアウトおよびスケールインイベントのログを記録します。

Lambda 関数を作成するには

1. AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。

2. 関数の作成 を選択します。
3. Author from scratch (製作者を最初から) を選択します。
4. Lambda 関数の名前を入力します。例えば、関数名を LogAutoScalingEvent とします。
5. 残りのオプションはデフォルトのまま、[Create function] (関数の作成) を選択します。
6. 関数ページの [Code] (コード) タブで、index.js をダブルクリックします。
7. 既存のコードを以下のコードに置き換えます。

```
'use strict';

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

8. [Deploy] (デプロイ) をクリックします。

ステップ 2: ルールを作成する

ステップ 1 で作成した Lambda 関数を実行するルールを作成します。ルールは、Auto Scaling グループがインスタンスを起動または停止したときに実行されます。

ルールを作成するには:

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで [Rules] (ルール) を選択します。
3. [Create rule] (ルールの作成) を選択します。
4. ルールの名前と説明を入力します。例えば、ルール TestRule に名前を付けます
5. [Event bus] (イベントバス) では、このルールに関連付けるイベントバスを選択します。このルールをアカウントからのイベントと一致させるには、[default] (デフォルト) を選択します。アカウントの AWS サービスがイベントを発行すると、常にアカウントのデフォルトのイベントバスに移動します。
6. [Rule type] (ルールタイプ) では、[Rule with an event pattern] (イベントパターンを持つルール) を選択します。
7. [Next] (次へ) をクリックします。
8. [Event source] (イベントソース) では、AWS[services] (サービス) を選択します。

9. [Event pattern] (イベントパターン) の場合は、次のいずれかを実行します。

    a. [Event source] (イベントソース) では、ドロップダウンリストから Auto Scaling (オートスケーリング) を選択します。

    b. [Event type] (イベントタイプ) では、ドロップダウンリストから[Instance Launch and Terminate] (インスタンスの起動と削除) を選択します。

    c. [Any instance event] (任意のインスタンスイベント) と [Any group name] (任意のグループ名) を選択します。

10. [Next] (次へ) をクリックします。

11. [Target types] (ターゲットタイプ) では、AWS[services] (サービス) を選択します。

12. ターゲットの選択では、ドロップダウンリストから [Lambda function] (Lambda 関数) を選択します。

13. [Function] (関数) で、[Step 1: Create a Lambda function] (ステップ 1 : Lambda 関数を作成する) セクションで作成した Lambda 関数を選択します。この例では、`LogAutoScalingEvent` を選択します。

14. [Next] (次へ) をクリックします。

15. [Next] (次へ) をクリックします。

16. ルールの詳細を確認し、[Create rule] (ルールの作成) を選択します。

ステップ 3: ルールをテストする

ルールをテストするには、インスタンスを起動するように Auto Scaling グループを手動でスケーリングします。スケールアウトイベントが発生するまで数分まってから、Lambda 関数が呼び出されたことを確認します。

Auto Scaling グループを使用してルールをテストするには

1. Auto Scaling グループのサイズを増やすには、以下の操作を実行します。

    a. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。

    b. ナビゲーションペインで、[Auto Scaling]、[Auto Scaling グループ] の順に選択します。

    c. Auto Scaling グループのチェックボックスを選択します。

    d. [Details (詳細)] タブで、[Edit (編集)] を選択します。[Desired] で、希望する容量を 1 つ増やします。たとえば、現在の値が 2 の場合は 3 と入力します。希望するキャパシティーは、グループの最大サイズと同じかそれ以下である必要があります。[Desired] の新しい値が、

[Max] よりも大きい場合、[Max] を更新する必要があります。完了したら、[Save] を選択します。

2. Lambda 関数からの出力を表示するには、以下の操作を実行します。

   a. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。

   b. ナビゲーションペインで [ログ] を選択します。

   c. Lambda 関数 (`/aws/lambda/function-name`) のロググループの名前を選択します。

   d. 起動したインスタンスの関数によって提供されるデータを表示するログのストリーム名を選択します。

3. (オプション) 終了すると、Auto Scaling グループが以前のサイズに戻るように、必要な容量を減らすことができます。

ステップ 4: 成功を確認する

CloudWatch ログに Lambda イベントがある場合、このチュートリアルは正常に完了しています。イベントが CloudWatch ログにない場合は、ルールが正常に作成されたことを確認してトラブルシューティングを開始し、ルールが正しく見える場合は、Lambda 関数のコードが正しいことを確認します。

ステップ 5: リソースをクリーンアップする

このチュートリアル用に作成したリソースは、保存を希望しない限り、すぐに削除できます。使用しなくなった AWS リソースを削除することで、AWS アカウントに請求される料金が発生しないようにできます。

EventBridge ルールを削除するには

1. Eventbridge コンソールの [Rules] (ルール) ページを開きます。
2. 作成したルールを選択します。
3. [Delete] (削除) をクリックします。
4. [Delete] (削除) をクリックします。

Lambda 関数を削除するには

1. Lambda コンソールの [Functions] (関数) ページを開きます。
2. 作成した関数を選択します。

3. [Actions] (アクション) で、[Delete] (削除) を選択します。
4. [削除] を選択します。

チュートリアル: を使用した AWS API コールのログ記録 EventBridge

Amazon EventBridge ルールを使用して、 によって記録された AWS サービスによって行われた API コールに応答できます AWS CloudTrail。

このチュートリアルでは、 EventBridge コンソールで証AWS CloudTrail跡、Lambda 関数、およびルールを作成します。このルールは、Amazon EC2 インスタンスが停止したときに Lambda 関数を呼び出します。

ステップ:

- ステップ 1: AWS CloudTrail 証跡を作成する
- ステップ 2: AWS Lambda 関数を作成する
- ステップ 3: ルールを作成する
- ステップ 4: ルールをテストする
- ステップ 5: 成功を確認する
- ステップ 6: リソースをクリーンアップする

ステップ 1: AWS CloudTrail 証跡を作成する

すでに証跡を設定している場合は、手順 2 に進んでください。

追跡を作成するには

1. https://console.aws.amazon.com/cloudtrail/ で CloudTrail コンソールを開きます。
2. [Trails (証跡)]、[Create trail (証跡の作成)] の順に選択します。
3. [Trail name] に、証跡の名前を入力します。
4. [Storage Location] (保存場所) の、[Create a new S3 bucket] (新しい S3 バケットを作成する) を実行します。
5. AWS KMS のエイリアスで、KMS キーのエイリアスを入力します。
6. [次へ] をクリックします。
7. [次へ] をクリックします。
8. [追跡の作成]を選択します。

ステップ 2: AWS Lambda 関数を作成する

Lambda 関数を作成して、API コールイベントのログを記録します。

Lambda 関数を作成するには

1. https://console.aws.amazon.com/lambda/ で AWS Lambda コンソールを開きます。
2. 関数の作成 を選択します。
3. Author from scratch (製作者を最初から) を選択します。
4. Lambda 関数の名前と説明を入力します。例えば、関数名を `LogEC2StopInstance` とします。
5. 残りのオプションはデフォルトのまま、[Create function] (関数の作成) を選択します。
6. 関数ページの [Code] (コード) タブで、index.js をダブルクリックします。
7. 既存のコードを以下のコードに置き換えます。

```
'use strict';

exports.handler = (event, context, callback) => {
    console.log('LogEC2StopInstance');
    console.log('Received event:', JSON.stringify(event, null, 2));
    callback(null, 'Finished');
};
```

8. [Deploy] (デプロイ) をクリックします。

ステップ 3: ルールを作成する

ステップ 2 で作成した、Amazon EC2 インスタンスを停止するたびに Lambda 関数を実行するルールを作成します。

ルールを作成するには:

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで Rules] (ルール) を選択します。
3. ルールの作成 を選択します。
4. ルールの名前と説明を入力します。例えば、ルール `TestRule` に名前を付けます
5. [Event bus] (イベントバス) では、このルールに関連付けるイベントバスを選択します。このルールをアカウントからのイベントと一致させるには、[default] (デフォルト) を選択します。ア

カウントの AWS サービスがイベントを発行すると、常にアカウントのデフォルトのイベントバスに移動します。

6. [Rule type] (ルールタイプ) では、[Rule with an event pattern] (イベントパターンを持つルール) を選択します。
7. 次へ をクリックします。
8. [Event source] (イベントソース) では、AWS [services] (サービス) を選択します。
9. [Event pattern] (イベントパターン) の場合は、次のいずれかを実行します。

   a. [Event source] (イベントソース) では、ドロップダウンリストから [EC2] を選択します。
   b. イベントタイプ で、ドロップダウンリストから AWS 経由で API コール CloudTrailを選択します。
   c. [Specific operations(s)] (特定のオペレーション) を選択し、`StopInstances` を入力します。
10. 次へ をクリックします。
11. [Target types] (ターゲットタイプ) では、AWS [services] (サービス) を選択します。
12. ターゲットの選択では、ドロップダウンリストから [Lambda function] (Lambda 関数) を選択します。
13. [Function] (関数) で、[Step 1: Create a Lambda function] (ステップ 1 : Lambda 関数を作成する) セクションで作成した Lambda 関数を選択します。この例では、`LogEC2StopInstance` を選択します。
14. [次へ] をクリックします。
15. 次へ をクリックします。
16. ルールの詳細を確認し、ルールの作成 を選択します。

ステップ 4: ルールをテストする

Amazon EC2 コンソールを使用して Amazon EC2 インスタンスを停止することで、ルールをテストできます。インスタンスが停止するまで数分待ってから、コンソールで AWS Lambda CloudWatch メトリクスをチェックして、関数が実行されたことを確認します。

インスタンスを停止してルールをテストするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. インスタンスを起動します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスの起動」を参照してください。 Amazon EC2

3. インスタンスを停止します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスの停止と開始」を参照してください。 Amazon EC2
4. Lambda 関数からの出力を表示するには、以下の操作を実行します。
   a. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
   b. ナビゲーションペインで [ログ] を選択します。
   c. Lambda 関数 (`/aws/lambda/function-name`) のロググループの名前を選択します。
   d. 停止したインスタンスの関数によって提供されるデータを表示するログのストリーム名を選択します。
5. (オプション) 終了したら、停止したインスタンスを終了します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスの終了」を参照してください。 Amazon EC2

ステップ 5: 成功を確認する

CloudWatch ログに Lambda イベントが表示された場合、このチュートリアルは正常に完了しています。イベントが CloudWatch ログにない場合は、ルールが正常に作成されたことを確認することでトラブルシューティングを開始し、ルールが正しい場合は Lambda 関数のコードが正しいことを確認します。

ステップ 6: リソースをクリーンアップする

このチュートリアル用に作成したリソースは、保持しない場合は削除できます。使用しなくなった AWS リソースを削除することで、 AWS アカウントへの不要な課金を防ぐことができます。

EventBridge ルールを削除するには (複数可)

1. EventBridge コンソールのルールページを開きます。
2. 作成したルールを選択します。
3. [Delete] (削除) を選択します。
4. [Delete] (削除) を選択します。

Lambda 関数を削除するには

1. Lambda コンソールの [関数ページ] を開きます。
2. 作成した関数を選択します。
3. [Actions] (アクション) で、[Delete] (削除) を選択します。

4. [Delete] (削除) を選択します。

CloudTrail 証跡を削除するには (複数可)

1. CloudTrail コンソールの証跡ページを開きます。
2. 作成した証跡を選択します。
3. [Delete] (削除) を選択します。
4. [Delete] (削除) を選択します。

チュートリアル: を使用して Amazon EC2 インスタンスの状態を ログに記録する EventBridge

Amazon EC2 インスタンスの状態の変化をログに記録する AWS Lambda 関数を作成できます。そうすると、状態の遷移や、関心のある 1 つ以上の状態への遷移があるたびに Lambda 関数を実行するルールを作成します。このチュートリアルでは、新しいインスタンスが起動されるたびにログに記録します。

ステップ

- ステップ 1: AWS Lambda 関数を作成する
- ステップ 2: ルールを作成する
- ステップ 3: ルールをテストする
- ステップ 4: 成功を確認する
- ステップ 5: リソースをクリーンアップする

ステップ 1: AWS Lambda 関数を作成する

状態変更イベントのログを記録する Lambda 関数を作成します。ステップ 2 でルールを作成するときに、この関数を指定します。

Lambda 関数を作成するには

1. https://console.aws.amazon.com/lambda/ で AWS Lambda コンソールを開きます。
2. 関数の作成 を選択します。
3. Author from scratch (製作者を最初から) を選択します。
4. Lambda 関数の名前と説明を入力します。例えば、関数名を `LogEC2InstanceStateChange` とします。
5. 残りのオプションはデフォルトのまま、[Create function] (関数の作成) を選択します。
6. 関数ページの [Code] (コード) タブで、index.js をダブルクリックします。
7. 既存のコードを以下のコードに置き換えます。

```
'use strict';

exports.handler = (event, context, callback) => {
    console.log('LogEC2InstanceStateChange');
```

```
    console.log('Received event:', JSON.stringify(event, null, 2));
    callback(null, 'Finished');
};
```

8. [Deploy] (デプロイ) をクリックします。

ステップ 2: ルールを作成する

ステップ 1 で作成した Lambda 関数を実行するルールを作成します。Amazon EC2 インスタンスを起動すると、このルールが実行されます。

EventBridge ルールを作成するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで Rules] (ルール) を選択します。
3. ルールの作成 を選択します。
4. ルールの名前と説明を入力します。例えば、ルール `TestRule` に名前を付けます
5. [Event bus] (イベントバス) では、このルールに関連付けるイベントバスを選択します。このルールをアカウントからのイベントと一致させるには、[default] (デフォルト) を選択します。アカウントの AWS サービスがイベントを発行すると、常にアカウントのデフォルトのイベントバスに移動します。
6. [Rule type] (ルールタイプ) では、[Rule with an event pattern] (イベントパターンを持つルール) を選択します。
7. 次へ をクリックします。
8. [Event source] (イベントソース) では、AWS [services] (サービス) を選択します。
9. [Event pattern] (イベントパターン) の場合は、次のいずれかを実行します。
    a. [Event source] (イベントソース) では、ドロップダウンリストから [EC2] を選択します。
    b. [Event type] (イベントタイプ) として、ドロップダウンリストから[EC2 Instance State-change Notification] (EC2 インスタンス状態変更通知) を選択します。
    c. [Specific states(s)] (特定の状態) を選択して、ドロップダウンリストから [running] (実行中) を選択します。
    d. [Any instance] (任意のインスタンス) を選択します。
10. 次へ をクリックします。
11. [Target types] (ターゲットタイプ) では、AWS [services] (サービス) を選択します。

12. ターゲットの選択では、ドロップダウンリストから [Lambda function] (Lambda 関数) を選択します。
13. [Function] (関数) で、[Step 1: Create a Lambda function] (ステップ 1 : Lambda 関数を作成する) セクションで作成した Lambda 関数を選択します。この例では、`LogEC2InstanceStateChange` を選択します。
14. [次へ] をクリックします。
15. 次へ をクリックします。
16. ルールの詳細を確認し、[Create rule] (ルールの作成) を選択します。

ステップ 3: ルールをテストする

Amazon EC2 コンソールを使用して Amazon EC2 インスタンスを停止することで、ルールをテストできます。インスタンスが停止するまで数分待ってから、コンソールで AWS Lambda CloudWatch メトリクスをチェックして、関数が実行されたことを確認します。

インスタンスを停止してルールをテストするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. インスタンスを起動します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスの起動」を参照してください。 Amazon EC2
3. インスタンスを停止します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスの停止と起動」を参照してください。 Amazon EC2
4. Lambda 関数からの出力を表示するには、以下の操作を実行します。
   a. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
   b. ナビゲーションペインで [ログ] を選択します。
   c. Lambda 関数 (`/aws/lambda/`*`function-name`*) のロググループの名前を選択します。
   d. 停止したインスタンスの関数によって提供されるデータを表示するログのストリーム名を選択します。
5. (オプション) 終了したら、停止したインスタンスを終了します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスの終了」を参照してください。 Amazon EC2

ステップ 4: 成功を確認する

CloudWatch ログに Lambda イベントが表示された場合、このチュートリアルは正常に完了しています。イベントが CloudWatch ログにない場合は、ルールが正常に作成されたことを確認することでトラブルシューティングを開始し、ルールが正しい場合は Lambda 関数のコードが正しいことを確認します。

ステップ 5: リソースをクリーンアップする

このチュートリアル用に作成したリソースは、保持しない場合は削除できます。使用しなくなった AWS リソースを削除することで、 AWS アカウントへの不要な課金を防ぐことができます。

EventBridge ルールを削除するには (複数可)

1. EventBridge コンソールのルールページを開きます。
2. 作成したルールを選択します。
3. [Delete] (削除) を選択します。
4. [Delete] (削除) を選択します。

Lambda 関数を削除するには

1. Lambda コンソールの [関数ページ] を開きます。
2. 作成した関数を選択します。
3. [Actions] (アクション) で、[Delete] (削除) を選択します。
4. [削除] を選択します。

チュートリアル: Eventbridge を使用して Amazon S3 オブジェクトレベル操作のログを記録する

Amazon S3 バケットにオブジェクトレベルの API 操作のログを記録することができます。Amazon EventBridge がこれらのイベントと一致するには、AWS CloudTrail を使用してこれらのイベントを受信する証跡を設定する必要があります。

このチュートリアルでは、CloudTrail 証跡を作成して AWS Lambda 関数を作成し、S3データイベントに応答してその関数を呼び出すルールを EventBridgeコンソール で作成します。

ステップ

- ステップ 1: AWS CloudTrail 証跡を設定する
- ステップ 2: AWS Lambda 関数を作成する
- ステップ 3: ルールを作成する
- ステップ 4 : ルールをテストする
- ステップ 5: 成功を確認する
- ステップ 6: リソースをクリーンアップする

ステップ 1: AWS CloudTrail 証跡を設定する

S3 バケットのデータイベントを AWS CloudTrail と Eventbridge に記録するには、まず証跡を作成する必要があります。証跡は、アカウントでの API コールと関連イベントをキャプチャし、指定した S3 バケットにログファイルを提供します。既存の証跡を更新するか、新しい証跡を作成できます。

詳細については、『AWS CloudTrail ユーザーガイド』の「データイベント」を参照してください。

追跡を作成するには

1. CloudTrail コンソール (https://console.aws.amazon.com/cloudtrail/) を開きます。
2. [Trails (証跡)]、[Create trail (証跡の作成)] の順に選択します。
3. [Trail name] に、証跡の名前を入力します。
4. [Storage Location] (保存場所) の、[Create a new S3 bucket] (新しい S3 バケットを作成する) を実行します。
5. AWS KMS のエイリアスで、KMS キーのエイリアスを入力します。

6. [Next] (次へ) をクリックします。
7. [Event type] (イベントタイプ) で [Data events] (データイベント) を選択します。
8. [Data events] (データイベント) で、次のいずれかの操作を実行します。
   - バケットのすべての Amazon S3 オブジェクトのデータイベントを記録するには、S3 バケットと空のプレフィックスを指定します。そのバケットのオブジェクトでイベントが発生すると、証跡がイベントを処理して記録します。
   - 特定の Amazon S3 オブジェクトのデータイベントをバケットに記録するには、S3 バケットとオブジェクトのプレフィックスを指定します。そのバケットのオブジェクトでイベントが発生し、オブジェクトが指定したプレフィックスで始まっていると、証跡がイベントを処理して記録します。
9. 各リソースについて、ログ記録の対象を [Read] (読み取り) イベントにするか、[Write] (書き込み) イベントにするか、または両方のタイプのイベントにするかを選択します。
10. [Next] (次へ) をクリックします。
11. [追跡の作成]を選択します。

ステップ 2: AWS Lambda 関数を作成する

S3 バケットのデータイベントのログを記録する Lambda 関数を作成します。

Lambda 関数を作成するには

1. AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. 関数の作成 を選択します。
3. Author from scratch (製作者を最初から) を選択します。
4. Lambda 関数の名前と説明を入力します。例えば、関数名を `LogS3DataEvents` とします。
5. 残りのオプションはデフォルトのまま、[Create function] (関数の作成) を選択します。
6. 関数ページの [Code] (コード) タブで、index.js をダブルクリックします。
7. 既存のコードを以下のコードに置き換えます。

```
'use strict';

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

```
};
```

8. [Deploy] (デプロイ) をクリックします。

ステップ 3: ルールを作成する

ステップ 2 で作成した Lambda 関数を実行するルールを作成します。このルールは、Amazon S3 データイベントに応答して実行されます。

ルールを作成するには:

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで [Rules] (ルール) を選択します。
3. [Create rule] (ルールの作成) を選択します。
4. ルールの名前と説明を入力します。例えば、ルール `TestRule` に名前を付けます
5. [Event bus] (イベントバス) では、このルールに関連付けるイベントバスを選択します。このルールをアカウントからのイベントと一致させるには、[default] (デフォルト) を選択します。アカウントの AWS サービスがイベントを発行すると、常にアカウントのデフォルトのイベントバスに移動します。
6. [Rule type] (ルールタイプ) では、[Rule with an event pattern] (イベントパターンを持つルール) を選択します。
7. [Next] (次へ) をクリックします。
8. [Event source] (イベントソース) では、AWS[services] (サービス) を選択します。
9. [Event pattern] (イベントパターン) の場合は、次のいずれかを実行します。

    a. [Event source] (イベントソース) として、ドロップダウンリストから [Simple Storage Service (S3)] を選択します。

    b. [Event type] (イベントタイプ) では、ドロップダウンリストから [Object-Level API Call via CloudTrail] (CloudTrail 経由のオブジェクトレベル API 呼び出し) を選択します。

    c. [Specific operation(s)] (特定のオペレーション) を選択したら、[PutObject] を選択します。

    d. デフォルトでは、このルールはリージョン内のすべてのバケットのデータイベントと一致します。特定のバケットのデータイベントに一致させるには、[特定のバケット (名前別)] で指定し、1 つ以上のバケットを指定します。

10. [Next] (次へ) をクリックします。
11. [Target types] (ターゲットタイプ) では、AWS[services] (サービス) を選択します。

12. ターゲットの選択では、ドロップダウンリストから [Lambda function] (Lambda 関数) を選択します。
13. [Function] (関数) として、ステップ 1 で作成した `LogS3DataEvents` Lambda 関数を選択します。
14. [Next] (次へ) をクリックします。
15. [Next] (次へ) をクリックします。
16. ルールの詳細を確認し、[Create rule] (ルールの作成) を選択します。

ステップ 4 : ルールをテストする

ルールをテストするには、オブジェクトを S3 バケットに配置します。Lambda 関数が呼び出されたことを確認できます。

Lambda 関数のログを表示するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで [ログ] を選択します。
3. Lambda 関数 (`/aws/lambda/`*`function-name`*) のロググループの名前を選択します。
4. 起動したインスタンスの関数によって提供されるデータを表示するログのストリーム名を選択します。

また、証跡に指定した S3 バケット内の CloudTrail ログを確認することもできます。詳細については、AWS CloudTrail ユーザーガイドの「CloudTrail ログファイルの取得と表示」を参照してください。

ステップ 5: 成功を確認する

CloudWatch ログに Lambda イベントがある場合、このチュートリアルは正常に完了しています。イベントが CloudWatch ログにない場合は、ルールが正常に作成されたことを確認してトラブルシューティングを開始し、ルールが正しく見える場合は、Lambda 関数のコードが正しいことを確認します。

ステップ 6: リソースをクリーンアップする

このチュートリアル用に作成したリソースは、保存を希望しない限り、すぐに削除できます。使用しなくなった AWS リソースを削除することで、AWS アカウントに請求される料金が発生しないようにできます。

EventBridge ルールを削除するには

1. Eventbridge コンソールの [Rules] (ルール) ページを開きます。
2. 作成したルールを選択します。
3. [Delete] (削除) をクリックします。
4. [Delete] (削除) をクリックします。

Lambda 関数を削除するには

1. Lambda コンソールの [Functions] (関数) ページを開きます。
2. 作成した関数を選択します。
3. [Actions] (アクション) で、[Delete] (削除) を選択します。
4. [削除] を選択します。

CloudTrail 証跡を削除するには

1. CloudTrail コンソールの [Trails] (追跡) ページを開きます。
2. 作成した証跡を選択します。
3. [Delete] (削除) をクリックします。
4. [Delete] (削除) を選択します。

チュートリアル: EventBridge とaws.eventsスキーマを使用して Amazon Kinesis ストリームにイベントを送信する

で AWS API コールイベント EventBridge を Amazon Kinesis ストリーム に送信したり、Kinesis Data Streams アプリケーションを作成したり、大量のデータを処理したりできます。このチュートリアルでは、Kinesis ストリームを作成し、Amazon EC2 インスタンスが停止したときにそのストリームにイベントを送信するルールを EventBridge コンソールで作成します。

ステップ:

- 前提条件
- ステップ 1: Amazon Kinesis ストリームを作成する
- ステップ 2: ルールを作成する
- ステップ 3: ルールをテストする
- ステップ 4: イベントが送信されたことを確認する
- ステップ 5: リソースをクリーンアップする

前提条件

このチュートリアルでは、以下を使用します。

- を使用して AWS CLI Kinesis ストリームを操作します。

  をインストールするには AWS CLI、「 AWS CLI バージョン 2 のインストール、更新、アンインストール」を参照してください。

> **Note**
>
> このチュートリアルでは、 AWS イベントと組み込みaws.eventsスキーマレジストリを使用します。カスタムイベントのスキーマに基づいて EventBridge ルールを作成するには、カスタムスキーマレジストリに手動で追加するか、スキーマ検出を使用します。
> スキーマの詳細については、「???」を参照してください。他のイベントパターンオプションを使用してルールを作成する方法の詳細については、「???」を参照してください。

ステップ 1: Amazon Kinesis ストリームを作成する

ストリームを作成するには、コマンドプロンプトで コマンドを使用しますcreate-stream AWS CLI 。

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

ストリームのステータスが ACTIVE の場合、ストリームは準備完了です。ストリームの状態を確認するには、describe-stream コマンドを使用します。

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

ステップ 2: ルールを作成する

Amazon EC2 インスタンスを停止したときにイベントをストリームに送信するルールを作成します。

ルールを作成するには:

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで Rules] (ルール) を選択します。
3. ルールの作成 を選択します。
4. ルールの名前と説明を入力します。例えば、ルール TestRule に名前を付けます
5. [Event bus] (イベントバス) として、[default] (デフォルト) を選択します。
6. ルールタイプ では、イベントパターンを持つルール] を選択します。
7. 次へ をクリックします。
8. イベントソース で、AWS イベント または EventBridge パートナーイベント を選択します。
9. [Creation method] (作成方法) で、[Use schema] (スキーマを使用する) を選択します。
10. [Event pattern] (イベントパターン) の場合は、次のいずれかを実行します。

    a. [Schema type] (スキーマの種類) で、[Select schema from Schema registry] (スキーマレジストリからスキーマを選択する) を選択します。

    b. [Schema registry] (スキーマレジストリ) で、ドロップダウンリストから [aws.events] を選択します。

    c. スキーマ で、ドロップダウンリストから aws.ec2@EC2InstanceStateChangeNotification を選択します。

EventBridge は、 モデル の下にイベントスキーマを表示します。

EventBridge は、イベントパターンではなく、イベント に必要なプロパティの横に赤いアスタリスクを表示します。

d. [Models] (モデル) で、以下のイベントフィルタープロパティを設定します。

i. [state] プロパティの横にある [+ Edit] (+ 編集) を選択します。

[Relationship] (関係) は空のままにします。[値] に「`running`」と入力します。[Set] (セット) を選択します。

ii. [source] プロパティの横にある [+ Edit] (+ 編集) を選択します。

[Relationship] (関係) は空のままにします。[値] に「`aws.ec2`」と入力します。[Set] (セット) を選択します。

iii. [detail-type] プロパティの横にある [+ Edit] (+ 編集) を選択します。

[Relationship] (関係) は空のままにします。[値] に「`EC2 Instance State-change Notification`」と入力します。[Set] (セット) を選択します。

e. 作成したイベントパターンを表示するには、[Generate event pattern in JSON] (JSON でイベントパターンを生成する) を選択します。

EventBridge はイベントパターンを JSON で表示します。

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

11. 次へ をクリックします。
12. ターゲットタイプ] では、AWS サービス] を選択します。
13. [Select a target] (ターゲットの選択) では、ドロップダウンリストから [Kinesis stream] (Kinesis ストリーム) を選択します。
14. [Stream] (ストリーム) として、[Step 1: Create an Amazon Kinesis stream] (ステップ 1: Amazon Kinesis ストリームの作成) セクションで作成した Kinesis ストリームを選択します。この例では、`test` を選択します。

15. [Execution role] (実行ロール) として、[Create a new for role for this specific resource] (この特定のリソースのための新しいロールを作成する) を選択します。
16. [次へ] をクリックします。
17. 次へ をクリックします。
18. ルールの詳細を確認し、[Create rule] (ルールの作成) を選択します。

ステップ 3: ルールをテストする

ルールをテストするには、Amazon EC2 インスタンスを停止します。インスタンスが停止するまで数分待ってから、 CloudWatch メトリクスをチェックして関数が実行されたことを確認します。

インスタンスを停止してルールをテストするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. インスタンスを起動します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスの起動」を参照してください。 Amazon EC2
3. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
4. ナビゲーションペインで [Rules (ルール)] を選択します。

   作成したルールの名前を選択し、[Metrics for the rule] (ルールのメトリクス) を選択します。
5. (オプション) 終了したら、インスタンスを終了します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスの終了」を参照してください。 Amazon EC2

ステップ 4: イベントが送信されたことを確認する

を使用してストリームからレコード AWS CLI を取得し、イベントが送信されたことを確認できます。

レコードを取得するには

1. Kinesis ストリームからの読み込みを開始するには、コマンドプロンプトで `get-shard-iterator` コマンドを使用します。

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

   以下は出力例です。

```
{
    "ShardIterator": "AAAAAAAAAAHSywljv0zEgPX4NyKdZ5wryMzP9yALs8NeKbUjp1IxtZs1Sp
+KEd9I6AJ9ZG4lNR1EMi+9Md/nHvtLyxpfhEzYvkTZ4D9DQVz/mBYWRO6OTZRKnW9gd
+efGN2aHFdkH1rJl4BL9Wyrk+ghYG22D2T1Da2EyNSH1+LAbK33gQweTJADBdyMwlo5r6PqcP2dzhg="
}
```

2. レコードを取得するには、次の get-records コマンドを使用します。前のステップの出力からシャードイテレーターを使用します。

```
aws kinesis get-records --shard-
iterator AAAAAAAAAAHSywljv0zEgPX4NyKdZ5wryMzP9yALs8NeKbUjp1IxtZs1Sp
+KEd9I6AJ9ZG4lNR1EMi+9Md/nHvtLyxpfhEzYvkTZ4D9DQVz/mBYWRO6OTZRKnW9gd
+efGN2aHFdkH1rJl4BL9Wyrk+ghYG22D2T1Da2EyNSH1+LAbK33gQweTJADBdyMwlo5r6PqcP2dzhg=
```

コマンドが成功すると、指定シャードのストリームからレコードをリクエストします。0 以上のレコードを受け取ることができます。返されるレコードは、ストリーム内のすべてのレコードを表すとは限りません。ご希望のデータを受け取っていない場合は、get-records を継続して呼び出します。

3. Kinesis のレコードは Base64 でエンコードされています。Base64 デコーダーを使用してデータを復号すめと、それが JSON 形式でストリームに送信されたイベントであることを確認できます。

ステップ 5: リソースをクリーンアップする

このチュートリアル用に作成したリソースは、保持しない場合は削除できます。使用しなくなった AWS リソースを削除することで、 AWS アカウントへの不要な課金を防ぐことができます。

EventBridge ルールを削除するには (複数可)

1. EventBridge コンソールのルールページを開きます。
2. 作成したルールを選択します。
3. [Delete] (削除) を選択します。
4. [Delete] (削除) を選択します。

Kinesis ストリームを削除するには

1. Kinesis コンソールの [Data streams] (データストリーム) ページを開きます。

2. 作成したストリームを選択します。
3. [アクション]、[削除] の順に選択します。
4. fiekd で delete と入力し、[Delete] (削除) を選択します。

チュートリアル: Eventbridge を使用した、自動化された Amazon EBS スナップショットのスケジュール

EventBridge ルール をスケジュールに従って実行できます。このチュートリアルでは、スケジュールに基づいて既存の Amazon Elastic Block Store (Amazon EBS) ボリュームのスナップショットを作成します。スナップショットは、一定の速度 (数分ごと) で作成することも、cron 式を使用して特定の時間帯で作成することもできます。

> **Important**
>
> 組み込みのターゲットにルールを作成するには、AWS Management Console を使用する必要があります。

ステップ

- ステップ 1: ルールを作成する
- ステップ 2: ルールをテストする
- ステップ 3: 成功を確認する
- ステップ 4: リソースをクリーンアップする

ステップ 1: ルールを作成する

スケジュールに従ってスナップショットを作成するルールを作成します。レート式または cron 式を使用してスケジュールを指定できます。詳細については、「スケジュールに従って実行する Amazon EventBridge ルールの作成」を参照してください。

ルールを作成するには:

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで [Rules] (ルール) を選択します。
3. [Create rule] (ルールの作成) を選択します。
4. ルールの名前と説明を入力します。

   ルールには、同じリージョン内および同じイベントバス上の別のルールと同じ名前を付けることはできません。

5. [Event bus] (イベントバス) では、このルールに関連付けるイベントバスを選択します。このルールをアカウントからのイベントと一致させるには、AWSデフォルトのイベントバスを選択します。アカウントの AWS サービスがイベントを発行すると、常にアカウントのデフォルトのイベントバスに移動します。
6. [Rule type] (ルールタイプ) では、[Schedule] (スケジュール) を選択します。
7. [Next] (次へ) をクリックします。
8. [Schedule pattern] (スケジュールパターン) については、[A schedule that runs at a regular rate, such as every 10 minutes.] (10 分ごとなど、定期的に実行されるスケジュール。) を選択して、**5** を入力し、ドロップダウンリストから [Minutes] (分) を選択します。
9. [Next] (次へ) をクリックします。
10. [Target types] (ターゲットタイプ) では、AWS[services] (サービス) を選択します。
11. [Select a target] (ターゲットの選択) では、ドロップダウンリストから [EBS Create Snapshot] (EBS によるスナップショットの作成) を選択します。
12. [Volume ID] (ボリューム ID) には、Amazon EBS ボリュームのボリューム ID を入力します。
13. [Execution role] (実行ロール) として、[Create a new for role for this specific resource] (この特定のリソースのための新しいロールを作成する) を選択します。
14. [Next] (次へ) をクリックします。
15. [Next] (次へ) をクリックします。
16. ルールの詳細を確認し、[Create rule] (ルールの作成) を選択します。

ステップ 2: ルールをテストする

最初のスナップショットを作成した後、そのスナップショットを表示することでルールの働きを検証できます。

ルールをテストするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで [Elastic Block Store]、[Snapshots] の順に選択します。
3. 最初のスナップショットがリストに表示されることを確認します。

ステップ 3: 成功を確認する

一覧にスナップショットが表示された場合、このチュートリアルは正常に完了しています。スナップショットが一覧にない場合は、ルールが正常に作成されたことを確認してトラブルシューティングを開始します。

ステップ 4: リソースをクリーンアップする

このチュートリアル用に作成したリソースは、保存を希望しない限り、すぐに削除できます。使用しなくなった AWS リソースを削除することで、AWS アカウントに請求される料金が発生しないようにできます。

EventBridge ルールを削除するには

1. Eventbridge コンソールの [Rules] (ルール) ページを開きます。
2. 作成したルールを選択します。
3. [Delete] (削除) をクリックします。
4. [Delete] (削除) を選択します。

チュートリアル: Amazon S3 オブジェクトが作成されたときに通知を送信する

Amazon Simple Storage Service (Amazon S3) オブジェクトが Amazon EventBridge と Amazon SNS を使用して作成されたときに、E メール通知を送信できます。このチュートリアルでは、SNS トピックとサブスクリプションを作成します。次に、EventBridge コンソールで、Amazon S3 `Object Created` イベントが受信されたときにそのトピックにイベントを送信するルールを作成します。

ステップ:

- 前提条件
- ステップ 1: Amazon SNS トピックを作成する
- ステップ 2: Amazon SNS サブスクリプションを作成する
- ステップ 3: ルールを作成する
- ステップ 4: ルールをテストする
- ステップ 5: リソースをクリーンアップする

前提条件

EventBridge で Amazon S3 イベントを受信するには、Amazon S3 コンソールで EventBridge を有効にする必要があります。このチュートリアルでは、EventBridge が有効であることを前提としています。詳細については、「S3 コンソールでの Amazon EventBridgeの有効化」を参照してください。

ステップ 1: Amazon SNS トピックを作成する

EventBridge からイベントを受信するトピックを作成します。

トピックを作成する

1. https://console.aws.amazon.com/sns/v3/home で Amazon SNS コンソールを開きます。
2. ナビゲーションペインで、[Topics] (トピック) を選択します。
3. [Create topic] (トピックの作成) を選択します。
4. [Type] (タイプ) で、[Standard] (標準) を選択します。
5. テーブルの名前として **`eventbridge-test`** を入力します。
6. [Create topic] (トピックの作成) を選択します。

ステップ 2: Amazon SNS サブスクリプションを作成する

トピックでイベントを受信したときに Amazon S3 から E メール通知を受け取るサブスクリプションを作成します。

サブスクリプションを作成するには

1. Amazon SNS コンソール (https://console.aws.amazon.com/sns/v3/home) を開きます。
2. ナビゲーションペインで [Subscriptions] (サブスクリプション) を選択します。
3. [Create subscription] を選択します。
4. [トピック ARN] で、ステップ 1 で作成したトピックを選択します。このチュートリアルでは、eventbridge-test を選択します。
5. [Protocol] (プロトコル) で [Email] (E メール) を選択します。
6. [エンドポイント] に E メールアドレスを入力します。
7. [Create subscription] (サブスクリプションの作成) を選択します。
8. AWS 通知から受信した E メールで、[サブスクリプションを確認] を選択してサブスクリプションを確認します。

ステップ 3: ルールを作成する

Amazon S3 オブジェクトが作成されたときにイベントをトピックに送信するルールを作成します。

ルールを作成するには:

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで [Rules] (ルール) を選択します。
3. [Create rule] (ルールの作成) を選択します。
4. ルールの名前と説明を入力します。例えば、ルール `s3-test` に名前を付けます
5. [Event bus] (イベントバス) として、[default] (デフォルト) を選択します。
6. [Rule type] (ルールタイプ) では、[Rule with an event pattern] (イベントパターンを持つルール) を選択します。
7. [Next] (次へ) をクリックします。
8. [Event source] (イベントソース) で、[AWS events or EventBridge partner events] (イベントまたは EventBridge パートナーイベント) を選択します。

9. [Creation method] (作成方法) で、[Use pattern form] (パターンフォームを使用する) を選択します。
10. [Event pattern] (イベントパターン) の場合は、次のいずれかを実行します。
    a. [Event source] (イベントソース) で、ドロップダウンリストから [AWS services] (AWS のサービス) を選択します。
    b. [AWS service] (AWS のサービス) で、ドロップダウンリストから [Simple Storage Service (S3)] を選択します。
    c. [Event type] (イベントタイプ) として、ドロップダウンリストから [Amazon S3 Event Notification] (Amazon S3 イベント通知) を選択します。
    d. [Specific events(s)] (特定のイベント) を選択し、ドロップダウンリストから [Object Created] (オブジェクトの作成) を選択します。
    e. [Any bucket] (任意のバケット) を選択します。
11. [Next] (次へ) をクリックします。
12. [Target types] (ターゲットタイプ) では、AWS[services] (サービス) を選択します。
13. ターゲットの選択では、ドロップダウンリストから [SNS topic] (SNS トピック) を選択します。
14. [Topic] (トピック) については、[Step 1: Create an SNS topic] (ステップ 1: SNS トピックの作成) セクションで作成した Amazon SNS トピックを選択します。この例では、`eventbridge-test` を選択します。
15. [Next] (次へ) をクリックします。
16. [Next] (次へ) をクリックします。
17. ルールの詳細を確認し、[Create rule] (ルールの作成) を選択します。

ステップ 4: ルールをテストする

ルールをテストするには、EventBridge 対応バケットにファイルをアップロードして Amazon S3 オブジェクトを作成します。次に、数分待ってからから、AWS 通知から E メールを受信するかどうかを確認します。

ステップ 5: リソースをクリーンアップする

このチュートリアル用に作成したリソースは、保存を希望しない限り、すぐに削除できます。使用しなくなった AWS リソースを削除することで、AWS アカウントに請求される料金が発生しないようにできます。

SNS トピックを削除するには

1. SNS コンソールの [トピック] ページを開きます。
2. 先ほど作成したトピックを選択します。
3. [Delete] (削除) をクリックします。
4. **`delete me`** と入力します。
5. [Delete] (削除) をクリックします。

SNS サブスクリプションを削除するには

1. SNS コンソールの [サブスクリプションページ] を開きます。
2. 作成したサブスクリプションを選択します。
3. [Delete] (削除) をクリックします。
4. [Delete] (削除) をクリックします。

EventBridge ルールを削除するには

1. Eventbridge コンソールの [Rules] (ルール) ページを開きます。
2. 作成したルールを選択します。
3. [Delete] (削除) をクリックします。
4. [Delete] (削除) を選択します。

チュートリアル: EventBridge を使用した AWS Lambda 関数のスケジュール

スケジュールに基づいて AWS Lambda 関数を実行するルールを設定できます。このチュートリアルでは、AWS Management Console または AWS CLI を使用してルールを作成する方法について説明します。AWS CLI を使用したいがインストールしていない場合は、「AWS CLI バージョン 2 のインストール、更新、アンインストール」を参照してください。

スケジュールについて、EventBridge のスケジュール式は、秒レベルの精度ではありません。cron 式を使用した最小の粒度は 1 分です。EventBridge とターゲットサービスは分散しているため、スケジュールされたルールがトリガーされてからターゲットサービスがターゲットリソースを実行するまでの間に数秒の遅延が発生することもあります。

ステップ:

- ステップ 1: Lambda 関数を作成する
- ステップ 2: ルールを作成する
- ステップ 3: ルールを確認する
- ステップ 4: 成功を確認する
- ステップ 5: リソースをクリーンアップする

ステップ 1: Lambda 関数を作成する

スケジュールされたイベントのログを記録する Lambda 関数を作成します。

Lambda 関数を作成するには

1. AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. 関数の作成 を選択します。
3. Author from scratch (製作者を最初から) を選択します。
4. Lambda 関数の名前と説明を入力します。例えば、関数名を `LogScheduledEvent` とします。
5. 残りのオプションはデフォルトのまま、[Create function] (関数の作成) を選択します。
6. 関数ページの [Code] (コード) タブで、index.js をダブルクリックします。
7. 既存のコードを以下のコードに置き換えます。

```
'use strict';
```

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

8. [Deploy] (デプロイ) をクリックします。

ステップ 2: ルールを作成する

ステップ 1 で作成した Lambda 関数をスケジュールに従って実行するルールを作成します。

ルールを作成するには、コンソールまたは AWS CLI を使用します。AWS CLI を使用するには、まず Lambda 関数を呼び出すためのアクセス許可をルールに付与します。次にルールを作成し、Lambda 関数をターゲットとして追加できます。

ルールを作成するには (コンソール)

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで [Rules] (ルール) を選択します。
3. [Create rule] (ルールの作成) を選択します。
4. ルールの名前と説明を入力します。

   ルールには、同じリージョン内および同じイベントバス上の別のルールと同じ名前を付けることはできません。
5. [Event bus] (イベントバス) では、このルールに関連付けるイベントバスを選択します。このルールをアカウントからのイベントと一致させるには、AWSデフォルトのイベントバスを選択します。アカウントの AWS サービスがイベントを発行すると、常にアカウントのデフォルトのイベントバスに移動します。
6. [Rule type] (ルールタイプ) では、[Schedule] (スケジュール) を選択します。
7. [Next] (次へ) をクリックします。
8. [Schedule pattern] (スケジュールパターン) については、[A schedule that runs at a regular rate, such as every 10 minutes.] (10 分ごとなど、定期的に実行されるスケジュール。) を選択して、**5** を入力し、ドロップダウンリストから [Minutes] (分) を選択します。
9. [Next] (次へ) をクリックします。
10. [Target types] (ターゲットタイプ) では、AWS[services] (サービス) を選択します。

11. ターゲットの選択では、ドロップダウンリストから [Lambda function] (Lambda 関数) を選択します。
12. [Function] (関数) で、[Step 1: Create a Lambda function] (ステップ 1 : Lambda 関数を作成する) セクションで作成した Lambda 関数を選択します。この例では、LogScheduledEvent を選択します。
13. [Next] (次へ) をクリックします。
14. [Next] (次へ) をクリックします。
15. ルールの詳細を確認し、[Create rule] (ルールの作成) を選択します。

ルール (AWS CLI) を作成するには

1. スケジュールに従って実行するルールを作成するには、put-rule コマンドを使用します。

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

このルールが実行されると、イベントが作成されてターゲットに送信されます。以下に示しているのは、イベントの例です。

```
{
    "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 サービスプリンシパル (events.amazonaws.com) のアクセス許可を使用してルールを実行するには、add-permission コマンドを使用します。

```
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. 次の内容で、targets.json ファイルを作成します。

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

4. ステップ 1 で作成した Lambda 関数をルールに追加するには、put-targets コマンドを使用します。

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

ステップ 3: ルールを確認する

ステップ 2 を完了してから少なくとも 5 分待ってから、Lambda 関数が呼び出されたことを確認できます。

Lambda 関数からの出力を表示する

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで [ログ] を選択します。
3. Lambda 関数 (/aws/lambda/*function-name*) のロググループの名前を選択します。
4. 起動したインスタンスの関数によって提供されるデータを表示するログのストリーム名を選択します。

ステップ 4: 成功を確認する

CloudWatch ログに Lambda イベントがある場合、このチュートリアルは正常に完了しています。イベントが CloudWatch ログにない場合は、ルールが正常に作成されたことを確認してトラブルシューティングを開始し、ルールが正しく見える場合は、Lambda 関数のコードが正しいことを確認します。

ステップ 5: リソースをクリーンアップする

このチュートリアル用に作成したリソースは、保存を希望しない限り、すぐに削除できます。使用しなくなった AWS リソースを削除することで、AWS アカウントに請求される料金が発生しないようにできます。

EventBridge ルールを削除するには

1. Eventbridge コンソールの [Rules] (ルール) ページを開きます。
2. 作成したルールを選択します。
3. [Delete] (削除) をクリックします。
4. [Delete] (削除) をクリックします。

Lambda 関数を削除するには

1. Lambda コンソールの [Functions] (関数) ページを開きます。
2. 作成した関数を選択します。
3. [Actions] (アクション) で、[Delete] (削除) を選択します。
4. [削除] を選択します。

SaaS プロバイダとの統合に関する Amazon EventBridge のチュートリアル

EventBridge は SaaS パートナーアプリケーションおよびサービスと直接連携して、イベントを送受信できます。以下のチュートリアルでは、EventBridge を SaaS パートナーと統合する方法を示します。

チュートリアル:

- チュートリアル: API 送信先として Datadog への接続を作成する
- チュートリアル: API 送信先として Salesforce への接続を作成する
- チュートリアル: API 送信先として Zendesk への接続を作成する

チュートリアル: API 送信先として Datadog への接続を作成する

EventBridge を使用して、Datadog などのサードパーティーサービスにイベントを送信できます。

このチュートリアルでは、EventBridge コンソールを使用して、Datadog への接続、Datadog を指す API 送信先、および Datadog にイベントを送信する ルールを作成します。

ステップ:

- 前提条件
- ステップ 1: 接続を作成する
- ステップ 2: API 送信先を作成する
- ステップ 3: ルールを作成する
- ステップ 4: ルールをテストする
- ステップ 5: リソースをクリーンアップする

前提条件

このチュートリアルを完了するには、以下のリソースが必要です。

- Datadog アカウント。
- Datadog API キー。
- EventBridge に対応した Amazon Simple Storage Service (Amazon S3) バケット。

ステップ 1: 接続を作成する

Datadog にイベントを送信するには、まず、Datadog API への接続を確立する必要があります。

接続を作成するには

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで、[API destinations] (API 送信先) を選択します。
3. [Connections (接続)] タブを選択し、[Create connection (接続の作成)] を選択します。
4. 接続の名前と説明を入力します。例えば、名前として「**`Datadog`**」、説明として「**`Datadog API Connection`**」と入力します。
5. 認証タイプには、API キーを選択します。

6. [API key name (API キー名)] に「**DD-API-KEY**」と入力します。
7. [Value] (値) に、Datadog シークレット API キーを貼り付けます。
8. [Create] (作成) を選択します。

ステップ 2: API 送信先を作成する

接続を作成したので、次に API 送信先を作成して、ルールのターゲットとして使用します。

API 送信先を作成するには

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで、[API destinations] (API 送信先) を選択します。
3. [Create API destination] (API 送信先の作成) を選択します。
4. API 送信先の名前と説明を入力します。この例では、名前には「**DatadogAD**」、説明には「**Datadog API Destination**」を入力します。
5. [API destination endpoint] (API 送信先エンドポイント) には、「**https://http-intake.logs.datadoghq.com/api/v2/logs**」と入力します。
6. [HTTP メソッド] で、[POST] を選択します。
7. [Invocation rate limit] (呼び出しレート制限) には、「**300**」と入力します。
8. [Connection](接続) で、[Use an existing connection] (既存の接続を使用する) を選択し、手順 1 で作成した Datadog 接続を選択します。
9. [Create] (作成) を選択します。

ステップ 3: ルールを作成する

次に、Amazon S3 オブジェクトが作成されたときにイベントを Datadog に送信するルールを作成します。

ルールを作成するには:

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで [Rules] (ルール) を選択します。
3. [Create rule] (ルールの作成) を選択します。
4. ルールの名前と説明を入力します。この例では、名前には「**DatadogRule**」、説明には「**Rule to send events to Datadog for S3 object creation**」を入力します。

5. [Event bus] (イベントバス) として、[default] (デフォルト) を選択します。
6. [Rule type] (ルールタイプ) では、[Rule with an event pattern] (イベントパターンを持つルール) を選択します。
7. [Next] (次へ) をクリックします。
8. [Event source] (イベントソース) では、[Other] (その他) を選択します。
9. [Event pattern] (イベントパターン) では、次のように入力します。

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

10. [Next] (次へ) をクリックします。
11. [Target types] (ターゲットタイプ) として、[EventBridge API destination] (EventBridge API 送信先) を選択します。
12. [API destination] (API 送信先) として、[Use an existing API destination] (既存の API 送信先を使用する) を選択し、ステップ 2 で作成した `DatadogAD` 送信先を選択します。
13. [Execution role] (実行ロール) として、[Create a new for role for this specific resource] (この特定のリソースのための新しいロールを作成する) を選択します。
14. [Additional settings] (追加設定) では、以下を実行します。

    a. ターゲット入力の設定では、ドロップダウンリストから[Input transformer] (インプットトランスフォーマー) を選択します。

    b. [Configure input transformer] (インプットトランスフォーマーの設定) を選択します。

    c. [Sample events] (イベント例) では、以下を入力します。

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

    d. [Target input transformer] (ターゲットインプットトランスフォーマー) では、以下を実行します。

        i. [Input Path] (入力パス) では、以下を入力します。

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

        ii. [Input Template] (入力テンプレート) では、以下を入力します。

```
{"message": <detail>}
```

e. [Confirm] (確認) を選択します。

15. [Next] (次へ) をクリックします。
16. [Next] (次へ) をクリックします。
17. ルールの詳細を確認し、[Create rule] (ルールの作成) を選択します。

ステップ 4: ルールをテストする

ルールをテストするには、EventBridge 対応バケットにファイルをアップロードして Amazon S3 オブジェクトを作成します。作成されたオブジェクトは Datadog Logs コンソールに記録されます。

ステップ 5: リソースをクリーンアップする

このチュートリアル用に作成したリソースは、保存を希望しない限り、すぐに削除できます。使用しなくなった AWS リソースを削除することで、AWS アカウントに請求される料金が発生しないようにできます。

EventBridge 接続を削除するには

1. Eventbridge コンソールの [API destination] (API 送信先) ページを開きます。
2. [Connections (接続)] タブを選択します。
3. 作成した接続を選択します。
4. [Delete] (削除) をクリックします。
5. 接続の名前を入力し、[Delete] (削除) を選択します。

EventBridge API の送信先を削除するには

1. Eventbridge コンソールの [API destination] (API 送信先) ページを開きます。
2. 作成した API の送信先を選択します。
3. [Delete] (削除) をクリックします。
4. API 送信先の名前を入力し、[Delete] (削除) を選択します。

EventBridge ルールを削除するには

1. Eventbridge コンソールの [Rules] (ルール) ページを開きます。
2. 作成したルールを選択します。
3. [Delete] (削除) をクリックします。
4. [Delete] (削除) を選択します。

チュートリアル: API 送信先として Salesforce への接続を作成する

EventBridge を使用して、 などのサードパーティサービスにイベントをルーティングできますSalesforce。

このチュートリアルでは、 EventBridge コンソールを使用して、 への接続Salesforce、 を指す API 送信先Salesforce、および にイベントをルーティングするルールを作成しますSalesforce。

ステップ:

- 前提条件
- ステップ 1: 接続を作成する
- ステップ 2: API 送信先を作成する
- ステップ 3: ルールを作成する
- ステップ 4: ルールをテストする
- ステップ 5: リソースをクリーンアップする

前提条件

このチュートリアルを完了するには、以下のリソースが必要です。

- Salesforce アカウント。
- Salesforce コネクテッドアプリケーション。
- Salesforce セキュリティトークン。
- Salesforce カスタムプラットフォームイベント。
- EventBridgeが有効な Amazon Simple Storage Service (Amazon S3) バケット。

ステップ 1: 接続を作成する

Salesforce にイベントを送信するには、まず、Salesforce API への接続を確立する必要があります。

接続を作成するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで、[API destinations] (API 送信先) を選択します。
3. [Connections (接続)] タブを選択し、[Create connection (接続の作成)] を選択します。

4. 接続の名前と説明を入力します。例えば、名前として「**Salesforce**」、説明として「**Salesforce API Connection**」と入力します。
5. [Destination type] (送信先タイプ) として、[Partners] (パートナー) を選択し、[Partner Destinations] (パートナー送信先) として、ドロップダウンリストから Salesforce を選択します。
6. [Authorization endpoint] (認可エンドポイント) として、以下のいずれかを入力します。
   - 実稼働組織を使用している場合は、**https://*MyDomainName*.my.salesforce.com./services/oauth2/token** を入力します。
   - 拡張ドメインのないサンドボックスを使用している場合は、**https://*MyDomainName*--*SandboxName*.my. salesforce.com/services / oauth2/token** を入力します。
   - 拡張ドメインのあるサンドボックスを使用している場合は、**https://*MyDomainName*--*SandboxName*.sandbox.my.salesforce.com/services/oauth2/token** を入力します。
7. [HTTP Method] (HTTP メソッド) として、ドロップダウンリストから [POST] を選択します。
8. [Client ID] (クライアント ID) として、Salesforce コネクテッドアプリケーションのクライアント ID を入力します。
9. [Client secret] (クライアントシークレット) として、Salesforce コネクテッドアプリケーションのクライアントシークレットを入力します。
10. OAuth Http パラメータ には、次のキーと値のペアを入力します。

キー	値
grant_type	client_credentials

11. [作成] を選択します。

ステップ 2: API 送信先を作成する

接続を作成したので、次に API 送信先を作成して、ルールのターゲットとして使用します。

API 送信先を作成するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで、[API destinations] (API 送信先) を選択します。

3. [Create API destination] (API 送信先の作成) を選択します。
4. API 送信先の名前と説明を入力します。この例では、名前には「**SalesforceAD**」、説明には「**Salesforce API Destination**」を入力します。
5. [API destination endpoint] (API 送信先エンドポイント) には、**https://*MyDomainName*.my.salesforce.com/services/data/v54.0/sobjects/*MyEvent__e*** を入力します。ここで、Myevent__e は、情報を送信するプラットフォームイベントです。
6. [HTTP Method] (HTTP メソッド) として、ドロップダウンリストから [POST] を選択します。
7. [Invocation rate limit] (呼び出しレート制限) には、「**300**」と入力します。
8. [Connection](接続) で、[Use an existing connection] (既存の接続を使用する) を選択し、手順 1 で作成した `Salesforce` 接続を選択します。
9. [作成] を選択します。

ステップ 3: ルールを作成する

次に、Amazon S3 オブジェクトが作成されたときにイベントを Salesforce に送信するルールを作成します。

ルールを作成するには:

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. ナビゲーションペインで Rules] (ルール) を選択します。
3. ルールの作成 を選択します。
4. ルールの名前と説明を入力します。この例では、名前には「**SalesforceRule**」、説明には「**Rule to send events to Salesforce for S3 object creation**」を入力します。
5. [Event bus] (イベントバス) として、[default] (デフォルト) を選択します。
6. [ルールタイプ] では、[イベントパターンを持つルール] を選択します。
7. 次へ をクリックします。
8. [Event source] (イベントソース) では、[Other] (その他) を選択します。
9. [Event pattern] (イベントパターン) では、次のように入力します。

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

10. [次へ] を選択します。
11. ターゲットタイプ で、EventBridge API 送信先 を選択します。
12. [API destination] (API 送信先) として、[Use an existing API destination] (既存の API 送信先を使用する) を選択し、ステップ 2 で作成した `SalesforceAD` 送信先を選択します。
13. [Execution role] (実行ロール) として、[Create a new for role for this specific resource] (この特定のリソースのための新しいロールを作成する) を選択します。
14. [Additional settings] (追加設定) では、以下を実行します。
    a. ターゲット入力の設定では、ドロップダウンリストから[Input transformer] (インプットトランスフォーマー) を選択します。
    b. [Configure input transformer] (インプットトランスフォーマーの設定) を選択します。
    c. [Sample events] (イベント例) では、以下を入力します。

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

    d. [Target input transformer] (ターゲットインプットトランスフォーマー) では、以下を実行します。
        i. [Input Path] (入力パス) では、以下を入力します。

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

        ii. [Input Template] (入力テンプレート) では、以下を入力します。

```
{"message": <detail>}
```

    e. [Confirm] (確認) を選択します。
15. [次へ] を選択します。
16. 次へ をクリックします。
17. ルールの詳細を確認し、ルールの作成 を選択します。

ステップ 4: ルールをテストする

ルールをテストするには、 EventBridgeが有効なバケットにファイルをアップロードして Amazon S3 オブジェクトを作成します。作成されたオブジェクトに関する情報は、Salesforce プラットフォームイベントに送信されます。

ステップ 5: リソースをクリーンアップする

このチュートリアル用に作成したリソースは、保持しない場合は削除できます。使用しなくなった AWS リソースを削除することで、 AWS アカウントに不要な料金が発生するのを防ぐことができます。

EventBridge Connections(s) を削除するには

1. EventBridge コンソールの API 送信先ページを開きます。
2. [Connections (接続)] タブを選択します。
3. 作成した接続を選択します。
4. [削除] をクリックします。
5. 接続の名前を入力し、[Delete] (削除) を選択します。

EventBridge API 送信先を削除するには (複数可)

1. EventBridge コンソールの API 送信先ページを開きます。
2. 作成した API の送信先を選択します。
3. [削除] をクリックします。
4. API 送信先の名前を入力し、[Delete] (削除) を選択します。

EventBridge ルールを削除するには (複数可)

1. EventBridge コンソールのルールページを開きます。
2. 作成したルールを選択します。
3. [Delete] (削除) を選択します。
4. [Delete] (削除) を選択します。

チュートリアル: API 送信先として Zendesk への接続を作成する

EventBridge を使用して、Zendesk などのサードパーティーサービスにイベントを送信できます。

このチュートリアルでは、EventBridge コンソールを使用して、Zendesk への接続、Zendesk を指す API 送信先、および Zendesk にイベントを送信する ルールを作成します。

ステップ:

前提条件

このチュートリアルを完了するには、以下のリソースが必要です。

- Zendesk アカウント。
- EventBridge に対応した Amazon Simple Storage Service (Amazon S3) バケット。

ステップ 1: 接続を作成する

Zendesk にイベントを送信するには、まず、Zendesk API への接続を確立する必要があります。

接続を作成するには

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで、[API destinations] (API 送信先) を選択します。
3. [Connections (接続)] タブを選択し、[Create connection (接続の作成)] を選択します。
4. 接続の名前と説明を入力します。この例では、名前には「**Zendesk**」、説明には「**Connection to Zendesk API**」を入力します。
5. [Authorization type] (認証タイプ) で、[Basic (Username/Password)] (基本 (ユーザー名/パスワード) を選択します。
6. [Username] (ユーザー名)] に、Zendesk ユーザー名を入力します。

7. [Password] (パスワード) に Zendesk パスワードを入力します。
8. [Create] (作成) を選択します。

ステップ 2: API 送信先を作成する

接続を作成したので、次に API 送信先を作成して、ルールのターゲットとして使用します。

API 送信先を作成するには

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで、[API destinations] (API 送信先) を選択します。
3. [Create API destination] (API 送信先の作成) を選択します。
4. API 送信先の名前と説明を入力します。この例では、名前には「**ZendeskAD**」、説明には「**Zendesk API destination**」を入力します。
5. [API destination endpoint] (API 送信先エンドポイント) に、**https://*your-subdomain*.zendesk.com/api/v2/tickets.json** と入力します。ここで、*your-subdomain* は Zendesk アカウントに関連付けられたサブドメインです。
6. [HTTP メソッド] で、[POST] を選択します。
7. [Invocation rate limit] (呼び出しレート制限) には、「**10**」と入力します。
8. [Connection](接続) で、[Use an existing connection] (既存の接続を使用する) を選択し、手順 1 で作成した Zendesk 接続を選択します。
9. [Create] (作成) を選択します。

ステップ 3: ルールを作成する

次に、Amazon S3 オブジェクトが作成されたときにイベントを Zendesk に送信するルールを作成します。

ルールを作成するには:

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで [Rules] (ルール) を選択します。
3. [Create rule] (ルールの作成) を選択します。
4. ルールの名前と説明を入力します。この例では、名前には「**ZendeskRule**」、説明には「**Rule to send events to Zendesk when S3 objects are created**」を入力します。

5. [Event bus] (イベントバス) として、[default] (デフォルト) を選択します。
6. [Rule type] (ルールタイプ) では、[Rule with an event pattern] (イベントパターンを持つルール) を選択します。
7. [Next] (次へ) をクリックします。
8. [Event source] (イベントソース) では、[Other] (その他) を選択します。
9. [Event pattern] (イベントパターン) では、次のように入力します。

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

10. [Next] (次へ) をクリックします。
11. [Target types] (ターゲットタイプ) として、[EventBridge API destination] (EventBridge API 送信先) を選択します。
12. [API destination] (API 送信先) として、[Use an existing API destination] (既存の API 送信先を使用する) を選択し、ステップ 2 で作成した ZendeskAD 送信先を選択します。
13. [Execution role] (実行ロール) として、[Create a new for role for this specific resource] (この特定のリソースのための新しいロールを作成する) を選択します。
14. [Additional settings] (追加設定) では、以下を実行します。

    a. ターゲット入力の設定では、ドロップダウンリストから[Input transformer] (インプットトランスフォーマー) を選択します。

    b. [Configure input transformer] (インプットトランスフォーマーの設定) を選択します。

    c. [Sample events] (イベント例) では、以下を入力します。

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

    d. [Target input transformer] (ターゲットインプットトランスフォーマー) では、以下を実行します。

        i. [Input Path] (入力パス) では、以下を入力します。

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

        ii. [Input Template] (入力テンプレート) では、以下を入力します。

```
{"message": <detail>}
```

e. [Confirm] (確認) を選択します。

15. [Next] (次へ) をクリックします。
16. [Next] (次へ) をクリックします。
17. ルールの詳細を確認し、[Create rule] (ルールの作成) を選択します。

ステップ 4: ルールをテストする

ルールをテストするには、EventBridge 対応バケットにファイルをアップロードして Amazon S3 オブジェクトを作成します。イベントがルールに一致すると、EventBridge は Zendesk Create Ticket API を呼び出します。Zendesk ダッシュボードに新しいチケットが表示されます。

ステップ 5: リソースをクリーンアップする

このチュートリアル用に作成したリソースは、保存を希望しない限り、すぐに削除できます。使用しなくなった AWS リソースを削除することで、AWS アカウントに請求される料金が発生しないようにできます。

EventBridge 接続を削除するには

1. Eventbridge コンソールの [API destination] (API 送信先) ページを開きます。
2. [Connections (接続)] タブを選択します。
3. 作成した接続を選択します。
4. [Delete] (削除) をクリックします。
5. 接続の名前を入力し、[Delete] (削除) を選択します。

EventBridge API の送信先を削除するには

1. Eventbridge コンソールの [API destination] (API 送信先) ページを開きます。
2. 作成した API の送信先を選択します。
3. [Delete] (削除) をクリックします。
4. API 送信先の名前を入力し、[Delete] (削除) を選択します。

EventBridge ルールを削除するには

1. Eventbridge コンソールの [Rules] (ルール) ページを開きます。
2. 作成したルールを選択します。
3. [Delete] (削除) をクリックします。
4. [Delete] (削除) を選択します。

AWS SDK EventBridge での の使用

AWS Software Development Kit (SDKsは、多くの一般的なプログラミング言語で使用できます。各 SDK には、デベロッパーが好みの言語でアプリケーションを簡単に構築できるようにする API、コード例、およびドキュメントが提供されています。

SDK ドキュメント	コード例
AWS SDK for C++	AWS SDK for C++ コード例
AWS CLI	AWS CLI コード例
AWS SDK for Go	AWS SDK for Go コード例
AWS SDK for Java	AWS SDK for Java コード例
AWS SDK for JavaScript	AWS SDK for JavaScript コード例
AWS SDK for Kotlin	AWS SDK for Kotlin コード例
AWS SDK for .NET	AWS SDK for .NET コード例
AWS SDK for PHP	AWS SDK for PHP コード例
AWS Tools for PowerShell	PowerShell コード例のツール
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) コード例
AWS SDK for Ruby	AWS SDK for Ruby コード例
AWS SDK for Rust	AWS SDK for Rust コード例
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コード例
AWS SDK for Swift	AWS SDK for Swift コード例

に固有の例については、 EventBridge「」を参照してくださいAWS SDKs EventBridge を使用するためのコード例。

可用性の例

必要なものが見つからなかった場合。このページの下側にある [Provide feedback (フィードバックを送信)] リンクから、コードの例をリクエストしてください。

AWS SDKs EventBridge を使用するためのコード例

次のコード例は、 Software AWS Development Kit (SDK) EventBridge で を使用する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

クロスサービスの例は、複数の AWS のサービスで動作するサンプルアプリケーションです。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK EventBridge での の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

開始方法

こんにち EventBridgeは

次のコード例は、 の使用を開始する方法を示しています EventBridge。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、「 API リファレンスListEventBuses」の「」を参照してください。AWS SDK for .NET

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * 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 の詳細については、「 API リファレンスListEventBuses」の「」を参照してください。AWS SDK for Java 2.x

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Kotlin API リファレンスListEventBusesの「」を参照してください。

コードの例

AWS SDKs EventBridge を使用するためのアクション

次のコード例は、 AWS SDKsで個々の EventBridgeアクションを実行する方法を示しています。これらの抜粋は EventBridge API を呼び出し、コンテキスト内で実行する必要がある大規模なプログラムからのコードの抜粋です。各例には GitHub、コードの設定と実行の手順を示す へのリンクが含まれています。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細なリストについては、「Amazon EventBridge API リファレンス」を参照してください。

例

- AWS SDK または CLI EnableRuleで を使用する
- AWS SDK または CLI ListRuleNamesByTargetで を使用する
- AWS SDK または CLI ListRulesで を使用する
- AWS SDK または CLI ListTargetsByRuleで を使用する
- AWS SDK または CLI PutEventsで を使用する
- AWS SDK または CLI PutRuleで を使用する
- AWS SDK または CLI PutTargetsで を使用する
- AWS SDK または CLI RemoveTargetsで を使用する

AWS SDK または CLI **DeleteRule**で を使用する

以下のコード例は、DeleteRule の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ルールとターゲットの使用開始

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

その名前でルールを削除します。

```
/// <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 の詳細については、「 API リファレンスDeleteRule」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

CloudWatch イベントルールを削除するには

この例では、EC2InstanceStateChanges という名前のルールを削除します。

```
aws events delete-rule --name "EC2InstanceStateChanges"
```

- API の詳細については、「 コマンドリファレンスDeleteRule」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    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 の詳細については、「 API リファレンスDeleteRule」の「」を参照してください。AWS SDK for Java 2.x

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 DeleteRule AWS SDK for Kotlin API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK EventBridge での の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeRule**で を使用する

以下のコード例は、DescribeRule の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ルールとターゲットの使用開始

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

ルールの説明を使用してルールの状態を取得します。

```
    /// <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 の詳細については、「 API リファレンスDescribeRule」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

CloudWatch イベントルールに関する情報を表示するには

この例では、 という名前のルールに関する情報を表示します DailyLambdaFunction。

```
aws events describe-rule --name "DailyLambdaFunction"
```

- API の詳細については、「 コマンドリファレンスDescribeRule」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   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 の詳細については、「 API リファレンスDescribeRule」の「」を参照してください。 AWS SDK for Java 2.x

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Kotlin API リファレンスDescribeRuleの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK EventBridge での の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DisableRule**で を使用する

以下のコード例は、DisableRule の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ルールとターゲットの使用開始

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

そのルール名でルールを無効化します。

```
/// <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 の詳細については、「 API リファレンスDisableRule」の「」を参照してください。AWS SDK for .NET

CLI

AWS CLI

CloudWatch イベントルールを無効にするには

この例では、 という名前のルールを無効にします DailyLambdaFunction。ルールは削除されません。

```
aws events disable-rule --name "DailyLambdaFunction"
```

- API の詳細については、「 コマンドリファレンスDisableRule」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

そのルール名を使用してルールを無効化します。

```
   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 の詳細については、「 API リファレンスDisableRule」の「」を参照してください。 AWS SDK for Java 2.x

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Kotlin API リファレンスDisableRuleの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK EventBridge での の使用。このトピックには、使用開始方法に関する情報と、以前のSDK バージョンの詳細も含まれています。

AWS SDK または CLI **EnableRule**で を使用する

以下のコード例は、EnableRule の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ルールとターゲットの使用開始

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

そのルール名でルールを有効化します。

```
/// <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 の詳細については、「 API リファレンスEnableRule」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

CloudWatch イベントルールを有効にするには

この例では DailyLambdaFunction、以前に無効にされていた という名前のルールを有効にします。

```
aws events enable-rule --name "DailyLambdaFunction"
```

- API の詳細については、「 コマンドリファレンスEnableRule」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

そのルール名を使用してルールを有効化します。

```
   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 の詳細については、「 API リファレンスEnableRule」の「」を参照してください。 AWS SDK for Java 2.x

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 EnableRule AWS SDK for Kotlin API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK EventBridge での の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI `ListRuleNamesByTarget` で を使用する

以下のコード例は、`ListRuleNamesByTarget` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ルールとターゲットの使用開始

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

ターゲットを使用してすべてのルール名を一覧表示します。

```
    /// <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 の詳細については、「 API リファレンスListRuleNamesByTarget」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

ターゲットが指定されているルールをすべて表示するには

この例では、「」という名前の Lambda 関数をターゲットMyFunctionNameとするすべてのルールを表示します。

```
aws events list-rule-names-by-target --target-arn "arn:aws:lambda:us-
east-1:123456789012:function:MyFunctionName"
```

- API の詳細については、「 コマンドリファレンスListRuleNamesByTarget」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

ターゲットを使用してすべてのルール名を一覧表示します。

```
   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 の詳細については、「 API リファレンスListRuleNamesByTarget」の「」を参照してください。 AWS SDK for Java 2.x

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Kotlin API リファレンスListRuleNamesByTargetの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK EventBridge での の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListRules**で を使用する

以下のコード例は、ListRules の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ルールとターゲットの使用開始

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

イベントバスのルールをすべて一覧表示します。

```
    /// <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 の詳細については、「 API リファレンスListRules」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

すべての CloudWatch イベントルールのリストを表示するには

この例では、リージョン内のすべての CloudWatch イベントルールを表示します。

```
aws events list-rules
```

特定の文字列で始まる CloudWatch イベントルールのリストを表示するには。

この例では、名前が「毎日」で始まるリージョン内のすべての CloudWatch イベントルールを表示します。

```
aws events list-rules --name-prefix "Daily"
```

- API の詳細については、「 コマンドリファレンスListRules」の「」を参照してください。AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

そのルール名を使用してルールを有効化します。

```
    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 の詳細については、「 API リファレンスListRules」の「」を参照してください。 AWS SDK for Java 2.x

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Kotlin API リファレンスListRulesの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK EventBridge での の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListTargetsByRule**で を使用する

以下のコード例は、ListTargetsByRule の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ルールとターゲットの使用開始

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

ルール名を使用してルールのすべてのターゲットを一覧表示します。

```
/// <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 の詳細については、「 API リファレンスListTargetsByRule」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

CloudWatch イベントルールのすべてのターゲットを表示するには

この例では、 という名前のルールのすべてのターゲットを表示します DailyLambdaFunction。

```
aws events list-targets-by-rule --rule  "DailyLambdaFunction"
```

- API の詳細については、「 コマンドリファレンスListTargetsByRule」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

ルール名を使用してルールのすべてのターゲットを一覧表示します。

```
   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 の詳細については、「 API リファレンスListTargetsByRule」の「」を参照してください。 AWS SDK for Java 2.x

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Kotlin API リファレンスListTargetsByRuleの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK EventBridge での の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **PutEvents**で を使用する

以下のコード例は、PutEvents の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ルールを作成してトリガーする
- ルールとターゲットの使用開始

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

ルールのカスタムパターンに一致するイベントを送信します。

```
    /// <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 の詳細については、「 API リファレンスPutEvents」の「」を参照してください。 AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

```
#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>
```

イベントを送信します。

```
        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 の詳細については、「 API リファレンスPutEvents」の「」を参照してください。 AWS SDK for C++

CLI

AWS CLI

カスタムイベントを CloudWatch イベントに送信するには

この例では、カスタムイベントを CloudWatch Events に送信します。このイベントは putevents.json ファイルに含まれています。

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

putevents.json ファイルの内容は次のとおりです。

```
[
  {
    "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 の詳細については、「 コマンドリファレンスPutEvents」の「」を参照してください。AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   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 の詳細については、「 API リファレンスPutEvents」の「」を参照してください。AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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 の詳細については、「 API リファレンスPutEvents」の「」を参照してください。 AWS SDK for JavaScript

SDK for JavaScript (v2)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// 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 の詳細については、「 API リファレンスPutEvents」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Kotlin API リファレンスPutEventsの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK EventBridge での の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **PutRule**で を使用する

以下のコード例は、PutRule の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ルールを作成してトリガーする
- ルールとターゲットの使用開始

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

Amazon Simple Storage Service バケットにオブジェクトが追加されたときにトリガーするルールを作成します。

```
    /// <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;
    }
```

カスタムパターンを使用するルールを作成します。

```
    /// <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 の詳細については、「 API リファレンスPutRule」の「」を参照してください。 AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

```
#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>
```

ルールを作成します。

```
        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 の詳細については、「 API リファレンスPutRule」の「」を参照してください。 AWS SDK for C++

CLI

AWS CLI

CloudWatch イベントルールを作成するには

この例は、毎日午前 9:00 (UTC) にトリガーされるルールを作成します。put-targets を使用して Lambda 関数をこのルールのターゲットとして追加すると、指定した時刻に Lambda 関数を毎日実行できます。

```
aws events put-rule --name "DailyLambdaFunction" --schedule-expression "cron(0 9
 * * ? *)"
```

この例は、リージョン内の任意の EC2 インスタンスの状態が変わったときにトリガーされるルールを作成します。

```
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"
```

この例は、リージョン内の任意の EC2 インスタンスが停止または終了したときにトリガーされるルールを作成します。

```
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 の詳細については、「 コマンドリファレンスPutRule」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

スケジュールルールを作成します。

```
   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);
       }
   }
```

Amazon Simple Storage Service バケットにオブジェクトが追加されたときにトリガーするルールを作成します。

```
   // 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 の詳細については、「 API リファレンスPutRule」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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 の詳細については、「 API リファレンスPutRule」の「」を参照してください。 AWS SDK for JavaScript

SDK for JavaScript (v2)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// 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 の詳細については、「 API リファレンスPutRule」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

スケジュールルールを作成します。

```
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}")
    }
}
```

Amazon Simple Storage Service バケットにオブジェクトが追加されたときにトリガーするルールを作成します。

```
// 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 の詳細については、 AWS SDK for Kotlin API リファレンスPutRuleの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK EventBridge での の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **PutTargets**で を使用する

以下のコード例は、PutTargets の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ルールとターゲットの使用開始

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

Amazon SNS トピックをルールのターゲットとして追加します。

```
    /// <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;
    }
```

ルールのターゲットにインプットトランスフォーマーを追加します。

```
    /// <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 の詳細については、「 API リファレンスPutTargets」の「」を参照してください。 AWS SDK for .NET

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

```
#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>
```

ターゲットとして追加します。

```
        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 の詳細については、「 API リファレンスPutTargets」の「」を参照してください。 AWS SDK for C++

CLI

AWS CLI

CloudWatch イベントルールのターゲットを追加するには

この例は、ルールのターゲットとして Lambda 関数を追加します。

```
aws events put-targets --rule DailyLambdaFunction --targets
 "Id"="1","Arn"="arn:aws:lambda:us-east-1:123456789012:function:MyFunctionName"
```

この例は、Amazon Kinesis ストリームをターゲットとして設定し、このルールによって捕捉されたイベントがストリームに中継されるようにします。

```
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"
```

この例は、2 つの Amazon Kinesis ストリームを 1 つのルールのターゲットとして設定します。

```
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 の詳細については、「 コマンドリファレンスPutTargets」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

Amazon SNS トピックをルールのターゲットとして追加します。

```
    // 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 + ".");
    }
```

ルールのターゲットにインプットトランスフォーマーを追加します。

```
    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 の詳細については、「 API リファレンスPutTargets」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SDK モジュールとクライアントモジュールをインポートし、API を呼び出します。

```
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 の詳細については、「 API リファレンスPutTargets」の「」を参照してください。 AWS SDK for JavaScript

SDK for JavaScript (v2)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// 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 の詳細については、「 API リファレンスPutTargets」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
// 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.")
    }
}
```

ルールのターゲットにインプットトランスフォーマーを追加します。

```
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 の詳細については、 AWS SDK for Kotlin API リファレンスPutTargetsの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK EventBridge での の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **RemoveTargets**で を使用する

以下のコード例は、RemoveTargets の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

ルール名を使用してルールのすべてのターゲットを削除します。

```
    /// <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 の詳細については、「 API リファレンスRemoveTargets」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

イベントのターゲットを削除するには

この例では、 MyStream1 という名前の Amazon Kinesis ストリームをルール のターゲットから削除します DailyLambdaFunction。 DailyLambdaFunction が作成されると、このストリームはターゲット 1 の ID を持つTarget1として設定されました。

```
aws events remove-targets --rule "DailyLambdaFunction" --ids "Target1"
```

- API の詳細については、「 コマンドリファレンスRemoveTargets」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

ルール名を使用してルールのすべてのターゲットを削除します。

```
   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 の詳細については、「 API リファレンスRemoveTargets」の「」を参照してください。 AWS SDK for Java 2.x

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Kotlin API リファレンスRemoveTargetsの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK EventBridge での の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDKs EventBridge を使用するシナリオ

次のコード例は、 AWS SDKs を使用して で EventBridge一般的なシナリオを実装する方法を示しています。これらのシナリオは、 内で複数の関数を呼び出して特定のタスクを実行する方法を示しています EventBridge。各シナリオには GitHub、コードのセットアップと実行の手順を示す へのリンクが含まれています。

例

- AWS SDK EventBridge を使用して Amazon でルールを作成してトリガーする
- AWS SDK を使用して EventBridge ルールとターゲットの使用を開始する

AWS SDK EventBridge を使用して Amazon でルールを作成してトリガーする

次のコード例は、Amazon でルールを作成してトリガーする方法を示しています EventBridge。

Ruby

SDK for Ruby

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

関数を正しい順序で呼び出します。

```
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"
```

指定された Amazon Simple Notification Service (Amazon SNS) トピックがこの関数に提供されているトピックの中に存在するかどうかをチェックします。

```
# 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 SNS で発信者が利用できるトピックの中に指定されたトピックが存在するかどうかをチェックします。

```
# 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 SNS でトピックを作成し、E メールアドレスをサブスクライブして、そのトピックに対する通知を受信します。

```
# 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
```

指定された AWS Identity and Access Management (IAM) ロールが、この関数に提供されるロールの中に存在するかどうかを確認します。

```
# 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
```

IAM で発信者が利用できるロールの中に指定されたロールが存在するかどうかをチェックします。

```
# 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
```

IAM でロールを作成します。

```
# 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
```

指定された EventBridge ルールが、この関数に提供されるルールの中に存在するかどうかを確認します。

```
# 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
```

指定されたルールが、 で発信者が利用できるルールの中に存在するかどうかを確認します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
```

でルールを作成します EventBridge。

```
# Creates a rule in Amazon EventBridge.
# This rule is triggered whenever an available instance in
# Amazon EC2 changes to the specified state.
# This rule is designed to be used specifically by this code example.
#
# Prerequisites:
#
# - A role in AWS Identity and Access Management (IAM) that is designed
#   to be used specifically by this code example.
# - A topic in Amazon SNS.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to create.
# @param rule_description [String] Some description for this rule.
# @param instance_state [String] The state that available instances in
#   Amazon EC2 must change to, to
#   trigger this rule.
# @param role_arn [String] The Amazon Resource Name (ARN) of the IAM role.
# @param target_id [String] Some identifying string for the rule's target.
# @param topic_arn [String] The ARN of the Amazon SNS topic.
# @return [Boolean] true if the rule was created; otherwise, false.
# @example
#   exit 1 unless rule_created?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     'Triggers when any available EC2 instance starts.',
#     'running',
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change',
#     'sns-topic',
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def rule_created?(
  cloudwatchevents_client,
```

```
  rule_name,
  rule_description,
  instance_state,
  role_arn,
  target_id,
  topic_arn
)
  puts "Creating rule with name '#{rule_name}'..."
  put_rule_response = cloudwatchevents_client.put_rule(
    name: rule_name,
    description: rule_description,
    event_pattern: {
      'source': [
        "aws.ec2"
      ],
      'detail-type': [
        "EC2 Instance State-change Notification"
      ],
      'detail': {
        'state': [
          instance_state
        ]
      }
    }.to_json,
    state: "ENABLED",
    role_arn: role_arn
  )
  puts "Rule created with ARN '#{put_rule_response.rule_arn}'."

  put_targets_response = cloudwatchevents_client.put_targets(
    rule: rule_name,
    targets: [
      {
        id: target_id,
        arn: topic_arn
      }
    ]
  )
  if put_targets_response.key?(:failed_entry_count) &&
      put_targets_response.failed_entry_count > 0
    puts "Error(s) adding target to rule:"
    put_targets_response.failed_entries.each do |failure|
      puts failure.error_message
    end
```

```
    return false
  else
    return true
  end
rescue StandardError => e
  puts "Error creating rule or adding target to rule: #{e.message}"
  puts "If the rule was created, you must add the target " \
    "to the rule yourself, or delete the rule yourself and try again."
  return false
end
```

指定されたロググループが、Amazon CloudWatch Logs で発信者が利用できるロググループの中に存在するかどうかを確認します。

```
# 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
```

Logs で CloudWatch ロググループを作成します。

```
# 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
```

CloudWatch Logs でログストリームにイベントを書き込みます。

```
# 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) インスタンスを再起動し、関連するアクティビティに関する情報を CloudWatch Logs のログストリームに追加します。

```
# 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
```

でルールのアクティビティに関する情報を表示します EventBridge。

```
# Displays information about activity for a rule in Amazon EventBridge.
#
# Prerequisites:
#
# - A rule in Amazon EventBridge.
#
# @param cloudwatch_client [Amazon::CloudWatch::Client] An initialized
#   Amazon CloudWatch client.
# @param rule_name [String] The name of the rule.
# @param start_time [Time] The timestamp that determines the first datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param end_time [Time] The timestamp that determines the last datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param period [Integer] The interval, in seconds, to check for activity.
# @example
#   display_rule_activity(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     Time.now - 600, # Start checking from 10 minutes ago.
#     Time.now, # Check up until now.
#     60 # Check every minute during those 10 minutes.
#   )
def display_rule_activity(
  cloudwatch_client,
  rule_name,
  start_time,
  end_time,
  period
)
  puts "Attempting to display rule activity..."
  response = cloudwatch_client.get_metric_statistics(
    namespace: "AWS/Events",
    metric_name: "Invocations",
    dimensions: [
      {
        name: "RuleName",
        value: rule_name
      }
    ],
    start_time: start_time,
    end_time: end_time,
    period: period,
    statistics: ["Sum"],
```

```
    unit: "Count"
  )

  if response.key?(:datapoints) && response.datapoints.count.positive?
    puts "The event rule '#{rule_name}' was triggered:"
    response.datapoints.each do |datapoint|
      puts "  #{datapoint.sum} time(s) at #{datapoint.timestamp}"
    end
  else
    puts "The event rule '#{rule_name}' was not triggered during the " \
      "specified time period."
  end
rescue StandardError => e
  puts "Error getting information about event rule activity: #{e.message}"
end
```

Logs ロググループ内のすべてのログストリームの CloudWatch ログ情報を表示します。

```
# 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
```

発信者にリマインダーを表示して、不要になった関連 AWS リソースを手動でクリーンアップします。

```
# 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 の詳細については、『AWS SDK for Ruby API リファレンス』の以下のトピックを参照してください。
  - PutEvents
  - PutRule

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK EventBridge での の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して EventBridge ルールとターゲットの使用を開始する

次のコード例は、以下を実行する方法を示しています。

- ルールを作成して、ターゲットを追加する。
- ルールを有効化および無効化する。
- ルールとターゲットを一覧表示して更新する。
- イベントを送信して、リソースをクリーンアップする。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
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 オペレーションをラップするクラスを作成します。

```
/// <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 の詳細については、『AWS SDK for .NET API リファレンス』の以下のトピックを参照してください。
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * 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 の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/*
 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 の詳細については、『AWS SDK for Kotlin API リファレンス』の以下のトピックを参照してください。
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK EventBridge での の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDKs EventBridge を使用するためのクロスサービスの例

次のサンプルアプリケーションでは、 AWS SDKs を使用して他の と組み合わせ EventBridgeます AWS のサービス。各例には GitHub、アプリケーションのセットアップと実行の手順を示す へのリンクが含まれています。

例

- スケジュールされたイベントを使用した Lambda 関数の呼び出し

スケジュールされたイベントを使用した Lambda 関数の呼び出し

次のコード例は、Amazon EventBridge のスケジュールされたイベントによって呼び出される AWS Lambda 関数を作成する方法を示しています。

Java

SDK for Java 2.x

AWS Lambda 関数を呼び出す Amazon EventBridge スケジュールされたイベントを作成する方法を示します。cron 式 EventBridge を使用して Lambda 関数が呼び出されるタイミングをスケジュールするように を設定します。この例では、Lambda Java ランタイム API を使用して Lambda 関数を作成します。この例では、さまざまな AWS サービスを呼び出して、特定のユースケースを実行します。この例では、年間の記念日に従業員を祝福するモバイルテキストメッセージを従業員に送信するアプリを作成する方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK for JavaScript (v3)

AWS Lambda 関数を呼び出す Amazon EventBridge スケジュールされたイベントを作成する方法を示します。cron 式 EventBridge を使用して Lambda 関数が呼び出されるタイミングをスケジュールするように を設定します。この例では、Lambda JavaScript ランタイム API を使用して Lambda 関数を作成します。この例では、さまざまな AWS サービスを呼び出して、特定のユースケースを実行します。この例では、年間の記念日に従業員を祝福するモバイルテキストメッセージを従業員に送信するアプリを作成する方法を示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例は、AWS SDK for JavaScript v3 デベロッパーガイドでも使用できます。

この例で使用されているサービス

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK for Python (Boto3)

この例では、スケジュールされた Amazon EventBridge イベントのターゲットとして AWS Lambda 関数を登録する方法を示します。Lambda ハンドラーは、わかりやすいメッセージと完全なイベントデータを Amazon CloudWatch Logs に書き込み、後で取得できるようにします。

- Lambda 関数をデプロイします。
- EventBridge スケジュールされたイベントを作成し、Lambda 関数をターゲットにします。
- Lambda 関数 EventBridge を呼び出すアクセス許可を に付与します。
- CloudWatch ログから最新のデータを出力して、スケジュールされた呼び出しの結果を表示します。
- デモ中に作成されたすべてのリソースをクリーンアップします。

この例は、 で表示するのが最適です GitHub。完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- CloudWatch ログ
- EventBridge
- Lambda

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK EventBridge での の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon EventBridge セキュリティ

Amazon EventBridge は AWS Identity and Access Management を使用して、他の AWS サービスやリソースへのアクセスを制御します。IAM の仕組みの概要については、「IAM ユーザーガイド」の「アクセス管理の概要」を参照してください。セキュリティ認証情報の概要については、「Amazon Web Services 全般のリファレンス」の「AWS セキュリティ認証情報」を参照してください。

トピック

Amazon でのデータ保護 EventBridge

責任 AWS 共有モデル、 でのデータ保護に適用されます Amazon EventBridge。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。お客様は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。また、使用する AWS のサービス のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「データプライバシーのよくある質問」を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された記事「AWS 責任共有モデルおよび GDPR」を参照してください。

データ保護の目的で、 認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須であり TLS 1.3 がお勧めです。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、コンソール、API、EventBridge または AWS のサービス SDK を使用して AWS CLIまたは他の を操作する場合も同様です。 AWS SDKs 名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

EventBridge イベントバスのデータ暗号化

EventBridge は、イベントデータを保護するために、保管時の暗号化と転送中の暗号化の両方を提供します。

- 保管中の暗号化

  EventBridge は AWS Key Management Service (KMS) と統合して、イベントバスに保存されているイベントデータを暗号化します。デフォルトでは、 EventBridge を使用してイベントデータを AWS 所有のキー 暗号化します。カスタムイベントとパートナーイベント カスタマー管理キー に を使用する EventBridge ように を指定することもできます。

- 転送中の暗号化

  EventBridge は、Transport Layer Security (TLS) を使用して EventBridge 、 と他の のサービスの間を通過するデータを暗号化します。イベントバスの場合、これには、イベントが に送信される間、 EventBridgeおよび がルールターゲットにイベント EventBridge を送信するときが含まれます。

イベントバスの保管時の暗号化

EventBridge は、 AWS Key Management Service (KMS) と統合することで、透過的なサーバー側の暗号化を提供します。保管中のデータをデフォルトで暗号化することで、機密データの保護におけるオーバーヘッドと複雑な作業を減らすのに役立ちます。同時に、セキュリティを重視したアプリケーションを構築して、暗号化のコンプライアンスと規制の厳格な要件を満たすことができます。

保管中のイベントバスデータの EventBridge 暗号化には以下が含まれます。

- AWS、カスタム 、およびパートナーイベントのイベントデータ。

  イベントバスの場合、イベントデータにはイベントの ???要素に含まれるすべてのフィールドが含まれます。

  EventBridge はイベントメタデータを暗号化しません。イベントメタデータの詳細については、「」を参照してください???。

- イベントパターン
- 入力トランスフォーマー

デフォルトでは、 EventBridge を使用してイベントデータを AWS 所有のキー 暗号化します。カスタムイベントとパートナーイベントに カスタマー管理キー を使用する EventBridge ように を指定することもできます。

イベントバス暗号化のセキュリティ上の考慮事項

機密情報や機密情報は保管時に暗号化されないため、以下のフィールドには入力しないことを強くお勧めします。

- イベントバス名
- ルール名
- タグでの などの共有リソース

KMS key イベントバス暗号化のオプション

EventBridge は AWS 所有のキー を使用して、イベントバスに保存されている AWS サービスイベントを暗号化します。

イベントバスごとに、そのバスに保存されているカスタムイベントとパートナーイベントを暗号化 EventBridge するために使用するタイプ KMS key を選択できます。

- AWS 所有のキー

  デフォルトでは、 は で 256 ビットの Advanced Encryption Standard (AES-256) を使用してデータを EventBridge 暗号化します。これにより AWS 所有のキー、不正アクセスからデータを保護できます。

  を表示、管理、使用したり AWS 所有のキー、その使用を監査したりすることはできません。ただし、データを暗号化するキーを保護するためのアクションの実施やプログラムの変更を行う必要はありません。

  一般に、 リソースを保護する暗号化キーを監査または制御する必要がない限り、 は適切な選択肢 AWS 所有のキー です。 AWS 所有のキー は完全に無料 (月額料金や使用料なし) で、アカウントの AWS KMS クォータにはカウントされません。キーまたはそのキーポリシーを作成または管理する必要はありません。

  詳細については、AWS Key Management Service デベロッパーガイドの「AWS 所有キー」を参照してください。
- カスタマー管理キー

  EventBridge は、 カスタマー管理キー ユーザーが作成、所有、管理する対称 の使用をサポートします。このタイプの を完全に制御できるため KMS key、次のようなタスクを実行できます。

  - キーポリシーの策定と維持

- IAM ポリシーとグラントの策定と維持
- キーポリシーの有効化と無効化
- 暗号化素材のローテーション
- タグの追加
- キーエイリアスの作成
- 削除のためのキースケジューリング

詳細については、「AWS Key Management Service デベロッパーガイド」の 「カスタマーマネージドキー」を参照してください。

EventBridge は、マルチリージョンキー とキー のクロスアカウントアクセスをサポートします。

カスタマーマネージドキー には月額料金が発生します。詳細については、「 AWS Key Management Service デベロッパーガイド」の「 のAWS Key Management Service 料金」、「クォータ」を参照してください。

> **Note**
>
> EventBridge は、 を使用して暗号化されたイベントバスで以下の機能をサポートしていません カスタマーマネージドキー。
>
> - アーカイブ
> - スキーマ検出

詳細については、「???」を参照してください。

によるイベントの暗号化 カスタマーマネージドキー

をデフォルト AWS 所有のキー として使用するのではなく、 EventBridge を使用して AWS KMS イベントバスに保存されているデータ (カスタムイベントとパートナーイベント) を カスタマー管理キー 暗号化するように指定できます。イベントバスを作成または更新 カスタマー管理キー するときに を指定できます。デフォルトのイベントバスを更新して、カスタムイベントとパートナーイベント カスタマー管理キー にも を使用することもできます。詳細については、「???」を参照してください。

イベントバス カスタマー管理キー に を指定する場合、イベントバスにデッドレターキュー (DLQ) を指定するオプションがあります。 EventBridge これにより、暗号化エラーまたは復号エラーを生

成するカスタムイベントまたはパートナーイベントがその DLQ に配信されます。詳細については、「???」を参照してください。

イベントバスの作成時の暗号化 カスタマー管理キー のための の指定 (コンソールを使用)

- 以下の手順に従います。

  ???.

イベントバスの作成時の暗号化 カスタマー管理キー のための の指定 (CLI を使用)

- を呼び出すときは`create-event-bus`、 `kms-key-identifier`オプションを使用して、イベントバスの暗号化 EventBridge に使用する カスタマー管理キー を指定します。

  必要に応じて、 `dead-letter-config`を使用してデッドレターキュー (DLQ) を指定します。

暗号化 カスタマー管理キー に を使用するようにイベントバスを更新する (コンソールを使用)

- 以下の手順に従います。

  ???.

暗号化 カスタマー管理キー に を使用するようにイベントバスを更新する (CLI を使用)

- を呼び出すときは`update-event-bus`、 `kms-key-identifier`オプションを使用して、イベントバスの暗号化 EventBridge に使用する カスタマー管理キー を指定します。

  必要に応じて、 `dead-letter-config`を使用してデッドレターキュー (DLQ) を指定します。

を使用した暗号化 カスタマー管理キー に を使用するようにデフォルトのイベントバスを更新する CloudFormation

はデフォルトのイベントバスを自動的にアカウントに EventBridge プロビジョニングするため、CloudFormation スタックに含めるリソースの場合と同様に、 CloudFormation テンプレートを使用して作成することはできません。 CloudFormation スタックにデフォルトのイベントバスを含めるには、まずスタックにインポートする必要があります。デフォルトのイベントバスをスタックにインポートしたら、必要に応じてイベントバスのプロパティを更新できます。

- 以下の手順に従います。

???.

の使用 EventBridge の許可 カスタマー管理キー

アカウント カスタマー管理キー で を使用して EventBridge イベントバスを保護する場合、 のポリシーは、ユーザーに代わってイベントバスを使用するアクセス EventBridge 許可を付与 KMS key する必要があります。これらのアクセス許可は、キーポリシー で指定します。

EventBridge は、 AWS アカウントの EventBridge リソースを保護するためにデフォルト を使用する AWS 所有のキー ための追加の認可を必要としません。

EventBridge には、 に対する次のアクセス許可が必要です カスタマーマネージドキー。

- kms:DescribeKey

  EventBridge は、指定されたキー ID の KMS key ARN を取得し、キーが対称であることを確認するために、このアクセス許可を必要とします。
- kms:GenerateDataKey

  EventBridge では、イベントデータの暗号化キーとしてデータキーを生成するために、このアクセス許可が必要です。
- kms:Decrypt

  EventBridge では、暗号化されたイベントデータで暗号化および保存されているデータキーを復号化するために、このアクセス許可が必要です。

  EventBridge はこれをルールマッチングに使用します。ユーザーはデータにアクセスできません。

次のキーポリシーの例は、必要なアクセス許可を提供します。

```
{
  "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 イベントバス暗号化 カスタマーマネージドキー に を使用する場合のセキュリティ

セキュリティのベストプラクティスとして、aws:SourceArn、、aws:sourceAccountまたは kms:EncryptionContext:aws:events:event-bus:arn条件キーを AWS KMS キーポリシーに追加します。 IAM グローバル条件キーは、 が指定されたバスまたはアカウントに対してのみ KMS キー EventBridge を使用するようにするのに役立ちます。

次の例は、 IAM ポリシーでこのベストプラクティスに従う方法を示しています。

```
{
      "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 イベントバス暗号化 カスタマーマネージドキー の管理

EventBridge が常に必要な へのアクセスを保持するようにするには カスタマー管理キー：

- で暗号化されたすべてのイベントが処理されたことを確認する カスタマー管理キー まで、 を削除しないでください。

  次のいずれかの操作を実行するときは、以前のキーマテリアルを保持して、 EventBridge が以前に暗号化されたイベントで引き続き使用できるようにします。

  - 自動キーローテーション
  - 手動キーローテーション
  - キーエイリアスの更新

  一般に、 AWS KMS キーの削除を検討している場合は、まずキーを無効にし、暗号化されたデータを復号するためにキーを使用する必要がないことを確認するようにCloudWatch アラームまたは同様のメカニズムを設定します。
- キーを使用するための EventBridge アクセス許可を提供するキーポリシーを削除しないでください。

その他の考慮事項は次のとおりです。

- 必要に応じて、ルールターゲット カスタマーマネージドキー に を指定します。

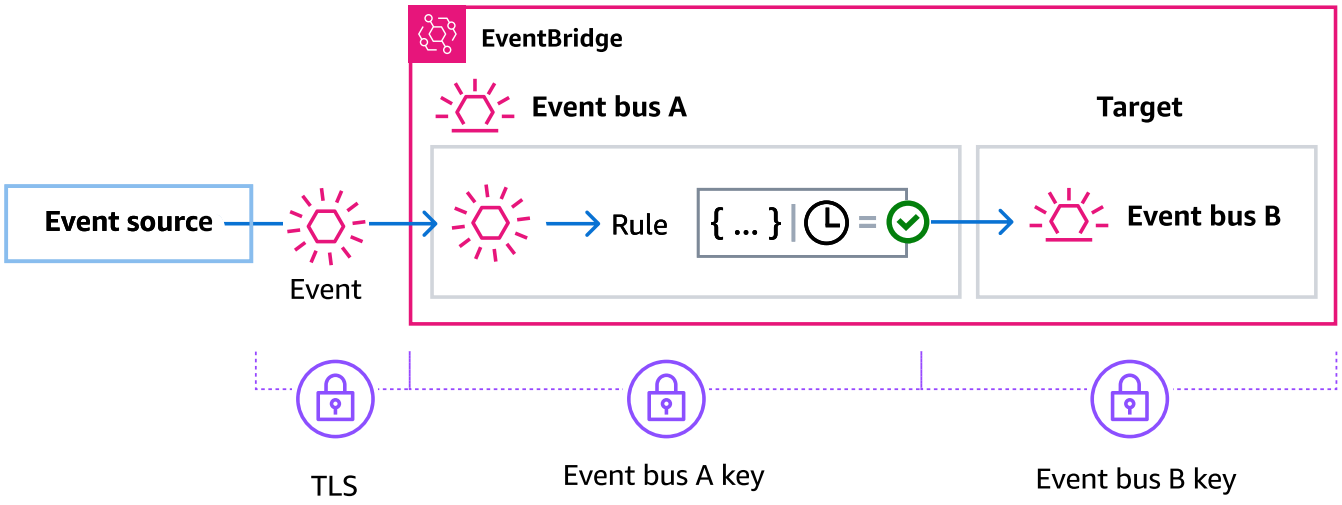
  がルールターゲットにイベント EventBridge を送信すると、イベントは Transport Layer Security (TLS) を使用して送信されます。ただし、ターゲットに保存されているイベントに適用される暗号化は、ターゲット自体で設定した暗号化によって異なります。

イベントバスがルールターゲットである場合のイベント暗号化

カスタムイベントまたはパートナーイベントがイベントバスに送信されると、 EventBridge はそのイベントバスの保管時の暗号化 KMS キー設定に従ってそのイベントを暗号化します。デフォルト AWS 所有のキー または カスタマー管理キーが指定されている場合は のいずれかです。イベントがルールと一致する場合、ルールターゲットが別のイベントバス でない限り、 はイベントがルールターゲットに送信されるまで、そのイベントバスの KMS キー設定でイベントを EventBridge 暗号化します。

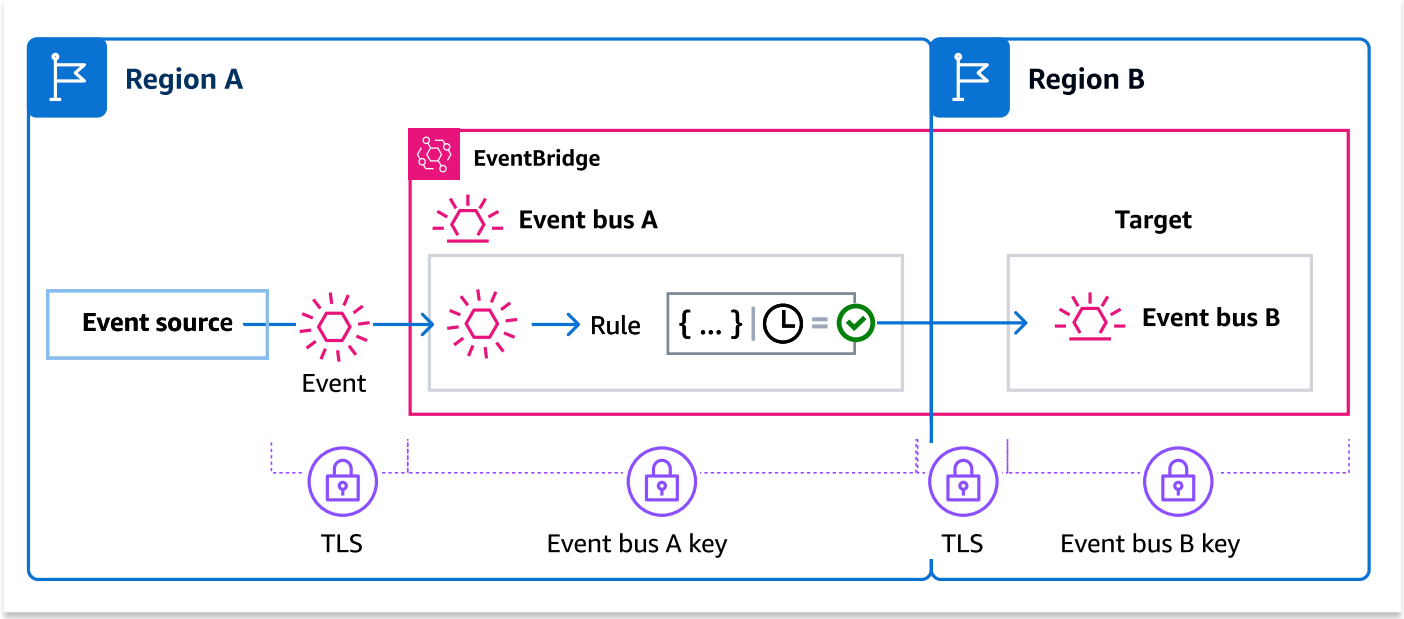
- ルールのターゲットが同じ AWS リージョン内の別のイベントバスである場合：

  ターゲットイベントバスに指定された がある場合 カスタマー管理キー、 は代わりにターゲットイベントバス カスタマー管理キー の でイベントを EventBridge 暗号化して配信します。



- ルールのターゲットが別の AWS リージョンの別のイベントバスである場合：

  EventBridge は、最初のイベントバスの KMS キー設定に従って保管中のイベントを暗号化します。 は TLS EventBridge を使用して、イベントを異なるリージョンの 2 番目のイベントバスに送信し、ターゲットイベントバスに指定された KMS キー設定に従って暗号化します。

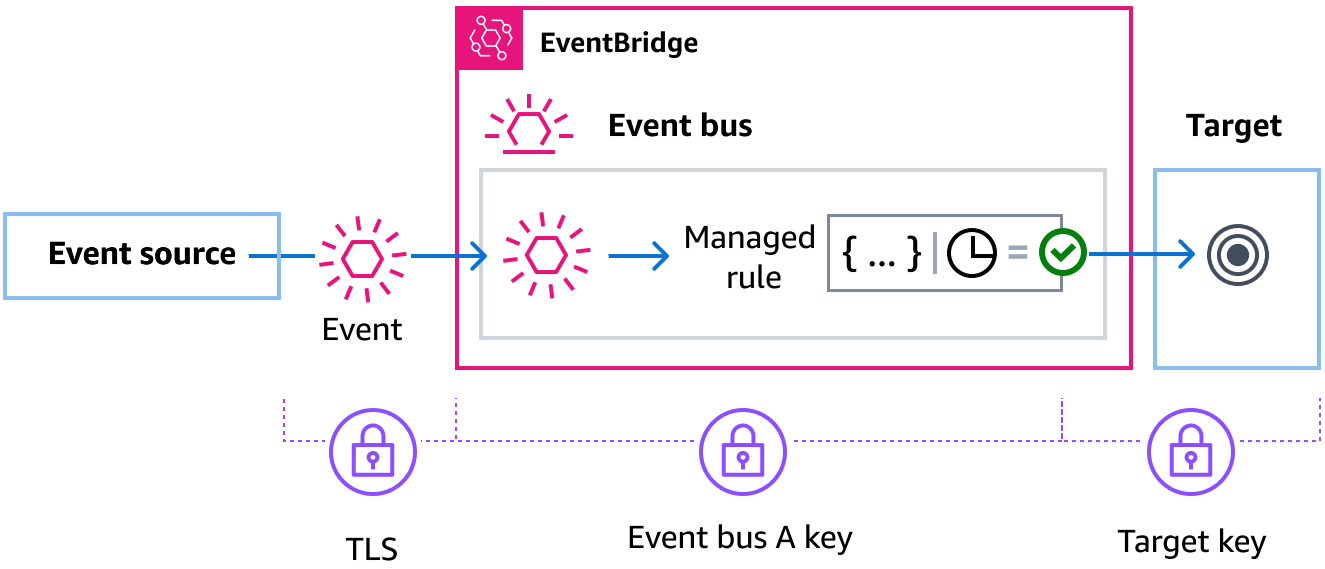


マネージドルールのイベント暗号化

AWS のサービスは、それらのサービスの特定の関数に必要なイベントバスルールを AWS アカウントに作成および管理できます。マネージドルールの一部として、 AWS サービスはルールターゲッ

トに カスタマー管理キー 指定された EventBridge を使用する を指定できます。これにより、ルールターゲットに基づいて カスタマー管理キー 使用する を柔軟に指定できます。

このような場合、カスタムイベントまたはパートナーイベントがマネージドルールと一致すると、はマネージドルールで カスタマー管理キー 指定されたターゲット EventBridge を使用して、ルールターゲットに送信されるまでイベントを暗号化します。これは、イベントバスが暗号化に独自の を使用するように設定されているかどうかにかかわらず カスタマー管理キー です。これは、マネージドルールのターゲットが別のイベントバスであり、そのイベントバスに暗号化用に独自の カスタマー管理キー が指定されている場合にも当てはまります。 は、イベントがイベントバスではないターゲットに送信されるまで、マネージドルールで カスタマー管理キー 指定されたターゲットの使用 EventBridge を続行します。



ルールターゲットが別のリージョンのイベントバスである場合は、マルチリージョンキー を指定する必要があります。最初のリージョンのイベントバスは、マネージドルールで指定された カスタマー管理キー を使用してイベントを暗号化します。次に、2 番目のリージョンのターゲットイベントバスにイベントを送信します。そのイベントバスは、イベントをターゲットに送信する カスタマー管理キー まで を引き続き使用できる必要があります。

EventBridge イベントバス暗号化コンテキスト

暗号化コンテキスト は、一連のキー値のペアおよび任意非シークレットデータを含みます。データを暗号化するリクエストに暗号化コンテキストを組み込むと、 AWS KMS は暗号化コンテキストを暗号化されたデータに暗号化してバインドします。データを復号するには、同じ暗号化コンテキストに渡す必要があります。

暗号化コンテキストは、ポリシーと許可の承認条件として使用することもできます。

イベントバスの場合、 はすべての暗号化オペレーションで同じ AWS KMS 暗号化コンテキスト EventBridge を使用します。カスタマーマネージドキーを使用して EventBridge リソースを保護する場合は、暗号化コンテキストを使用して、監査レコードとログ KMS key での の使用を特定できます。また、AWS CloudTrail や Amazon CloudWatch Logs などのログにもプレーンテキストで表示されます。

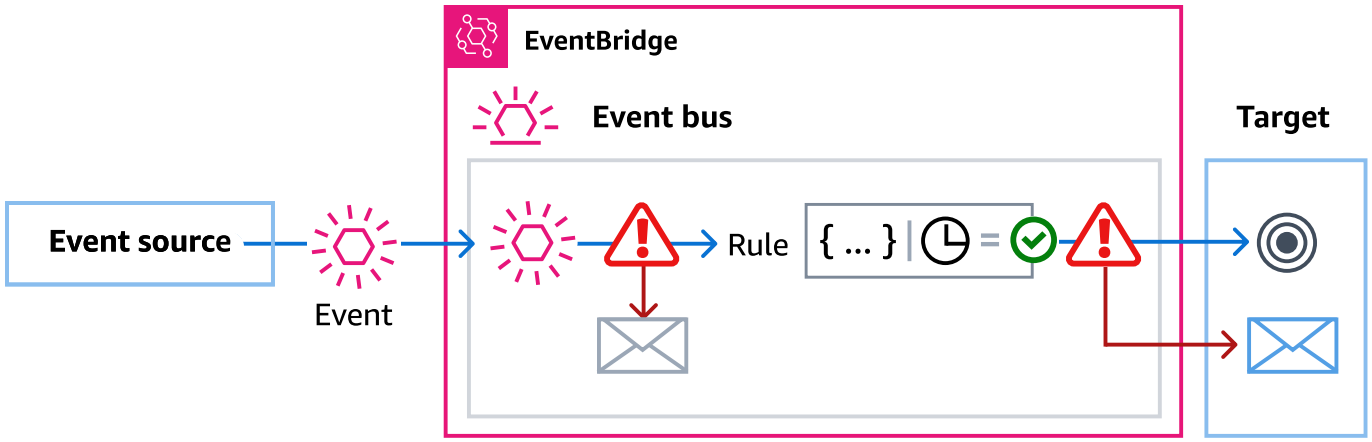
へのリクエストで AWS KMS、 はイベントバス ARN を含む単一のキーと値のペアを持つ暗号化コンテキスト EventBridge を使用します。

```
"encryptionContext": {
    "kms:EncryptionContext:aws:events:event-bus:arn": "event-bus-arn"
}
```

デッドレターキューを使用して暗号化されたイベントエラーをキャプチャする

イベントバスで カスタマー管理キー 暗号化を設定する場合は、そのイベントバスにデッドレターキュー (DLQ) を指定することをお勧めします。 は、イベントバスでイベントを処理するときに再試行不可能なエラーが発生した場合、カスタムイベントとパートナーイベントをこの DLQ EventBridge に送信します。再試行不可能なエラーとは、指定された が無効 カスタマー管理キー になっている、または欠落しているなど、根本的な問題を解決するためにユーザーアクションが必要なエラーです。

- EventBridge がイベントバスでイベントを処理しているときに、取得不可能な暗号化または復号エラーが発生した場合、イベントはイベントバス の DLQ に送信されます。
- EventBridge がターゲットにイベントを送信しようとしたときに、取得不可能な暗号化または復号エラーが発生した場合、イベントはターゲット の DLQ に送信されます。



DLQs「」を参照してください???。

EventBridge デッドレターキューでのイベントの復号化

再試行不可能なエラーの原因となっている根本的な問題を解決したら、イベントバスまたはターゲット DLQsに送信されたイベントを処理できます。暗号化されたイベントの場合は、まずイベントを復号化して処理する必要があります。

次の例は、 がイベントバスまたはターゲット DLQ に配信 EventBridge したイベントを復号する方法を示しています。

```
// 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);
```

タグベースのポリシー

Amazon EventBridge では、タグベースのポリシーを使用して、リソースへのアクセスを制御できます。

たとえば、キー environment および値 production のタグを含むすべてのリソースへのアクセスを制限できます。以下のポリシー例では、このタグを持つリソースが、environment/production というタグが付けられたリソースのタグ、ルール、イベントバスを作成、削除、または変更することを拒否します。

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

タグ付けの詳細については、以下を参照してください。

- Amazon EventBridge タグ
- IAM タグを使用したアクセスの制御

Amazon EventBridge と AWS Identity and Access Management

Amazon にアクセスするには EventBridge、 AWSがリクエストの認証に使用できる認証情報が必要です。認証情報には、他の AWS リソースからのイベントデータの取得などの AWS リソースへのアクセス権限が必要です。以下のセクションでは、 AWS Identity and Access Management (IAM) と を使用して、リソースにアクセスできるユーザーを制御することでリソース EventBridge を保護する方法について詳しく説明します。

トピック

認証

AWS には、次のタイプのアイデンティティでアクセスできます。

- AWS アカウントのルートユーザー – AWS にサインアップするときは、アカウントに関連付けられた E メールアドレスとパスワードを指定します。これらはルート認証情報であり、これらの情報を使用すると、すべての AWS リソースへの完全なアクセスが可能になります。

> **Important**
>
> セキュリティ上の理由から、アカウントへの完全な許可を持つ管理者 (IAM ユーザー) を作成するためにのみ、ルート認証情報を使用することをお勧めします。その後、この管理者を使用して、制限されたアクセス権限を持つ他の ユーザーとロールを作成できます。詳細については、「IAM User Guide」(IAM ユーザーガイド) で「IAM Best Practices」(IAM ベ

ストプラクティス) および「Creating an Admin User and Group」(管理者のユーザーおよびグループの作成) を参照してください。

- IAM ユーザー – IAM ユーザーは、 のターゲットにイベントデータを送信するアクセス許可など、特定のアクセス許可を持つ アカウント内のアイデンティティです EventBridge。IAM のサインイン認証情報を使用したサインインにより、AWS Management Console、AWS ディスカッションフォーラム、AWS Support センターなどの AWS ウェブページを保護できます。

  サインイン認証情報に加えて、ユーザーごとにアクセスキーを生成することもできます。複数の SDK の 1 つを通して、または AWS Command Line Interface (AWS CLI) を使用し、プログラムで AWS サービスにアクセスして暗号でリクエストに署名するときに、これらのキーを使用します。AWS ツールを使用しない場合は、リクエストを署名バージョン 4 で署名する必要があります。これは、インバウンド API リクエストを認証するためのプロトコルです。リクエストの認証の詳細については、『』の「署名バージョン 4 の署名プロセスAmazon Web Services 全般のリファレンス」を参照してください。

- IAM ロール - IAM ロールは、アカウントで作成して特定のアクセス許可を付与できるもうひとつの IAM アイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーに関連付けられていません。IAM ロールを使用すると、AWS サービスおよびリソースにアクセスできる一時的なアクセスキーを取得することができます。IAM ロールと一時的な認証情報は、次のような状況で役立ちます。

  - フェデレーションユーザーアクセス – ユーザーを作成する代わりに、AWS Directory Service、エンタープライズユーザーディレクトリ、またはウェブ ID プロバイダー (IdP) のアイデンティティを使用することもできます。このようなユーザーはフェデレーティッドユーザーと呼ばれます。AWS では、ID プロバイダーを通じてユーザーがアクセスをリクエストしたとき、フェデレーティッドユーザーにロールを割り当てます。フェデレーションユーザーの詳細については、「IAM ユーザーガイド」の「フェデレーションユーザーとロール」を参照してください。
  - クロスアカウントアクセス – アカウントの IAM ロールを使用して、アカウントのリソースにアクセスするためのアクセス許可を別のアカウントに付与することができます。この例については、「IAM ユーザーガイド」の「チュートリアル: AWS アカウント間の IAM ロールを使用したアクセスの委任」を参照してください。
  - AWS のサービスのアクセス – アカウントで IAM ロールを使用して、アカウントのリソースにアクセスするための、AWS のサービスのアクセス許可を付与できます。例えば、Amazon Redshift が Amazon S3 バケットに保存されたデータを Amazon Redshift クラスターにロードすることを許可するロールを作成できます。詳細については、IAM ユーザーガイドの「AWS のサービスにアクセス権限を委任するロールの作成」を参照してください。

- Amazon EC2 で実行されているアプリケーション – へのアクセスを必要とする Amazon EC2 アプリケーションの場合 EventBridge、EC2 インスタンスにアクセスキーを保存するか、IAM ロールを使用して一時的な認証情報を管理できます。AWS ロールを EC2 インスタンスに割り当てるには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルはロールを含み、EC2 インスタンスで実行されるアプリケーションに一時アクセスキーを提供します。詳細については、「IAM User Guide」(IAM ユーザーガイド) の「Using Roles for Applications on Amazon EC2」(Amazon EC2 上のアプリケーションに対するロールの使用) を参照してください。

アクセスコントロール

EventBridge リソースを作成またはアクセスするには、有効な認証情報とアクセス許可の両方が必要です。例えば、AWS Lambda、Amazon Simple Notification Service (Amazon SNS)、および Amazon Simple Queue Service (Amazon SQS) ターゲットを呼び出すには、それらのサービスに対するアクセス許可が必要です。

Amazon EventBridge リソースへのアクセス許可の管理

アイデンティティベースまたはリソースベースのポリシーを使用して、ルールやイベントなどの EventBridge リソースへのアクセスを管理します。

EventBridge リソース

Eventbridge のリソースとサブリソースには、一意の Amazon リソースネーム (ARN) が関連付けられています。イベントパターンを作成するには、EventBridge で ARN を使用します。ARN の詳細については、AWSの「Amazon リソースネーム (ARN) および Amazon Web Services 全般のリファレンス サービスの名前空間」を参照してください。

EventBridge がリソースを操作する際のオペレーションのリストについては、「Amazon EventBridge アクセス許可のリファレンス」を参照してください。

> **Note**
>
> AWS のほとんどのサービスでは、ARN 内のコロン (:) またはスラッシュ (/) は同じ文字として扱われます。ただし、EventBridge では、イベントパターンおよびルールで完全一致が使用されます。イベントパターンの作成時に正しい ARN 文字を使用して、一致させるイベント内の ARN 構文とそれらの文字が一致するようにしてください。

次の表に、EventBridge のリソースを示します。

リソースタイプ	ARN 形式
アーカイブ	`arn:aws:events:` *`region`*`:`*`account`*`:archive/` *`archive-name`*
再生	`arn:aws:events:` *`region`*`:`*`account`*`:replay/`*`replay-name`
ルール	`arn:aws:events:` *`region`*`:`*`account`*`:rule/[event-bus-name]/`*`rule-name`
イベントバス	`arn:aws:events:` *`region`*`:`*`account`*`:event-bus/` *`event-bus-name`*

リソースタイプ	ARN 形式
全ての EventBridge リソース	`arn:aws:events:*`
特定リージョンの特定アカウントが所有するすべての Eventbridge リソース	`arn:aws:events: region:account:*`

次の例では、ARN を使用してステートメント内で特定のルール (*myRule*) を指定する方法を示しています。

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

特定のアカウントに属するすべてのルールを指定するには、以下のようにアスタリスク (*) ワイルドカードを使用します。

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

すべてのリソースを指定する場合、または特定の API アクションが ARN をサポートしていない場合は、以下のように Resource 要素内でアスタリスク(*) ワイルドカードを使用します。

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

1 つのステートメントで複数のリソースまたは PutTargets を指定するには、次のように ARN をカンマで区切ります。

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

リソースの所有権

アカウントは、誰がリソースを作成したかにかかわらず、そのアカウント内のリソースを所有します。リソース所有者は、リソースの作成リクエストを認証するプリンシパルエンティティ、アカウントルートユーザー、IAM ユーザーまたはロールのアカウントです。次の例は、この仕組みを示しています。

- アカウントのルートユーザー認証情報を使用してルールを作成する場合、アカウントは Eventbridge リソースの所有者です。
- アカウントでユーザーを作成し、このユーザーに Eventbridge リソースを作成するアクセス許可を付与すると、このユーザーは Eventbridge リソースを作成できるようになります。ただし、ユーザーが属するアカウントが Eventbridge リソースの所有者になります。
- Eventbridge リソースを作成するためのアクセス許可を持つアカウントに IAM ロールを作成する場合は、ロールを引き受けることのできるいずれのユーザーも Eventbridge リソースを作成できます。ロールが属するアカウントは Eventbridge リソースを所有しているとします。

リソースへのアクセスの管理

アクセスポリシーでは、誰が何にアクセスできるかを記述します。以下のセクションで、アクセス許可ポリシーを作成するために使用可能なオプションについて説明します。

> **Note**
>
> このセクションでは、Eventbridge のコンテキストでの IAM の使用について説明します。これは、IAM サービスに関する詳細情報を取得できません。IAM に関する詳細なドキュメントについては、「IAM ユーザーガイド」の「What is IAM?」(IAM とは？) を参照してください。IAM ポリシー構文の詳細と説明については、「IAM ユーザーガイド」の「IAM ポリシーリファレンス」を参照してください。

IAM ID にアタッチされたポリシーは ID ベースのポリシー (IAM ポリシー) と呼ばれ、リソースにアタッチされたポリシーはリソースベースのポリシーと呼ばれます。EventBridge では、アイデンティティベースのポリシー (IAM ポリシー) とリソースベースのポリシーの両方が使用できます。

トピック

- アイデンティティベースのポリシー (IAM ポリシー)
- リソースベースのポリシー (IAM ポリシー)

アイデンティティベースのポリシー (IAM ポリシー)

ポリシーを IAM アイデンティティにアタッチできます。例えば、次の操作を実行できます。

- アカウントのユーザーまたはグループにアクセス権限ポリシーをアタッチする – Amazon CloudWatch コンソールでルールを表示するアクセス権限を付与するには、ユーザーが所属するユーザーまたはグループにアクセス許可のポリシーをアタッチします。
- 許可ポリシーをロールにアタッチする (クロスアカウントの許可を付与) - ID ベースのアクセス許可ポリシーを IAM ロールにアタッチして、クロスアカウントの許可を付与することができます。例えば、アカウント A の管理者は、次のように別のアカウント B または AWS サービスにクロスアカウント許可を付与するロールを作成できます。
  1. アカウント A の管理者は、IAM ロールを作成して、アカウント A のリソースに許可を付与するロールに許可ポリシーを添付します。
  2. アカウント A の管理者は、アカウント B をそのロールを引き受けるプリンシパルとして識別するロールに、信頼ポリシーをアタッチします。
  3. アカウント B の管理者は、アカウント B のユーザーにロールを引き受ける権限を委任できるようになります。これにより、アカウント B のユーザーにアカウント A のリソースの作成とアクセスが許可されます。ロールを引き受けるために必要なアクセス権限を AWS のサービスに付与するには、信頼ポリシー内のプリンシパルも、AWS のサービスのプリンシパルとなることができます。

  IAM を使用した許可の委任の詳細については、「IAM ユーザーガイド」の「アクセス管理」を参照してください。

アカウントのユーザーがアクセスできる呼び出しやリソースを制限する特定の IAM ポリシーを作成し、これらのポリシーをユーザーにアタッチできます。IAM ロールを作成する方法、および Eventbridge の IAM ポリシーステートメントの例を調べる方法の詳細については、Amazon EventBridge リソースへのアクセス許可の管理 を参照してください。

リソースベースのポリシー (IAM ポリシー)

EventBridge でルールが実行されると、そのルールに関連するすべてのターゲットが呼び出されます。つまり、AWS Lambda 関数の呼び出し、Amazon SNS トピックへの公開、または Amazon Kinesis ストリームへのイベントの中継です。所有するリソースで API をコールするには、EventBridge に適切な許可が必要です。Lambda、Amazon SNS、および Amazon SQS リソースの場合、EventBridge はリソースベースのポリシーを使用します。Kinesis ストリームの場合、EventBridge は IAM ロールを使用します。

IAM ロールの作成方法、および Eventbridge のリソースベースのポリシーステートメント例の活用方法に関する詳細は、「Amazon EventBridge のリソースベースのポリシーを使用する」を参照してください。

ポリシー要素 (アクション、効果、プリンシパル) の指定

EventBridge リソースごとに、EventBridge は一連の API オペレーションを定義します。こうした API オペレーションへのアクセス権限を付与するために、EventBridge は一連のアクションをポリシーに定義します。一部の API オペレーションは、API オペレーションを実行するために複数のアクションに対するアクセス許可を要求します。リソースおよび API オペレーションに関する詳細については、「EventBridge リソース」および「Amazon EventBridge アクセス許可のリファレンス」を参照してください。

以下は、基本的なポリシーの要素です。

- リソース - Amazon リソースネーム (ARN) を使用して、ポリシーを適用するリソースを識別します。詳細については、「EventBridge リソース」を参照してください。
- アクション – キーワードを使用して、許可または拒否するリソースオペレーションを識別します。例えば、`events:Describe` 権限は、`Describe` オペレーションの実行をユーザーに許可します。
- 効果— 許可または拒否 のいずれかを指定します。リソースへのアクセスを明示的に許可していない場合、アクセスは拒否されます。また、明示的にリソースへのアクセスを拒否すると、別のポリシーによってアクセスが許可されている場合でも、ユーザーがそのリソースにアクセスできるかどうかは不確実になります。
- プリンシパル - ID ベースのポリシー (IAM ポリシー) で、ポリシーがアタッチされているユーザーが黙示的なプリンシパルとなります。リソースベースのポリシーでは、権限 (リソースベースのポリシーにのみ適用) を受け取りたいユーザー、アカウント、サービス、またはその他のエンティティを指定します。

IAM ポリシー構文の詳細と説明については、IAM ユーザーガイドの「IAM JSON ポリシーリファレンス」を参照してください。

EventBridge API アクションとそれらが適用されるリソースの詳細については、「Amazon EventBridge アクセス許可のリファレンス」を参照してください。

ポリシーでの条件の指定

アクセス権限を付与するとき、アクセスポリシー言語を使用して、ポリシーが有効になる必要がある条件を指定できます。例えば、特定の日付の後にのみ適用されるポリシーが必要になる場合があります。ポリシー言語での条件の指定の詳細については、「IAM ユーザーガイド」の「条件」を参照してください。

条件を定義には、条件キーを使用します。AWS 条件キーと EventBridge 固有のキーがあり、必要に応じて使用できます。AWS キーの完全な一覧については、IAM ユーザーガイドの条件に利用可能なキーを参照してください。EventBridge 固有のキーの一覧については、「詳細に設定されたアクセスコントロールのための IAM ポリシー条件の使用」を参照してください。

Amazon でのアイデンティティベースのポリシー (IAM ポリシー) の使用 EventBridge

アイデンティティベースのポリシーは、IAM アイデンティティにアタッチするアクセス許可ポリシーです。

トピック

AWS の マネージドポリシー EventBridge

AWS は、 によって作成および管理されるスタンドアロン IAM ポリシーを提供することで、多くの一般的なユースケースに対処します AWS。管理ポリシー、つまり事前定義ポリシーは、一般的ユースケースに必要なアクセス許可を付与するため、どの許可が必要なのかをユーザーが調査する必要はありません。詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

アカウントのユーザーにアタッチできる以下の AWS マネージドポリシーは、 に固有です EventBridge。

- AmazonEventBridgeFullAccess — EventBridge パイプ EventBridge、 EventBridge スキーマ、ス EventBridge ケジューラを含む へのフルアクセスを許可します。
- AmazonEventBridgeReadOnlyAccess — EventBridge Pipes EventBridge、 EventBridge Schemas、ス EventBridge ケジューラを含む への読み取り専用アクセスを許可します。

AmazonEventBridgeFullAccess ポリシー

この AmazonEventBridgeFullAccess ポリシーは、すべての EventBridge アクションを使用するアクセス許可と、次のアクセス許可を付与します。

- `iam:CreateServiceLinkedRole` - API EventBridge 送信先のアカウントにサービスロールを作成するには、このアクセス許可が必要です。このアクセス許可は、API 送信先専用のロールをアカウントに作成するための IAM サービスアクセス許可のみを付与します。

- iam:PassRole - 呼び出しロールを に渡し、ルールのターゲットを呼び出す EventBridge には、このアクセス許可 EventBridge が必要です。
- Secrets Manager のアクセス許可 – EventBridge では、API 送信先を承認するために接続リソースを介して認証情報を提供するときに、アカウントのシークレットを管理するためにこれらのアクセス許可が必要です。

次の JSON は、 AmazonEventBridgeFullAccess ポリシーを示しています。

```
{
    "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**
>
> このセクションの情報は CloudWatchEventsFullAccess ポリシーにも適用されます。ただし、Amazon CloudWatch Events EventBridge の代わりに Amazon を使用することを強くお勧めします。

AmazonEventBridgeReadOnlyAccess ポリシー

この AmazonEventBridgeReadOnlyAccess ポリシーは、すべての読み取り EventBridge アクションを使用するアクセス許可を付与します。

次の JSON は、 AmazonEventBridgeReadOnlyAccess ポリシーを示しています。

```
{
    "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**
>
> このセクションの情報は CloudWatchEventsReadOnlyAccess ポリシーにも適用されます。ただし、Amazon CloudWatch Events EventBridge の代わりに Amazon を使用することを強くお勧めします。

EventBridge スキーマ固有の管理ポリシー

スキーマは、 に送信されるイベントの構造を定義します EventBridge。 EventBridge は、 AWS サービスによって生成されるすべてのイベントのスキーマを提供します。 EventBridge スキーマに固有の以下の AWS 管理ポリシーを使用できます。

- AmazonEventBridgeSchemasServiceRolePolicy
- AmazonEventBridgeSchemasFullAccess

- AmazonEventBridgeSchemasReadOnlyAccess

EventBridge スケジューラ固有の管理ポリシー

Amazon EventBridge Scheduler はサーバーレススケジューラで、1 つの中央マネージドサービスからタスクを作成、実行、管理できます。ス EventBridge ケジューラに固有の AWS 管理ポリシーについては、スAWS EventBridge ケジューラユーザーガイドのスケジューラの 管理ポリシーを参照してください。 EventBridge

EventBridge パイプ固有の管理ポリシー

Amazon EventBridge Pipes はイベントソースをターゲットに接続します。パイプを使用すると、イベント駆動型アーキテクチャを開発する際に専門知識や統合コードが不要になります。これにより、会社のアプリケーション全体の一貫性が確保されます。 EventBridge Pipes に固有の以下の AWS マネージドポリシーを使用できます。

- AmazonEventBridgePipesFullAccess

  Amazon EventBridge Pipes へのフルアクセスを提供します。

  > **Note**
  >
  > このポリシーは `iam:PassRole` を提供します。 EventBridge パイプは、パイプを作成および開始するために呼び出しロールを に渡す EventBridge ためにこのアクセス許可を必要とします。

- AmazonEventBridgePipesReadOnlyAccess

  Amazon EventBridge Pipes への読み取り専用アクセスを提供します。

- AmazonEventBridgePipesOperatorAccess

  Amazon EventBridge Pipes への読み取り専用アクセスとオペレーター (つまり、Pipes の実行を停止および開始する機能) アクセスを提供します。

イベントを送信する IAM ロール

イベントをターゲットに中継するには、IAM ロール EventBridge が必要です。

にイベントを送信するための IAM ロールを作成するには EventBridge

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. IAM ロールを作成するには、「IAM ユーザーガイド」の AWS 「サービスにアクセス許可を委任するロールの作成」のステップに従います。 その手順で、次のように実行します。

   - [Role Name] (ロール名) で、アカウント内で一意の名前を使用します。
   - 「ロールタイプの選択」でAWS 「サービスロール」を選択し、「Amazon EventBridge」を選択します。これにより、ロールを引き受ける EventBridge アクセス許可が付与されます。
   - 「ポリシーのアタッチ」で、「」を選択しますAmazonEventBridgeFullAccess。

独自のカスタム IAM ポリシーを作成して、 EventBridge アクションとリソースのアクセス許可を許可することもできます。これらのカスタムポリシーは、それらのアクセス許可が必要な IAM ユーザーまたはグループにアタッチできます。IAM ポリシーの詳細については、「IAM ユーザーガイド」の「IAM ポリシーの概要」を参照してください。カスタム IAM ポリシーの管理と作成の詳細については、『IAM ユーザーガイド』の「IAM ポリシーの管理」を参照してください。

が IAM ロールを使用してターゲットにアクセス EventBridge するために必要なアクセス許可

EventBridge ターゲットには通常、ターゲットを呼び出す EventBridge アクセス許可を付与する IAM ロールが必要です。以下は、さまざまな AWS サービスとターゲットの例です。それ以外の場合は、EventBridge コンソールを使用してルールを作成し、適切な範囲のアクセス許可が事前設定されたポリシーで作成される新しいロールを作成します。

Amazon SQS 、Amazon SNS 、Lambda、 CloudWatch Logs、および EventBridge バスターゲットはロールを使用しないため、 へのアクセス許可はリソースポリシーを介して付与 EventBridge する必要があります。API Gateway ターゲットは、リソースポリシーまたは IAM ロールのいずれかを使用できます。

ターゲットが API 送信先の場合、指定するロールには次のポリシーを含める必要があります。

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

```
        }
    ]
}
```

ターゲットが Kinesis ストリームの場合、そのターゲットにイベントデータを送信するために使用されるロールは、次のポリシーを含める必要があります。

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

ターゲットが Systems Manager Run Command で、コマンドに 1 つ以上の `InstanceIds` 値を指定する場合は、指定するロールに次のポリシーを含める必要があります。

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

ターゲットが Systems Manager Run Command で、コマンドに 1 つ以上のタグを指定する場合は、指定するロールに次のポリシーを含める必要があります。

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

ターゲットが AWS Step Functions ステートマシンの場合、指定するロールには次のポリシーを含める必要があります。

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

ターゲットが Amazon ECS タスクの場合は、指定するロールに次のポリシーを含める必要があります。

```
{
```

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

次のポリシーでは、 の組み込みターゲット EventBridge がユーザーに代わって Amazon EC2 アクションを実行することを許可します。組み込みターゲットでルールを作成するには AWS Management Console 、 を使用する必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TargetInvocationAccess",
            "Effect": "Allow",
            "Action": [
                "ec2:Describe*",
                "ec2:RebootInstances",
                "ec2:StopInstances",
```

```
                "ec2:TerminateInstances",
                "ec2:CreateSnapshot"
            ],
            "Resource": "*"
        }
    ]
}
```

次のポリシーでは EventBridge 、 が アカウントの Kinesis ストリームにイベントをリレーすることを許可します。

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

カスタマー管理ポリシーの例 : タグ付けを使用してルールへのアクセスを制御する

次の例は、 EventBridge アクションのアクセス許可を付与するユーザーポリシーを示しています。このポリシーは、 EventBridge API、 AWS SDKs、または を使用する場合に機能します AWS CLI。

ユーザーに特定の EventBridge ルールへのアクセスを許可しながら、他のルールへのアクセスを禁止できます。そのためには、両方のルールにタグ付けし、そのタグを参照する IAM ポリシーを使用します。 EventBridge リソースのタグ付けの詳細については、「」を参照してくださいAmazon EventBridge タグ。

特定のタグでそのルールにのみアクセスを許可する IAM ポリシーをユーザーに付与することができます。アクセスを許可するルールは、その特定のタグでタグ付けすることで選択します。例えば次のポリシーでは、タグキー `Stack` の値が `Prod` のルールにユーザーアクセス権が付与されます。

```
{
    "Statement": [
        {
```

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

IAM ポリシーステートメントの詳細については、『IAM ユーザーガイド』の「ポリシーを使用したアクセス制御」を参照してください。

AWS マネージドポリシーに対する Amazon EventBridge の更新

このサービスがこれらの変更の追跡を開始した EventBridge 以降の の AWS マネージドポリシーの更新に関する詳細を表示します。このページの変更に関する自動通知については、 EventBridge ドキュメント履歴ページの RSS フィードをサブスクライブしてください。

変更	説明	日付
AmazonEventBridgeFullAccess - ポリシーを更新	AWS GovCloud (US) Regions のみ 次のアクセス許可は使用されないため、含まれません。 • `iam:CreateServiceLinkedRole` EventBridge Schema Registry の アクセス許可	2024 年 5 月 9 日
AmazonEventBridgeSchemasFullAccess - ポリシーを更新	AWS GovCloud (US) Regions のみ 次のアクセス許可は使用されないため、含まれません。	2024 年 5 月 9 日

変更	説明	日付
	• `iam:CreateServiceLinkedRole` EventBridge Schema Registry の アクセス許可	
AmazonEventBridgePipesFullAccess – 新しいポリシーが追加されました	EventBridge EventBridge Pipes を使用するための完全なアクセス許可のための マネージドポリシーが に追加されました。	2022 年 12 月 1 日
AmazonEventBridgePipesReadOnlyAccess – 新しいポリシーが追加されました	EventBridge に EventBridge 、パイプ情報リソースを表示するアクセス許可の マネージドポリシーが追加されました。	2022 年 12 月 1 日
AmazonEventBridgePipesOperatorAccess – 新しいポリシーが追加されました	EventBridge は EventBridge 、パイプ情報を表示したり、パイプの実行を開始および停止したりするためのアクセス許可の マネージドポリシーを追加しました。	2022 年 12 月 1 日
AmazonEventBridgeFullAccess – 既存ポリシーへの更新	EventBridge EventBridge Pipes 機能を使用するために必要なアクセス許可を含めるようにポリシーを更新しました。	2022 年 12 月 1 日

変更	説明	日付
AmazonEventBridgeReadOnlyAccess – 既存ポリシーへの更新	EventBridge で EventBridge 、パイプ情報リソースを表示するために必要なアクセス許可が追加されました。 以下のアクションが追加されました。 • `pipes:DescribePipe` • `pipes:ListPipes` • `pipes:ListTagsForResource`	2022 年 12 月 1 日
CloudWatchEventsReadOnlyAccess – 既存ポリシーへの更新	と一致するように更新されました AmazonEventBridgeReadOnlyAccess。	2022 年 12 月 1 日
CloudWatchEventsFullAccess – 既存ポリシーへの更新	と一致するように更新されました AmazonEventBridgeFullAccess。	2022 年 12 月 1 日

変更	説明	日付
AmazonEventBridgeFullAccess – 既存ポリシーへの更新	EventBridge は、スキーマとスケジューラ機能を使用するために必要なアクセス許可を含めるようにポリシーを更新しました。 次のアクセス許可が追加されました。 • EventBridge スキーマレジストリアクション • EventBridge スケジューラアクション • `iam:CreateServiceLinkedRole` EventBridge Schema Registry の アクセス許可 • `iam:PassRole` ス EventBridge ケジューラのアクセス許可	2022 年 11 月 10 日

変更	説明	日付
AmazonEventBridgeReadOnlyAccess – 既存ポリシーへの更新	EventBridge は、スキーマとスケジューラの情報リソースを表示するために必要なアクセス許可を追加しました。 以下のアクションが追加されました。 • schemas:DescribeCodeBinding • schemas:DescribeDiscoverer • schemas:DescribeRegistry • schemas:DescribeSchema • schemas:ExportSchema • schemas:GetCodeBindingSource • schemas:GetDiscoveredSchema • schemas:GetResourcePolicy • schemas:ListDiscoverers • schemas:ListRegistries • schemas:ListSchemas • schemas:ListSchemaVersions	2022 年 11 月 10 日

変更	説明	日付
	• schemas:ListTagsForResource • schemas:SearchSchemas • scheduler:GetSchedule • scheduler:GetScheduleGroup • scheduler:ListSchedules • scheduler:ListScheduleGroups • scheduler:ListTagsForResource	
AmazonEventBridgeReadOnlyAccess – 既存ポリシーへの更新	EventBridge は、エンドポイント情報を表示するために必要なアクセス許可を追加しました。 以下のアクションが追加されました。 • events:ListEndpoints • events:DescribeEndpoint	2022 年 4 月 7 日

変更	説明	日付
AmazonEventBridgeReadOnlyAccess – 既存ポリシーへの更新	EventBridge は、接続と API 送信先情報の表示に必要なアクセス許可を追加しました。 以下のアクションが追加されました。 • `events:DescribeConnection` • `events:ListConnections` • `events:DescribeApiDestination` • `events:ListApiDestinations`	2021 年 3 月 4 日

変更	説明	日付
AmazonEventBridgeFullAccess – 既存ポリシーへの更新	EventBridge API 送信先を使用するために必要な `iam:CreateServiceLinkedRole` および AWS Secrets Manager アクセス許可を含めるようにポリシーを更新しました。 以下のアクションが追加されました。 • `secretsmanager:CreateSecret` • `secretsmanager:UpdateSecret` • `secretsmanager:DeleteSecret` • `secretsmanager:GetSecretValue` • `secretsmanager:PutSecretValue`	2021 年 3 月 4 日
EventBridge が変更の追跡を開始しました	EventBridge が AWS マネージドポリシーの変更の追跡を開始しました。	2021 年 3 月 4 日

Amazon EventBridge のリソースベースのポリシーを使用する

Eventbridge でルールが実行されると、このルールに関連付けられているすべてのターゲットが呼び出されます。ルールは、AWS Lambda 関数を呼び出す、Amazon SNS トピックを発行する、または Kinesis ストリームにイベントを中継することができます。所有しているリソースに対して API コールを行うには、Eventbridge に適切なアクセス許可が必要です。Lambda、Amazon SNS、Amazon SQS、および Amazon CloudWatch Logs リソースの場合、EventBridge はリソースベースのポリシーを使用します。Kinesis ストリームの場合、EventBridge はアイデンティティベースのポリシーを使用します。

AWS CLI を使用して、ターゲットにアクセス許可を追加します。AWS CLI をインストールして設定する方法については、『AWS Command Line Interface ユーザーガイド』の「AWS Command Line Interface でのセットアップ」を参照してください。

トピック

- Amazon API Gateway のアクセス許可
- CloudWatch Logs のアクセス許可
- AWS Lambda のアクセス許可
- Amazon SNS のアクセス許可
- Amazon SQS のアクセス許可
- EventBridge Pipes の詳細

Amazon API Gateway のアクセス許可

EventBridge ルールを使用して Amazon API Gateway エンドポイントを呼び出すには、API Gateway エンドポイントのポリシーに以下のアクセス許可を追加します。

```
{
  "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 のアクセス許可

CloudWatch Logs がルールのターゲットである場合、EventBridge がログストリームを作成し、CloudWatch Logs がログエントリとしてイベントからテキストを保存します。EventBridge がログストリームを作成してイベントを記録するためには、EventBridge が CloudWatch Logs に書き込むことを可能にするリソースベースポリシーを CloudWatch Logs に含める必要があります。

AWS Management Console を使用して、CloudWatch Logs をルールのターゲットとして追加する場合、リソースベースのポリシーは自動的に作成されます。AWS CLI を使用してターゲットを追加し、ポリシーがまだ存在しない場合は、作成する必要があります。

次の例では、Eventbridge が、/aws/events/ で始まる名前を持つすべてのロググループに書き込むことができます。これらのタイプのログに別の命名ポリシーを使用する場合は、それに応じて例を調整します。

```
{
    "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"
```

```
}
```

詳細については、Amazon CloudWatch Logs API リファレンスガイドの「PutResourcePolicy」を参照してください。

AWS Lambda のアクセス許可

EventBridge ルールを使用して AWS Lambda 関数を呼び出すには、Lambda 関数のポリシーに次のアクセス許可を追加します。

```
{
  "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"
}
```

EventBridge が AWS CLI を使用して Lambda 関数を呼び出せるように上記のパーミッションを追加するには

- コマンドプロンプトで、次のコマンドを入力します。

```
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"
```

EventBridge が Lambda 関数を呼び出せるようにするアクセス許可の詳細な設定方法については、[AWS Lambda 開発者ガイド』の「AddPermission」と「スケジュールされたイベントでの Lambda の使用」を参照してください。

Amazon SNS のアクセス許可

EventBridge が Amazon SNS トピックを発行できるようにするには、`aws sns get-topic-attributes` および `aws sns set-topic-attributes` コマンドを使用します。

> **Note**
>
> EventBridge の Amazon SNS トピックポリシーでは、`Condition` ブロックを使用できません。

Eventbridge に SNS トピックの発行を可能にするアクセス許可を追加するには

1. SNS トピックの属性を一覧表示するには、次のコマンドを使用します。

   ```
   aws sns get-topic-attributes --topic-arn "arn:aws:sns:region:account-id:topic-name"
   ```

   次の例は、新しい SNS トピックの結果を示しています。

   ```
   {
       "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 から文字列へのコンバーターを使用して、次のステートメントを文字列に変換します。

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

文字列に変換したステートメントは、次の例のようになります。

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

3. 前のステップで作成した文字列を、"Policy" 属性の中の "Statement" コレクションに追加します。
4. 新しいポリシーを設定するには、aws sns set-topic-attributes コマンドを使用します。

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

詳細については、『Amazon Simple Notification Service API リファレンス』の「SetTopicAttributes アクション」を参照してください。

Amazon SQS のアクセス許可

EventBridge ルールに Amazon SQS キューの呼び出しを許可するには、`aws sqs get-queue-attributes` コマンドと `aws sqs set-queue-attributes` コマンドを使用します。

SQS キューのポリシーが空の場合は、まずポリシーを作成してから、それに許可ステートメントを追加する必要があります。新しい SQS キューには空のポリシーがあります。

SQS キューに既にポリシーがある場合は、元のポリシーをコピーして、新しいステートメントと組み合わせ、そこに許可ステートメントを追加する必要があります。

EventBridge ルールに SQS キューの呼び出しを可能にするアクセス許可を追加するには

1. SQS キューの属性を一覧表示するには、コマンドプロンプトで、次のコマンドを入力します。

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

2. 次のステートメントを追加します。

```
{
      "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 から文字列へのコンバーターを使用して、上のステートメントを文字列に変換します。文字列に変換したポリシーは、次のようになります。

```
{\"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. set-queue-attributes.json というファイルを次の内容で作成します。

```
{
    "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. 次のコマンドに示すように、先ほど作成した set-queue-attributes.json ファイルを入力として使用して、ポリシー属性を設定します。

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

詳細については、『Amazon Simple Queue サービスデベロッパーガイド』の「Amazon SQS ポリシーの例」を参照してください。

EventBridge Pipes の詳細

EventBridge Pipes は、リソースベースのポリシーをサポートしていません。また、リソースベースのポリシー条件をサポートする API もありません。

サービス間の混乱した代理の防止

混乱した代理問題とは、アクションを実行する許可を持たないエンティティが、より高い特権を持つエンティティにそのアクションの実行を強制できるというセキュリティ問題です。AWS では、サービス間でのなりすましが、混乱した代理問題を生じさせることがあります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別の顧客のリソースに対する処理を実行するように操作される場合があります。これを防ぐために AWS では、顧客のすべてのサービスのデータを保護するのに役立つツールを提供しています。これには、アカウントのリソースへのアクセス許可が付与されたサービスプリンシパルを使用します。

リソースポリシー内では aws:SourceArn および aws:SourceAccount のグローバル条件コンテキストキーを使用して、Amazon EventBridge が別のサービスに付与する、リソースへのアクセス許可を制限することをお勧めします。クロスサービスのアクセスにリソースを 1 つだけ関連付けたい場合は、aws:SourceArn を使用します。クロスサービスが使用できるように、アカウント内の任意のリソースを関連づけたい場合は、aws:SourceAccount を使用します。

混乱した代理問題から保護するための最も効果的な方法は、リソースの完全な ARN を指定しながら、aws:SourceArn グローバル条件コンテキストキーを使用することです。リソースの完全な ARN が不明な場合や、複数のリソースを指定する場合には、グローバルコンテキスト条件キー aws:SourceArn で、ARN の未知部分を示すためにワイルドカード文字 (*) を使用します。例えば、arn:aws:*servicename*:*:*123456789012*:* です。

aws:SourceArn の値に Amazon S3 バケット ARN などのアカウント ID が含まれていない場合は、両方のグローバル条件コンテキストキーを使用して、アクセス許可を制限する必要があります。

イベントバス

EventBridge のイベントバスルールターゲットの場合は、aws:SourceArn の値がルール ARN でなければなりません。

次の例では、EventBridge の aws:SourceArn と aws:SourceAccount グローバル条件コンテキストキーを使用して、混乱した代理問題を回避する方法を示します。この例は、EventBridge ルールで使用されるロールのロール信頼ポリシーで使用するためのものです。

```
{
  "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 Pipes の場合、aws:SourceArn の値がパイプ ARN でなければなりません。

次の例では、EventBridge の aws:SourceArn と aws:SourceAccount グローバル条件コンテキストキーを使用して、混乱した代理問題を回避する方法を示します。この例は、EventBridge Pipes で使用されるロールのロール信頼ポリシーで使用するためのものです。

```
{
  "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 スキーマのリソースベースのポリシー

EventBridge のスキーマレジストリは、リソースベースのポリシーをサポートしています。リソースベースのポリシーは、IAM アイデンティティではなくリソースにアタッチされるポリシーです。例えば、Amazon Simple Storage Service (Amazon S3) では、Amazon S3 バケットにリソースポリシーがアタッチされます。

EventBridge Schemas とリソースベースポリシーの詳細については、以下を参照してください。

- Amazon EventBridge スキーマ REST API リファレンス
- 『IAM ユーザーガイド』の「アイデンティティベースおよびリソースベースのポリシー」

リソースベースのポリシーでサポートされている API

EventBridge スキーマレジストリのリソースベースのポリシーでは、次の API を使用できます。

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

サポートされるすべてのアクションを AWS アカウントに付与するポリシーの例

EventBridge スキーマレジストリの場合、必ずリソースベースのポリシーをレジストリにアタッチする必要があります。スキーマへのアクセスを付与するには、ポリシーでスキーマ ARN とレジストリ ARN を指定します。

EventBridge スキーマの利用可能なすべての API へのアクセスをユーザーに付与するには、以下のようなポリシーを使用し、アクセスを付与したいアカウントのアカウント ID に `"Principal"` を置き換えます。

```
{
    "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 アカウントに読み取り専用のアクションを付与するポリシーの例

次の例では、EventBridge スキーマの読み取り専用 API のみのアクセスをアカウントに付与しています。

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

組織にすべてのアクションを付与するポリシーの例

EventBridge スキーマレジストリでリソースベースポリシーを使用して、組織へのアクセスを付与することができます。詳細については、AWS Organizations ユーザーガイドを参照してください。次の例では、o-a1b2c3d4e5 という ID の組織にスキーマレジストリへのアクセス許可を付与します。

```
{
    "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 アクセス許可のリファレンス

EventBridge ポリシーでアクションを指定するには、次の例のように events: というプレフィックスの後に API オペレーション名を使用します。

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

単一のステートメントに複数のアクションを指定するには、以下のようにコンマで区切ります。

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

複数のアクションを指定するには、ワイルドカードを使用することもできます。例えば、名前が "Put" という単語で始まるすべてのアクションは、以下のように指定できます。

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

Eventbridge API アクションをすべて指定するには、* ワイルドカードを以下のように使用します。

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

次の表は、IAM ポリシーで指定できる EventBridge API オペレーションとそれに対応するアクションの一覧です。

EventBridge API オペレーション	必要なアクセス許可	説明
DeleteRule	events:DeleteRule	ルールを削除するのに必要です。
DescribeEventBus	events:DescribeEventBus	現在のアカウントのイベントバスにイベントを書き込むことが許可されているアカウントを一覧表示するために必要です。
DescribeRule	events:DescribeRule	ルールについての詳細を一覧表示するのに必要です。

EventBridge API オペレーション	必要なアクセス許可	説明
DisableRule	events:DisableRule	ルールを無効にするのに必要です。
EnableRule	events:EnableRule	ルールを有効にするのに必要です。
ListRuleNamesByTarget	events:ListRuleNamesByTarget	ターゲットと関連付けられるルールを一覧表示するために必要です。
ListRules	events:ListRules	アカウントの全グループを一覧表示するために必要です。
ListTagsForResource	events:ListTagsForResource	EventBridge リソースに関連付けられているすべてのタグを一覧表示するために必要です。現在、タグ付けできるのはルールのみです。
ListTargetsByRule	events:ListTargetsByRule	ルールと関連付けられるすべてのターゲットを一覧表示するために必要です。
PutEvents	events:PutEvents	ルールと一致するカスタムイベントを追加するために必要です。
PutPermission	events:PutPermission	このアカウントのデフォルトのイベントバスにイベントを書き込むアクセス権限を別のアカウントに付与するために必要です。
PutRule	events:PutRule	ルールを作成または更新するために必要です。

EventBridge API オペレーション	必要なアクセス許可	説明
PutTargets	events:PutTargets	ルールにターゲットを追加するために必要です。
RemovePermission	events:RemovePermission	このアカウントのデフォルトのイベントバスにイベントを書き込むアクセス権限を別のアカウントから取り消すために必要です。
RemoveTargets	events:RemoveTargets	ターゲットをルールから削除するために必要です。
TestEventPattern	events:TestEventPattern	特定のイベントに対してイベントパターンをテストするために必要です。

詳細に設定されたアクセスコントロールのための IAM ポリシー条件の使用

アクセス権限を付与するには、ポリシーステートメントで IAM ポリシー言語を使用して、ポリシーが有効になる必要がある条件を指定できます。例えば、特定の日付の後にのみ適用されるポリシーを設定することができます。

ポリシーの条件は、キーと値のペアで構成されます。条件キーは大文字小文字を区別しません。

1 つの条件に複数の条件またはキーを指定した場合、EventBridge が許可を付与するにはすべての条件およびキーを満たす必要があります。1 つのキーに複数の値を持つ 1 つの条件を指定した場合、EventBridge は値の 1 つが満たされた場合にアクセス許可を付与します。

条件を指定する際にプレースホルダー、つまりポリシー変数も使用できます。詳細については、IAM ユーザーガイド の「ポリシー変数」を参照してください。IAM ポリシー言語での条件の指定の詳細については、IAM ユーザーガイドの「条件」を参照してください。

デフォルトでは、IAM ユーザー/ロールはお客様のアカウントのイベントにアクセスすることはできません。イベントにアクセスするには、ユーザーは PutRule API アクションに対するアクセス権限が必要です。ユーザーまたはロールは、`events:PutRule` アクションが許可されている場合、特定のイベントに一致するルールを作成できます。ただし、ルールが有用であるためには、ユーザーは `events:PutTargets` アクションに対するアクセス許可も必要です。なぜなら、ルールによって CloudWatch メトリクスの発行以上のことを行いたい場合は、ルールにターゲットを追加する必要もあるからです。

IAM ユーザー/ロールのポリシーステートメントに、特定の一連のソースとイベントタイプにのみ一致するルールの作成を許可する条件を追加できます。特定タイプのソースやイベントへのアクセスを許可するには、`events:source` および `events:detail-type` の条件キーを使用します。

同様に、IAM ユーザー/ロールのポリシーステートメントに、特定のリソースとアカウントにのみ一致するルールの作成を許可する条件を追加できます。特定のリソースへのアクセスを許可するには、`events:TargetArn` の条件キーを使用します。

次の例は、`PutRule` API アクションで deny ステートメントを使用して、ユーザーが EventBridge の Amazon EC2 イベント以外のすべてのイベントにアクセスすることを許可するポリシーです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyPutRuleForAllEC2Events",
            "Effect": "Deny",
```

```
            "Action": "events:PutRule",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "events:source": "aws.ec2"
                }
            }
        }
    ]
}
```

EventBridge の条件キー

EventBridge のポリシーで使用できる条件キーおよび、キーと値のペアは、次の表のとおりです。

条件キー	キーと値のペア	評価の種類
aws:SourceAccount	aws:SourceArn によって指定されたルールが存在するアカウント。	アカウント ID、Null
aws:SourceArn	イベントを送信しているルールの ARN。	ARN、Null
events:creatorAccount	"events:creatorAccount":" *creatorAccount* " *creatorAccount* には、ルールを作成したアカウントのアカウント ID を使用します。特定のアカウントからのルールに対する API コールを認可するには、この条件を使用します。	creatorAccount、Null
events:detail-type	"events:detail-type":" *detail-type* " ここで、*detail-type* は、イベントの [detail-type] フィールドのリテラル文字列 ("AWS API	詳細タイプ、Null

条件キー	キーと値のペア	評価の種類
	Call via CloudTrail" や "EC2 Instance State-change Notification" など) です。	
events: detail.eventTypeCode	"events:detail.eventTypeCode":" *eventTypeCode* " *eventTypeCode* には、イベントの detail.eventTypeCode フィールドのリテラル文字列、例えば "AWS_ABUSE_DOS_REPORT" などを使用します。	eventTypeCode、Null
events: detail.service	"events:detail.service":" *service*" *service* には、イベントの detail.service フィールドのリテラル文字列、例えば "ABUSE" などを使用します。	サービス、Null
events: detail.userIdentity.principalId	"events:detail.userIdentity.principalId":" *principal-id* " *principal-id* には、detail-type が "AWS API Call via CloudTrail" のイベントの detail.userIdentity.principalId フィールドのリテラル文字列、例えば "AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName." などを使用します。	プリンシパル ID、Null

条件キー	キーと値のペア	評価の種類
events:eventBusInvocation	"events:eventBusInvocation":" *boolean*" *boolean* には、ルールが他のアカウントのイベントバスであるターゲットにイベントを送信する場合、true を使用します。PutEvents API コールを使用する場合は、false を使用します。	eventBusInvocation、Null
events:ManagedBy	AWS のサービスによって内部で使用されます。お客様の代わりに AWS のサービスによってルールが作成された場合、値はルールを作成したサービスのプリンシパル名です。	カスタマーポリシーで使用することは想定されていません。
events:source	"events:source":" *source*" イベントのソースフィールドを表すリテラル文字列 ("aws.ec2" または "aws.s3") には *source* を使用します。*source* に指定できるその他の値については、AWS サービスからのイベント のイベント例を参照してください。	ソース、Null

条件キー	キーと値のペア	評価の種類
events:TargetArn	`"events:TargetArn" :" target-arn "` *target-arn* には、ルールのターゲットの ARN、例えば `"arn:aws:lambda:*:*:function:*"` などを使用します。	ArrayofArn、Null

Eventbridge のポリシーステートメントの例については、Amazon EventBridge リソースへのアクセス許可の管理 を参照してください。

トピック

- EventBridge Pipes の詳細
- 例: creatorAccount 条件を使用する
- 例: eventBusInvocation 条件を使用する
- 例: 特定のソースへのアクセスの制限
- 例: イベントパターンで個々に使用できる複数のソースの定義
- 例: イベントパターンで使用できるソースと DetailType を定義する
- 例: ソースがイベントパターンで定義されていることを確認する
- 例: イベントパターンに複数のソースがある場合にのみ許可されるソースのリストを定義する
- 例: detail.service による PutRule アクセスの制限
- 例: detail.eventTypeCode による PutRule アクセスの制限
- 例: 特定の PrincipalId からの API コールの AWS CloudTrail イベントのみが許可されていることを確認する
- 例: ターゲットへのアクセスの制限

EventBridge Pipes の詳細

EventBridge パイプは追加の IAM ポリシー条件キーをサポートしていません。

例: **creatorAccount** 条件を使用する

次のポリシーステートメントの例では、ポリシーで creatorAccount 条件を使用し、creatorAccount として指定されているアカウントがルールを作成したアカウントである場合にのみ、ルールの作成を許可する方法を示しています。

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

例: **eventBusInvocation** 条件を使用する

eventBusInvocation は、呼び出しがクロスアカウントターゲットまたは PutEvents API リクエストから実行されるかどうかを示します。この値は、ターゲットが他のアカウントのイベントバスである場合など、クロスアカウントターゲットを含むルールからの呼び出しの結果である場合、true になります。PutEvents API リクエストからの呼び出しの場合は、値は false となります。次の例は、クロスアカウントターゲットからの呼び出しを示しています。

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

```
        }
      }
    }
  ]
}
```

例: 特定のソースへのアクセスの制限

以下のポリシー例は IAM ユーザーにアタッチできます。ポリシー A では、すべてのイベントに対する PutRule API アクションを許可します。一方、ポリシー B では、作成するルールのイベントパターンが Amazon EC2 イベントに一致する場合にのみ PutRule を許可します。

ポリシー A: すべてのイベントを許可

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

ポリシー B: Amazon EC2 からのイベントのみを許可

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

```
    ]
}
```

EventPattern は PutRule の必須引数です。そのため、ポリシー B が適用されるユーザーが以下のようなイベントパターンで PutRule を呼び出すとします。

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

この場合、ポリシーではこの特定のソース "aws.ec2" を許可するため、ルールが作成されます。ただし、ポリシー B のユーザーが次のようなイベントパターンで PutRule を呼び出した場合、ポリシーはこの特定のソース "aws.s3" を許可しないため、ルールの作成は拒否されます。

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

基本的に、ポリシー B が適用されるユーザーのみが、Amazon EC2 から発生したイベントに一致するルールの作成を許可されます。そのため、Amazon EC2 から発生したイベントのみへのアクセスを許可されます。

ポリシー A とポリシー B の比較については、以下の表を参照してください。

イベントパターン	ポリシー A で許可	ポリシー B で許可
`{ "source": [ "aws.ec2" ] }`	はい	はい
`{ "source": [ "aws.ec2", "aws.s3" ] }`	はい	いいえ (ソース aws.s3 は許可されない)
`{`	はい	はい

イベントパターン	ポリシー A で許可	ポリシー B で許可

<pre>"source":
["aws.ec2"],
"detail-type":
["EC2 Instance
State-change
Notification"]
}</pre>		

<pre>{
"detail-type":
["EC2 Instance
State-change
Notification"]
}</pre>
	はい	いいえ (ソースの指定が必要)

例: イベントパターンで個々に使用できる複数のソースの定義

次のポリシーでは、IAM ユーザーまたはロールは、EventPattern のソースが Amazon EC2 または Amazon ECS のいずれかであるルールを作成することができます。

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

以下の表で、このポリシーで許可または拒否されるイベントパターンを示しています。

イベントパターン	ポリシーで許可
`{ "source": [ "aws.ec2" ] }`	はい
`{ "source": [ "aws.ecs" ] }`	はい
`{ "source": [ "aws.s3" ] }`	いいえ
`{ "source": [ "aws.ec2", "aws.ecs" ] }`	いいえ
`{ "detail-type": [ "AWS API Call via CloudTrail" ] }`	いいえ

例: イベントパターンで使用できるソースと **DetailType** を定義する

以下のポリシーでは、DetailType が EC2 instance state change notification に等しい aws.ec2 ソースからのイベントのみを許可します。

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

以下の表で、このポリシーで許可または拒否されるイベントパターンを示しています。

イベントパターン	ポリシーで許可
`{ "source": [ "aws.ec2" ] }`	いいえ
`{ "source": [ "aws.ecs" ] }`	いいえ
`{ "source": [ "aws.ec2" ], "detail-type": [ "EC2 Instance State-change Notification" ] }`	はい
`{ "source": [ "aws.ec2" ], "detail-type": [ "EC2 Instance Health Failed" ] }`	いいえ
`{`	いいえ

イベントパターン	ポリシーで許可

<pre>"detail-type": ["EC2 Instance State-change Notification"]
}</pre>	

例: ソースがイベントパターンで定義されていることを確認する

以下のポリシーでは、ソースフィールドがある EventPatterns に対するルールの作成のみを許可します。このポリシーでは、IAM ユーザーまたはロールは、特定のソースを指定しない EventPattern を持つルールを作成することはできません。

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

以下の表で、このポリシーで許可または拒否されるイベントパターンを示しています。

イベントパターン	ポリシーで許可

<pre>{
"source": ["aws.ec2"],
"detail-type": ["EC2 Instance State-change Notification"]</pre>
	はい

イベントパターン	ポリシーで許可
} 	

{
"source": ["aws.ecs",
"aws.ec2"]
}
	はい

{
"detail-type": ["EC2
Instance State-change Notificat
ion"]
}
	いいえ

例: イベントパターンに複数のソースがある場合にのみ許可されるソースのリストを定義する

以下のポリシーでは、複数のソースがある EventPatterns に対してのみルールの作成を許可します。イベントパターンにある各ソースは、条件に指定されたリストに含まれている必要があります。ForAllValues 条件を使用するときは、条件リストの少なくとも 1 つの項目が定義されていることを確認してください。

```
{
    "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"
                }
            }
```

```
        }
    ]
}
```

以下の表で、このポリシーで許可または拒否されるイベントパターンを示しています。

イベントパターン	ポリシーで許可
`{ "source": [ "aws.ec2" ] }`	はい
`{ "source": [ "aws.ec2", "aws.s3" ] }`	はい
`{ "source": [ "aws.ec2", "aws.autoscaling" ] }`	いいえ
`{ "detail-type": [ "EC2 Instance State-change Notification" ] }`	いいえ

例: **detail.service** による **PutRule** アクセスの制限

IIAM ユーザーまたはロールを制限して、events:details.service フィールドに特定の値を持つイベントのみにルールの作成を制限することができます。events:details.service の値は AWS サービスの名前であるとは限りません。

このポリシー条件は、セキュリティまたは不正使用に関連する AWS Health からのイベントを処理するときに役立ちます。このポリシー条件を使用すると、これらの機密性の高いアラートへのアクセスを、必要があるユーザーのみに制限することができます。

たとえば、次のポリシーでは、events:details.service の値が ABUSE であるイベントのみにルールを作成することができます。

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

例: **detail.eventTypeCode** による **PutRule** アクセスの制限

IIAM ユーザーまたはロールを制限して、events:details.eventTypeCode フィールドに特定の値を持つイベントのみにルールの作成を制限することができます。このポリシー条件は、セキュリティまたは不正使用に関連する AWS Health からのイベントを処理するときに役立ちます。このポリシー条件を使用すると、これらの機密性の高いアラートへのアクセスを、必要があるユーザーのみに制限することができます。

たとえば、次のポリシーでは、events:details.eventTypeCode の値が AWS_ABUSE_DOS_REPORT であるイベントのみにルールを作成することができます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutRuleEventsWithDetailServiceEC2",
            "Effect": "Allow",
            "Action": "events:PutRule",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
```

```
                    "events:detail.eventTypeCode": "AWS_ABUSE_DOS_REPORT"
                }
            }
        }
    ]
}
```

例: 特定の **PrincipalId** からの API コールの AWS CloudTrail イベントのみが許可されていることを確認する

すべての AWS CloudTrail イベントの `detail.userIdentity.principalId` パスには、API コールを行ったユーザーのプリンシパル ID があります。`events:detail.userIdentity.principalId` 条件キーを利用して、IAM ユーザーまたはロールのアクセス先を、特定のアカウントからのみ発生した CloudTrail イベントに制限できます。

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

以下の表で、このポリシーで許可または拒否されるイベントパターンを示しています。

イベントパターン	ポリシーで許可
`{ "detail-type": [ "AWS API Call via CloudTrail" ]`	いいえ

イベントパターン	ポリシーで許可
}	
{ "detail-type": ["AWS API Call via CloudTrail"], "detail.userIdentity.principalId": ["AIDAJ45Q7YFFAREXAMPLE"] }	はい
{ "detail-type": ["AWS API Call via CloudTrail"], "detail.userIdentity.principalId": ["AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName"] }	いいえ

例: ターゲットへのアクセスの制限

IAM ユーザーまたはロールに events:PutTargets アクセス許可がある場合は、アクセスできるルールと同じアカウントでターゲットを追加できます。次のポリシーでは、ターゲットを特定のルール (アカウント 123456789012 のMyRule) のみに追加することができます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutTargetsOnASpecificRule",
            "Effect": "Allow",
            "Action": "events:PutTargets",
            "Resource": "arn:aws:events:us-east-1:123456789012:rule/MyRule"
        }
    ]
}
```

ルールに追加できるターゲットを制限するには、events:TargetArn 条件キーを使用します。次のように、ターゲットを Lambda 関数だけに制限できます。

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

EventBridge のサービスにリンクされたロールの使用

Amazon EventBridge は、AWS Identity and Access Management (IAM) のサービスにリンクされたロールを使用します。サービスリンクロールは、EventBridge に直接リンクされた一意のタイプの IAM ロールです。サービスリンクロールは、EventBridge による事前定義済みのロールであり、ユーザーに代わってサービスから他の AWS のサービスを呼び出すために必要なすべてのアクセス許可を備えています。

トピック

- API 送信先のシークレットを作成するためのロールの使用
- スキーマ検出にロールを使用する

API 送信先のシークレットを作成するためのロールの使用

Amazon EventBridge は AWS Identity and Access Management (IAM) サービスリンクロールを使用します。サービスリンクロールは、EventBridge に直接リンクされた一意のタイプの IAM ロールです。サービスリンクロールは、EventBridge による事前定義済みのロールであり、ユーザーに代わってサービスから他の AWS のサービスを呼び出すために必要なすべての権限を備えています。

サービスリンクロールを使用すると、必要なアクセス許可を手動で追加する必要がなくなるため、EventBridge の設定が簡単になります。このサービスリンクロールのアクセス許可は EventBridge で定義します。特に定義されている場合を除き、EventBridge のみがそのロールを引き受けることができます。定義された権限には、信頼ポリシーと権限ポリシーに含まれており、その権限ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスリンクロールを削除するには、まずその関連リソースを削除します。これにより、リソースへの意図しないアクセスによる権限の削除が防止され、EventBridge リソースは保護されます。

サービスリンクロールをサポートする他のサービスについては、「IAM と連動する AWS のサービス」を参照し、[Service-linked role (サービスリンクロール)] の列内で [Yes (はい)] と表記されたサービスを確認してください。そのサービスに関するサービスリンクロールのドキュメントを表示するには、リンクが設定されている [Yes (はい)] を選択します。

EventBridge のサービスリンクロールのアクセス許可

EventBridge は、 という名前のサービスにリンクされたロールを使用しますAWSServiceRoleForAmazonEventBridgeApiDestinations。 によって作成された Secrets Manager シークレットへのアクセスを有効にしますEventBridge。

AWSServiceRoleForAmazonEventBridgeApiDestinations サービスにリンクされたロールは、ロールの引き受けについて以下のサービスを信頼します。

- `apidestinations.events.amazonaws.com`

Policy という名前のロール許可ポリシーAmazonEventBridgeApiDestinationsServiceRoleは EventBridge、 が指定されたリソースに対して以下のアクションを実行することを許可します。

- アクション: `secrets created for all connections by EventBridge` 上で `create, describe, update and delete secrets; get and put secret values`

ユーザー、グループ、ロールなどがサービスにリンクされたロールを作成、編集、削除できるようにするには、アクセス権限を設定する必要があります。詳細については、IAM ユーザーガイド の「サービスリンクロールの権限」を参照してください。

EventBridge のサービスリンクロールの作成

サービスリンクロールを手動で作成する必要はありません。AWS Management Console、、AWS CLIまたは AWS API で接続を作成すると、 によってサービスにリンクされたロールEventBridgeが作成されます。

> **Important**
>
> このサービスリンクロールは、このロールでサポートされている機能を使用する別のサービスでアクションが完了した場合にアカウントに表示されます。EventBridge サービスにリンクされたロールのサポートが開始された 2021 年 2 月 11 日より前にサービスを使用していた場合、 はアカウントにAWSServiceRoleForAmazonEventBridgeApiDestinationsロールEventBridgeを作成しました。詳細については、「AWS アカウント に新しいロールが表示される」を参照してください。

このサービスリンクロールを削除した後で再度作成する必要が生じた場合は、同じ方法でアカウントにロールを再作成できます。接続を作成すると、 によってサービスにリンクされたロールが再度EventBridge作成されます。

EventBridge のサービスにリンクされたロールの編集

EventBridge では、AWSServiceRoleForAmazonEventBridgeApiDestinations のサービスにリンクされたロールを編集することはできません。サービスリンクロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロールの説明の編集はできます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの編集」を参照してください。

EventBridge のサービスリンクロールの削除

サービスリンクロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、モニタリングや保守が積極的に行われていない未使用のエンティティを排除できます。ただし、手動で削除する前に、サービスリンクロールをクリーンアップする必要があります。

サービスリンクロールのクリーンアップ

IAM を使用してサービスリンクロールを削除するには、最初に、そのロールで使用されているリソースをすべて削除する必要があります。

> **Note**
>
> リソースを削除する際に、EventBridge のサービスでロールが使用されている場合、削除は失敗することがあります。失敗した場合は、数分待ってから操作を再試行してください。

AWSServiceRoleForAmazonEventBridgeApiDestinations で使用されている EventBridge リソースを削除するには (コンソール)

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. 統合 で API 送信先 を選択し、接続 タブを選択します。
3. 接続を選択し、削除を選択します。

AWSServiceRoleForAmazonEventBridgeApiDestinations で使用されている EventBridge リソースを削除するには (AWS CLI)

- 次のコマンドを使用します: `delete-connection`。

AWSServiceRoleForAmazonEventBridgeApiDestinations で使用されている EventBridge リソースを削除するには (API)

- 次のコマンドを使用します: `DeleteConnection`。

サービスリンクロールの手動による削除

IAM コンソール、AWS CLI、または AWS API を使用して、AWSServiceRoleForAmazonEventBridgeApiDestinations サービスにリンクされたロールを削除します。詳細については、IAM ユーザーガイド の「サービスリンクロールの削除」を参照してください。

EventBridge のサービスにリンクされたロールをサポートするリージョン

EventBridge では、このサービスが利用可能なすべてのリージョンで、サービスにリンクされたロールの使用をサポートしています。詳細については、「AWS リージョンとエンドポイント」を参照してください。

スキーマ検出にロールを使用する

Amazon EventBridge は AWS Identity and Access Management (IAM) サービスリンクロールを使用します。サービスリンクロールは、EventBridge に直接リンクされた一意のタイプの IAM ロールです。サービスリンクロールは、EventBridge による事前定義済みのロールであり、ユーザーに代わってサービスから他の AWS のサービスを呼び出すために必要なすべての権限を備えています。

サービスリンクロールを使用すると、必要なアクセス許可を手動で追加する必要がなくなるため、EventBridge の設定が簡単になります。このサービスリンクロールのアクセス許可は

EventBridge で定義します。特に定義されている場合を除き、EventBridge のみがその口ールを引き受けることができます。定義された権限には、信頼ポリシーと権限ポリシーに含まれており、その権限ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスリンクロールを削除するには、まずその関連リソースを削除します。これにより、リソースへの意図しないアクセスによる権限の削除が防止され、EventBridge リソースは保護されます。

サービスリンクロールをサポートする他のサービスについては、「IAM と連動する AWS のサービス」を参照し、[Service-linked role (サービスリンクロール)] の列内で [Yes (はい)] と表記されたサービスを確認してください。そのサービスに関するサービスリンクロールのドキュメントを表示するには、リンクが設定されている [Yes (はい)] を選択します。

EventBridge のサービスリンクロールのアクセス許可

EventBridge は、 という名前のサービスにリンクされたロールを使用します。AWSServiceRoleForSchemas Amazon EventBridgeスキーマによって作成された マネージドルールにアクセス許可を付与します。

AWSServiceRoleForSchemas サービスにリンクされたロールは、ロールの引き受けについて以下のサービスを信頼します。

- `schemas.amazonaws.com`

という名前のロールのアクセス許可ポリシーAmazonEventBridgeSchemasServiceRolePolicyは EventBridge、 が指定されたリソースに対して以下のアクションを実行することを許可します。

- アクション: `all managed rules created by EventBridge 上で put, enable, disable, and delete rules; put and remove targets; list targets per rule`

ユーザー、グループ、ロールなどがサービスにリンクされたロールを作成、編集、削除できるようにするには、アクセス権限を設定する必要があります。詳細については、IAM ユーザーガイド の「サービスリンクロールの権限」を参照してください。

EventBridge のサービスリンクロールの作成

サービスリンクロールを手動で作成する必要はありません。AWS Management Console、、AWS CLIまたは AWS API でスキーマ検出を実行すると、 によってサービスにリンクされたロール EventBridgeが作成されます。

> **Important**
>
> このサービスリンクロールは、このロールでサポートされている機能を使用する別のサービスでアクションが完了した場合にアカウントに表示されます。EventBridge サービスにリンクされたロールのサポートが開始された 2019 年 11 月 27 日より前に サービスを使用していた場合、 はアカウントにAWSServiceRoleForSchemasロールEventBridgeを作成しました。詳細については、「AWS アカウント に新しいロールが表示される」を参照してください。

このサービスリンクロールを削除した後で再度作成する必要が生じた場合は、同じ方法でアカウントにロールを再作成できます。Schema Discovery を実行すると、 によってサービスにリンクされたロールが再度EventBridge作成されます。

EventBridge のサービスにリンクされたロールの編集

EventBridge では、AWSServiceRoleForSchemas のサービスにリンクされたロールを編集することはできません。サービスリンクロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロールの説明の編集はできます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの編集」を参照してください。

EventBridge のサービスリンクロールの削除

サービスリンクロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、モニタリングや保守が積極的に行われていない未使用のエンティティを排除できます。ただし、手動で削除する前に、サービスリンクロールをクリーンアップする必要があります。

サービスリンクロールのクリーンアップ

IAM を使用してサービスリンクロールを削除するには、最初に、そのロールで使用されているリソースをすべて削除する必要があります。

> **Note**
>
> リソースを削除する際に、EventBridge のサービスでロールが使用されている場合、削除は失敗することがあります。失敗した場合は、数分待ってから操作を再試行してください。

AWSServiceRoleForSchemas で使用されている EventBridge リソースを削除するには (コンソール)

1. https://console.aws.amazon.com/events/ で Amazon EventBridge コンソールを開きます。
2. バス でイベントバス を選択し、イベントバスを選択します。
3. 検出を停止を選択します。

AWSServiceRoleForSchemas で使用されている EventBridge リソースを削除するには (AWS CLI)

- 次のコマンドを使用します: `delete-discoverer`。

AWSServiceRoleForSchemas で使用されている EventBridge リソースを削除するには (API)

- 次のコマンドを使用します: `DeleteDiscoverer`。

サービスリンクロールの手動による削除

IAM コンソール、AWS CLI、または AWS API を使用して、AWSServiceRoleForSchemas サービスにリンクされたロールを削除します。詳細については、IAM ユーザーガイド の「サービスリンクロールの削除」を参照してください。

EventBridge のサービスにリンクされたロールをサポートするリージョン

EventBridge では、このサービスが利用可能なすべてのリージョンで、サービスにリンクされたロールの使用をサポートしています。詳細については、「AWS リージョンとエンドポイント」を参照してください。

を使用した Amazon EventBridge API コールのログ記録 AWS CloudTrail

Amazon EventBridge は、ユーザーAWS CloudTrail、ロール、または によって実行されたアクションを記録するサービスである と統合されています AWS のサービス。 は、 のすべての API コールをイベント EventBridge として CloudTrail キャプチャします。キャプチャされた呼び出しには、EventBridge コンソールからの呼び出しと EventBridge API オペレーションへのコード呼び出しが含まれます。で収集された情報を使用して CloudTrail、 に対するリクエスト EventBridge、リクエスト元の IP アドレス、リクエスト日時などの詳細を確認できます。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するために役立ちます。

- ルートユーザーまたはユーザー認証情報のどちらを使用してリクエストが送信されたか
- リクエストが IAM Identity Center ユーザーに代わって行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS のサービスによって送信されたかどうか。

CloudTrail アカウント AWS アカウント を作成すると、 が でアクティブになり、 CloudTrail イベント履歴 に自動的にアクセスできます。 CloudTrail イベント履歴は、 に記録された過去 90 日間の管理イベントの表示、検索、ダウンロード、およびイミュータブルな記録を提供します AWS リージョン。詳細については、「 ユーザーガイド」の CloudTrail 「イベント履歴の使用AWS CloudTrail 」を参照してください。イベント履歴の表示には料金はかかりません CloudTrail。

AWS アカウント 過去 90 日間のイベントを継続的に記録するには、証跡または CloudTrail Lake イベントデータストアを作成します。

CloudTrail 証跡

証跡により、 はログファイル CloudTrail を Amazon S3 バケットに配信できます。を使用して作成された証跡はすべてマルチリージョン AWS Management Console です。 AWS CLIを使用する際は、単一リージョンまたは複数リージョンの証跡を作成できます。 AWS リージョン アカウントのすべての でアクティビティをキャプチャするため、マルチリージョンの証跡を作成することをお勧めします。単一リージョンの証跡を作成する場合、証跡の AWS リージョンに記録されたイベントのみを表示できます。証跡の詳細については、「AWS CloudTrail ユーザーガイド」の「AWS アカウントの証跡の作成」および「組織の証跡の作成」を参照してください。

証跡を作成 CloudTrail することで、 から進行中の管理イベントのコピーを 1 つ無料で Amazon S3 バケットに配信できますが、Amazon S3 ストレージ料金が発生します。 CloudTrail 料金の詳細については、AWS CloudTrail 「 の料金」を参照してください。Amazon S3 の料金に関する詳細については、「Amazon S3 の料金」を参照してください。

CloudTrail Lake イベントデータストア

CloudTrail Lake では、イベントに対して SQL ベースのクエリを実行できます。 CloudTrail Lake は、既存のイベントを行ベースの JSON 形式で Apache ORC 形式に変換します。ORC は、データを高速に取得するために最適化された単票ストレージ形式です。イベントはイベントデータストアに集約されます。イベントデータストアは、高度なイベントセレクタを適用することによって選択する条件に基いた、イベントのイミュータブルなコレクションです。どのイベントが存続し、クエリに使用できるかは、イベントデータストアに適用するセレクタが制御します。CloudTrail Lake の詳細については、「 ユーザーガイド」の AWS CloudTrail 「Lake の使用AWS CloudTrail 」を参照してください。

CloudTrail Lake イベントデータストアとクエリにはコストが発生します。イベントデータストアを作成する際に、イベントデータストアに使用する料金オプションを選択します。料金オプションによって、イベントの取り込みと保存にかかる料金、および、そのイベントデータストアのデフォルトと最長の保持期間が決まります。 CloudTrail 料金の詳細については、AWS CloudTrail 「 の料金」を参照してください。

EventBridge での データイベント CloudTrail

データイベントでは、リソース上またはリソース内で実行されるリソースオペレーション (Amazon S3 オブジェクトの読み取りまたは書き込みなど) についての情報が得られます。これらのイベントは、データプレーンオペレーションとも呼ばれます。データイベントは、多くの場合、高ボリュームのアクティビティです。デフォルトでは、 CloudTrail はデータイベントを記録しません。 CloudTrail イベント履歴にはデータイベントは記録されません。

追加の変更がイベントデータに適用されます。 CloudTrail 料金の詳細については、AWS CloudTrail 「 の料金」を参照してください。

CloudTrail コンソール、、または CloudTrail API オペレーションを使用して AWS CLI、 EventBridge リソースタイプのデータイベントをログに記録できます。データイベントをログに記録する方法の詳細については、「 AWS CloudTrail ユーザーガイド」の「 AWS Management Consoleを使用したデータイベントのログ記録」および「AWS Command Line Interfaceを使用したデータイベントのログ記録」を参照してください。

次の表に、データイベントをログに記録できる EventBridge リソースタイプを示します。データイベントタイプ (コンソール) 列には、 CloudTrail コンソールのデータイベントタイプリストから選択する値が表示されます。resources.type 値列には、 AWS CLI または CloudTrail APIsを使用して高度なイベントセレクタを設定するときに指定する `resources.type`値が表示されます。列にログ記録された Data APIs CloudTrail には、リソースタイプ CloudTrail について にログ記録された API コールが表示されます。

データイベントタイプ (コンソール)	resources.type 値	にログ記録APIs CloudTrail
イベントバス	`AWS::Events::EventBus`	• DescribeEventBus
イベントバスルール	`AWS::Events::Rule`	• DeleteRule • DescribeRule • DisableRule • EnableRule • ListRuleNamesByTarget • ListRules • ListTargetsByRule • PutRule • PutTargets • RemoveTargets • TestEventPattern
パイプ	`AWS::Pipes::Pipe`	• CreatePipe • DeletePipe • DescribePipe • ListPipes • StartPipe • StopPipe • UpdatePipe

eventName、readOnly、および resources.ARN フィールドでフィルタリングして、自分にとって重要なイベントのみをログに記録するように高度なイベントセレクタを設定できます。オブジェクトの詳細については、「AWS CloudTrail API リファレンス」の「AdvancedFieldSelector」を参照してください。

EventBridge での 管理イベント CloudTrail

管理イベントは、 のリソースで実行される管理オペレーションに関する情報を提供します AWS アカウント。これらのイベントは、コントロールプレーンオペレーションとも呼ばれます。デフォルトでは、 は管理イベント CloudTrail を記録します。

Amazon EventBridge は、すべての EventBridge コントロールプレーンオペレーションを管理イベントとして記録します。が に記録する Amazon EventBridge コントロールプレーンオペレーションのリストについては CloudTrail、 EventBridge Amazon EventBridge 「 API リファレンス」を参照してください。

EventBridge イベントの例

イベントは任意の送信元からの単一のリクエストを表し、リクエストされた API オペレーション、オペレーションの日時、リクエストパラメータなどに関する情報が含まれます。 CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、イベントは特定の順序では表示されません。

次の例は、 PutRuleオペレーションを示す CloudTrail イベントを示しています。

```
{
  "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 レコードの内容については、「 ユーザーガイド」の「 CloudTrailレコードの内容」を参照してください。 AWS CloudTrail

CloudTrail EventBridge Pipes によって実行されたアクションの ログエントリ

EventBridge Pipes は、ソースからイベントを読み取るとき、エンリッチメントを呼び出すとき、またはターゲットを呼び出すときに、指定された IAM ロールを引き受けます。すべてのエンリッチメント、ターゲット、および Amazon SQS、Kinesis、DynamoDB ソースでアカウントで実行されたアクションに関連する CloudTrail エントリの場合、 `sourceIPAddress` および `invokedBy`フィールドには が含まれます`pipes.amazonaws.com`。

すべてのエンリッチメント、ターゲット、Amazon SQS、Kinesis、DynamoDB ソースのサンプル CloudTrail ログエントリ

```
{
  "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"
}
```


他のすべてのソースでは、 CloudTrail ログエントリの sourceIPAddressフィールドには動的な IP アドレスが割り当てられるため、統合やイベントの分類には使用できません。また、これらのエントリには invokedBy フィールドがありません。

その他すべてのソースのサンプル CloudTrail ログエントリ

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    ...
  },
  "eventTime": ",,,",
  "eventName": "...",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "Python-httplib2/0.8 (gzip)",
}
```

Amazon EventBridge でのコンプライアンス検証

SOC、PCI、FedRAMP、HIPAA などのサードパーティーの監査人は、複数の AWS コンプライアンスプログラムの一環として、AWS サービスのセキュリティとコンプライアンスを評価します。

特定のコンプライアンスプログラムの範囲内の AWS サービスのリストについては、「コンプライアンスプログラムによる範囲内の AWS サービス」「」 を参照してください。一般的な情報については、「AWS コンプライアンスプログラム」「」「」を参照してください。

AWS Artifact を使用して、サードパーティーの監査レポートをダウンロードできます。詳細については、「AWS アーティファクトでレポートをダウンロードする」「」を参照ください。

Eventbridge を使用する際のお客様のコンプライアンス責任は、データの機密性、企業のコンプライアンス目的、適用法規によって決まります。AWS は、コンプライアンスに役立つ以下のリソースを提供しています。

- セキュリティ & コンプライアンスクイックスタートガイド – アーキテクチャ上の考慮事項の説明と、AWS でセキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイするためのステップが記載されています。
- HIPAA のセキュリティとコンプライアンスに関するホワイトペーパーを作成する - 企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。
- AWS コンプライアンスのリソース - ワークブックおよびガイドのコレクションです。
- AWS Config デベロッパーガイドの「ルールでのリソースの評価」 - リソース設定が社内のプラクティス、業界のガイドライン、規制にどの程度準拠しているかについて AWS Config が評価する方法に関する情報です。
- AWS Security Hub – セキュリティに関する業界標準およびベストプラクティスへの準拠を確認するのに役立つ、AWS 内でのセキュリティ状態を包括的に表示したもの。

Amazon EventBridge の耐障害性

AWS のグローバルインフラストラクチャは AWS リージョンとアベイラビリティーゾーンを中心として構築されます。AWSリージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立· 隔離されたアベイラビリティーゾーンがあります。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、および拡張性が優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、「AWS グローバルインフラストラクチャ」を参照してください。

Amazon EventBridge でのインフラストラクチャセキュリティ

マネージドサービスである Amazon EventBridge は、AWS グローバルネットワークセキュリティで保護されています。AWSセキュリティサービスと AWS がインフラストラクチャを保護する方法については、「AWS クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「セキュリティの柱 - AWS Well-Architected Framework」の「インフラストラクチャ保護」を参照してください。

AWS が発行している APIコールを使用して、ネットワーク経由で Eventbridge にアクセスします。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS) TLS 1.2 および TLS 1.3 をお勧めします。
- DHE (Ephemeral Diffie-Hellman) や ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートです。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストは、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時セキュリティ認証情報を生成し、リクエストに署名することもできます。

これらの API オペレーションは、任意のネットワークの場所から呼び出すことができ、EventBridge でリソースベースのアクセスポリシーを使用して、ソース IP アドレスに基づく制限を含めることができます。また、Eventbridge ポリシーを使用して、特定の Amazon Virtual Private Cloud (Amazon VPC) エンドポイントまたは特定の VPC からのアクセスを制御することもできます。これにより、実質的に AWS ネットワーク内の特定の VPC からの特定の Eventbridge リソースへのネットワークアクセスが分離されます。

Amazon EventBridge での設定と脆弱性の分析

設定および IT 管理は、AWS とお客様の間で共有される責任です。詳細については、AWS 責任共有モデルを参照してください。

Amazon のモニタリング EventBridge

EventBridge は、一致したイベントの数から、ルール によってターゲットが呼び出された回数まで、すべてのメトリクスを 1 分 CloudWatch ごとに Amazon に送信します。 ???

次の動画では、 によるモニタリングと監査 EventBridge の動作を確認します CloudWatch: イベントのモニタリングと監査

トピック

- EventBridge メトリクス
- EventBridge メトリクスのディメンション

EventBridge メトリクス

AWS/Events 名前空間には、次のメトリクスが含まれます。

Count を単位として使用するメトリクスでは、Sum と が最も有用な統計になる SampleCount 傾向があります。

RuleName ディメンションのみを指定するメトリクスは、デフォルトのイベントバスを参照します。ディメンションEventBusNameと RuleNameディメンションの両方を指定するメトリクスは、カスタムイベントバスを参照します。

メトリクス	説明	ディメンション	単位
DeadLetterInvocations	イベントに反応してルールのターゲットが呼び出されなかった回数。これには、呼び出しによって同じルールが再度実行され、無限ループが発生したものが含まれます。	RuleName	カウント
Events	によって取り込まれたパートナーイベントの数 EventBridge。	EventSourceName	カウント
FailedInvocations	完全に失敗した呼び出しの回数。これには、再試行された呼び出しや、再試行の後に成功した	RuleName	カウント

メトリクス	説明	ディメンション	単位
	呼び出しは含まれません。また、DeadLetterInvocations にカウントされる失敗した呼び出しもカウントされません。 **Note** EventBridge は、このメトリクスがゼロ CloudWatch でない場合にのみ に送信します。		
Invocations	イベントに反応してルールのターゲットが呼び出された回数。これには、成功した呼び出しと失敗した呼び出しが含まれますが、スロットルされた試行と再実行された試行は完全に失敗するまで含められません。DeadLetterInvocations は含まれません。 **Note** EventBridge は、このメトリクスがゼロ CloudWatch でない場合にのみ に送信します。	なし、RuleName	カウント
InvocationAttempts	ターゲットの呼び出しを EventBridge 試行した回数。	なし	カウント
InvocationsCreated	各イベントに応答して作成された呼び出しの総数。 このメトリクスは、1 秒あたりのトランザクションの呼び出しスロットリング制限の使用率をモニタリングするためによく使用されますEventBridge 。	なし	カウント

メトリクス	説明	ディメンション	単位
InvocationsFailedToBeSentToDlq	デッドレターキューに移動できなかった呼び出しの数。デッドレターキューエラーは、アクセス許可エラー、使用できないリソース、またはサイズ制限が原因で発生する可能性があります。 **Note** EventBridge は、このメトリクスがゼロ CloudWatch でない場合にのみ に送信します。	RuleName	カウント
IngestionToInvocationCompleteLatency	イベントの取り込みから最初の呼び出し試行が成功して完了するまでにかかった時間。	EventBusName、なし、RuleName	ミリ秒
IngestionToInvocationStartLatency	イベントが によって取り込まれた時点からターゲットの最初の呼び出し EventBridge まで、イベントを処理する時間。	EventBusName、なし、RuleName	ミリ秒
InvocationsSentToDlq	デッドレターキューに移動された呼び出しの数。 **Note** EventBridge は、このメトリクスがゼロ CloudWatch でない場合にのみ に送信します。	RuleName	カウント

メトリクス	説明	ディメンション	単位
`MatchedEvents`	EventBusName または EventSourceName が指定されている場合、任意のルールに一致したイベントの数。 RuleName が指定されている場合、特定のルールに一致したイベントの数。	EventBusName, EventSourceName, RuleName	カウント
`RetryInvocationAttempts`	Target の呼び出しを再試行した回数。 **Note** EventBridge は、このメトリクスがゼロ CloudWatch でない場合にのみ に送信します。	なし	カウント
`SuccessfulInvocationAttempts`	ターゲットが正常に呼び出された回数。	なし	カウント
`ThrottledRules`	ルール実行がスロットリングされた回数。これらのルールの呼び出しは遅延する場合があります。 詳細については、「???」の「Invocations スロットリング制限 (トランザクション/秒)」を参照してください。	EventBusName、なし、RuleName	カウント
`TriggeredRules`	実行されて任意のイベントに一致したルールの数。 ルールがトリガー CloudWatch されるまで、このメトリクスは に表示されません。	EventBusName、なし、RuleName	カウント

EventBridge PutEvents メトリクス

AWS/Events 名前空間には、PutEvents API リクエストに関する以下のメトリクスが含まれます。

Count を単位として使用するメトリクスでは、Sum と が最も有用な統計になる SampleCount 傾向があります。

メトリクス	説明	ディメンション	単位
PutEventsApproximateCallCount	受信した PutEvents リクエストの概数。	なし	カウント
PutEventsApproximateFailedCount	失敗した PutEvents リクエストの概数。	なし	カウント
PutEventsApproximateSuccessCount	成功した PutEvents リクエストの概数。	なし	カウント
PutEventsApproximateThrottledCount	スロットリングにより拒否された PutEvents リクエストの数。	なし	カウント
PutEventsEntriesCount	PutEvents リクエストに含まれるイベントエントリの数。	なし	カウント
PutEventsFailedEntriesCount	取り込みに失敗した PutEvents リクエスト内のイベントエントリの数。	なし	カウント

メトリクス	説明	ディメンション	単位
PutEvents Latency	PutEvents リクエスト 1 回あたりにかかった時間。	なし	ミリ秒
PutEvents RequestSize	PutEvents リクエストのサイズ。	なし	バイト

EventBridge PutPartnerEvents メトリクス

AWS/Events 名前空間には、PutPartnerEvents API リクエストに関する以下のメトリクスが含まれます。

> **Note**
>
> EventBridge には、イベントを送信する SaaS パートナーアカウントのPutPartnerEventsリクエストに関連するメトリクスのみが含まれます。詳細については、「???」を参照してください。

Count を単位として使用するメトリクスでは、Sum と が最も有用な統計になる SampleCount 傾向があります。

メトリクス	説明	ディメンション	単位
PutPartnerEventsApproximateCallCount	受信した PutPartnerEvents リクエストの概数。	なし	カウント
PutPartnerEventsApproximate	失敗した PutPartnerEvents リクエストの概数。	なし	カウント

メトリクス	説明	ディメンション	単位
FailedCount			
PutPartnerEventsApproximateThrottledCount	スロットリングにより拒否された PutPartnerEvents リクエストの数。	なし	カウント
PutPartnerEventsApproximateSuccessCount	成功した PutPartnerEvents リクエストの概数。	なし	カウント
PutPartnerEventsEntriesCount	PutPartnerEvents リクエストに含まれるイベントエントリの数。	なし	カウント
PutPartnerEventsFailedEntriesCount	取り込みに失敗した PutPartnerEvents リクエスト内のイベントエントリの数。	なし	カウント
PutPartnerEventsLatency	PutPartnerEvents リクエスト 1 回あたりにかかった時間。	なし	ミリ秒

EventBridge メトリクスのディメンション

EventBridge メトリクスには、ディメンション またはソート可能な属性があり、以下にリストされています。

ディメンション	説明
EventBusName	使用可能なメトリクスをイベントバス名でフィルター処理します。
EventSourceName	使用可能なメトリクスをパートナーイベントソース名でフィルター処理します。
RuleName	使用可能なメトリクスをルール名でフィルター処理します。

Amazon のトラブルシューティング EventBridge

このセクションのステップを使用して、Amazon のトラブルシューティングを行うことができます EventBridge。

トピック

ルールは実行されましたが、Lambda 関数が呼び出されませんでした

Lambda 関数が実行されない理由としては、適切なアクセス許可がないことが考えられます。

Lambda 関数のアクセス許可を確認するには

1. を使用して AWS CLI、関数と AWS リージョンで次のコマンドを実行します。

```
aws lambda get-policy --function-name MyFunction --region us-east-1
```

次のような出力が表示されます。

```
{
    "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. 以下のメッセージが表示される場合。

```
A client error (ResourceNotFoundException) occurred when calling the GetPolicy
 operation: The resource you requested does not exist.
```

または、出力が表示されたが、信頼できるエンティティとして events.amazonaws.com がポリシーにない場合は、以下のコマンドを実行します。

```
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. 出力に SourceAccount フィールドが含まれている場合は、削除する必要があります。SourceAccount 設定により、 EventBridge は関数を呼び出せなくなります。

> **Note**
>
> ポリシーが正しくない場合は、ルールを削除してからルールに追加することで、EventBridge コンソールでルールを編集できます。次に、 EventBridge コンソールはターゲット に正しいアクセス許可を設定します。
> 特定の Lambda エイリアスまたはバージョンを使用する場合は、次に示すコマンドを使用して、`aws lambda get-policy` および `aws lambda add-permission` コマンドで `--qualifier` パラメータを追加する必要があります。

```
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
```

ルールを作成または修正したが、テストイベントと一致しない

ルールまたはそのターゲットを変更すると、受信イベントはすぐに、新しい、またはか更新されたルールへのマッチングを開始/停止しないことがあります。変更が有効になるまで、しばらくお待ちください。

イベントが短期間経過しても一致しない場合は、`FailedInvocations`ルールの CloudWatch メトリクス `TriggeredRules`、`Invocations`、および を確認します。これらのメトリクスの詳細については、「Amazon のモニタリング EventBridge」を参照してください。

ルールが AWS サービスのイベントと一致することを意図している場合は、次のいずれかを実行します。

- `TestEventPattern` アクションを使用して、ルールのイベントパターンがテストイベントに一致するかどうかをテストします。詳細については、「Amazon EventBridge API リファレンスTestEventPattern」の「」を参照してください。
- EventBridge コンソール でサンドボックス を使用します。

ルールが、ScheduleExpression で指定した時間に実行されませんでした

ルールのスケジュールが、UTC+0 タイムゾーンで設定されていることを確認します。ScheduleExpression が正しい場合は、「ルールを作成または修正したが、テストイベントと一致しない」の手順に従います。

良きした時刻にルールが実行されなかった

EventBridge は、設定した開始時刻から 1 分以内にルールを実行します。実行時間へのカウントダウンは、ルールを作成するとすぐに開始されます。

> **Note**
>
> スケジュールされたルールは、配信タイプが guaranteed で、予定された時間ごとに少なくとも一度はイベントがトリガーされることを意味します。

cron 式を使用して、指定した時間にターゲットを起動できます。4 時間ごとの 0 分に実行するルールを作成するには、次のいずれかを実行します。

- EventBridge コンソールでは、cron 式 を使用します`0 0/4 * * ? *`。
- を使用して AWS CLI、式 を使用しますcron(`0 0/4 * * ? *`)。

例えば、 を使用して 4 時間ごとにTestRule実行される という名前のルールを作成するには AWS CLI、次のコマンドを使用します。

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

5 分ごとにルールを実行するには、次の cron 式を使用します。

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

cron 式を使用する EventBridge ルールの最小解像度は 1 分です。スケジュールされたルールは、その分のうちに実行されますが、正確に 0 秒に実行されるわけではありません。

EventBridge および ターゲットサービスは分散されるため、スケジュールされたルールの実行からターゲットサービスがターゲットリソースでアクションを実行するまでに数秒の遅延が発生する可能性があります。

ルールが AWS グローバルサービス API コールと一致したが、実行されなかった

AWS IAM や Amazon Route 53 などの グローバルサービスは、米国東部 (バージニア北部) リージョンでのみ利用できるため、グローバルサービスからの AWS API コールからのイベントはそのリージョンでのみ使用できます。詳細については、「AWS サービスからのイベント」を参照してください。

ルールが実行されるとき、ルールに関連付けられた IAM ロールが無視されます

EventBridge は、Kinesis ストリームにイベントを送信するルールにのみ IAM ロールを使用します。Lambda 関数と Amazon SNS トピックを呼び出すルールの場合、リソースベースのアクセス許可を付与する必要があります。

指定した IAM ロールを引き受けるときに AWS STS が EventBridge 使用できるように、リージョンエンドポイントが有効になっていることを確認します。詳細については、IAM ユーザーガイドの「 AWS リージョン AWS STS での のアクティブ化と非アクティブ化」を参照してください。

ルールにはリソースに一致することを条件とするイベントパターンがありますが、一致するイベントがありません

のほとんどのサービスは、Amazon リソースネーム (ARN) でコロン (:) またはスラッシュ (/) を同じ文字として AWS 扱います。ただし、 はイベントパターンとルール で完全一致 EventBridge を使用します。 ARNs イベントパターンの作成時に正しい ARN 文字を使用して、一致させるイベント内の ARN 構文とそれらの文字が一致するようにしてください。

からの AWS API コールイベントなど、一部のイベントでは CloudTrail、リソースフィールドに何も表示されません。

ターゲットへのイベントの配信で遅延が発生する

EventBridge は、ターゲットリソースが制約されているシナリオを除き、最大 24 時間ターゲットにイベントを配信しようとします。最初の試行は、イベントがイベントストリームに到達するとすぐに行われます。ターゲットサービスに問題がある場合、 EventBridge は別の配信を自動的に再スケジュールします。イベントが到着してから 24 時間が経過すると、 EventBridge はイベントの配信を停止し、 に FailedInvocationsメトリクスを発行します CloudWatch。ターゲットに正常に配信できなかったイベントを保存するように DLQ を設定することをお勧めします。詳細については、「デッドレターキューを使用した未配信イベントの処理」を参照してください。

一部のイベントがターゲットに配信されない

EventBridge ルールのターゲットが長時間制約されている場合、配信を再試行しない EventBridge 可能性があります。例えば、ターゲットが受信イベントトラフィックを処理するようにプロビジョニングされておらず、ターゲットサービスがユーザーに代わって が EventBridge 行うリクエストをスロットリングしている場合、 は配信を再試行しない EventBridge 可能性があります。

1 つのイベントに応じてルールが複数回トリガーされた

まれに、単一のイベントまたはスケジュールされた期間に対して同じルールが複数回実行されたり、トリガーされる特定のルールに対して同じターゲットを複数回呼び出されたりする場合があります。

無限ループの防止

では EventBridge、ルールが繰り返し実行される無限ループにつながるルールを作成できます。無限ループに陥るルールがある場合は、そのルールが実行するアクションが同じルールに一致しないように書き換えてください。

例えば、Amazon S3 バケットで ACL が変更されたことを検出してから、ソフトウェアを実行してそれを新しい状態に変更するルールは、無限ループの原因となります。これを解決する 1 つの方法は、不正な状態の ACL のみに一致するようにルールを書き換えることです。

無限ループにより、予想よりも高い料金がすぐに発生する可能性があります。指定した制限を料金が超えるとアラートで知らせる予算設定を使用することをお勧めします。詳細については、「予算によるコストの管理」を参照してください。

マイイベントがターゲットの Amazon SQS キューに配信されない

Amazon SQS キューが暗号化されている場合は、顧客管理 KMS キーを作成し、KMS キーポリシーに次の許可セクションを含める必要があります。詳細については、「アクセスAWS KMS 許可の設定」を参照してください。

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

ルールは実行されるが、Amazon SNS トピックにいずれのメッセージもパブリッシュされない

シナリオ 1

Amazon SNS トピックにメッセージを発行するには、アクセス許可が必要です。を使用して次のコマンドを使用し AWS CLI、us-east-1 をリージョンに置き換え、トピック ARN を使用します。

```
aws sns get-topic-attributes --region us-east-1 --topic-arn "arn:aws:sns:us-
east-1:123456789012:MyTopic"
```

正しいアクセス許可を取得するには、ポリシーが次のようになります。

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

Publish アクセス許可のある events.amazonaws.com が表示されていない場合は、まず現在のポリシーをコピーして、以下のステートメントを追加してください。

```
{\"Sid\":\"Allow_Publish_Events\",
\"Effect\":\"Allow\",\"Principal\":{\"Service\":\"events.amazonaws.com\"},
\"Action\":\"sns:Publish\",
\"Resource\":\"arn:aws:sns:us-east-1:123456789012:MyTopic\"}
```

次に、 を使用してトピック属性を設定し AWS CLI、次のコマンドを使用します。

```
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**
>
> ポリシーが正しくない場合は、ルールを削除して rule. EventBridge sets に追加することで、EventBridge コンソールでルールを編集することもできます。ターゲット に対する正しいアクセス許可を設定します。

シナリオ 2

SNS トピックが暗号化されている場合は、KMS キーポリシーに次のセクションを含める必要があります。

```
{
```

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

Amazon SNS トピックに関連付けられたルールを削除した後でも、Amazon SNS トピックには の EventBridgeアクセス許可が引き続き付与されます。

Amazon SNS をターゲット としてルールを作成すると、 はユーザーに代わって Amazon SNS トピックにアクセス許可 EventBridge を追加します。 ???ルールの作成直後にルールを削除した場合、Amazon SNS トピックからアクセス許可を削除しない EventBridge 可能性があります。その場合は、`aws sns set-topic-attributes` コマンドを使用してトピックからアクセス許可を削除できます。イベントを送信するためのリソースベースのアクセス権限については、「Amazon EventBridge のリソースベースのポリシーを使用する」を参照してください。

で使用できる IAM 条件キー EventBridge

EventBridge は、 AWS全体の条件キー (「IAM ユーザーガイド」の「IAM および AWS STS 条件コンテキストキー」を参照) と、「」に記載されているキーをサポートします詳細に設定されたアクセスコントロールのための IAM ポリシー条件の使用。

EventBridge ルールが壊れているタイミングはどうすればわかりますか？

次のアラームを使用して、 EventBridge ルールが壊れたときに通知できます。

ルールが壊れているときに警告するアラームを作成するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。

2. [アラームの作成] を選択します。CloudWatch カテゴリ別のメトリクスペインで、イベントメトリクス を選択します。
3. メトリクスのリストで、 を選択しますFailedInvocations。
4. グラフの上で、[Statistic]、[Sum] を選択します。
5. [Period] で、値 (例: 5 分) を選択します。[次へ] をクリックします。
6. アラームしきい値 で、名前 にアラームの一意の名前を入力します。例えば、 ですmyFailedRules。[Description] (説明) に、アラームの説明として、例えば「Rules are not delivering events to targets」と入力します。
7. [is] で [>=] および [1] を選択します。[for] に「10」と入力します。
8. [アクション] の [アラームが次の時:] で、[状態: 警告] を選択します。
9. [Send notification to (通知の送信先)] で、既存の Amazon SNS トピックを選択するか、新しいトピックを作成します。新しいトピックを作成するには、[新しいリスト] を選択します。新しい Amazon SNS トピックの名前を入力します。例: myFailedRules。
10. [Email list] に、アラームが ALARM 状態に変わったら通知する E メールアドレスを、カンマ区切りのリストに入力します。
11. [アラームの作成] を選択します。

Amazon EventBridge クォータ

EventBridge のほとんどの側面にはクォータがあります。

トピック

> **Note**
>
> EventBridge スケジューラのクォータの一覧については、「EventBridge スケジューラユーザーガイド」の「EventBridge スケジューラのクォータ」を参照してください。

EventBridge クォータ

EventBridge には、以下のクォータがあります。

Service Quotas コンソールには、EventBridge のクォータに関する情報が表示されます。デフォルトのクォータの表示に加えて、Service Quotas コンソールを使用して、調整可能なクォータの引き上げをリクエストできます。

名前	デフォルト	引き上げ可能	説明
API の送信先	サポートされている各リージョン: 3,000	はい	各リージョンでアカウントごとの API 送信先の最大数

名前	デフォルト	引き上げ可能	説明
接続	サポートされている各リージョン: 3,000	はい	リージョンごとのアカウントあたりの接続の最大数。
CreateEndpoint スロットリング制限 (トランザクション/秒)	サポートされている各リージョン: 5/秒	いいえ	CreateEndpoint API の 1 秒あたりのリクエストの最大数。追加のリクエストは調整されます。
DeleteEndpoint スロットリング制限 (トランザクション/秒)	サポートされている各リージョン: 5/秒	いいえ	DeleteEndpoint API の 1 秒あたりのリクエストの最大数。追加のリクエストは調整されます。
エンドポイント	サポートされている各リージョン: 100	はい	リージョンごとのアカウントあたりのエンドポイントの最大数。
イベントバスポリシーのサイズ	サポートされている各リージョン: 10,240	はい	ポリシーの最大サイズ (文字数)。このポリシーサイズは、別のアカウントにアクセスを許可するたびに増えます。現在のポリシーとそのサイズを確認するには、DescribeEventBus API を使用します。
イベントバス	サポートされている各リージョン: 100	はい	アカウントあたりの最大イベントバス。

名前	デフォルト	引き上げ可能	説明
イベントパターンのサイズ	サポートされている各リージョン: 2,048	はい	イベントパターンの最大サイズ (文字数)。

名前	デフォルト	引き上げ可能	説明
Invocations スロットリング制限 (トランザクション/秒)	us-east-1: 18,750/秒 us-east-2: 4,500/秒 us-west-1: 2,250/秒 us-west-2: 18,750/秒 af-south-1: 750/秒 ap-northeast-1: 2,250/秒 ap-northeast-3: 750/秒 ap-southeast-1: 2,250/秒 ap-southeast-2: 2,250/秒 ap-southeast-3: 750/秒 eu-central-1: 4,500/秒 eu-south-1: 750/秒	はい	呼び出しはルールに一致するイベントで、ルールのターゲットに対して送信されます。制限に到達後、呼び出しが調整されます。つまり、引き続き呼び出しは行われますが、遅延が発生します。

名前	デフォルト	引き上げ可能	説明
	eu-west-1: 18,750/秒 eu-west-2: 2,250/秒 他のサポートされている各リージョン: 1,100/秒		
ルールの数	af-south-1: 100 eu-south-1: 100 他のサポートされている各リージョン: 300	はい	イベントバスごとにアカウントが保持できるルールの最大数

名前	デフォルト	引き上げ可能	説明
PutEvents スロットリング制限 (トランザクション/秒)	us-east-1: 10,000/秒 us-east-2: 2,400/秒 us-west-1: 1,200/秒 us-west-2: 10,000/秒 af-south-1: 400/秒 ap-northeast-1: 1,200/秒 ap-northeast-3: 400/秒 ap-southeast-1: 1,200/秒 ap-southeast-2: 1,200/秒 ap-southeast-3: 400/秒 eu-central-1: 2,400/秒 eu-south-1: 400/秒	はい	PutEvents API の 1 秒あたりのリクエストの最大数。追加のリクエストは調整されます。

名前	デフォルト	引き上げ可能	説明
	eu-west-1: 10,000/秒 eu-west-2: 1,200/秒 他のサポートされている各リージョン: 600/秒		
API 送信先あたりの呼び出しのレート	サポートされている各リージョン: 300/秒	はい	各リージョンでアカウントごとに各 API 送信先エンドポイントに送信する 1 秒あたりの呼び出しの最大数。クォータが満たされると、その API エンドポイントへの今後の呼び出しがスロットリングされます。呼び出しは引き続き発生しますが、遅延します。
ルールあたりのターゲット	サポートされている各リージョン：5	いいえ	ルールに関連付けることができるターゲットの最大数。

名前	デフォルト	引き上げ可能	説明
スロットリング制限 (トランザクション/秒)	サポートされている各リージョン：50/秒	はい	PutEvents を除くすべての EventBridge API オペレーションの 1 秒あたりのリクエストの最大数。追加のリクエストは調整されます。
UpdateEndpoint スロットリング制限 (トランザクション/秒)	サポートされている各リージョン: 5/秒	いいえ	UpdateEndpoint API の 1 秒あたりのリクエストの最大数。追加のリクエストは調整されます。

また、EventBridge には、Service Quotas コンソールでは管理されない以下のクォータがあります。

名前	デフォルト	[Description] (説明)
イベントバス	サポートされている各リージョン: 100	アカウントあたりの最大イベントバス。
イベントバスポリシーのサイズ	サポートされている各リージョン: 10,240	ポリシーの最大サイズ (文字数)。このポリシーサイズは、別のアカウントにアクセスを許可するたびに増えます。現在のポリシーとそのサイズを確認するには、`DescribeEventBus` API を使用します。
イベントパターンのサイズ	サポートされている各リージョン: 2,048	イベントパターンの最大サイズ (文字数)。

名前	デフォルト	[Description] (説明)
		これは最大 4096 文字まで調整可能です。制限をより高くする必要がある場合は、サポートにお問い合わせください。
ワイルドカードを含むルール	サポートされている各リージョン: イベントバスあたり 30 ルール	ワイルドカードを含むイベントフィルターを含めることができる、アカウントごとのイベントバスあたりの最大ルール数。このクォータは変更できません。 イベントパターンでのワイルドカードの使用について詳しくは、「???」を参照してください。
スキーマ検出レベル	サポートされている各リージョン: 255 レベル	スキーマ検出の最大レベル数では、ネストされたイベントを推測します。255 レベルを超えるイベントは無視されます。

リージョン別の PutPartnerEvents クォータ

制限をより高くする必要がある場合は、サポートにお問い合わせください。

リージョン	1 秒あたりのトランザクション
• AWS GovCloud (米国西部) • AWS GovCloud (米国東部) • 米国東部 (バージニア北部) • 米国東部 (オハイオ) • 米国西部 (北カリフォルニア) • 米国西部 (オレゴン) • アフリカ (ケープタウン) • アジアパシフィック (香港) • アジアパシフィック (ムンバイ)	すべてのリージョンで、PutPartnerEvents には、デフォルトで毎秒 1,400 のスループットリクエスト、3,600 のバーストリクエストのソフト制限があります。

リージョン	1 秒あたりのトランザクション
• アジアパシフィック (大阪) • アジアパシフィック (ソウル) • アジアパシフィック (シンガポール) • アジアパシフィック (シドニー) • アジアパシフィック (東京) • カナダ (中部) • 欧州 (フランクフルト) • 欧州 (アイルランド) • ヨーロッパ (ロンドン) • ヨーロッパ (ミラノ) • ヨーロッパ (パリ) • 欧州 (ストックホルム) • 欧州 (ミラノ) • 南米 (サンパウロ) • 中国 (寧夏) • 中国 (北京)	

EventBridge スキーマレジストリのクォータ

EventBridge スキーマレジストリには、以下のクォータがあります。

Service Quotas コンソールには、EventBridge のクォータに関する情報が表示されます。デフォルトのクォータの表示に加えて、Service Quotas コンソールを使用して、調整可能なクォータの引き上げをリクエストできます。

名前	デフォルト	引き上げ可能	説明
DiscoveredSchemas	サポートされている各リージョン: 200	はい	現在のリージョンで作成できる検出スキーマレジストリのスキーマの最大数
Discoverers	サポートされている各リージョン: 10	はい	現在のリージョンで作成できるディスカバーの最大数。
レジストリ	サポートされている各リージョン: 10	はい	現在のリージョンで作成できるレジストリの最大数。
SchemaVersions	サポートされている各リージョン: 100	はい	現在のリージョンで作成できるスキーマあたりのバージョンの最大数。
スキーマ	サポートされている各リージョン: 100	はい	現在のリージョンで作成できるレジストリあたりのスキーマの最大数。(検出スキーマレジストリを除く)

EventBridge Pipes クォータ

EventBridge Pipes には、以下のクォータがあります。制限をより高くする必要がある場合は、サポートにお問い合わせください。

[リソース]	リージョン	デフォルトの制限
アカウントあたりの同時パイプ実行数	• AWS GovCloud (米国西部) • AWS GovCloud (米国東部) • 中国 (寧夏) • 中国 (北京) • アジアパシフィック (大阪) • アフリカ (ケープタウン) • 欧州 (ミラノ) • 米国東部 (オハイオ) • 欧州 (フランクフルト) • 米国西部 (北カリフォルニア) • 欧州 (ロンドン) • アジアパシフィック (シドニー) • アジアパシフィック (東京) • アジアパシフィック (シンガポール) • カナダ (中部) • 欧州 (パリ) • 欧州 (ストックホルム) • 南米 (サンパウロ) • アジアパシフィック (ソウル) • アジアパシフィック (ムンバイ) • アジアパシフィック (香港) • 中東 (バーレーン) • 中国 (寧夏) • 中国 (北京)	1,000

[リソース]	リージョン	デフォルトの制限
	• アジアパシフィック (大阪) • アフリカ (ケープタウン) • 欧州 (ミラノ)	
アカウントあたりの同時パイプ実行数	• 米国東部 (バージニア北部) • 米国西部 (オレゴン) • 欧州 (アイルランド)	3000
アカウントごとのパイプ数	すべて	1,000

Amazon EventBridge タグ

タグは、お客様または が AWS リソース AWS に割り当てるカスタム属性ラベルです。では EventBridge、ルールとイベントバスにタグを割り当てることができます。各リソースには、最大 50 個のタグを設定できます。

タグを使用して、 AWS リソースを識別して整理します。多くの AWS サービスはタグ付けをサポートしているため、異なるサービスのリソースに同じタグを割り当てて、リソースが関連していることを示すことができます。例えば、EC2 インスタンスに割り当てるのと同じタグを EventBridge ルールに割り当てることができます。

各タグは 2 つの部分で構成されます。

- タグキー (例: `CostCenter`、`Environment`、または `Project`)。
  - タグキーでは、大文字と小文字が区別されます。
  - タグキーの最大長は UTF-8 で 128 Unicode 文字です。
  - リソースごとに各タグキーを一意にする必要があります。
  - 使用できる文字は、UTF-8 対応の文字、数字、スペースと、文字 (. : + = @ _ / -) (ハイフン) です。
  - `aws:` プレフィックスは AWS 用に予約されているため、タグで使用することはできません。このプレフィックスを持つタグのキーや値を編集または削除することはできません。このプレフィックスを持つタグは、リソースあたりのタグ数の制限にはカウントされません。
- オプションのタグ値フィールド (例: `111122223333` または `Production`)。
  - 各タグキーが保持できる値は 1 つのみです。
  - タグ値は大文字と小文字が区別されます。
  - タグ値を省略すると、空の文字列を使用した場合と同じになります。
  - タグ値の最大長は UTF-8 で 256 Unicode 文字です。
  - 使用できる文字は、UTF-8 対応の文字、数字、スペースと、文字 (. : + = @ _ / -) (ハイフン) です。

> **Tip**
>
> ベストプラクティスとして、タグでの大文字の使用方針を決定し、その方針をすべてのリソースタイプで一貫して実装することをお勧めします。たとえ

ば、Costcenter、costcenter、CostCenter を使用するかどうかを決定し、すべてのタグに同じ規則を使用します。

EventBridge コンソール、 EventBridge API、または AWS CLI を使用して、タグを追加、編集、または削除できます。詳細については、次を参照してください。

- TagResource「Amazon EventBridge API リファレンス」のListTagsForResource「」、UntagResource「」、および「」
- AWS CLI リファレンスlist-tags-for-resourceの tag-resource、untag-resource、および
- Resource Groups ユーザーガイドの「タグエディタの使用」

ドキュメント履歴

次の表は、2019 年 7 月以降の「Amazon EventBridge ユーザーガイド」の各リリースにおける重要な変更点を示しています。このドキュメントの更新に関する通知を受け取るには、RSS フィードにサブスクライブできます。

変更	説明	リリース日
AWS 管理ポリシーを更新しました。	AWS GovCloud (US) Regions のみ `AmazonEventBridgeFullAccess` および `AmazonEventBridgeSchemasFullAccess` ポリシーには、使用されないため`iam:CreateServiceLinkedRole` 、 は含まれません。 • the section called “ポリシーの更新”	2024 年 5 月 9 日
イベントバスとルールから AWS CloudFormation テンプレートを生成します。	既存の Amazon EventBridge イベントバスとルールから AWS CloudFormation テンプレートを生成できるようになりました。 • Amazon EventBridge イベントバスから AWS CloudFormation テンプレートを生成する	2022 年 11 月 18 日
EventBridge Pipes ドキュメントを起動しました。	オプションのフィルタリングとエンリッチメントを使用して、ソースをターゲットに接続するパイプを作成できるようになりました。 • パイプ	2022 年 12 月 1 日
イベントバスとルールから AWS CloudFormation テンプレートを生成します。	既存の Amazon EventBridge イベントバスとルールから AWS CloudFormation テンプレートを生成できるようになりました。 • Amazon EventBridge イベントバスから AWS CloudFormation テンプレートを生成する	2022 年 11 月 18 日

変更	説明	リリース日
AmazonEventBridgePipesFullAccess ポリシーを追加しました。	Amazon EventBridge Pipes へのフルアクセスを提供します。 • EventBridge パイプ固有の管理ポリシー	2022 年 12 月 1 日
AmazonEventBridgePipesReadOnlyAccess ポリシーを追加しました。	Amazon EventBridge Pipes への読み取り専用アクセスを提供します。 • EventBridge パイプ固有の管理ポリシー	2022 年 12 月 1 日
AmazonEventBridgePipesOperatorAccess ポリシーを追加しました。	Amazon EventBridge Pipes への読み取り専用アクセスとオペレーター (つまり、Pipes の実行を停止および開始する機能) アクセスを提供します。 • EventBridge パイプ固有の管理ポリシー	2022 年 12 月 1 日
CloudWatchEventsFullAccess ポリシーを更新しました。	`AmazonEventBridgeFullAccess` と一致するように更新されました。 • AmazonEventBridgeFullAccess ポリシー	2022 年 12 月 1 日
CloudWatchEventsReadOnlyAccess ポリシーを更新しました。	`AmazonEventBridgeReadOnlyAccess` と一致するように更新されました。 • AmazonEventBridgeReadOnlyAccess ポリシー	2022 年 12 月 1 日

変更	説明	リリース日
イベントパターンのコンテンツフィルタリングが更新されました。	`suffix`、`equals-ignore-case` 、および `$or` フィルタリングオプションを使用してイベントパターンを作成できるようになりました。 • Amazon EventBridge イベントパターンでのコンテンツフィルタリング	2022 年 11 月 14 日
AmazonEventBridgeFullAccess ポリシーを更新しました。	Schema Registry と EventBridge Scheduler EventBridge を使用するために必要なアクセス許可を追加しました。 • AmazonEventBridgeFullAccess ポリシー	2022 年 11 月 10 日
AmazonEventBridgeReadOnlyAccess ポリシーを更新しました。	EventBridge スキーマレジストリとスケ EventBridge ジューラの情報を表示できるようになりました。 • AmazonEventBridgeReadOnlyAccess ポリシー	2022 年 11 月 10 日
イベントパターンのコンテンツフィルタリングが更新されました。	`suffix`、`equals-ignore-case` 、および `$or` フィルタリングオプションを使用してイベントパターンを作成できるようになりました。 • Amazon EventBridge イベントパターンでのコンテンツフィルタリング	2022 年 11 月 14 日
AmazonEventBridgeFullAccess ポリシーを更新しました。	Schema Registry と EventBridge Scheduler EventBridge を使用するために必要なアクセス許可を追加しました。 • AmazonEventBridgeFullAccess ポリシー	2022 年 11 月 10 日

変更	説明	リリース日
AmazonEventBridgeReadOnlyAccess ポリシーを更新しました。	EventBridge スキーマレジストリとスケ EventBridge ジューラの情報を表示できるようになりました。 • AmazonEventBridgeReadOnlyAccess ポリシー	2022 年 11 月 10 日
AmazonEventBridgeReadOnlyAccess ポリシーを更新しました。	エンドポイント情報を表示できるようになりました。 • AmazonEventBridgeReadOnlyAccess ポリシー	2022 年 4 月 7 日
グローバルエンドポイントのサポートを追加しました。	Amazon では、グローバルエンドポイントの使用がサポートされる EventBridge ようになりました。これにより、アプリケーションのリージョンフォールトトレラントが追加コストなしで実現されます。詳細については、以下をご覧ください。 • グローバルエンドポイントとイベントレプリケーションにより、アプリケーションをリージョンフォールトトレラントにする • `CreateEndpoint`	2022 年 4 月 7 日
アーカイブとイベント再生のサポートが追加されました。	Amazon EventBridge では、アーカイブを使用してイベントを保存し、イベントリプレイを使用してアーカイブからイベントをリプレイできるようになりました。詳細については、以下をご覧ください。 • Amazon EventBridge イベントのアーカイブ. • `CreateArchive` • `StartReplay`	2020 年 11 月 5 日

変更	説明	リリース日
デッドレターキューとターゲットの再試行ポリシーのサポートが追加されました。	Amazon では、デッドレターキューの使用とターゲットの再試行ポリシーの定義がサポートされる EventBridge ようになりました。詳細については、以下をご覧ください。 • デッドレターキューを使用した未配信イベントの処理. • `PutTargets`	2020 年 10 月 12 日
JSONSchema Draft4 形式のスキーマのサポートが追加されました。	Amazon は、JSONSchema Draft 4 形式のスキーマをサポートする EventBridge ようになりました。 EventBridge API を使用してスキーマをエクスポートできるようになりました。詳細については、以下をご覧ください。 • Amazon EventBridge スキーマ • `Export` EventBridge 「 Schema Registry API リファレンス」の「」。	2020 年 9 月 28 日
EventBridge Schema Registry のリソースベースのポリシー	Amazon EventBridge Schema Registry で、リソースベースのポリシーがサポートされるようになりました。詳細については、以下を参照してください。 • Amazon EventBridge スキーマのリソースベースのポリシー • `Policy` スキーマレジストリ API EventBridge リファレンスの • AWS CloudFormation ユーザーガイドのRegistryPolicy リソースタイプ	2020 年 4 月 30 日

変更	説明	リリース日
イベントバスのタグ	このリリースにより、イベントバスのタグを作成および管理できるようになりました。イベントバスの作成時にタグを追加したり、関連する API を呼び出すことで既存のタグを追加または管理できます。詳細については、以下を参照してください。 • Amazon EventBridge タグ • タグベースのポリシー • TagResource • UntagResource • ListTagsForResource	2020 年 2 月 24 日
増加したサービスクォータ	Amazon EventBridge では、呼び出しと のクォータが増加しましたPutEvents 。クォータはリージョンによって異なり、必要に応じて増やすことができます。	2020 年 2 月 11 日
ターゲット入力の変換に関する新しいトピックを追加し、Application Auto Scaling イベントへのリンクを追加しました。	入力トランスのドキュメントが改善されました。 • Amazon EventBridge 入力変換 • インプットトランスフォーマーを使用してイベントからデータを抽出し、そのデータをターゲットに入力する • チュートリアル: Eventbridge がイベントターゲットに渡すものを Input Transformer を使用してカスタマイズする Application Auto Scaling イベントへのリンクを追加しました。 • Application Auto Scaling イベントと EventBridge • AWS サービスからのイベント	2019 年 12 月 20 日

変更	説明	リリース日
コンテンツベースのフィルタリング		2019 年 12 月 19 日
Amazon Augmented AI イベント例へのリンクを追加しました。	Amazon Augmented AI のイベント例を提供する「Amazon SageMaker デベロッパーガイド」の「Amazon Augmented AI トピックへのリンクを追加しました。詳細については、以下を参照してください。 • Amazon Augmented AI でイベントを使用する • AWS サービスからのイベント	2019 年 12 月 13 日
Amazon Chime イベント例へのリンクを追加しました。	そのサービスのイベント例を示す Amazon Chime トピックへのリンクを追加しました。詳細については、以下を参照してください。 • による Amazon Chime の自動化 EventBridge • AWS サービスからのイベント	2019 年 12 月 12 日
Amazon EventBridge スキーマ	Amazon でスキーマを管理し、イベントのコードバインディングを生成できるようになりましたEventBridge。詳細については、以下を参照してください。 • Amazon EventBridge スキーマ • EventBridge スキーマ API リファレンス • のEventSchemas リソースタイプリファレンス AWS CloudFormation	2019年12月1日
AWS CloudFormation イベントバスのサポート	AWS CloudFormation が EventBus リソースをサポートするようになりました。また、 EventBusPolicy および ルールリソースの両方で EventBusName パラメータもサポートしています。詳細については、「Amazon EventBridge リソースタイプリファレンス」を参照してください。	2019 年 10 月 7 日

変更	説明	リリース日
新しいサービス	Amazon の初回リリース EventBridge。	2019 年 7 月 11 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。