aws

ユーザーガイド

EventBridge スケジューラ

EventBridge スケジューラ: ユーザーガイド

Table of Contents

Amazon EventBridge スケジューラとは

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 スケジューラは拡張性が高く、270 を超える AWS サービスと 6,000 を超える API オペレーションを呼び出すことができる何百万ものタスクをスケジュールできます。EventBridge スケジューラでは、インフラストラクチャをプロビジョニングして管理したり、複数のサービスと統合したりすることなく、スケジュールを大規模に配信してメンテナンスコストを削減できます。

EventBridge スケジューラは、ダウンストリームのターゲットの空き状況に基づいてスケジュールを調整する組み込みメカニズムにより、タスクを確実に配信します。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。配信の時間枠を柔軟に設定したり、再試行制限を定義したり、失敗したトリガーの最大保持時間を設定したりできます。

トピック

EventBridge スケジューラの主な機能

EventBridge スケジューラには、ターゲットの設定やスケジュールの拡張に使用できる以下の主要機能があります。

- テンプレート化されたターゲット — EventBridge スケジューラは、Amazon SQS、Amazon SNS、Lambda、EventBridge を使用して、一般的な API オペレーションを実行するテンプレート化されたターゲットをサポートしています。定義済みのターゲットを使用すると、EventBridge スケジューラのコンソール、EventBridge スケジューラ SDK、または AWS CLI を使用してスケジュールをすばやく設定できます。
- ユニバーサルターゲット — EventBridge スケジューラにはユニバーサルターゲットパラメーター (UTP) が用意されています。これを使用して、スケジュール上で 270を超える AWS サービスと 6,000 を超える API オペレーションをターゲットとするカスタマイズされたトリガーを作成できます。UTP では、EventBridge スケジューラのコンソール、EventBridge スケジューラ SDK、または AWS CLI を使用して、カスタマイズされたトリガーを設定できます。

- 柔軟な時間枠 — EventBridge スケジューラは柔軟な時間枠をサポートしているため、ターゲットの呼び出しを正確にスケジュールする必要のないユースケースでも、スケジュールを分散してトリガーの信頼性を高めることができます。
- 再試行 — EventBridge スケジューラは、ターゲットに少なくとも 1 回のイベント配信を行います。つまり、ターゲットからの応答で少なくとも 1 つの配信が成功します。EventBridge スケジューラでは、失敗したタスクのスケジュールの再試行回数を設定できます。EventBridge スケジューラは、スケジュールの信頼性を高め、ターゲットが確実に利用できるようにするために、失敗したタスクを遅延させて再試行します。

EventBridge スケジューラへのアクセス

EventBridge スケジューラは、EventBridge コンソール、EventBridge スケジューラ SDK、AWS CLI を介して使用するか、EventBridge スケジューラ API から直接使用できます。

Amazon EventBridge スケジューラのセットアップ

EventBridge スケジューラを使用するには、事前に以下のステップを完了する必要があります。

トピック

にサインアップする AWS

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、電話またはテキストメッセージを受け取り、電話キーパッドで検証コードを入力します。

   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。セキュリティベストプラクティスとして、ユーザーに管理アクセス権を割り当て、ルートユーザーアクセスが必要なタスクの実行にはルートユーザーのみを使用するようにしてください。

IAM ユーザーの作成

管理者ユーザーを作成するには、以下のいずれかのオプションを選択します。

管理者を管理する方法を 1 つ選択します	目的	方法	以下の操作も可能
IAM アイデンティティセンター内 (推奨)	短期の認証情報を使用して AWSにアクセスします。 これはセキュリティのベストプラクティスと一致しています。ベストプラクティスの詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。	AWS IAM アイデンティティセンター ユーザーガイドの「開始方法」の手順に従います。	AWS Command Line Interface ユーザーガイドの を使用する AWS CLI ように を設定 AWS IAM アイデンティティセンターして、プログラムによるアクセスを設定します。
IAM 内 (非推奨)	長期認証情報を使用して AWSにアクセスします。	IAM ユーザーガイドの「緊急アクセス用の IAM ユーザーを作成する」の手順に従います。	IAM ユーザーガイドの「IAM ユーザーのアクセスキーを管理する」の手順に従って、プログラムによるアクセスを設定します。

マネージドポリシーを使用する

前のステップでは、 AWS リソースにアクセスするための認証情報を使用して IAM ユーザーを設定します。ほとんどの場合、EventBridge スケジューラを安全に使用するために、EventBridge スケジューラを使用するために必要な権限のみを持つユーザー、グループ、またはロールを個別に作成することをお勧めします。EventBridge スケジューラは、一般的なユースケースに対して以下のマネージドポリシーをサポートしています。

- AmazonEventBridgeSchedulerFullAccess - コンソールと API を使用した EventBridge スケジューラへのフルアクセスを付与します。
- AmazonEventBridgeSchedulerReadOnlyAccess – EventBridge スケジューラへの読み取り専用アクセスを許可します。

これらのマネージドポリシーは、前のステップで `AdministratorAccess` ポリシーをアタッチしたのと同じ方法で IAM プリンシパルにアタッチできます。アイデンティティベースの IAM ポリシーを使用した EventBridge スケジューラへのアクセス管理の詳細については、「the section called “アイデンティティベースのポリシーを使用する”」を参照してください。

実行ロールを設定する

実行ロールは、EventBridge スケジューラが AWS のサービス ユーザーに代わって他の とやり取りするために引き受ける IAM ロールです。このロールにアクセス権限ポリシーをアタッチして、EventBridge スケジューラにターゲットを呼び出すアクセス権を付与します。

コンソールを使用して新しいスケジュールを作成するときに、新しい実行ロールを作成することもできます。コンソールを使用する場合、EventBridge スケジューラは、選択したターゲットに基づく権限を持つロールをユーザーに代わって作成します。EventBridge スケジューラがロールを作成すると、ロールの信頼ポリシーには、ユーザーに代わってロールを引き受けることができるプリンシパルを制限する条件キーが含まれます。これにより、混乱した代理のセキュリティ問題を防ぐことができます。

次のステップでは、新しい実行ロールを作成する方法と、EventBridge スケジューラにターゲットを呼び出すアクセス許可を付与する方法について説明します。このトピックでは、一般的なテンプレート化されたターゲットの権限について説明します。他のターゲットに権限を追加する方法については、「the section called “テンプレート化されたターゲットの使用”」を参照してください。

を使用して実行ロールを作成するには AWS CLI

1. 次のロール引き受け JSON ポリシーをコピーし、`Scheduler-Execution-Role.json` という名前でローカルに保存します。この信頼ポリシーにより、EventBridge スケジューラはユーザーに代わってロールを引き受けることを許可されます。

   JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "scheduler.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

> **Important**
>
> 実稼働環境で実行ロールを設定するには、混乱した代理の問題を防ぐための追加の保護手段を導入することをお勧めします。詳細およびポリシーの例については、「the section called “混乱した代理の防止”」を参照してください。

2. AWS Command Line Interface (AWS CLI) から、次のコマンドを入力して新しいロールを作成します。*SchedulerExecutionRole* をこのロールに割り当てる名前に置き換えます。

```
$ aws iam create-role --role-name SchedulerExecutionRole --assume-role-policy-
document file://Scheduler-Execution-Role.json
```

成功すると、次の出力が表示されます。

```
{
    "Role": {
        "Path": "/",
        "RoleName": "Scheduler-Execution-Role",
        "RoleId": "BR1L2DZK3K4CTL5ZF9EIL",
        "Arn": "arn:aws:iam::123456789012:role/SchedulerExecutionRole",
        "CreateDate": "2022-03-10T18:45:01+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "scheduler.amazonaws.com"
```

```
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
}
```

3. EventBridge スケジューラがターゲットを呼び出すことを許可する新しいポリシーを作成するには、次の共通ターゲットのいずれかを選択します。JSON アクセス権限ポリシーをコピーし、.json ファイルとしてローカルに保存します。

   Amazon SQS – SendMessage

   以下により、EventBridge スケジューラはアカウントのすべての Amazon SQS キューに対して sqs:SendMessage アクションを呼び出すことができます。

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Action": [
                   "sqs:SendMessage"
               ],
               "Effect": "Allow",
               "Resource": "*"
           }
       ]
   }
   ```

   Amazon SNS – Publish

   以下により、EventBridge スケジューラはアカウント内のすべての Amazon SNS トピックに対して sns:Publish アクションを呼び出すことができます。

   JSON

   ```
   {
   ```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Lambda – Invoke

以下により、EventBridge スケジューラはアカウント内のすべての Lambda 関数に対して `lambda:InvokeFunction` アクションを呼び出すことができます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

4. 次のコマンドを実行して、新しいアクセス許可ポリシーを作成します。*PolicyName* をこのポリシーに割り当てる名前に置き換えます。

```
$ aws iam create-policy --policy-name PolicyName --policy-document file://
PermissionPolicy.json
```

成功すると、次の出力が表示されます。ポリシーの ARN を書き留めておきます。この ARN を使用して、次のステップで、このポリシーを実行ロールにアタッチします。

```
{
    "Policy": {
        "PolicyName": "PolicyName",
        "CreateDate": "2022-03-015T19:31:18.620Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "ZXR6A36LTYANPAI7NJ5UV",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:policy/PolicyName",
        "UpdateDate": "2022-03-015T19:31:18.620Z"
    }
}
```

5. 次のコマンドを実行して、ポリシーを実行ロールにアタッチします。*your-policy-arn* を、前のステップで作成したポリシーの ARN に置き換えます。*SchedulerExecutionRole* を実行ロールの名前に置き換えます。

```
$ aws iam attach-role-policy --policy-arn your-policy-arn --role-
name SchedulerExecutionRole
```

attach-role-policy オペレーションはコマンドラインにレスポンスを返しません。

ターゲットをセットアップする

EventBridge スケジューラのスケジュールを作成する前に、スケジュールが呼び出すターゲットを少なくとも 1 つ用意する必要があります。既存のリソースを使用するか、新しい AWS リソースを作成できます。次の手順は、 を使用して新しい標準 Amazon SQS キューを作成する方法を示しています CloudFormation。

新しい Amazon SQS キューを作成する方法

1. 次の JSON CloudFormation テンプレートをコピーし、 としてローカルに保存しますScheduler-Target-SQS.json。

```
{
   "AWSTemplateFormatVersion": "2010-09-09",
   "Resources": {
      "MyQueue": {
```

```
            "Type": "AWS::SQS::Queue",
            "Properties": {
               "QueueName": "MyQueue"
                }
            }
         },
      "Outputs": {
         "QueueName": {
            "Description": "The name of the queue",
            "Value": {
               "Fn::GetAtt": [
                  "MyQueue",
                  "QueueName"
               ]
            }
         },
         "QueueURL": {
            "Description": "The URL of the queue",
            "Value": {
               "Ref": "MyQueue"
            }
         },
         "QueueARN": {
            "Description": "The ARN of the queue",
            "Value": {
               "Fn::GetAtt": [
                  "MyQueue",
                  "Arn"
               ]
            }
         }
      }
}
```

2. から次のコマンドを実行して AWS CLI、Scheduler-Target-SQS.jsonテンプレートから CloudFormation スタックを作成します。

```
$ aws cloudformation create-stack --stack-name Scheduler-Target-SQS --template-body
 file://Scheduler-Target-SQS.json
```

成功すると、次の出力が表示されます。

```
{
```

```
    "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/Scheduler-
Target-SQS/1d2af345-a121-12eb-abc1-012e34567890"
}
```

3. 次のコマンドを実行して、 CloudFormation スタックの概要情報を表示します。この情報には、スタックの状態とテンプレートで指定されている出力が含まれます。

```
$ aws cloudformation describe-stacks --stack-name Scheduler-Target-SQS
```

成功すると、コマンドは Amazon SQS キューを作成し、次の出力を返します。

```
{
    "Stacks": [
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/
Scheduler-Target-SQS/1d2af345-a121-12eb-abc1-012e34567890",
            "StackName": "Scheduler-Target-SQS",
            "CreationTime": "2022-03-17T16:21:29.442000+00:00",
            "RollbackConfiguration": {},
            "StackStatus": "CREATE_COMPLETE",
            "DisableRollback": false,
            "NotificationARNs": [],
            "Outputs": [
                {
                    "OutputKey": "QueueName",
                    "OutputValue": "MyQueue",
                    "Description": "The name of the queue"
                },
                {
                    "OutputKey": "QueueARN",
                    "OutputValue": "arn:aws:sqs:us-west-2:123456789012:MyQueue",
                    "Description": "The ARN of the queue"
                },
                {
                    "OutputKey": "QueueURL",
                    "OutputValue": "https://sqs.us-
west-2.amazonaws.com/123456789012/MyQueue",
                    "Description": "The URL of the queue"
                }
            ],
            "Tags": [],
            "EnableTerminationProtection": false,
            "DriftInformation": {
```

```
                "StackDriftStatus": "NOT_CHECKED"
            }
        }
    ]
}
```

このガイドの後半では、QueueARN の値を使用してキューを EventBridge スケジューラのターゲットとして設定します。

次のステップ

セットアップステップが完了したら、入門ガイドを使用して最初の EventBridge スケジューラのスケジュールを作成し、ターゲットを呼び出します。

EventBridge スケジューラの開始方法

このトピックでは、新しい EventBridge スケジューラのスケジュールの作成について説明します。EventBridge スケジューラのコンソール、AWS Command Line Interface (AWS CLI)、または AWS SDK を使用して、テンプレート化された Amazon SQS ターゲットを使用してスケジュールを作成します。次に、ロギングを設定し、再試行回数を設定し、失敗したタスクの最大保持時間を設定します。スケジュールを作成したら、スケジュールがターゲットを正常に呼び出し、ターゲットキューにメッセージを送信することを確認します。

> **Note**
>
> このガイドに従うには、「the section called “アイデンティティベースのポリシーを使用する”」で説明されている最低限の権限で IAM ユーザーを設定することをお勧めします。ユーザーを作成して設定したら、次のコマンドを実行してアクセス認証情報を設定します。AWS CLI を設定するには、アクセスキー ID と シークレットアクセスキーが必要です。
>
> ```
> $ aws configure
> AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
> AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
> Default region name [None]: us-west-2
> Default output format [None]: json
> ```
>
> 認証情報を設定するさまざまな方法について詳しくは、「バージョン 2 用 AWS Command Line Interface ユーザーガイド」の「構成設定と優先順位」を参照してください。

トピック

前提条件

このセクションの手順を開始する前に、次を実行します。

- 「設定」で説明されている各タスクを実行します。

EventBridge スケジューラのコンソールを使用してスケジュールを作成する

コンソールを使用して新しいスケジュールを作成するには

1. AWS マネジメントコンソール にサインインし、次のリンクを選択して EventBridge コンソールの EventBridge スケジューラのセクションを開きます。https://us-west-2.console.aws.amazon.com/scheduler/home?region=us-west-2#home

   > **Note**
   >
   > AWS マネジメントコンソール のリージョンセレクターを使用して AWS リージョン を切り替えることができます。

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: **`MyTestSchedule`**

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

   c. [スケジュールグループ] で、ドロップダウンオプションからスケジュールグループを選択します。スケジュールグループをまだ作成していない場合は、スケジュールの `default` グループを選択できます。新しいスケジュールグループを作成するには、コンソールの説明にある [独自のスケジュールを作成] リンクを選択します。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. [スケジュールのパターン] セクションで、次の操作を行います。

   a. [頻度] で、以下のいずれかのパターンオプションを選択します。設定オプションは、選択したパターンによって変わります。

      - [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] には、有効な日付を YYYY/MM/DD 形式で入力します。次に、24 時間の hh:mm 形式でタイムスタンプを指定します。最後に、ドロップダウンオプションからタイムゾーンを選択します。

- [繰り返しのスケジュール] – 繰り返しのスケジュールは、cron 式または rate 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設定するには、[Cron ベースのスケジュール] を選択します。rate 式を使用するには、[レートベースのスケジュール] を選択し、[値] に正の数を入力し、ドロップダウンオプションから [単位] を選択します。

cron 式と rate 式の詳細については、「スケジュールタイプ」を参照してください。

b. [柔軟な時間枠] で、[オフ] を選択してオプションをオフにするか、ドロップダウンリストから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 15 分以内に実行されます。

5. 前のステップで [繰り返しのスケジュール] を選択した場合は、[時間枠] セクションでタイムゾーンを指定し、必要に応じてスケジュールの開始日時と終了日時を設定します。開始日のない繰り返しのスケジュールは、作成されて利用可能になるとすぐに開始されます。終了日のない繰り返しのスケジュールは、そのターゲットを無期限に呼び出し続けます。
6. [次へ] を選択します。
7. [ターゲットを選択] ページで、次の操作を行います。

   a. [テンプレート化されたターゲット] を選択し、ターゲット API を選択します。この例では、Amazon SQS **SendMessage** のテンプレート化されたターゲットを選択します。

   b. [SendMessage] セクションの [SQS キュー] で、ドロップダウンリストから `arn:aws:sqs:us-west-2:123456789012:TestQueue` などの既存の Amazon SQS キュー ARN を選択します。新しいキューを作成するには、[新しい SQS キューの作成] を選択して Amazon SQS コンソールに移動します。キューの作成が完了したら、EventBridge スケジューラのコンソールに戻り、ドロップダウンを更新します。新しいキュー ARN が表示され、選択できるようになります。

   c. [ターゲット] には、EventBridge スケジューラがターゲットに配信するペイロードを入力します。この例では、次のメッセージをターゲットキューに送信します: **`Hello, it's EventBridge Scheduler.`**
8. [次へ] を選択し、[設定 - オプション] ページで次の操作を行います。

9.

a. [スケジュールの状態] セクションの [スケジュールを有効にする] で、スイッチを使って機能のオンとオフを切り替えます。デフォルトでは、EventBridge スケジューラはスケジュールを有効にします。

b. [スケジュール完了後のアクション] セクションで、スケジュール完了後に EventBridge スケジューラが実行するアクションを設定します。

- スケジュールを自動的に削除したい場合は、[削除] を選択します。1 回限りのスケジュールの場合は、スケジュールがターゲットを一度呼び出した後に削除が実行されます。繰り返しのスケジュールの場合は、スケジュールが最後に予定されていた呼び出しの後に削除が実行されます。自動削除の詳細については、「the section called “スケジュール完了後の削除”」を参照してください。
- スケジュールの完了後に EventBridge スケジューラにアクションを実行させない場合は [なし] を選択するか、値を選択しないでください。

c. [再試行ポリシーとデッドレターキュー (DLQ)] セクションの [再試行ポリシー] で [再試行] をオンにして、スケジュールの再試行ポリシーを設定します。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。

d. [イベントの最大有効期間 - オプション] 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。

> **Note**
>
> 最大値は 24 時間です。

e. [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。

> **Note**
>
> 再試行の最大値は 185 です。

f. [デッドレターキュー (DLQ)] で、次のオプションから選択します。

- [なし] — DLQ を設定しない場合は、このオプションを選択してください。

- [自分の AWS アカウントの Amazon SQS キューを DLQ として選択] — このオプションを選択し、ドロップダウンリストからキュー ARN を選択し、スケジュールを作成しているのと同じ AWS アカウント に DLQ を設定します。
- [他の AWS アカウントの Amazon SQS キューを DLQ として指定] — キューが別の AWS アカウント にある場合は、このオプションを選択し、DLQ として設定するキューの ARN を入力します。このオプションを使用するには、キューの正確な ARN を入力する必要があります。

g. カスタマーマネージド KMS 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] セクションで [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 キーを作成] 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、「the section called “保管中の暗号化”」を参照してください。

h. [アクセス許可] で [既存のロールを使用] を選択し、セットアップ手順中に作成したロールをドロップダウンリストから選択します。[IAM コンソールに移動] を選択して新しいロールを作成することもできます。

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追加します。

10. [次へ] を選択します。
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] を選択して、新しいスケジュールの作成を完了します。[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。
13. スケジュールが Amazon SQS ターゲットを呼び出すことを確認するには、Amazon SQS コンソールを開いて以下を実行します。

a. [キュー] リストからターゲットキューを選択します。

b. [メッセージの送信と受信] を選択します。

c. [メッセージの送信と受信] ページの [メッセージの受信] で [メッセージのポーリング] を選択し、スケジュールによってターゲットキューに送信されたテストメッセージを取得します。

AWS CLI を使用してスケジュールを作成する

次の例は、AWS CLI コマンド `create-schedule` を使用して、テンプレート化された Amazon SQS ターゲットを使用して EventBridge スケジューラのスケジュールを作成する方法を示しています。次のパラメータのプレースホルダ値を自分自身の情報へと置き換えます。

- --name – スケジュールの名前を入力します。
- RoleArn — スケジュールに関連付ける実行ロールの ARN を入力します。
- Arn — ターゲットの ARN を入力します。この場合、ターゲットは Amazon SQS キューです。
- Input – EventBridge スケジューラがターゲットキューに配信するメッセージを入力します。

```
$ aws scheduler create-schedule --name sqs-templated-schedule --schedule-expression
 'rate(5 minutes)' \
--target '{"RoleArn": "ROLE_ARN", "Arn": "QUEUE_ARN", "Input": "TEST_PAYLOAD" }' \
--flexible-time-window '{ "Mode": "OFF"}'
```

EventBridge スケジューラ SDK を使用してスケジュールタスクを作成する

次の例では、EventBridge スケジューラ SDK を使用して、テンプレート化された Amazon SQS ターゲットを使用して EventBridge スケジューラのスケジュールを作成します。

Example Python SDK

```
import boto3
scheduler = boto3.client('scheduler')

flex_window = { "Mode": "OFF" }

sqs_templated = {
    "RoleArn": "<ROLE_ARN>",
    "Arn": "<QUEUE_ARN>",
    "Input": "Message for scheduleArn: '<aws.scheduler.schedule-arn>', scheduledTime:
 '<aws.scheduler.scheduled-time>'"
}

scheduler.create_schedule(
```

```
    Name="sqs-python-templated",
    ScheduleExpression="rate(5 minutes)",
    Target=sqs_templated,
    FlexibleTimeWindow=flex_window)
```

Example Java SDK

```
package com.example;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.scheduler.SchedulerClient;
import software.amazon.awssdk.services.scheduler.model.*;


public class MySchedulerApp {

    public static void main(String[] args) {

        final SchedulerClient client = SchedulerClient.builder()
                .region(Region.US_WEST_2)
                .build();

        Target sqsTarget = Target.builder()
                .roleArn("<ROLE_ARN>")
                .arn("<QUEUE_ARN>")
                .input("Message for scheduleArn: '<aws.scheduler.schedule-arn>',
 scheduledTime: '<aws.scheduler.scheduled-time>'")
                .build();

        CreateScheduleRequest createScheduleRequest = CreateScheduleRequest.builder()
                .name("<SCHEDULE NAME>")
                .scheduleExpression("rate(10 minutes)")
                .target(sqsTarget)
                .flexibleTimeWindow(FlexibleTimeWindow.builder()
                        .mode(FlexibleTimeWindowMode.OFF)
                        .build())
                .build();

        client.createSchedule(createScheduleRequest);
        System.out.println("Created schedule with rate expression and an Amazon SQS
 templated target");
    }
}
```

次のステップ

- コンソール、AWS CLI、または EventBridge スケジューラ SDK を使用してスケジュールを管理する方法の詳細については、「スケジュールの管理」を参照してください。
- テンプレート化されたターゲットの設定方法とユニバーサルターゲットパラメータの使用方法の詳細については、「ターゲットの管理」を参照してください。
- EventBridge スケジューラのデータ型と API オペレーションの詳細については、「EventBridge スケジューラ API リファレンス」を参照してください。

EventBridge スケジューラのスケジュールタイプ

以下のトピックでは、Amazon EventBridge スケジューラがサポートするさまざまなスケジュールタイプと、EventBridge スケジューラが夏時間やさまざまなタイムゾーンでのスケジューリングを管理する方法について説明します。スケジュールを設定する際には、レートベース、cron ベース、および 1 回限りのスケジュールの 3 つのスケジュールタイプから選択できます。

レートベースのスケジュールと cron ベースのスケジュールはどちらも繰り返しのスケジュールです。設定するスケジュールのタイプに対応するスケジュール式を使用し、EventBridge スケジューラが式を評価するタイムゾーンを指定して、繰り返しのスケジュールタイプをそれぞれ設定します。

1 回限りのスケジュールは、ターゲットを 1 回だけ呼び出すスケジュールです。1 回限りのスケジュールを設定するには、EventBridge スケジューラがスケジュールを評価する時刻、日付、およびタイムゾーンを指定します。

> **Note**
>
> EventBridge スケジューラのすべてのスケジュールタイプは、60 秒の精度でターゲットを呼び出します。つまり、`1:00` にスケジュールを実行するように設定すると、柔軟な時間枠を設定していない限り、ターゲット API は `1:00:00` と `1:00:59` の間に呼び出されます。

以下のセクションでは、繰り返しのスケジュールタイプごとにスケジュール式を設定する方法と、EventBridge スケジューラで 1 回限りのスケジュールを設定する方法について説明します。

トピック

レートベースのスケジュール

レートベースのスケジュールは、スケジュールに指定した開始日の後に開始され、スケジュールの終了日までユーザーが定義した標準レートで実行されます。一般的な繰り返しのスケジュールのユー

スケースのほとんどは、レートベースのスケジュールを使用して設定できます。たとえば、15 分ごと、2 時間に 1 回、または 5 日に 1 回ターゲットを呼び出すスケジュールが必要な場合は、レートベースのスケジュールを使用してこれを達成できます。レートベースのスケジュールは、rate 式を使用して設定します。

レートベースのスケジュールでは、StartDate プロパティを使用してスケジュールが最初に出現する日を設定します。レートベースのスケジュールに StartDate を指定しない場合、スケジュールはただちにターゲットを呼び出します。

rate 式には次のように 2 つの必須フィールドがあり、空白で区切られます。

構文

```
rate(value unit)
```

値

正数。

単位

スケジュールでターゲットを呼び出す時間単位。

有効な入力: minutes | hours | days

例

次の例は、 コマンドで AWS CLI create-schedule rate 式を使用してレートベースのスケジュールを設定する方法を示しています。この例では、5 分ごとに実行されるスケジュールを作成し、テンプレート化された SqsParameters ターゲットタイプを使用して Amazon SQS キューにメッセージを配信します。

この例では --start-date パラメータの値を設定していないため、スケジュールを作成してアクティブ化した直後に、ターゲットの呼び出しが開始されます。

```
$ aws scheduler create-schedule --schedule-expression 'rate(5 minutes)' --
name schedule-name \
--target '{"RoleArn": "role-arn", "Arn": "QUEUE_ARN", "Input": "TEST_PAYLOAD" }' \
--flexible-time-window '{ "Mode": "OFF"}'
```

Cron ベースのスケジュール

cron 式は、指定した日時に実行されるきめ細かい繰り返しのスケジュールを作成します。EventBridge スケジューラは、協定世界時 (UTC) またはスケジュール作成時に指定したタイムゾーンでの cron ベースのスケジュール設定をサポートしています。cron ベースのスケジュールでは、スケジュールを実行するタイミングと頻度をより細かく制御できます。EventBridge スケジューラの rate 式ではサポートされていないカスタマイズされた繰り返しのスケジュールが必要な場合は、cron ベースのスケジュールを使用してください。例えば、毎月第 1 月曜日の午前 8 時 (PST) に実行する cron ベースのスケジュールを作成できます。cron ベースのスケジュールは、cron 式を使用して設定します。

cron 式は、次に示すように、空白で区切られた 5 つの必須フィールド (分、時間、日、月、曜日) と 1 つのオプションフィールド (年) で構成されます。

構文

```
cron(minutes hours day-of-month month day-of-week year)
```

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。例えば、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。

例

次の例は、 コマンドで AWS CLI `create-schedule` cron 式を使用して cron ベースのスケジュールを設定する方法を示しています。この例では、2022 ~ 2023 年の各月の最終金曜日の午前 10 時 15 分 (UTC+0) に実行されるスケジュールを作成し、テンプレート化された `SqsParameters` ターゲットタイプを使用して Amazon SQS キューにメッセージを配信します。

```
$ aws scheduler create-schedule --schedule-expression "cron(15 10 ? * 6L 2022-2023)" --
name schedule-name \
--target '{"RoleArn": "role-arn", "Arn": "QUEUE_ARN", "Input": "TEST_PAYLOAD" }' \
--flexible-time-window '{ "Mode": "OFF"}'
```

1 回限りのスケジュール

1 回限りのスケジュールは、有効な日付とタイムスタンプを使用して指定した日時に 1 回だけターゲットを呼び出します。EventBridge スケジューラは、協定世界時 (UTC) またはスケジュール作成時に指定したタイムゾーンでのスケジュール設定をサポートしています。

> **Note**
>
> 1 回限りのスケジュールは、ターゲットの実行と呼び出しが完了した後も、アカウントクォータに対してカウントされます。1 回限りのスケジュールは、実行が完了した後に削除することをおすすめします。

1 回限りのスケジュールは at 式を使用して設定します。at 式は、以下に示すように、EventBridge スケジューラでスケジュールを呼び出したい日付と時刻で構成されます。

構文

```
at(yyyy-mm-ddThh:mm:ss)
```

1 回限りのスケジュールを設定すると、EventBridge スケジューラはスケジュールに指定された `StartDate` と `EndDate` を無視します。

例

次の例は、 コマンドを使用して AWS CLI `create-schedule` 式で を使用して 1 回限りのスケジュールを設定する方法を示しています。この例では、2022 年 11 月 20 日の午後 1 時 (UTC-8) に実行されるスケジュールを作成し、テンプレート化された `SqsParameters` ターゲットタイプを使用して Amazon SQS キューにメッセージを配信します。

```
$ aws scheduler create-schedule --schedule-expression "at(2022-11-20T13:00:00)" --
name schedule-name \
--target '{"RoleArn": "role-arn", "Arn": "QUEUE_ARN", "Input": "TEST_PAYLOAD" }' \
--schedule-expression-timezone "America/Los_Angeles"
--flexible-time-window '{ "Mode": "OFF"}'
```

EventBridge スケジューラのタイムゾーン

EventBridge スケジューラでは、指定したタイムゾーンで cron ベースのスケジュールと 1 回限りのスケジュールを設定できます。EventBridge スケジューラは、Internet Assigned Numbers Authority (IANA) が管理するタイムゾーンデータベースを使用します。

では AWS CLI、 `--schedule-expression-timezone`パラメータを使用して EventBridge スケジューラでスケジュールを評価するタイムゾーンを設定できます。例えば、次のコマンドは、毎日午前 8 時 30 分にアメリカ/ニューヨークでテンプレート化された Amazon SQS SendMessage ターゲットを呼び出す cron ベースのスケジュールを作成します。

```
$ aws scheduler create-schedule --schedule-expression "cron(30 8 * * ? *)" --name
 schedule-in-est \
    --target '{"RoleArn": "role-arn", "Arn": "QUEUE_ARN", "Input": "This schedule runs
in the America/New_York time zone." }' \
    --schedule-expression-timezone "America/New_York"
    --flexible-time-window '{ "Mode": "OFF"}'
```

EventBridge スケジューラの夏時間

EventBridge スケジューラは、夏時間に合わせてスケジュールを自動的に調整します。春に時間が進むときに、cron 式が存在しない日時に該当すると、スケジュールの呼び出しはスキップされます。秋に時間が戻ったとき、スケジュールは 1 回だけ実行され、その呼び出しは繰り返されません。次の呼び出しは、通常、指定された日時に行われます。

EventBridge スケジューラは、スケジュールを作成するときに指定したタイムゾーンに応じてスケジュールを調整します。アメリカ/ニューヨークでスケジュールを設定した場合、そのタイムゾーンの時刻が変更されるとスケジュールが調整されます。一方、アメリカ/ロサンゼルスのスケジュールは、西海岸で時刻が変更されると 3 時間後に調整されます。

`rate(1 days)` など、単位として `days` を使用するレートベースのスケジュールの場合、`days` は時計上の 24 時間を表します。つまり、夏時間によって 1 日が 23 時間に短縮されたり、25 時間に延長されたりしても、EventBridge スケジューラはスケジュールが最後に呼び出されてから 24 時間後に rate 式を評価します。

> **Note**
>
> 地域の規則や規制により、一部のタイムゾーンでは夏時間が適用されません。夏時間に対応していないタイムゾーンでスケジュールを作成した場合、EventBridge スケジューラはスケ

ジュールを調整しません。夏時間の調整は、協定世界時 (UTC) のスケジュールには適用されません。

例

アメリカ/ロサンゼルスで次の cron 式を使用してスケジュールを作成するシナリオを考えてみましょう: `cron(30 2 * * ? *)` このスケジュールは、指定されたタイムゾーンで毎日午前 2 時 30 分に実行されます。

- スプリングフォワード — 春に午前 1 時 59 分から午前 3 時に時間が進むと、EventBridge スケジューラはその日のスケジュール呼び出しをスキップし、翌日には通常どおりスケジュールの実行を再開します。
- フォールバック — 秋に午前 2 時 59 分から午前 2 時に時間が戻ると、EventBridge スケジューラは時間が戻る前の午前 2 時 30 分に 1 回だけスケジュールを実行しますが、時間が変わった後の午前 2 時 30 分にはスケジュールの呼び出しを繰り返しません。

EventBridge スケジューラでのスケジュールの管理

スケジュールとは、Amazon EventBridge スケジューラを使用して作成、設定、管理する主なリソースです。

すべてのスケジュールには、スケジュールをいつ、どの頻度で実行するかを決定するスケジュール式があります。EventBridge スケジューラは、レート、cron、1 回限りのスケジュールの 3 種類のスケジュールをサポートしています。さまざまなスケジュールタイプの詳細については、「スケジュールタイプ」を参照してください。

スケジュールを作成するときは、そのスケジュールが呼び出すターゲットを設定します。ターゲットとは、スケジュールが実行されるたびに EventBridge スケジューラがユーザーに代わって呼び出す API オペレーションです。EventBridge スケジューラ は 2 種類のターゲットをサポートします。1 つはコアサービスグループ全体で共通の API オペレーションを呼び出すテンプレート化されたターゲットで、もう 1 つは 270 を超えるサービスにわたって 6,000 を超えるオペレーションを呼び出すことができるユニバーサルターゲットパラメーター (UTP) です。ターゲットの設定の詳細については、「ターゲットの管理」を参照してください。

EventBridge スケジューラがイベントをターゲットに正常に配信できなかった場合の、スケジュールで障害を処理する方法を設定するには、再試行ポリシーとデッドレターキュー (DLQ) という 2 つの主要なメカニズムを使用します。再試行ポリシーは、EventBridge スケジューラが失敗したイベントを再試行する回数と、未処理のイベントを保持する期間を決定します。DLQ は、EventBridge スケジューラが再試行ポリシーを使い果たした後に、失敗したイベントを配信するために使用する標準の Amazon SQS キューです。DLQ を使用して、スケジュールやダウンストリームターゲットの問題をトラブルシューティングできます。詳細については、「the section called “DLQ の設定”」を参照してください。

このセクションでは、コンソール、、および EventBridge スケジューラ SDKs を使用して AWS CLI EventBridge スケジューラのスケジュールを管理する例を示します。

トピック

EventBridge スケジューラでのスケジュール状態の変更

EventBridge スケジューラのスケジュールには、enabled と disabled の 2 つの状態があります。次の例では、UpdateSchedule を使用して、5 分ごとに起動して Lambda ターゲットを呼び出すスケジュールを無効にします。

UpdateSchedule を使用するときは、必要なパラメータをすべて指定する必要があります。EventBridge スケジューラは、ユーザーが提供した情報でスケジュールを置き換えます。以前に設定したパラメータを指定しないと、デフォルトの null に設定されます。

Example AWS CLI

```
$ aws scheduler update-schedule --name lambda-universal --schedule-expression 'rate(5
 minutes)' \
--target '{"RoleArn": "ROLE_ARN", "Arn":"arn:aws:scheduler:::aws-sdk:lambda:invoke"
 "Input": "{\"FunctionName\":\"arn:aws:lambda:REGION:123456789012:function:HelloWorld
\",\"InvocationType\":\"Event\",\"Payload\":\"{\\\"message\\\":\\\"testing function\\
\"}\"}" }' \
--flexible-time-window '{ "Mode": "OFF"}' \
--state DISABLED
```

```
{
    "ScheduleArn": "arn:aws:scheduler:us-west-2:123456789012:schedule/default/lambda-
universal"
}
```

次の例では、Python SDK と UpdateSchedule オペレーションを使用して、テンプレート化されたターゲットを使用して Amazon SQS をターゲットとするスケジュールを無効にします。

Example Python SDK

```
import boto3
scheduler = boto3.client('scheduler')

sqs_templated = {
    "RoleArn": "<ROLE_ARN>",
    "Arn": "<QUEUE_ARN>",
    "Input": "{}"}

flex_window = { "Mode": "OFF" }
```

```
scheduler.update_schedule(Name="your-schedule",
    ScheduleExpression="rate(5 minutes)",
    Target=sqs_templated,
    FlexibleTimeWindow=flex_window,
    State='DISABLED')
```

EventBridge スケジューラでの柔軟な時間枠の設定

柔軟な時間枠を使用してスケジュールを設定すると、EventBridge スケジューラは設定された時間枠内でターゲットを呼び出します。これは、ターゲットの呼び出しを正確にスケジュールする必要がない場合に便利です。柔軟な時間枠を設定すると、ターゲットの呼び出しが分散されるため、スケジュールの信頼性が向上します。

例えば、1 時間ごとに実行されるスケジュールに 15 分の柔軟な時間枠を設定すると、スケジュールされた時間から 15 分以内にターゲットを呼び出します。次の および EventBridge スケジューラ SDK の例では AWS CLI、 を使用してUpdateSchedule、1 時間に 1 回実行されるスケジュールに 15 分の柔軟な時間枠を設定します。

> **Note**
>
> 柔軟な時間枠を設定するかどうかを指定する必要があります。このオプションを設定しない場合は、OFF を指定します。値を FLEXIBLE に設定した場合は、スケジュールを実行する最大時間枠を指定する必要があります。

Example AWS CLI

```
$ aws scheduler update-schedule --name lambda-universal --schedule-expression 'rate(1
 hour)' \
--target '{"RoleArn": "ROLE_ARN", "Arn":"arn:aws:scheduler:::aws-sdk:lambda:invoke"
 "Input": "{\"FunctionName\":\"arn:aws:lambda:REGION:123456789012:function:HelloWorld
\",\"InvocationType\":\"Event\",\"Payload\":\"{\\\"message\\\":\\\"testing function\\
\"}\"}" }' \
--flexible-time-window '{ "Mode": "FLEXIBLE", "MaximumWindowInMinutes": 15} \
```

```
{
    "ScheduleArn": "arn:aws:scheduler:us-west-2:123456789012:schedule/lambda-universal"
}
```

Example Python SDK

```
import boto3
scheduler = boto3.client('scheduler')

sqs_templated = {
    "RoleArn": "<ROLE_ARN>",
    "Arn": "<QUEUE_ARN>",
    "Input": "{}"}

flex_window = { "Mode": "FLEXIBLE", "MaximumWindowInMinutes": 15}

scheduler.update_schedule(Name="your-schedule",
    ScheduleExpression="rate(1 hour)",
    Target=sqs_templated,
    FlexibleTimeWindow=flex_window)
```

EventBridge スケジューラにおけるスケジュールのデッドレターキューの設定

Amazon EventBridge スケジューラでは、Amazon Simple Queue Service を使用してデッドレターキュー (DLQ) をサポートしています。スケジュールがターゲットの呼び出しに失敗すると、EventBridge スケジューラは、呼び出しの詳細とターゲットから受信したレスポンスを含む JSON ペイロードを、指定した Amazon SQS 標準キューに配信します。

次のトピックでは、この JSON をデッドレターイベントと呼んでいます。デッドレターイベントを使用すると、スケジュールやターゲットに関する問題をトラブルシューティングできます。スケジュールに再試行ポリシーを設定すると、EventBridge スケジューラは、設定された最大再試行回数を使い切ったデッドレターイベントを配信します。

以下のトピックでは、Amazon SQS キューをスケジュールの DLQ として設定する方法、EventBridge スケジューラが Amazon SQS にメッセージを配信するために必要な権限を設定する方法、および DLQ からデッドレターイベントを受信する方法について説明します。

トピック

- Amazon SQS キュー を作成する
- 実行ロールのアクセス許可を設定する
- デッドレターキューを指定する

- デッドレターイベントの取得

Amazon SQS キュー を作成する

スケジュールに DLQ を設定する前に、標準の Amazon SQS キューを作成する必要があります。Amazon SQS コンソールを使用してキューを作成する手順については、「Amazon Simple Queue Service デベロッパーガイド」の「Amazon SQS キューの作成」を参照してください。

> **Note**
>
> EventBridge スケジューラは、スケジュールの DLQ として FIFO キューを使用することをサポートしていません。

次の AWS CLI コマンドを使用して、標準キューを作成します。

```
$ aws sqs create-queue --queue-name queue-name
```

成功すると、出力に QueueURL が表示されます。

```
{
    "QueueUrl": "https://sqs.us-west-2.amazonaws.com/123456789012/scheduler-dlq-test"
}
```

キューを作成したら、キュー ARN を書き留めておきます。EventBridge スケジューラのスケジュールに DLQ を指定するときに、ARN が必要になります。キュー ARN は、Amazon SQS コンソールで、または get-queue-attributes AWS CLI コマンドを使用して見つけることができます。

```
$ aws sqs get-queue-attributes --queue-url your-dlq-url --attribute-names QueueArn
```

成功すると、出力にキュー ARN が表示されます。

```
{
    "Attributes": {
        "QueueArn": "arn:aws:sqs:us-west-2:123456789012:scheduler-dlq-test"
    }
}
```

次のセクションでは、スケジュール実行ロールに必要なアクセス権限を追加して、EventBridge スケジューラがデッドレターイベントを Amazon SQS に配信できるようにします。

実行ロールのアクセス許可を設定する

EventBridge スケジューラがデッドレターイベントを Amazon SQS に配信できるようにするには、スケジュール実行ロールに以下のアクセス権限ポリシーが必要です。スケジュール実行ロールに新しいアクセス権限ポリシーをアタッチする方法の詳細については、「実行ロールの設定」を参照してください。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sqs:SendMessage"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> EventBridge スケジューラを使用して Amazon SQS API ターゲットを呼び出す場合、スケジュール実行ロールには必要な権限がすでにアタッチされている可能性があります。

次のセクションでは、EventBridge スケジューラのコンソールを使用して、スケジュールの DLQ を指定します。

デッドレターキューを指定する

DLQ を指定するには、EventBridge スケジューラコンソールまたは AWS CLI を使用して既存のスケジュールを更新するか、新しいスケジュールを作成します。

Console

コンソールを使用して DLQ を指定する方法

1. にサインインし AWS マネジメントコンソール、次のリンクを選択して EventBridge コンソールの EventBridge スケジューラセクションを開きます。https://console.aws.amazon.com/scheduler/home
2. EventBridge スケジューラのコンソールで、新しいスケジュールを作成するか、スケジュールのリストから既存のスケジュールを選択して編集します。
3. [設定] ページの [デッドレターキュー (DLQ)] で、次のいずれかを実行します。
   - AWS アカウントの Amazon SQS キューを DLQ として選択を選択し、ドロップダウンリストから DLQ のキュー ARN を選択します。
   - 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、DLQ のキュー ARN を入力します。別の AWS アカウントのキューを選択した場合、EventBridge スケジューラのコンソールはキュー ARN をドロップダウンリストに表示できません。
4. 選択内容を確認し、[スケジュールを作成] または [スケジュールを保存] を選択して DLQ の設定を完了します。
5. (オプション) スケジュールの DLQ の詳細を表示するには、一覧からスケジュールの名前を選択し、[スケジュール詳細] ページの [デッドレターキュー] タブを選択します。

AWS CLI

を使用して既存のスケジュールを更新するには AWS CLI

- update-schedule コマンドを使用してスケジュールを更新します。以前に作成した Amazon SQS キューを DLQ として指定します。必要な Amazon SQS アクセス権限をアタッチした IAM ロール ARN を実行ロールとして指定します。他のすべてのプレースホルダー値を、ユーザー自身の情報に置き換えます。

```
$ aws scheduler update-schedule --name existing-schedule \
    --schedule-expression 'rate(5 minutes)' \
    --target '{"DeadLetterConfig": {"Arn": "DLQ_ARN"}, "RoleArn": "ROLE_ARN",
 "Arn":"QUEUE_ARN", "Input": "Hello world!" }' \
    --flexible-time-window '{ "Mode": "OFF"}'
```

を使用して DLQ で新しいスケジュールを作成するには AWS CLI

- スケジュールを作成するには、create-schedule コマンドを使用します。すべてのプレースホルダー値を、ユーザー自身の情報に置き換えます。

```
$ aws scheduler create-schedule --name new-schedule \
    --schedule-expression 'rate(5 minutes)' \
    --target '{"DeadLetterConfig": {"Arn": "DLQ_ARN"}, "RoleArn": "ROLE_ARN",
 "Arn":"QUEUE_ARN", "Input": "Hello world!" }' \
    --flexible-time-window '{ "Mode": "OFF"}'
```

次のセクションでは、 AWS CLI を使用して DLQ からデッドレターイベントを受信します。

デッドレターイベントの取得

次に示す receive-message コマンドを使用して、DLQ からデッドレターイベントを取得します。--max-number-of-messages 属性を使用して、取得するメッセージの数を設定できます。

```
$ aws sqs receive-message --queue-url your-dlq-url --attribute-names All --message-
attribute-names All --max-number-of-messages 1
```

成功すると、次のような出力が表示されます。

```
{
    "Messages": [
        {
            "MessageId": "2aeg3510-fe3a-4f5a-ab6a-6906560eaf7e",
            "ReceiptHandle": "AQEBkNKTdOMrWgHKPoITRBwrPoK3eCSZIcZwVqCY0BZ
+FfTcORFpopJbtCqj36VbBTlHreM8+qM/m5jcwqSlAlGmIJO/hYmMgn/
+dwIty9izE7HnpvRhhEyHxbeTZ5V05RbeasYaBdNyi9WLcnAHviDh6MebLXXNWoFyYNsxdwJuG0f/
w3htX6r3dxpXvvFNPGoQb8ihY37+u0gtsbuIwhLtUSmE8rbldEEwiUfi3IJ1zEZpUS77n/k1GWrMrnYg0Gx/
BuaLzOrFi2F738XI/
Hnh45uv3ca6OYwS1ojPQ1LtX2URg1haV5884FYlaRvY8jRlpCZabTkYRTZKSXG5KNgYZnHpmsspii6JNkjitYVFKPo0H91w
            "MD5OfBody": "07adc3fc889d6107d8bb8fda42fe0573",
            "Body": "{\"MessageBody\":\"Hello, world!\",\"QueueUrl\":\"https://sqs.us-
west-2.amazonaws.com/123456789012/does-not-exist\"}",
            "Attributes": {
                "SenderId": "AROA2DZE3W4CTL5ZR7EIN:ff00212d8c453aaaae644bc6846d4723",
                "ApproximateFirstReceiveTimestamp": "1652499058144",
                "ApproximateReceiveCount": "2",
                "SentTimestamp": "1652490733042"
```

```
            },
            "MD5OfMessageAttributes": "f72c1d78100860e00403d849831d4895",
            "MessageAttributes": {
                "ERROR_CODE": {
                    "StringValue": "AWS.SimpleQueueService.NonExistentQueue",
                    "DataType": "String"
                },
                "ERROR_MESSAGE": {
                    "StringValue": "The specified queue does not exist for this wsdl
 version.",
                    "DataType": "String"
                },
                "EXECUTION_ID": {
                    "StringValue": "ad06616e51cdf74a",
                    "DataType": "String"
                },
                "EXHAUSTED_RETRY_CONDITION": {
                    "StringValue": "MaximumEventAgeInSeconds",
                    "DataType": "String"
                }
                "IS_PAYLOAD_TRUNCATED": {
                    "StringValue": "false",
                    "DataType": "String"
                },
                "RETRY_ATTEMPTS": {
                    "StringValue": "0",
                    "DataType": "String"
                },
                "SCHEDULED_TIME": {
                    "StringValue": "2022-05-14T01:12:00Z",
                    "DataType": "String"
                },
                "SCHEDULE_ARN": {
                    "StringValue": "arn:aws:scheduler:us-west-2:123456789012:schedule/
DLQ-test",
                    "DataType": "String"
                },
                "TARGET_ARN": {
                    "StringValue": "arn:aws:scheduler:::aws-sdk:sqs:sendMessage",
                    "DataType": "String"
                }
            }
        }
    ]
```

```
}
```

デッドレターイベントの以下の属性に注目しておくと、ターゲットの呼び出しが失敗した原因として考えられるものの特定とトラブルシューティングに役立ちます。

- **ERROR_CODE** — EventBridge スケジューラがターゲットのサービス API から受け取るエラーコードが含まれます。前述の例では、Amazon SQS によって返されるエラーコードは `AWS.SimpleQueueService.NonExistentQueue` です。EventBridge スケジューラの問題によりスケジュールがターゲットの呼び出しに失敗した場合、代わりに次のエラーコードが表示されます: `AWS.Scheduler.InternalServerError`
- **ERROR_MESSAGE** — EventBridge スケジューラがターゲットのサービス API から受け取るエラーメッセージが含まれます。前述の例では、Amazon SQS によって返されるエラーメッセージは `The specified queue does not exist for this wsdl version` です。EventBridge スケジューラの問題によりスケジュールが失敗した場合、代わりに次のエラーメッセージが表示されます: `Unexpected error occurred while processing the request`
- **TARGET_ARN** — スケジュールが呼び出すターゲットの ARN。サービス ARN 形式は次のとおりです: `arn:aws:scheduler:::aws-sdk:service:apiAction`
- **EXHAUSTED_RETRY_CONDITION** — イベントが DLQ に配信された理由を示します。この属性は、ターゲット API からのエラーが永続的なエラーではなく、再試行可能なエラーである場合に表示されます。この属性には、スケジュールに設定した最大再試行回数を超えた後に EventBridge スケジューラが DLQ に送信した場合は `MaximumRetryAttempts` の値、またはイベントがスケジュールに設定した最大経過時間よりも古く、依然として配信に失敗している場合は `MaximumEventAgeInSeconds` の値が含まれる場合があります。

前の例では、エラーコードとエラーメッセージに基づいて、スケジュールに指定したターゲットキューが存在しないと判断できます。

> **Note**
>
> ユニバーサルターゲットを使用する場合、EventBridge スケジューラはスケジュール作成時に`Input`フィールドの内容を検証しないことに注意してください。無効な入力パラメータを含むスケジュールは正常に作成されますが、呼び出しのたびに失敗します。DLQ メッセージには、ターゲットサービスからのエラーコードとメッセージが含まれており、無効なパラメータを識別するのに役立ちます。詳細については、「無効なユニバーサルターゲット入力設定」を参照してください。

EventBridge スケジューラでのスケジュールの削除

スケジュールを削除するには、自動削除を設定するか、個々のスケジュールを手動で削除します。次のトピックでは、両方の方法を使用してスケジュールを削除する方法と、1 つの方法を選択する理由について説明します。

トピック

スケジュール完了後の削除

EventBridge スケジューラでスケジュールリソースを個別に管理する必要がないようにするには、スケジュール完了後に自動削除を設定します。一度に数千のスケジュールを作成し、必要に応じてスケジュール数を柔軟にスケールアップする必要がある用途では、自動削除を設定することで、指定したリージョンのスケジュール数に対するアカウントクォータに達しないようにすることができます。

スケジュールの自動削除を設定すると、EventBridge スケジューラは最後のターゲット呼び出しの後にスケジュールを削除します。1 回限りのスケジュールの場合は、スケジュールがターゲットを一度呼び出した後に削除が実行されます。rate 式または cron 式を使用して設定した繰り返しのスケジュールの場合、スケジュールは最後の呼び出し後に削除されます。繰り返しのスケジュールの最後の呼び出しは、指定した `EndDate` に最も近い呼び出しです。スケジュールに自動削除を設定しても、`EndDate` の値を指定しなかった場合、EventBridge スケジューラはスケジュールを自動的に削除しません。

スケジュールを最初に作成するときに自動削除を設定したり、既存のスケジュールの設定を更新したりできます。次のステップでは、既存のスケジュールの自動削除を設定する方法について説明します。

AWS マネジメントコンソール

1. EventBridge スケジューラのコンソール (https://console.aws.amazon.com/scheduler/) を開きます。
2. スケジュールされたリストから、編集するスケジュールを選択し、[編集] を選択します。
3. 左のナビゲーションリストから、[設定] を選択します。
4. [スケジュール完了後のアクション] セクションで、ドロップダウンリストから [削除] を選択し、変更を保存します。

AWS CLI

1. 新しいプロンプトウィンドウを開きます。
2. update-schedule AWS CLI コマンドを使用して、次に示す既存のスケジュールを更新します。このコマンドは `--action-after-completion` を `DELETE` に設定します。この例では、ターゲット設定が JSON ファイルでローカルに定義されていることを前提としています。スケジュールを更新するには、ターゲットのほか、既存のスケジュールに設定したいその他のスケジュールパラメータを指定する必要があります。

   これは 1 時間に 1 回の頻度で呼び出しを行う繰り返しのスケジュールです。そのため、`--action-after-completion` パラメータを設定する際に終了日を指定します。

```
$ aws scheduler update-schedule --name schedule-name
 \
--action-after-completion 'DELETE' \
--schedule-expression 'rate(1 hour)' \
--end-date '2024-01-01T00:00:00'
--target file://target-configuration.json \
--flexible-time-window '{ "Mode": "OFF"}' \
```

手動削除

スケジュールが不要になった場合には、DeleteSchedule オペレーションを使用して削除することができます。

Example AWS CLI

```
$ aws scheduler delete-schedule --name your-schedule
```

Example Python SDK

```
import boto3
scheduler = boto3.client('scheduler')

scheduler.delete_schedule(Name="your-schedule")
```

次のステップ

- Lambda と Step Functions のテンプレート化されたターゲットを設定する方法の詳細と、ユニバーサルターゲットパラメータの使用方法については、「ターゲットの管理」を参照してください。
- EventBridge スケジューラのデータ型と API オペレーションの詳細については、「EventBridge スケジューラ API リファレンス」を参照してください。

EventBridge スケジューラでのスケジュールグループの管理

スケジュールグループとは、スケジュールを整理するために使用する Amazon EventBridge スケジューラのリソースです。

AWS アカウント には `default` スケジューラグループが付属しています。新しいスケジュールは、`default` グループ、または自分で作成して管理するスケジュールグループに関連付けることができます。AWS アカウント には最大 500 個のスケジュールグループを作成できます。EventBridge スケジューラでは、タグを適用することで、個々のスケジュールではなく、スケジュールグループを整理できます。

タグとは、ユーザーが定義する大文字と小文字を区別するキーと値で構成されるラベルです。タグを作成して、目的、所有者、環境などの基準に基づいてスケジュールを分類できます。例えば、次のタグを使用して、スケジュールが属する環境を特定できます: `environment:production`

> **Important**
>
> 個人情報 (PII) などの機密情報や秘匿性の高い情報はタグに追加しないようにします。タグは、多くの AWS のサービス (請求など) からアクセスできます。タグは、プライベートデータや機密データに使用することを意図していません。

スケジュールグループには、ACTIVE と DELETING という 2 つの状態があります。

最初にグループを作成すると、デフォルトでは `ACTIVE` になっています。スケジュールは `ACTIVE` グループに追加できます。グループを削除すると、EventBridge スケジューラが関連付けれらたスケジュールの削除を完了するまで状態は `DELETING` に変わります。EventBridge スケジューラがグループ内のスケジュールを削除すると、そのグループはアカウントで使用できなくなります。

以下のトピックでは、スケジュールグループを作成し、タグを適用します。また、スケジュールをグループに関連付けます。最後に、グループを削除します。

トピック

EventBridge スケジューラでのスケジュールグループの作成

スケジュールグループとタグ付けを使用して、共通の目的を共有するスケジュールや同じ環境に属するスケジュールを整理します。次の手順では、新しいスケジュールグループを作成し、タグを使用してラベルを付けます。次に、新しいスケジュールをそのグループに関連付けます。

> **Note**
>
> グループを作成すると、そのグループからスケジュールを削除したり、そのスケジュールを別のグループに関連付けることはできません。スケジュールをグループに関連付けることができるのは、最初にスケジュールを作成したときだけです。

ステップ 1: 新しいスケジュールグループを作成する

以下のトピックでは、新しいスケジュールグループを作成して次のタグを使用してラベルを付ける方法について説明します: `environment:development`

AWS マネジメントコンソール

AWS マネジメントコンソール を使用して新しいグループを作成する方法

1. AWS マネジメントコンソール にサインインし、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、[スケジュールグループ] をクリックします。
3. [スケジュールグループ] ページで、[スケジュールグループを作成] を選択します。
4. [スケジュールグループ詳細] セクションの [名前] に、グループの名前を入力します。例えば、**`TestGroup`**。
5. [タグ] セクションで、次の操作を行います。
   a. [新しいタグを追加] をクリックします。
   b. [キー] には、このキーに割り当てる名前を入力します。このチュートリアルでは、このスケジュールグループが属する環境にラベルを付けるには、**`environment`** と入力します。
   c. [値 - オプション] には、このキーに割り当てる値を入力します。このチュートリアルでは、環境キーの値 **`development`** を入力します。

> **Note**
>
> グループを作成した後に、タグを追加できます。

6. [スケジュールグループを作成] を選択して終了します。新しいグループが [スケジュールグループ] リストに表示されます。
7. (オプション) グループを編集したりタグを管理したりするには、新しいグループのチェックボックスを選択して [編集] を選択します。

> **Note**
>
> `default` スケジュールグループは編集できません。

AWS CLI

AWS CLI を使用して新しいグループを作成する方法

1. 新しいコマンドプロンプトウィンドウを開きます。
2. AWS Command Line Interface (AWS CLI) から、次の `create-schedule-group` コマンドを入力して、新しいグループを作成します。このコマンドは、1 つのタグ `environment:development` を使用してグループを作成します。このタグまたは類似のタグ付けシステムを使用して、スケジュールグループが属する環境に応じてラベルを付けることができます。

   スケジュール名とタグのキーと値を自分自身の情報に置き換えます。

```
$ aws scheduler create-schedule-group --name TestGroup --tags
 Key=environment,Value=development
```

デフォルトでは、新しいグループの状態は `ACTIVE` になります。これで、作成した新しいグループに新しいスケジュールを関連付けることができます。

ステップ 2: スケジュールをグループに関連付ける

次の手順を使用して、前のステップで作成したグループに新しいスケジュールを関連付けます。

AWS マネジメントコンソール

AWS マネジメントコンソール を使用してスケジュールをグループに関連付ける方法

1. AWS マネジメントコンソール にサインインし、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で、[スケジュール] を選択します。
3. [スケジュール] テーブルから [スケジュールを作成] を選択し、新しいスケジュールを作成します。
4. [スケジュールの詳細を指定] ページの [スケジュールグループ] で、ドロップダウンリストから新しいグループの名前を選択します。例えば、TestGroup を選択します。
5. スケジュールのパターン、ターゲット、設定を指定し、[スケジュールの確認と保存] ページで選択内容を確認します。新しいスケジュールの設定の詳細については、「開始方法」を参照してください。
6. スケジュールを終了して保存するには、[スケジュールを保存] を選択します。

AWS CLI

AWS CLI を使用してスケジュールをグループに関連付ける方法

1. 新しいコマンドプロンプトウィンドウを開きます。
2. AWS Command Line Interface (AWS CLI) から、次の `create-schedule` コマンドを入力します。これにより、スケジュールが作成され、前のステップの `sqs-test-schedule` という名前のグループに関連付けられます。このスケジュールでは、テンプレート化された Amazon SQS ターゲットタイプを使用して SendMessage オペレーションを呼び出します。スケジュール名、ターゲット、およびグループ名を自分自身の情報に置き換えます。

```
$ aws scheduler create-schedule --name sqs-test-schedule --schedule-expression
 'rate(5 minutes)' \
--target '{"RoleArn": "ROLE_ARN", "Arn": "QUEUE_ARN", "Input": "TEST_PAYLOAD" }'
 \
--group-name TestGroup
--flexible-time-window '{ "Mode": "OFF"}'
```

これで、新しいスケジュールが TestGroup スケジュールグループに関連付けられました。

EventBridge スケジューラでのスケジュールグループの削除

以下では、AWS マネジメントコンソール と AWS Command Line Interface を使用してスケジュールグループを削除する方法を説明します。グループを削除すると、EventBridge スケジューラがグループ内のすべてのスケジュールを削除するまでグループの状態は `DELETING` になります。EventBridge スケジューラがグループ内のスケジュールを削除すると、そのグループはアカウントで使用できなくなります。

> **Note**
>
> グループを作成すると、そのグループからスケジュールを削除したり、そのスケジュールを別のグループに関連付けることはできません。スケジュールをグループに関連付けることができるのは、最初にスケジュールを作成したときだけです。

AWS マネジメントコンソール

AWS マネジメントコンソール を使用してグループを削除する方法

1. AWS マネジメントコンソール にサインインし、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で、[スケジュールグループ] を選択します。
3. [スケジュールグループ] ページで、現在の AWS リージョン の既存のグループのリストから、削除するグループを探します。探しているグループが表示されない場合は、別の AWS リージョン グループを選択してください。

   > **Note**
   >
   > default グループを削除したり編集したりすることはできません。

4. 削除するグループのチェックボックスをオンにします。
5. [削除] を選択します。
6. [スケジュールグループの削除] ダイアログボックスで、グループの名前を入力して選択を確定し、[削除] を選択します。
7. [スケジュールグループ] リストの [ステータス] 列が変わり、グループが削除中であることが示されます。EventBridge スケジューラがグループに関連付けられているスケジュールをすべて削除するまで、グループはこの状態のままになります。

8. リストを更新してグループが削除されたことを確認するには、[更新] アイコンを選択します。

AWS CLI

AWS CLI を使用してグループを削除する方法

1. 新しいコマンドプロンプトウィンドウを開きます。
2. AWS Command Line Interface(AWS CLI) から、次の delete-schedule-group コマンドを入力してスケジュールグループを削除します。--name の値を自分自身の情報に置き換えます。

```
$ aws scheduler delete-schedule-group --name TestGroup
```

成功すると、この AWS CLI オペレーションは応答を返しません。

3. グループが DELETING の状態にあることを確認するには、以下の get-schedule-group コマンドを実行します。

```
$ aws scheduler get-schedule-group --name TestGroup
```

成功すると、次のような出力が表示されます。

```
{
    "Arn": "arn:aws::scheduler:us-west-2:123456789012:schedule-group/TestGroup",
    "CreationDate": "2023-01-01T09:00:00.000000-07:00",
    "LastModificationDate": "2023-01-01T09:00:00.000000-07:00",
    "Name": "TestGroup",
    "State": "DELETING"
}
```

EventBridge スケジューラは、グループに関連付けられているスケジュールを削除した後でグループを削除します。get-schedule-group をもう一度実行すると、次の ResourceNotFoundException 応答が返されます。

```
An error occurred (ResourceNotFoundException) when calling the GetScheduleGroup
 operation: Schedule group TestGroup does not exist.
```

関連リソース

スケジュールグループの詳細については、以下のリソースを参照してください。

- 「EventBridge スケジューラ API リファレンス」の CreateScheduleGroup オペレーション。
- 「EventBridge スケジューラ API リファレンス」の DeleteScheduleGroup オペレーション。

EventBridge スケジューラでのターゲットの管理

以下のトピックでは、EventBridge スケジューラでテンプレート化されたユニバーサルターゲットを使用する方法について説明し、EventBridge スケジューラのユニバーサルターゲットパラメータを使用して設定できるサポートされている AWS サービスのリストを提供します。

テンプレート化されたターゲットは、Amazon SQS、Step Functions などのコア AWS サービスのグループにわたる一般的な API オペレーションのセットです。例えば、関数 ARN を指定することで Lambda の Invoke API オペレーションをターゲットにしたり、ターゲットのキュー ARN を使用して Amazon SQS の `SendMessage` オペレーションをターゲットにしたりできます。

ユニバーサルターゲットはカスタマイズ可能なパラメータのセットであり、多くの AWS サービスに対してより広範な API オペレーションのセットを呼び出すことができます。例えば、EventBridge スケジューラのユニバーサルターゲットパラメータ (UTP) を使用して、`CreateQueue` オペレーションを使用して新しい Amazon SQS キューを作成できます。

テンプレート化されたターゲットまたはユニバーサルターゲットのいずれかを設定するには、ターゲットとして設定した API オペレーションを呼び出す権限がスケジュールに含まれている必要があります。そのためには、スケジュールの実行ロールに必要なアクセス許可をアタッチします。例えば、Amazon SQS の `SendMessage` オペレーションをターゲットにするには、実行ロールに `sqs:SendMessage` アクションを実行する権限を付与します。ほとんどの場合、ターゲットサービスがサポートする AWS マネージドポリシーを使用して必要なアクセス権限を追加できます。ただし、独自のカスタマー管理ポリシーを作成したり、実行ロールにアタッチされた既存のポリシーにインライン権限を追加したりすることもできます。以下のトピックでは、テンプレート化されたターゲットタイプとユニバーサルターゲットタイプの両方にアクセス権限を追加する例を示しています。

スケジュールの実行ロールのセットアップについては、「the section called “実行ロールを設定する”」を参照してください。

トピック

EventBridge スケジューラでのテンプレート化されたターゲットの使用

テンプレート化されたターゲットは、Amazon SQS、Step Functions などのコア AWS サービスのグループ全体の一般的な API オペレーションのセットです。例えば、関数 ARN を指定することで Lambda の `Invoke` オペレーションをターゲットにしたり、キュー ARN を使用して Amazon SQS の `SendMessage` オペレーションをターゲットにしたりできます。テンプレート化されたターゲットを設定するには、ターゲットの API オペレーションを実行するためのアクセス権限をスケジュールの実行ロールに付与する必要もあります。

AWS CLI または EventBridge スケジューラ SDKs のいずれかを使用してプログラムでテンプレート化されたターゲットを設定するには、実行ロールの ARN、ターゲットリソースの ARN、EventBridge スケジューラがターゲットに配信するオプションの入力、および一部のテンプレート化されたターゲットでは、そのターゲットに追加の設定オプションを含む一意のパラメータセットを指定する必要があります。テンプレート化されたターゲットリソースの ARN を指定すると、EventBridge スケジューラは、そのサービスでサポートされている API オペレーションを呼び出したいと自動的に想定します。EventBridge スケジューラでサービスの別の API オペレーションをターゲットにする場合は、ターゲットをユニバーサルターゲットとして設定する必要があります。

以下は、EventBridge スケジューラがサポートしているすべてのテンプレート化されたターゲット、および該当する場合は各ターゲット固有の関連パラメータの完全なリストです。各パラメータセットのリンクを選択すると、EventBridge スケジューラ API リファレンスの必須フィールドとオプションフィールドが表示されます。

- CodeBuild – `StartBuild`
- CodePipeline – `StartPipelineExecution`
- Amazon ECS – `RunTask`
  - パラメータ: `EcsParameters`
- EventBridge – `PutEvents`
  - パラメータ: `EventBridgeParameters`
- Amazon Inspector – `StartAssessmentRun`
- Kinesis – `PutRecord`
  - パラメータ: `KinesisParameters`
- Firehose – `PutRecord`
- Lambda – `Invoke`

- SageMaker AI – StartPipelineExecution
  - パラメータ: SageMakerPipelineParameters
- Amazon SNS – Publish
- Amazon SQS – SendMessage
  - パラメータ: SqsParameters
- Step Functions – StartExecution

以下の例では、さまざまなテンプレート化されたターゲットを設定する方法と、説明されている各ターゲットに必要な IAM アクセス許可について説明します。

Amazon SQS **SendMessage**

Example実行ロールのアクセス権限ポリシー

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sqs:SendMessage"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Example AWS CLI

```
$ aws scheduler create-schedule --name sqs-templated --schedule-expression 'rate(5
 minutes)' \
--target '{"RoleArn": "ROLE_ARN", "Arn":"QUEUE_ARN", "Input": "Message for scheduleArn:
 '<aws.scheduler.schedule-arn>', scheduledTime: '<aws.scheduler.scheduled-time>'" }' \
--flexible-time-window '{ "Mode": "OFF"}'
```

Example Python SDK

```
import boto3
scheduler = boto3.client('scheduler')

flex_window = { "Mode": "OFF" }

sqs_templated = {
    "RoleArn": "<ROLE_ARN>",
    "Arn": "<QUEUE_ARN>",
    "Input": "Message for scheduleArn: '<aws.scheduler.schedule-arn>', scheduledTime:
 '<aws.scheduler.scheduled-time>'"
}

scheduler.create_schedule(
    Name="sqs-python-templated",
    ScheduleExpression="rate(5 minutes)",
    Target=sqs_templated,
    FlexibleTimeWindow=flex_window)
```

Example Java SDK

```
package com.example;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.scheduler.SchedulerClient;
import software.amazon.awssdk.services.scheduler.model.*;


public class MySchedulerApp {

    public static void main(String[] args) {

        final SchedulerClient client = SchedulerClient.builder()
                .region(Region.US_WEST_2)
                .build();

        Target sqsTarget = Target.builder()
                .roleArn("<ROLE_ARN>")
                .arn("<QUEUE_ARN>")
                .input("Message for scheduleArn: '<aws.scheduler.schedule-arn>',
 scheduledTime: '<aws.scheduler.scheduled-time>'")
                .build();
```

```
        CreateScheduleRequest createScheduleRequest = CreateScheduleRequest.builder()
                .name("<SCHEDULE NAME>")
                .scheduleExpression("rate(10 minutes)")
                .target(sqsTarget)
                .flexibleTimeWindow(FlexibleTimeWindow.builder()
                        .mode(FlexibleTimeWindowMode.OFF)
                        .build())
                .build();

        client.createSchedule(createScheduleRequest);
        System.out.println("Created schedule with rate expression and an Amazon SQS
 templated target");
    }
}
```

Lambda **Invoke**

Example実行ロールのアクセス権限ポリシー

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Example AWS CLI

```
$ aws scheduler create-schedule --name lambda-templated-schedule --schedule-expression
 'rate(5 minutes)' \
--target '{"RoleArn": "ROLE_ARN", "Arn":"FUNCTION_ARN", "Input": "{ \"Payload\":
 \"TEST_PAYLOAD\" }" }' \
```

```
--flexible-time-window '{ "Mode": "OFF"}'
```

Example Python SDK

```
import boto3
scheduler = boto3.client('scheduler')

flex_window = { "Mode": "OFF" }

lambda_templated = {
    "RoleArn": "<ROLE_ARN>",
    "Arn": "<LAMBDA_ARN>",
    "Input": "{ 'Payload': 'TEST_PAYLOAD' }"}
}

scheduler.create_schedule(
    Name="lambda-python-templated",
    ScheduleExpression="rate(5 minutes)",
    Target=lambda_templated,
    FlexibleTimeWindow=flex_window)
```

Example Java SDK

```
package com.example;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.scheduler.SchedulerClient;
import software.amazon.awssdk.services.scheduler.model.*;


public class MySchedulerApp {

    public static void main(String[] args) {

        final SchedulerClient client = SchedulerClient.builder()
                .region(Region.US_WEST_2)
                .build();

        Target lambdaTarget = Target.builder()
                .roleArn("<ROLE_ARN>")
                .arn("<Lambda ARN>")
                .input("{ 'Payload': 'TEST_PAYLOAD' }")
                .build();
```

```
        CreateScheduleRequest createScheduleRequest = CreateScheduleRequest.builder()
                .name("<SCHEDULE_NAME>")
                .scheduleExpression("rate(10 minutes)")
                .target(lambdaTarget)
                .flexibleTimeWindow(FlexibleTimeWindow.builder()
                        .mode(FlexibleTimeWindowMode.OFF)
                        .build())
                .clientToken("<Token GUID>")
                .build();

        client.createSchedule(createScheduleRequest);
        System.out.println("Created schedule with rate expression and Lambda templated
 target");
    }
}
```

Step Functions **StartExecution**

Example実行ロールのアクセス権限ポリシー

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "states:StartExecution"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Example AWS CLI

```
$ aws scheduler create-schedule --name sfn-templated-schedule --schedule-expression
 'rate(5 minutes)' \
```

```
--target '{"RoleArn": "ROLE_ARN", "Arn":"STATE_MACHINE_ARN", "Input": "{ \"Payload\":
 \"TEST_PAYLOAD\" }" }' \
--flexible-time-window '{ "Mode": "OFF"}'
```

Example Python SDK

```
import boto3
scheduler = boto3.client('scheduler')

flex_window = { "Mode": "OFF" }

sfn_templated= {
    "RoleArn": "<ROLE_ARN>",
    "Arn": "<STATE_MACHINE_ARN>",
    "Input": "{ 'Payload': 'TEST_PAYLOAD' }"
}

scheduler.create_schedule(Name="sfn-python-templated",
    ScheduleExpression="rate(5 minutes)",
    Target=sfn_templated,
    FlexibleTimeWindow=flex_window)
```

Example Java SDK

```
package com.example;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.scheduler.SchedulerClient;
import software.amazon.awssdk.services.scheduler.model.*;


public class MySchedulerApp {

    public static void main(String[] args) {

        final SchedulerClient client = SchedulerClient.builder()
                .region(Region.US_WEST_2)
                .build();

        Target stepFunctionsTarget = Target.builder()
                .roleArn("<ROLE_ARN>")
                .arn("<STATE_MACHINE_ARN>")
                .input("{ 'Payload': 'TEST_PAYLOAD' }")
```

```
                .build();

        CreateScheduleRequest createScheduleRequest = CreateScheduleRequest.builder()
                .name("<SCHEDULE_NAME>")
                .scheduleExpression("rate(10 minutes)")
                .target(stepFunctionsTarget)
                .flexibleTimeWindow(FlexibleTimeWindow.builder()
                        .mode(FlexibleTimeWindowMode.OFF)
                        .build())
                .clientToken("<Token GUID>")
                .build();

        client.createSchedule(createScheduleRequest);
        System.out.println("Created schedule with rate expression and Step Function
 templated target");
    }
}
```

EventBridge スケジューラでのユニバーサルターゲットの使用

ユニバーサルターゲットはカスタマイズ可能なパラメータのセットであり、多くの AWS サービスに対してより広範な API オペレーションのセットを呼び出すことができます。例えば、ユニバーサルターゲットパラメータ (UTP) を使用して、CreateQueue オペレーションを使用して新しい Amazon SQS キューを作成できます。

AWS CLIまたは EventBridge スケジューラ SDKs のいずれかを使用してスケジュールのユニバーサルターゲットを設定するには、次の情報を指定する必要があります。

- RoleArn — ターゲットに使用したい実行ロールの ARN。指定する実行ロールには、スケジュールの対象とする API オペレーションを呼び出す権限が必要です。
- Arn — ターゲットとする API オペレーションを含むサービス ARN 全体を、次の形式で示します: `arn:aws:scheduler:::aws-sdk:service:apiAction`

  例えば、Amazon SQS の場合、指定するサービス名は `arn:aws:scheduler:::aws-sdk:sqs:sendMessage` です。

> **Note**
>
> ユニバーサルターゲット ARN #####値は、ターゲットサービスの AWS SDK サービス識別子と一致する必要があります。この識別子は、サービスのエンドポイントプレフィッ

クスとは異なる場合があります。例えば、Amazon Cognito ID プロバイダーの場合は、`cognitoidentityprovider` (ではなく) を使用しますcognito-idp。正しいサービス識別子を見つけるには、ターゲットにするサービスの AWS SDK ドキュメントを参照してください。

- Input — EventBridge スケジューラがターゲット API に送信するリクエストパラメータで指定する、正しい形式の JSON。`Input` に設定する JSON のパラメータと形状は、スケジュールが呼び出すサービス API によって決まります。この情報については、対象とするサービスの API リファレンスをご覧ください。

サポートされていないアクション

EventBridge スケジューラは、以下のプレフィックスのリストで始まる一般的な `GET` オペレーションなどの読み取り専用 API アクションをサポートしていません。

```
get
describe
list
poll
receive
search
scan
query
select
read
lookup
discover
validate
batchGet
batchDescribe
batchRead
transactGet
adminGet
adminList
testMigration
retrieve
testConnection
translateDocument
isAuthorized
invokeModel
```

例えば、GetQueueUrl API アクションのサービス ARN は次のようになります: arn:aws:scheduler:::aws-sdk:sqs:**getQueueURL** API アクションは get プレフィックスで始まるため、EventBridge スケジューラはこのターゲットをサポートしていません。同様に、Amazon MQ アクション ListBrokers には list プレフィックスが付いているため、これはターゲットとしてサポートされません。

ユニバーサルターゲットの使用例

スケジュールの Input フィールドに渡すパラメータは、呼び出したいサービス API が受け入れるリクエストパラメータによって異なります。例えば、Lambda Invoke をターゲットにするには、「AWS Lambda API リファレンス」に記載されているパラメータを設定できます。これには、Lambda 関数に渡すことができるオプションの JSON ペイロードが含まれます。

さまざまな API に設定できるパラメータを確認するには、そのサービスの API リファレンスを参照してください。Lambda Invoke と同様に、一部の API は URI パラメータとリクエスト本文のペイロードを受け入れます。このような場合は、URI パスパラメータと JSON ペイロードをスケジュールの Input に指定します。

以下の例は、ユニバーサルターゲットを使用して Lambda、Amazon SQS、およびStep Functions で一般的な API オペレーションを呼び出す方法を示しています。

Example Lambda

```
$ aws scheduler create-schedule --name lambda-universal-schedule --schedule-expression
 'rate(5 minutes)' \
--target '{"RoleArn": "ROLE_ARN", "Arn":"arn:aws:scheduler:::aws-sdk:lambda:invoke"
 "Input": "{\"FunctionName\":\"arn:aws:lambda:REGION:123456789012:function:HelloWorld
\",\"InvocationType\":\"Event\",\"Payload\":\"{\\\"message\\\":\\\"testing function\\
\"}\"}" }' \
--flexible-time-window '{ "Mode": "OFF"}'
```

Example Amazon SQS

```
import boto3
scheduler = boto3.client('scheduler')

flex_window = { "Mode": "OFF" }

sqs_universal= {
    "RoleArn": "<ROLE_ARN>",
    "Arn": "arn:aws:scheduler:::aws-sdk:sqs:sendMessage",
```

```
    "Input": "{\"MessageBody\":\"My message\",\"QueueUrl\":\"<QUEUE_URL>\"}"}
}

scheduler.create_schedule(
    Name="sqs-sdk-test",
    ScheduleExpression="rate(5 minutes)",
    Target=sqs_universal,
    FlexibleTimeWindow=flex_window)
```

Example Step Functions

```
package com.example;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.scheduler.SchedulerClient;
import software.amazon.awssdk.services.scheduler.model.*;


public class MySchedulerApp {

    public static void main(String[] args) {

        final SchedulerClient client = SchedulerClient.builder()
                .region(Region.US_WEST_2)
                .build();

        Target stepFunctionsUniversalTarget = Target.builder()
                .roleArn("<ROLE_ARN>")
                .arn("arn:aws:scheduler:::aws-sdk:sfn:startExecution")
                .input("{\"Input\":\"{}\",\"StateMachineArn\":\"<STATE_MACHINE_ARN>
\"}")
                .build();

        CreateScheduleRequest createScheduleRequest = CreateScheduleRequest.builder()
                .name("<SCHEDULE_NAME>")
                .scheduleExpression("rate(10 minutes)")
                .target(stepFunctionsUniversalTarget)
                .flexibleTimeWindow(FlexibleTimeWindow.builder()
                        .mode(FlexibleTimeWindowMode.OFF)
                        .build())
                .clientToken("<Token GUID>")
                .build();
```

```
        client.createSchedule(createScheduleRequest);
        System.out.println("Created schedule with rate expression and Step Function
 universal target");
    }
}
```

EventBridge スケジューラでのコンテキスト属性の追加

ターゲットに渡すペイロードで以下のキーワードを使用して、スケジュールに関するメタデータを収集する。スケジュールがターゲットを呼び出すと、EventBridge スケジューラは各キーワードをそれぞれの値に置き換える。

- **<aws.scheduler.schedule-arn>** — スケジュールの ARN。
- **<aws.scheduler.scheduled-time>** — スケジュールがターゲットを呼び出すために指定した時間 (例: `2022-03-22T18:59:43Z`)。
- **<aws.scheduler.execution-id>** — EventBridge スケジューラがターゲットの呼び出しを試みるたびに割り当てる固有の ID (例: `d32c5kddcf5bb8c3`)。
- **<aws.scheduler.attempt-number>** — 現在の呼び出しの試行回数を識別するカウンター (例: 1)。

この例では、5 分ごとに起動し、Amazon SQS の `SendMessage` オペレーションをユニバーサルターゲットとして呼び出すスケジュールの作成を示しています。メッセージ本文には `schedule-time` の値が含まれています。

Example AWS CLI

```
$ aws scheduler create-schedule --name your-schedule \
    --schedule-expression 'rate(5 minutes)' \
    --target '{"RoleArn": "ROLE_ARN", \
        "Arn": "arn:aws:scheduler:::aws-sdk:sqs:sendMessage", \
        "Input": "{\"MessageBody\":\"<aws.scheduler.scheduled-time>\",\"QueueUrl\":
\"https://sqs.us-west-2.amazonaws.com/123456789012/scheduler-cli-test\"}"}' \
    --flexible-time-window '{ "Mode": "OFF"}'
```

Example Python SDK

```
import boto3
scheduler = boto3.client('scheduler')
```

```
sqs_universal= {
    "RoleArn": "<ROLE_ARN>",
    "Arn": "arn:aws:scheduler:::aws-sdk:sqs:sendMessage",
    "Input": "{\"MessageBody\":\"<aws.scheduler.scheduled-time>\",\"QueueUrl\":
\"https://sqs.us-west-2.amazonaws.com/123456789012/scheduler-cli-test\"}"
}

flex_window = { "Mode": "OFF" }


scheduler.update_schedule(Name="your-schedule",
    ScheduleExpression="rate(5 minutes)",
    Target=sqs_universal,
    FlexibleTimeWindow=flex_window)
```

次のステップ

EventBridge スケジューラのデータ型と API オペレーションの詳細については、「EventBridge スケジューラ API リファレンス」を参照してください。

インターフェイスエンドポイント (AWS PrivateLink) を使用して Amazon EventBridge スケジューラにアクセスする

AWS PrivateLink を使用して、VPC と Amazon EventBridge スケジューラの間にプライベート接続を作成できます。インターネットゲートウェイ、NAT デバイス、VPN 接続、または Direct Connect 接続を使用せずに、VPC 内にあるかのように EventBridge スケジューラにアクセスできます。VPC のインスタンスは、パブリック IP アドレスがなくても EventBridge スケジューラにアクセスできます。

このプライベート接続を確立するには、AWS PrivateLink を利用したインターフェイスエンドポイントを作成します。インターフェイスエンドポイントに対して有効にする各サブネットにエンドポイントネットワークインターフェイスを作成します。これらは、EventBridge スケジューラ宛てのトラフィックのエントリポイントとして機能するリクエスタ管理型ネットワークインターフェイスです。

詳細については、「AWS PrivateLinkガイド」の「AWS PrivateLinkから AWS のサービスにアクセスする」を参照してください。

EventBridge スケジューラに関する考慮事項

EventBridge スケジューラのインターフェイスエンドポイントを設定する前に、「AWS PrivateLink ガイド」の「考慮事項」を確認してください。

EventBridge スケジューラは、インターフェイスエンドポイントを介してすべての API アクションの呼び出しをサポートしています。

EventBridge スケジューラのインターフェイスエンドポイントを作成する

Amazon VPC コンソールまたは AWS Command Line Interface (AWS CLI) を使用して、EventBridge スケジューラのインターフェイスエンドポイントを作成できます。詳細については、「AWS PrivateLink ガイド」の「インターフェイスエンドポイントを作成」を参照してください。

以下のサービス名を使用して、EventBridge スケジューラ用のインターフェイスエンドポイントを作成します。

```
com.amazonaws.region.scheduler
```

インターフェイスエンドポイントのプライベート DNS を有効にすると、リージョンのデフォルト DNS 名を使用して、EventBridge スケジューラへの API リクエストを実行できます。例えば、`scheduler.us-east-1.amazonaws.com`。

インターフェイスエンドポイントのエンドポイントポリシーを作成する

エンドポイントポリシーは、インターフェイスエンドポイントにアタッチできる IAM リソースです。デフォルトのエンドポイントポリシーでは、インターフェイスエンドポイントを介した EventBridge スケジューラへのフルアクセスが許可されています。VPC から EventBridge スケジューラへの許可されたアクセスをコントロールするには、カスタムエンドポイントポリシーをインターフェイスエンドポイントにアタッチします。

エンドポイントポリシーは以下の情報を指定します。

- アクションを実行できるプリンシパル (AWS アカウント、IAM ユーザー、IAM ロール)。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、AWS PrivateLink ガイドのControl access to services using endpoint policies (エンドポイントポリシーを使用してサービスへのアクセスをコントロールする)を参照してください。

例: EventBridge スケジューラアクションの VPC エンドポイントポリシー

以下は、カスタムエンドポイントポリシーの例です。インターフェイスエンドポイントにアタッチされると、このポリシーは、すべてのリソースですべてのプリンシパルに、リストされている EventBridge スケジューラアクションへのアクセス権を付与します。

```
{
   "Statement": [
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "scheduler:GetSchedule",
            "scheduler:ListSchedules",
            "scheduler:GetScheduleGroup",
            "scheduler:ListScheduleGroups"
         ],
```

```
            "Resource":"*"
        }
    ]
}
```

Amazon EventBridge スケジューラでのセキュリティ

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、 AWS お客様とお客様の間の責任共有です。責任共有モデルではこれをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ – AWS は、 で AWS サービスを実行するインフラストラクチャを保護する責任を担います AWS クラウド。 は、お客様が安全に使用できるサービス AWS も提供します。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。Amazon EventBridge スケジューラに適用されるコンプライアンスプログラムの詳細については、「コンプライアンスAWS プログラムによる対象範囲内のサービスコンプライアンス」を参照してください。
- クラウドのセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、ユーザーは、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、EventBridge スケジューラ使用時に責任共有モデルが適用されるしくみを理解するうえで役立ちます。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するように EventBridge スケジューラを設定する方法について説明します。また、EventBridge スケジューラリソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

トピック

- Amazon EventBridge スケジューラへのアクセスの管理
- Amazon EventBridge スケジューラでのデータ保護
- Amazon EventBridge スケジューラでのコンプライアンス検証
- Amazon EventBridge スケジューラでの耐障害性
- Amazon EventBridge スケジューラでのインフラストラクチャセキュリティ

Amazon EventBridge スケジューラへのアクセスの管理

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に

EventBridge スケジューラのリソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

オーディエンス

AWS Identity and Access Management (IAM) の使用方法は、ロールによって異なります。

- サービスユーザー - 機能にアクセスできない場合は、管理者にアクセス許可をリクエストします (「Amazon EventBridge スケジューラのアイデンティティとアクセスのトラブルシューティング」を参照)。
- サービス管理者 - ユーザーアクセスを決定し、アクセス許可リクエストを送信します (「EventBridge スケジューラが IAM と連動する方法」を参照)
- IAM 管理者 - アクセスを管理するためのポリシーを作成します (「EventBridge スケジューラでのアイデンティティベースのポリシーの使用」を参照)

アイデンティティを使用した認証

認証は、ID 認証情報 AWS を使用して にサインインする方法です。、IAM ユーザー AWS アカウントのルートユーザー、または IAM ロールを引き受けることで認証される必要があります。

AWS IAM アイデンティティセンター (IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーティッド ID としてサインインできます。サインインの詳細については、「AWS サインイン ユーザーガイド」の「AWS アカウントにサインインする方法」を参照してください。

プログラムによるアクセスの場合、 は SDK と CLI AWS を提供してリクエストを暗号化して署名します。詳細については、「IAM ユーザーガイド」の「API リクエストに対するAWS 署名バージョン 4」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、まず、すべての AWS のサービス および リソースへの完全なアクセス権を持つ AWS アカウント root ユーザーと呼ばれる 1 つのサインインアイデンティティから始めます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザー認証情報を必要とするタスクについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーテッドアイデンティティ

ベストプラクティスとして、人間のユーザーが一時的な認証情報 AWS のサービス を使用して にアクセスするには、ID プロバイダーとのフェデレーションを使用する必要があります。

フェデレーティッド ID は、エンタープライズディレクトリ、ウェブ ID プロバイダー、または ID Directory Service ソースからの認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーテッドアイデンティティは、一時的な認証情報を提供するロールを引き受けます。

アクセスを一元管理する場合は、 AWS IAM アイデンティティセンターをお勧めします。詳細については、「AWS IAM アイデンティティセンター ユーザーガイド」の「IAM アイデンティティセンターとは」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、特定の個人やアプリケーションに対する特定のアクセス許可を持つアイデンティティです。長期認証情報を持つ IAM ユーザーの代わりに一時的な認証情報を使用することをお勧めします。詳細については、AWS IAM ユーザーガイドの「ID プロバイダーとのフェデレーションを使用して にアクセスする必要がある」を参照してください。

IAM グループは、IAM ユーザーの集合を指定し、大量のユーザーに対するアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可を持つアイデンティであり、一時的な認証情報を提供します。ユーザーから IAM ロール (コンソール) に切り替えるか、 または API オペレーションを呼び出す

ことで、ロールを引き受けることができます。 AWS CLI AWS 詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールは、フェデレーションユーザーアクセス、一時的な IAM ユーザーのアクセス許可、クロスアカウントアクセス、クロスサービスアクセス、および Amazon EC2 で実行するアプリケーションに役立ちます。詳細については、IAM ユーザーガイド の IAM でのクロスアカウントリソースアクセス を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは、アイデンティティまたはリソースに関連付けられたときにアクセス許可を定義します。は、プリンシパルがリクエストを行うときにこれらのポリシー AWS を評価します。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は、ポリシーを使用して、どのプリンシパルがどのリソースに対して、どのような条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

デフォルトでは、ユーザーやロールにアクセス許可はありません。IAM 管理者は IAM ポリシーを作成してロールに追加し、このロールをユーザーが引き受けられるようにします。IAM ポリシーは、オペレーションの実行方法を問わず、アクセス許可を定義します。

アイデンティティベースのポリシー

アイデンティティベースのポリシーは、アイデンティティ (ユーザー、グループ、またはロール) にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティがどのリソースに対してどのような条件下でどのようなアクションを実行できるかを制御します。アイデンティティベースポリシーの作成方法については、IAM ユーザーガイド の カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する を参照してください。

アイデンティティベースのポリシーは、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーとインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシーのいずれかを選択する」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。例としては、IAM ロール信頼ポリシーや Amazon S3 バケットポリシーなどがあります。リソースベースのポ

リシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

その他のポリシータイプ

AWS は、より一般的なポリシータイプによって付与されるアクセス許可の最大数を設定できる追加のポリシータイプをサポートしています。

- アクセス許可の境界 – アイデンティティベースのポリシーで IAM エンティティに付与することのできるアクセス許可の数の上限を設定します。詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可境界」を参照してください。
- サービスコントロールポリシー (SCP) - AWS Organizations内の組織または組織単位の最大のアクセス許可を指定します。詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー」を参照してください。
- リソースコントロールポリシー (RCP) – は、アカウント内のリソースで利用できる最大数のアクセス許可を定義します。詳細については、「AWS Organizations ユーザーガイド」の「リソースコントロールポリシー (RCP)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを作成する際にパラメータとして渡される高度なポリシーです。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成されるアクセス許可を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

EventBridge スケジューラが IAM と連動する方法

IAM を使用して EventBridge スケジューラへのアクセスを管理する前に、EventBridge スケジューラで利用できる IAM の機能について説明します。

Amazon EventBridge スケジューラで利用できる IAM の機能

IAM 機能	EventBridge スケジューラのサポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	なし
ポリシーアクション	あり
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
ACL	なし
ABAC (ポリシー内のタグ)	部分的
一時認証情報	あり
プリンシパルアクセス権限	あり
サービスロール	あり
サービスリンクロール	いいえ

EventBridge スケジューラおよびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要については、IAM ユーザーガイドのAWS 「IAM と連携する のサービス」を参照してください。

EventBridge スケジューラのアイデンティティベースのポリシー

アイデンティティベースのポリシーのサポート: あり

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。JSON ポリシーで使用できるすべての要素に

ついて学ぶには、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

EventBridge スケジューラのアイデンティティベースのポリシーの例

EventBridge スケジューラのアイデンティティベースのポリシーの例を表示するには、「EventBridge スケジューラでのアイデンティティベースのポリシーの使用」を参照してください。

EventBridge スケジューラ内のリソースベースのポリシー

リソースベースのポリシーのサポート: なし

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーで、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、全体のアカウント、または別のアカウントの IAM エンティティを、リソースベースのポリシーのプリンシパルとして指定します。詳細については、IAM ユーザーガイドのIAM でのクロスアカウントリソースアクセスを参照してください。

EventBridge スケジューラのポリシーアクション

ポリシーアクションのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。このアクションは関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

EventBridge スケジューラのアクションのリストを確認するには、「サービス認可リファレンス」の「Amazon EventBridge スケジューラで定義されるアクション」を参照してください。

EventBridge スケジューラのポリシーアクションは、アクションの前に以下のプレフィックスを使用します:

```
scheduler
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

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

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、List という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": [
   "scheduler:List*"
]
```

EventBridge スケジューラのポリシーリソース

ポリシーリソースのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。リソースレベルのアクセス許可をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

EventBridge スケジューラのリソースタイプとその ARN のリストを確認するには、「サービス認可リファレンス」の「Amazon EventBridge スケジューラで定義されるリソース」を参照してください。各リソースの ARN を指定できるアクションについては、「Amazon EventBridge スケジューラで定義されているアクション」を参照してください。

EventBridge スケジューラのアイデンティティベースのポリシーの例を表示するには、「EventBridge スケジューラでのアイデンティティベースのポリシーの使用」を参照してください。

EventBridge スケジューラのポリシー条件キー

サービス固有のポリシー条件キーのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

`Condition` 要素は、定義された基準に基づいてステートメントが実行される時期を指定します。イコールや未満などの条件演算子を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「グローバル条件コンテキストキー」を参照してください。

EventBridge スケジューラの条件キーのリストを確認するには、「サービス認可リファレンス」の「Amazon EventBridge スケジューラの条件キー」を参照してください。条件キーを使用できるアクションとリソースについては、「Amazon EventBridge スケジューラで定義されるアクション」を参照してください。

EventBridge スケジューラのアイデンティティベースのポリシーの例を表示するには、「EventBridge スケジューラでのアイデンティティベースのポリシーの使用」を参照してください。

EventBridge スケジューラでの ACL

ACL のサポート: なし

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするためのアクセス許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

EventBridge スケジューラでの ABAC

ABAC (ポリシー内のタグ) のサポート: 一部

属性ベースのアクセスコントロール (ABAC) は、タグと呼ばれる属性に基づいてアクセス許可を定義する認可戦略です。IAM エンティティと AWS リソースにタグをアタッチし、プリンシパルのタグがリソースのタグと一致するときにオペレーションを許可するように ABAC ポリシーを設計できます。

タグに基づいてアクセスを管理するには、aws:ResourceTag/*key-name*、aws:RequestTag/*key-name*、または aws:TagKeys の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、「IAM ユーザーガイド」の「ABAC 認可でアクセス許可を定義する」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「IAM ユーザーガイド」の「属性ベースのアクセスコントロール (ABAC) を使用する」を参照してください。

EventBridge スケジューラでの一時的な認証情報の使用

一時的な認証情報のサポート: あり

一時的な認証情報は AWS 、リソースへの短期的なアクセスを提供し、フェデレーションまたはスイッチロールの使用時に自動的に作成されます。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「IAM ユーザーガイド」の「IAM の一時的な認証情報」および「AWS のサービス と IAM との連携」を参照してください。

EventBridge スケジューラのクロスサービスプリンシパル許可

転送アクセスセッション (FAS) のサポート: あり

転送アクセスセッション (FAS) は、 を呼び出すプリンシパルのアクセス許可と AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする を使用します。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

EventBridge スケジューラのサービスロール

サービスロールのサポート: あり

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、IAM ユーザーガイドの AWS のサービスに許可を委任するロールを作成するを参照してください。

> **Warning**
>
> サービスロールの許可を変更すると、EventBridge スケジューラの機能が破損する可能性があります。EventBridge スケジューラが指示する場合以外は、サービスロールを編集しないでください。

EventBridge スケジューラのサービスにリンクされたロール

サービスにリンクされたロールのサポート: なし

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

サービスにリンクされたロールの作成または管理の詳細については、「IAM と提携するAWS のサービス」を参照してください。表の「サービスリンクロール」列に Yes と記載されたサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

EventBridge スケジューラでのアイデンティティベースのポリシーの使用

デフォルトでは、ユーザーおよびロールには、EventBridge スケジューラのリソースを作成または変更するアクセス許可がありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。

これらのサンプルの JSON ポリシードキュメントを使用して IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーを作成する (コンソール)」を参照してください。

EventBridge スケジューラが定義するアクションとリソースタイプ (リソースタイプごとの ARN の形式を含む) の詳細については、「サービス認可リファレンス」の「Amazon EventBridge スケジューラのアクション、リソース、および条件キー」を参照してください。

トピック

- ポリシーに関するベストプラクティス

- EventBridge スケジューラのアクセス許可
- AWS EventBridge スケジューラの マネージドポリシー
- EventBridge スケジューラのカスタマー管理ポリシー
- AWS マネージドポリシーの更新

ポリシーに関するベストプラクティス

アイデンティティベースのポリシーは、ユーザーのアカウントで誰かが EventBridge スケジューラのリソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションでは、AWS アカウントに費用が発生する場合があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。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 または ジョブ機能のAWS マネージドポリシー を参照してください。
- 最小特権を適用する – IAM ポリシーでアクセス許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、IAM ユーザーガイド の IAM でのポリシーとアクセス許可 を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。たとえば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます CloudFormation。詳細については、IAM ユーザーガイド の IAM JSON ポリシー要素:条件 を参照してください。
- IAM アクセスアナライザー を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM アクセスアナライザー は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、IAM ユーザーガイド の IAM Access Analyzer でポリシーを検証する を参照してください。

- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、MFA をオンにしてセキュリティを強化します。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、IAM ユーザーガイド の MFA を使用した安全な API アクセス を参照してください。

IAM でのベストプラクティスの詳細については、IAM ユーザーガイド の IAM でのセキュリティのベストプラクティス を参照してください。

EventBridge スケジューラのアクセス許可

IAM プリンシパル (ユーザー、グループ、またはロール) が EventBridge スケジューラでスケジュールを作成し、コンソールまたは API から EventBridge スケジューラのリソースにアクセスするには、プリンシパルのアクセス権限ポリシーに一連の権限を追加する必要があります。これらの権限は、プリンシパルの職務に応じて設定できます。例えば、EventBridge スケジューラのコンソールのみを使用して既存のスケジュールのリストを表示するユーザーまたはロールには、`CreateSchedule` API オペレーションを呼び出すために必要な権限は必要ありません。アイデンティティベースの権限を調整して、最も権限の低いアクセス権のみを提供することをおすすめします。

次のリストは、EventBridge スケジューラのリソースと、それに対応してサポートされているアクションを示しています。

- スケジュール
  - `scheduler:ListSchedules`
  - `scheduler:GetSchedule`
  - `scheduler:CreateSchedule`
  - `scheduler:UpdateSchedule`
  - `scheduler:DeleteSchedule`
- スケジュールグループ
  - `scheduler:ListScheduleGroups`
  - `scheduler:GetScheduleGroup`
  - `scheduler:CreateScheduleGroup`
  - `scheduler:DeleteScheduleGroup`
  - `scheduler:ListTagsForResource`
  - `scheduler:TagResource`

- `scheduler:UntagResource`

EventBridge スケジューラの権限を使用して、EventBridge スケジューラで使用する独自のカスタマー管理ポリシーを作成できます。次のセクションで説明する AWS 管理ポリシーを使用して、独自のポリシーを管理することなく、一般的なユースケースに必要なアクセス許可を付与することもできます。

AWS EventBridge スケジューラの マネージドポリシー

AWS は、 が AWS 作成および管理するスタンドアロン 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。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 管理ポリシーは、EventBridge スケジューラに固有です。

- AmazonEventBridgeSchedulerFullAccess

  スケジュールおよびスケジュールグループのすべての EventBridge スケジューラアクションを使用する権限を付与します。

  このポリシーの許可を確認するには、「AWS マネージドポリシーリファレンス」の「AmazonEventBridgeSchedulerFullAccess」を参照してください。
- AmazonEventBridgeSchedulerReadOnlyAccess

  スケジュールとスケジュールグループに関する詳細を表示する読み取り専用の権限を付与します。

  このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンス」の「AmazonEventBridgeSchedulerReadOnlyAccess」を参照してください。

EventBridge スケジューラのカスタマー管理ポリシー

以下の例を使用して、EventBridge スケジューラの独自のカスタマー管理ポリシーを作成します。カスタマー管理ポリシーを使用すると、プリンシパルの職務に応じて、チーム内のアプリケーションとユーザーに必要なアクションとリソースのみにアクセス許可を付与できます。

トピック

- 例: CreateSchedule

- 例: GetSchedule
- 例: UpdateSchedule
- 例: DeleteScheduleGroup

例: `CreateSchedule`

新しいスケジュールを作成するときに、EventBridge スケジューラ上のデータを暗号化するのに AWS 所有のキー を使用するか、またはカスタマーマネージドキーを使用するかを選択します。

次のポリシーでは、プリンシパルはスケジュールを作成し、 AWS 所有のキーを使用して暗号化を適用できます。を使用すると AWS 所有のキー、 は AWS Key Management Service (AWS KMS) のリソース AWS を管理するため、 を操作するための追加のアクセス許可は必要ありません AWS KMS。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Action":
            [
                "scheduler:CreateSchedule"
            ],
            "Effect": "Allow",
            "Resource":
            [
                "arn:aws:scheduler:us-west-2:123456789012:schedule/my-group/my-schedule-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "scheduler.amazonaws.com"
                }
            }
```

```
        }
    ]
}
```

次のポリシーを使用して、プリンシパルがスケジュールを作成し、 AWS KMS カスタマーマネージドキーを暗号化に使用できるようにします。カスタマーマネージドキーを使用するには、プリンシパルにアカウントの AWS KMS リソースへのアクセス許可が必要です。このポリシーは、EventBridge スケジューラ上のデータを暗号化するために使用される単一の指定された KMS キーへのアクセスを許可します。または、ワイルドカード (*) 文字を使用して、アカウント内のすべてのキー、または特定の名前パターンに一致するサブセットへのアクセスを許可することもできます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Action":
            [
                "scheduler:CreateSchedule"
            ],
            "Effect": "Allow",
            "Resource":
            [
            "arn:aws:scheduler:us-east-1:123456789012:schedule/my-group/my-
schedule-name"
            ]
        },
        {
            "Action":
            [
                "kms:DescribeKey",
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Effect": "Allow",
            "Resource":
            [
                "arn:aws:kms:us-west-2:123456789012:key/my-key-id"
            ],
```

```
            "Condition": {
                "StringLike": {
                "kms:ViaService": "scheduler.us-east-1.amazonaws.com",
                    "kms:EncryptionContext:aws:scheduler:schedule:arn":
 "arn:aws:scheduler:us-west-2:123456789012:schedule/my-group/my-schedule-name"
            }
        }
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "scheduler.amazonaws.com"
                }
            }
        }
    ]
}
```

例: **GetSchedule**

以下のポリシーを使用して、プリンシパルがスケジュールに関する情報を取得できるようにします。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Action":
            [
                "scheduler:GetSchedule"
            ],
            "Effect": "Allow",
            "Resource":
            [
                "arn:aws:scheduler:us-west-2:123456789012:schedule/my-group/my-
schedule-name"
            ]
```

```
        }
    ]
}
```

例: **UpdateSchedule**

以下のポリシーを使用して、プリンシパルが scheduler:UpdateSchedule アクションを呼び出してスケジュールを更新できるようにします。と同様にCreateSchedule、ポリシーは、スケジュールが暗号化に AWS KMS AWS 所有のキー を使用するか、カスタマーマネージドキーを使用するかによって異なります。で設定されたスケジュールの場合は AWS 所有のキー、次のポリシーを使用します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Action":
            [
                "scheduler:UpdateSchedule"
            ],
            "Effect": "Allow",
            "Resource":
            [
                "arn:aws:scheduler:us-west-2:123456789012:schedule/my-group/my-
schedule-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "scheduler.amazonaws.com"
                }
            }
        }
    ]
```

```
}
```

カスタマーマネージドキーを使用して設定されたスケジュールの場合は、以下のポリシーを使用します。このポリシーには、プリンシパルがアカウントの AWS KMS リソースにアクセスできるようにする追加のアクセス許可が含まれています。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Action":
            [
                "scheduler:UpdateSchedule"
            ],
            "Effect": "Allow",
            "Resource":
            [
            "arn:aws:scheduler:us-east-1:123456789012:schedule/my-group/my-
schedule-name"
    ]
        },
        {
            "Action":
            [
                "kms:DescribeKey",
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Effect": "Allow",
            "Resource":
            [
                "arn:aws:kms:us-west-2:123456789012:key/my-key-id"
            ],
            "Condition": {
                "StringLike": {
                "kms:ViaService": "scheduler.us-east-1.amazonaws.com",
                    "kms:EncryptionContext:aws:scheduler:schedule:arn":
 "arn:aws:scheduler:us-east-1:123456789012:schedule/my-group/my-schedule-name"
```

```
            }
        }
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "scheduler.amazonaws.com"
                }
            }
        }
    ]
}
```

例: **DeleteScheduleGroup**

以下のポリシーを使用して、プリンシパルがスケジュールグループを削除できるようにします。グループを削除すると、そのグループに関連付けられているスケジュールも削除されます。グループを削除するプリンシパルには、そのグループに関連付けられているスケジュールも削除する権限が必要です。このポリシーは、指定されたスケジュールグループとグループ内のすべてのスケジュールに対して scheduler:DeleteScheduleGroup アクションを呼び出す権限をプリンシパルに付与します。

> **Note**
>
> EventBridge スケジューラは、個々のスケジュールにリソースレベルの権限を指定することをサポートしていません。例えば、次の記述は無効であり、ポリシーには含めないでください。
> "Resource": "arn:aws:scheduler:us-west-2:123456789012:schedule/*my-group*/*my-schedule-name*"

JSON

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

```
        {
            "Effect": "Allow",
            "Action": "scheduler:DeleteSchedule",
            "Resource": "arn:aws:scheduler:us-west-2:123456789012:schedule/my-
group/*"
        },
        {
            "Effect": "Allow",
            "Action": "scheduler:DeleteScheduleGroup",
            "Resource": "arn:aws:scheduler:us-west-2:123456789012:schedule/my-
group/*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "scheduler.amazonaws.com"
                }
            }
        }
    ]
}
```

AWS マネージドポリシーの更新

変更	説明	日付
AmazonEventBridgeSchedulerFullAccess – 新しい管理ポリシー	EventBridge スケジューラは、スケジュールやスケジュールグループを含むすべてのリソースへのフルアクセスをユーザーに許可する新しいマネージドポリシーのサポートを追加しました。	2022 年 11 月 10 日
AmazonEventBridgeSchedulerReadOnlyAccess – 新しい管理ポリシー	EventBridge スケジューラは、スケジュールやスケジュールグループを含むすべて	2022 年 11 月 10 日

変更	説明	日付
	のリソースへの読み取り専用アクセスをユーザーに許可する新しいマネージドポリシーのサポートを追加しました。	
EventBridge スケジューラが変更の追跡を開始	EventBridge スケジューラは AWS 、管理ポリシーの変更の追跡を開始しました。	2022 年 11 月 10 日

EventBridge スケジューラでの混乱した代理の防止

混乱した代理問題は、アクションを実行する許可を持たないエンティティが、より特権のあるエンティティにアクションを実行するように強制できるセキュリティの問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。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、別のお客様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、 AWS では、アカウントのリソースへのアクセス権が付与されたサービスプリンシパルで、すべてのサービスのデータを保護するために役立つツールを提供しています。

スケジュール実行ロール内では `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、EventBridge スケジューラ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。そのアカウント内のリソースをクロスサービスの使用に関連付けることを許可する場合は、`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 を指定して、`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。以下の条件は個々のスケジュールグループに限定されます: `arn:aws:scheduler:*:`*`123456789012`*`:schedule-group/`*`your-schedule-group`*

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: `arn:aws:scheduler:*:`*`123456789012`*`:schedule-group/*`。

aws:SourceArn の値は、この条件の範囲を設定したい EventBridge スケジューラのスケジュールグループ ARN でなければなりません。

> **Important**
>
> aws:SourceArn ステートメントの範囲を特定のスケジュールやスケジュール名のプレフィックスに限定しないでください。指定する ARN はスケジュールグループである必要があります。

次の例では、aws:SourceArn および aws:SourceAccount グローバル条件コンテキストキーを実行ロールの信頼ポリシーで使用して、混乱した代理問題を回避する方法を示します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "scheduler.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012",
                    "aws:SourceArn": "arn:aws:scheduler:us-
west-2:123456789012:schedule-group/your-schedule-group"
                }
            }
        }
    ]
}
```

の値にワイルドカード文字を使用する場合はaws:SourceArn、条件演算子StringEqualsとしてArnLikeの代わりに を使用する必要があります。たとえば、次の信頼ポリシーでは、 ArnLikeを使用してアカウントの任意のスケジュールグループを照合します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "scheduler.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:scheduler:*:123456789012:schedule-group/
*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

Amazon EventBridge スケジューラのアイデンティティとアクセスのトラブルシューティング

次の情報は、EventBridge スケジューラと IAM の使用に伴って発生する可能性がある一般的な問題の診断や修復に役立ちます。

トピック

- EventBridge スケジューラでアクションを実行する権限がない
- iam:PassRole を実行する権限がありません
- 自分の 以外のユーザーに EventBridge スケジューラリソース AWS アカウント へのアクセスを許可したい

EventBridge スケジューラでアクションを実行する権限がない

アクションを実行する権限がないというエラーが表示された場合は、そのアクションを実行できるようにポリシーを更新する必要があります。

以下のエラー例は、mateojackson IAM ユーザーがコンソールを使用して架空の *my-example-widget* リソースに関する詳細情報を表示しようとしているが、架空の scheduler:*GetWidget* 権限がないという場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 scheduler:GetWidget on resource: my-example-widget
```

この場合、Mateo のポリシーでは、*my-example-widget* アクションを使用して scheduler:*GetWidget* リソースにアクセスすることを許可するように更新する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

iam:PassRole を実行する権限がありません

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して EventBridge スケジューラにロールを渡すことができるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、既存のロールをそのサービスに渡すことができます。そのためには、サービスにロールを渡すアクセス許可が必要です。

以下の例では、marymajor という IAM ユーザーがコンソールを使用して EventBridge スケジューラでアクションを実行しようする場合にエラーが発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与されたアクセス許可が必要です。Mary には、ロールをサービスに渡すアクセス許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分の 以外のユーザーに EventBridge スケジューラリソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまた

はアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- EventBridge スケジューラがこれらの機能をサポートしているかどうかを確認するには、「EventBridge スケジューラが IAM と連動する方法」を参照してください。
- 所有 AWS アカウント する のリソースへのアクセスを提供する方法については、IAM ユーザーガイドの「所有 AWS アカウント する別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイド の 外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可 を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、IAM ユーザーガイド の IAM でのクロスアカウントのリソースへのアクセス を参照してください。

Amazon EventBridge スケジューラでのデータ保護

責任 AWS 共有モデル、Amazon EventBridge スケジューラでのデータ保護に適用されます。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。ユーザーは、このインフラストラクチャでホストされ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に関する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「AWS 責任共有モデルおよび GDPR」のブログ記事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 アイデンティティセンター 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 は必須ですが、TLS 1.3 を推奨します。
- 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。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- 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 などの高度な管理された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 140-3 検証済み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) 140-3」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を、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含めないことを強くお勧めします。これは、コンソール、API、AWS CLIまたは SDK を使用して EventBridge スケジューラまたは他の AWS のサービス を使用する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

EventBridge スケジューラでの保管中の暗号化

このセクションでは、Amazon EventBridge スケジューラが保管中のデータを暗号化および復号化する方法について説明します。保管中のデータとは、EventBridge スケジューラとサービスの基盤となるコンポーネントに保存されているデータです。EventBridge スケジューラは AWS Key Management Service (AWS KMS) と統合され、 を使用してデータを暗号化および復号化しますAWS KMS key。EventBridge スケジューラは、AWS 所有のキー とカスタマーマネージドキーの 2 種類の KMS キーをサポートしています。

> **Note**
>
> EventBridge スケジューラは、対称暗号化 KMS キーの使用のみをサポートしています。

AWS 所有のキー は、 AWS サービスが複数の AWS アカウントで使用するために所有および管理する KMS キーです。 AWS 所有のキー EventBridge スケジューラが使用する は AWS アカウント

に保存されませんが、EventBridge スケジューラはそれらを使用してデータとリソースを保護します。デフォルトでは、EventBridge スケジューラは AWS 所有キーを使用してすべてのデータを暗号化および復号します。自分の AWS 所有のキー またはアクセスポリシーを管理する必要はありません。EventBridge スケジューラがデータ AWS 所有のキー を保護するために を使用する場合、料金は発生せず、その使用量はアカウントの AWS KMS クォータの一部としてカウントされません。

カスタマーマネージドキーは、作成、所有、管理する AWS アカウントに保存されている KMS キーです。特定のユースケースで、EventBridge スケジューラでデータを保護する暗号化キーを管理および監査する必要がある場合は、カスタマーマネージドキーを使用できます。カスタマーマネージドキーを選択した場合、キーポリシーを管理する必要があります。カスタマーマネージドキーの使用には、月額料金と、無料利用枠を超えた使用に対する料金がかかります。カスタマーマネージドキーの使用も AWS KMS クォータの一部としてカウントされます。料金の詳細については、「AWS Key Management Service の料金」を参照してください。

トピック

- 暗号化アーティファクト
- KMS キーの管理
- CloudTrail イベントの例

暗号化アーティファクト

次の表は、EventBridge スケジューラが保管時に暗号化するさまざまなタイプのデータと、各カテゴリでサポートされる KMS キーの種類を示しています。

データ型	説明	AWS 所有のキー	カスタマーマネージドキー
ペイロード (最大 256 KB)	スケジュールをターゲットに配信するように設定するときに、スケジュールの `TargetInput` パラメータで指定するデータ。	サポート対象	サポート

データ型	説明	AWS 所有のキー	カスタマーマネージドキー
識別子と状態	スケジュールの固有の名前と状態 (有効、無効)。	サポート	サポートされていません
スケジューリング設定	繰り返しのスケジュールの場合は rate 式や cron 式などのスケジューリング式、1 回限りの呼び出しの場合はタイムスタンプ、スケジュールの開始日、終了日、タイムゾーン。	サポート	サポートされていません
ターゲット設定	ターゲットの Amazon リソースネーム (ARN)、およびその他のターゲットに関連する設定の詳細。	サポート	サポートされていません
呼び出しと障害動作の設定	柔軟な時間枠設定、スケジュールの再試行ポリシー、配信失敗時に使用するデッドレターキューの詳細。	サポート	サポートされていません

EventBridge スケジューラは、前の表で説明したように、ターゲットペイロードを暗号化および復号化する場合にのみカスタマーマネージドキーを使用します。カスタマーマネージドキーを使用することを選択した場合、EventBridge スケジューラはペイロードを 2 回暗号化および復号します。1 回はデフォルトを使用し AWS 所有のキー、もう 1 回は指定したカスタマーマネージドキーを使用します。他のすべてのデータ型では、EventBridge スケジューラは保管中のデータ AWS 所有のキー を保護するためにデフォルトのみを使用します。

以下の the section called “KMS キーの管理” セクションでは、EventBridge スケジューラでカスタマーマネージドキーを使用するために IAM リソースとキーポリシーを管理する方法を説明します。

KMS キーの管理

スケジュールがターゲットに配信するペイロードを暗号化および復号化するためのカスタマーマネージドキーをオプションで提供できます。EventBridge スケジューラは、最大 256 KB のデータのペイロードを暗号化および復号化します。カスタマーマネージドキーの使用には、月額料金と、無料利用枠を超えた使用に対する料金がかかります。カスタマーマネージドキーの使用は AWS KMS クォータの一部としてカウントされます。料金の詳細については、「AWS Key Management Service の料金」を参照してください。

EventBridge スケジューラは、データを暗号化するスケジュールを作成するプリンシパルに関連付けられた IAM アクセス許可を使用します。つまり、EventBridge スケジューラ API を呼び出すユーザーまたはロールに必要な AWS KMS 関連アクセス許可をアタッチする必要があります。さらに、EventBridge スケジューラはリソースベースのポリシーを使用してデータを復号化します。つまり、スケジュールに関連付けられた実行ロールには、データの復号時に AWS KMS API を呼び出すために必要な AWS KMS 関連アクセス許可も必要です。

> **Note**
>
> EventBridge スケジューラは、一時的な権限の付与の使用をサポートしていません。

次のセクションでは、 AWS KMS キーポリシーを管理する方法と、EventBridge スケジューラでカスタマーマネージドキーを使用するために必要な IAM アクセス許可について説明します。

トピック

- IAM アクセス許可を追加する
- キーポリシーの管理

IAM アクセス許可を追加する

カスタマーマネージドキーを使用するには、スケジュールを作成するアイデンティティベースの IAM プリンシパルに次の権限と、スケジュールに関連付ける実行ロールを追加する必要があります。

カスタマーマネージドキーに対するアイデンティティベースのアクセス許可

スケジュールの作成時に EventBridge スケジューラ API を呼び出すプリンシパル (ユーザー、グループ、またはロール) に関連付けられたアクセス許可ポリシーに、次の AWS KMS アクションを追加する必要があります。

- **kms:DescribeKey** — 指定したキーが対称暗号化 KMS キーであることを検証するために必要です。
- **kms:GenerateDataKey** — EventBridge スケジューラがクライアント側の暗号化を実行するために使用するデータキーを生成するために必要です。
- **kms:Decrypt** — EventBridge スケジューラが暗号化されたデータとともに保存する暗号化されたデータキーを復号化するために必要です。

これらは、次のアクションに加えて行われます。

- **scheduler:***
- **iam:PassRole** – 実行ロールを渡すために必要です。

カスタマーマネージドキーの実行ロール権限

データを復号するときに AWS KMS API を呼び出す EventBridge スケジューラへのアクセスを提供するには、スケジュールの実行ロールのアクセス許可ポリシーに次のアクションを追加する必要があります。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEventBridgeSchedulerToDecryptDataUsingCMKMS",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-east-1:123456789012:key/your-key-id"
        }
    ]
```

```
}
```

- **kms:Decrypt** — EventBridge スケジューラが暗号化されたデータとともに保存する暗号化されたデータキーを復号化するために必要です。

新しいスケジュールを作成するときに EventBridge スケジューラのコンソールを使用して新しい実行ロールを作成すると、EventBridge スケジューラは必要な権限を自動的に実行ロールにアタッチします。ただし、既存の実行ロールを選択した場合、カスタマーマネージドキーを使用できるようにするには、必要な権限をロールに追加する必要があります。

キーポリシーの管理

デフォルトでは AWS KMS、 を使用してカスタマーマネージドキーを作成すると、スケジュールの実行ロールへのアクセスを提供する次のキーポリシーがキーに含まれます。

オプションで、キーポリシーの範囲を実行ロールへのアクセスのみに限定することもできます。これは、カスタマーマネージドキーを EventBridge スケジューラのリソースでのみ使用したい場合などに行います。次のキーポリシーの例を使用して、キーを使用できる EventBridge スケジューラのリソースを制限します。

CloudTrail イベントの例

AWS CloudTrail は、すべての API コールイベントをキャプチャします。これには、EventBridge スケジューラがカスタマーマネージドキーを使用してデータを復号化するときの API コールが含まれます。次の例は、カスタマーマネージドキーを使用して `kms:Decrypt` アクションを使用する EventBridge スケジューラを示す CloudTrail イベントエントリを示しています。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
      "type": "AssumedRole",
      "principalId": "ABCDEABCD1AB12ABABAB0:70abcd123a123a12345a1aa12aa1bc12",
      "arn": "arn:aws:sts::123456789012:assumed-role/execution-
role/70abcd123a123a12345a1aa12aa1bc12",
      "accountId": "123456789012",
      "accessKeyId": "ABCDEFGHI1JKLMNOP2Q3",
      "sessionContext": {
          "sessionIssuer": {
              "type": "Role",
              "principalId": "ABCDEABCD1AB12ABABAB0",
```

```
                "arn": "arn:aws:iam::123456789012:role/execution-role",
                "accountId": "123456789012",
                "userName": "execution-role"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-10-31T21:03:15Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-10-31T21:03:15Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "eu-north-1",
    "sourceIPAddress": "13.50.87.173",
    "userAgent": "aws-sdk-java/2.17.295 Linux/4.14.291-218.527.amzn2.x86_64 OpenJDK_64-
Bit_Server_VM/11.0.17+9-LTS Java/11.0.17 kotlin/1.3.72-release-468 (1.3.72) vendor/
Amazon.com_Inc. md/internal exec-env/AWS_ECS_FARGATE io/sync http/Apache cfg/retry-
mode/standard AwsCrypto/2.4.0",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-west-2:123456789012:key/2321abab-2110-12ab-a123-
a2b34c5abc67",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "encryptionContext": {
            "aws:scheduler:schedule:arn": "arn:aws:scheduler:us-
west-2:123456789012:schedule/default/execution-role"
        }
    },
    "responseElements": null,
    "requestID": "request-id",
    "eventID": "event-id",
    "readOnly": true,
    "resources": [
        {
            "accountId": "123456789012",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:123456789012:key/2321abab-2110-12ab-a123-
a2b34c5abc67"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
```

```
    "eventCategory": "Management",
    "tlsDetails": {
      "tlsVersion": "TLSv1.3",
      "cipherSuite": "TLS_AES_256_GCM_SHA384",
      "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
  }
}
```

EventBridge スケジューラでの転送中の暗号化

EventBridge スケジューラは、転送中のデータがネットワークを移動する際に暗号化されます。Transport Layer Security (TLS) は、EventBridge スケジューラ API オペレーションを呼び出すときと、EventBridge スケジューラがスケジュールを呼び出すときにターゲット API を呼び出すときに、データを暗号化します。EventBridge スケジューラはデフォルトでは TLS 1.2 を使用して、転送中のデータを暗号化します。EventBridge スケジューラを使用する場合、転送中に暗号化を設定する必要はなく、別の TLS バージョンを選択することもできません。

EventBridge スケジューラ API の使用 — `CreateSchedule` などの API オペレーションを実行すると、EventBridge スケジューラはリクエスト本文とヘッダーを含む HTTP リクエスト全体を暗号化します。EventBridge スケジューラは、API から受け取るレスポンスオブジェクト全体も暗号化します。

ターゲット API の使用 — EventBridge スケジューラがスケジュールを呼び出すと、スケジュールの作成時に指定したターゲット API が呼び出されます。イベントをターゲットに配信する際、EventBridge スケジューラは、リクエスト本文とすべてのヘッダーを含むリクエスト全体、およびターゲットから受信する応答を暗号化します。

Amazon EventBridge スケジューラでのコンプライアンス検証

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、AWS のサービス「コンプライアンスプログラムによる対象範囲内」の「コンプライアンス」を参照し、関心のあるコンプライアンスプログラムを選択します。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。

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 AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。を使用する際のコンプ

ライアンス責任の詳細については AWS のサービス、AWS 「 セキュリティドキュメント」を参照してください。

Amazon EventBridge スケジューラでの耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。 は、低レイテンシー、高スループット、高度に冗長なネットワークで接続された、物理的に分離および分離された複数のアベイラビリティーゾーン AWS リージョン を提供します。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ー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、フォールトトレランス、および拡張性が優れています。

AWS リージョン および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。

EventBridge スケジューラには、 AWS グローバルインフラストラクチャに加えて、データの耐障害性とバックアップのニーズをサポートするのに役立つ機能がいくつか用意されています。

Amazon EventBridge スケジューラ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 スケジューラは、 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 がインフラストラクチャ AWS 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環境を AWS 設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS 公開された API コールを使用して、ネットワーク経由で EventBridge スケジューラにアクセスします。クライアントは次をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは Java 7 以降など、ほとんどの最新システムでサポートされています。

Amazon EventBridge スケジューラのモニタリングとメトリクス

モニタリングは、Amazon EventBridge スケジューラとその他 AWS ソリューションの信頼性、可用性、およびパフォーマンスの維持における重要な要素です。AWS は、EventBridge スケジューラをモニタリングし、問題が発生した場合には報告を行い、必要に応じて自動アクションを実行するために以下のモニタリングツールを提供しています。

- Amazon CloudWatch は、AWS のリソースおよび AWS で実行しているアプリケーションをリアルタイムでモニタリングします。メトリクスを収集および追跡し、カスタマイズされたダッシュボードを作成し、指定されたメトリックが指定したしきい値に達したときに通知またはアクションを実行するアラームを設定できます。詳細については、『Amazon CloudWatch ユーザーガイド』を参照してください。
- AWS CloudTrail は、AWSアカウントにより、またはそのアカウントに代わって行われた API コールや関連イベントを取得し、指定した Amazon S3 バケットにログファイルを配信します。AWS を呼び出したユーザーとアカウント、呼び出し元の IP アドレス、および呼び出しの発生日時を特定できます。詳細については、AWS CloudTrailユーザーガイドを参照してください。

トピック

Amazon CloudWatch による Amazon EventBridge スケジューラのモニタリング

Amazon CloudWatch を使用して Amazon EventBridge スケジューラをモニタリングすることで、raw データを収集し、ほぼリアルタイムに処理して読み取り可能なメトリクスにできます。EventBridge スケジューラは、すべてのスケジュールのメトリクスセットと、デッドレターキュー (DLQ) が関連付けられているスケジュールの追加のメトリクスセットを出力します。スケジュールに DLQ を設定すると、スケジュールが再試行ポリシーを使い果たしたときに、EventBridge スケジューラは追加のメトリクスを公開します。

これらの統計は 15 か月間保持されるため、履歴情報にアクセスして、スケジュールが失敗する理由をより的確に把握し、根本的な問題のトラブルシューティングを行うことができます。また、特定の

しきい値を監視するアラームを設定し、これらのしきい値に達したときに通知を送信したりアクションを実行したりできます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

トピック

用語

名前空間

名前空間は、 AWS サービスの CloudWatch メトリクスのコンテナです。EventBridge スケジューラの場合、名前空間は `AWS/Scheduler` です。

CloudWatch メトリクス

CloudWatch メトリクスは、CloudWatch に特異的な時系列のデータポイントのセットを表します。

ディメンション

ディメンションは、メトリクスのアイデンティティの一部である名前と値のペアです。

Unit

統計には、測定単位があります。EventBridge スケジューラの場合、単位には Count (カウント) が含まれます。

ディメンション

このセクションでは、CloudWatch のEventBridge スケジューラメトリクスの CloudWatch ディメンショングループについて説明します。

ディメンション	説明
ScheduleGroup	CloudWatch を使用してメトリクスを表示するスケジュールのグループ。まだグループを作成していない場合、EventBridge スケジューラはスケジュールを `default` グループに関連付けます。

メトリクスへのアクセス

このセクションでは、特定の EventBridge スケジューラのスケジュールについて CloudWatch のパフォーマンスメトリクスにアクセスする方法について説明します。

ディメンションのパフォーマンスメトリクスを表示する方法

1. CloudWatch コンソールで [メトリクス] ページを開きます。
2. AWS リージョンセレクタを使用してスケジュールのリージョンを選択する
3. [スケジューラ] 名前空間を選択します。
4. [すべてのメトリクス]タブで、[スケジュールグループメトリクス] などのディメンションを選択します。選択したリージョンで作成したすべてのスケジュールのメトリクスを表示するには、[アカウントメトリクス] を選択します。
5. ディメンションの CloudWatch メトリクスを選択します。例えば、[InvocationAttemptCount] や [InvocationDroppedCount] を選択し、[グラフ検索] を選択します。
6. EventBridge スケジューラメトリクスのパフォーマンス統計を表示するには、[グラフ化したメトリクス] タブを選択します。

メトリクスの一覧

次の表は、すべての EventBridge スケジューラのスケジュールのメトリクスと、DLQ を設定したスケジュールの追加のメトリクスを示しています。

すべてのスケジュールのメトリクス

名前空間	メトリクス	単位	説明
AWS/Scheduler	InvocationAttemptCount	カウント	呼び出しを試みるたびに発生します。このメトリクスを使用して、EventBridge スケジューラがスケジュールを呼び出そうとしていることを確認し、呼び出しがアカウントのクォータに近づく時期を確認します。
AWS/Scheduler	TargetErrorCount	カウント	EventBridge スケジューラがターゲット API を呼び出した後にターゲットが例外を返したときに発生します。これを使用して、ターゲットへの配信がいつ失敗したかを確認します。
AWS/Scheduler	TargetErrorThrottledCount	カウント	ターゲットによる API スロットリングによりターゲットの呼び出しが失敗した場合に発生します。EventBridge スケジューラによるターゲット API スロットリング呼び出しが根本的な原因である場合に、こ

名前空間	メトリクス	単位	説明
			れを使用して配信障害を診断します。
AWS/Scheduler	InvocationThrottleCount	カウント	EventBridge スケジューラによって設定されたサービスクォータを超過したために、EventBridge スケジューラがターゲットの呼び出しを抑制したときに発生します。これを使用して、呼び出しスロットリング制限クォータをいつ超過したかを判断します。サービスクォータの詳細については、「クォータ」を参照してください。
AWS/Scheduler	InvocationDroppedCount	カウント	スケジュールの再試行ポリシーが使い果たされた後に、EventBridge スケジューラがターゲットを呼び出す試みを停止したときに発生します。再試行ポリシーの詳細については、「EventBridge スケジューラ API リファレンス」の「RetryPolicy」を参照してください。

DLQ を含むスケジュールのメトリクス

名前空間	メトリクス	単位	説明
AWS/Scheduler	InvocationsSentToDeadLetterCount	カウント	スケジュールの DLQ への配信が成功するたびに発生します。これを使用して、イベントが DLQ に送信されるタイミングを判断し、スケジュールの DLQ に配信されたイベントをチェックして、障害の原因を特定するのに役立つ詳細情報を確認します。
AWS/Scheduler	InvocationsFailedToBeSentToDeadLetterCount	カウント	EventBridge スケジューラが DLQ にイベントを配信できない場合に発生します。この 2 つのメトリクスを使用して、EventBridge スケジューラが
AWS/Scheduler	InvocationsFailedToBeSentToDeadLetterCount_<error_code>	カウント	

名前空間	メトリクス	単位	説明
			DLQ にイベントを送信できない理由を特定し、DLQ 設定を変更して問題を解決します。 以下は、DLQ として指定した Amazon SQS キューが存在しない場合の `InvocationsFailedToBeSentToDeadLetterCount_<error_code>` メトリクスの例です: `InvocationsFailedToBeSentToDeadLetterCount_` **`AWS.SimpleQueueService.NonExistentQueue`**

名前空間	メトリクス	単位	説明
AWS/Scheduler	InvocationsSentToDeadLetterCount_Truncated_MessageSizeExceeded	カウント	DLQ に送信されたイベントのペイロードが Amazon SQS で許可されている最大サイズを超え、EventBridge スケジューラがスケジュールの Input 属性で指定されたペイロードを切り捨てたときに発生します。

EventBridge スケジューラの使用状況メトリクス

CloudWatch は、一部の AWS リソースの使用状況を追跡するメトリクスを収集します。これらのメトリクスは、 AWS サービスクォータに対応しています。これらのメトリクスを追跡することで、クォータを積極的に管理できます。サービスクォータの詳細については、「クォータ」を参照してください。

これらのメトリクスは ではなく AWS/Usage名前空間に含まれAWS/Scheduler、1 分ごとに収集されます。 は、ディメンション Service、、Class、および を使用してTypeこの名前空間にメトリクス CloudWatch を発行しますResource。

使用状況メトリクスの一般的なディメンション

ディメンション	説明
Service	リソースを含む AWS サービスの名前。 EventBridge スケジューラ 使用状況メトリクスの場合、値は ですScheduler 。

ディメンション	説明
Class	追跡されているリソースのクラス。 EventBridge スケジューラ 使用状況メトリクスの場合、値は ですNone。
Type	追跡されるリソースのタイプ。API 使用状況メトリクスの場合、値は ですAPI。リソース数メトリクスの場合、値は ですResource。
Resource	追跡される特定のリソース。API 使用状況メトリクスの場合、これは API オペレーション名です。リソース数メトリクスの場合、これはカウントされるリソースタイプです。

API 使用状況メトリクス

API 使用状況メトリクスは、アカウントで実行された API オペレーションの数を追跡します。これらのメトリクスを使用して、API コールボリュームをモニタリングし、API レートクォータを管理します。

メトリクス名は ですCallCount。このメトリクスで最も有用な統計は です。これはSUM、1 分間の合計オペレーション数を表します。

API 使用状況メトリクス

メトリクス	[リソース]	説明
CallCount	CreateSchedule	アカウントで実行された CreateSchedule API オペレーションの数。
CallCount	CreateSch eduleGroup	アカウントで実行された CreateSch eduleGroup API オペレーションの数。
CallCount	DeleteSchedule	アカウントで実行された DeleteSchedule API オペレーションの数。
CallCount	DeleteSch eduleGroup	アカウントで実行された DeleteSch eduleGroup API オペレーションの数。
CallCount	GetSchedule	アカウントで実行された GetSchedule API オペレーションの数。

メトリクス	[リソース]	説明
CallCount	GetScheduleGroup	アカウントで実行された GetScheduleGroup API オペレーションの数。
CallCount	ListScheduleGroups	アカウントで実行された ListScheduleGroups API オペレーションの数。
CallCount	ListSchedules	アカウントで実行された ListSchedules API オペレーションの数。
CallCount	ListTagsForResource	アカウントで実行された ListTagsForResource API オペレーションの数。
CallCount	TagResource	アカウントで実行された TagResource API オペレーションの数。
CallCount	UntagResource	アカウントで実行された UntagResource API オペレーションの数。
CallCount	UpdateSchedule	アカウントで実行された UpdateSchedule API オペレーションの数。

たとえば、次のディメンションを持つ CallCountメトリクスは、アカウントで CreateSchedule API オペレーションが呼び出された回数を示します。

- "Service": "Scheduler"
- 「クラス」：「なし」
- "Type": "API"
- "Resource": "CreateSchedule"

リソース数メトリクス

リソース数メトリクスは、アカウントのリソースのおおよその数を追跡します。これらのメトリクスを使用して、サービスクォータの制限に近づいているタイミングをモニタリングし、容量が不足する前にクォータの引き上げをリクエストできます。

メトリクス名は ですResourceCount。このメトリクスの最も有用な統計は Maximum です。

リソース数メトリクス

メトリクス	[リソース]	説明
ResourceCount	ApproximateSchedule	アカウント内のスケジュールの概算数。このメトリクスを使用して、スケジュールのクォータ制限に近づいているタイミングをモニタリングし、 を呼び出す際のServiceQuotaExceededException エラーを回避しますCreateSchedule 。 スケジュールの数が 100 万未満の場合、このメトリクスは 0 と表示される場合があります。スケジュールクォータに近づいたときに通知するアラームについては、デフォルトのスケジュールクォータが 1,000 万であるため、しきい値が 500 万以上の Maximum 統計を使用することをお勧めします。
ResourceCount	ApproximateScheduleGroup	アカウント内のスケジュールグループのおおよその数。このメトリクスを使用して、スケジュールグループのクォータ制限に近づいているタイミングをモニタリングし、を呼び出す際のServiceQuotaExceededException エラーを回避するのに役立ちますCreateScheduleGroup 。

たとえば、次のディメンションを持つ ResourceCountメトリクスと Maximum 統計は、アカウントのスケジュールのおおよその数を示します。

- "Service": "Scheduler"
- 「クラス」：「なし」
- 「タイプ」：「リソース」
- 「リソース」：「ApproximateSchedule」

AWS CloudTrail を使用して Amazon EventBridge スケジューラ API コールをログ記録する

Amazon EventBridge スケジューラは、EventBridge スケジューラのユーザー、ロール、または AWS のサービス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 AWS CloudTrail と統合されています。CloudTrail は、EventBridge スケジューラのすべての API コールをイベントとしてキャプチャします。キャプチャされる呼び出しには、EventBridge スケジューラのコンソールからの呼び出しと、EventBridge スケジューラ API オペレーションへのコード呼び出しが含まれます。証跡を作成する場合は、EventBridge スケジューラのイベントなど、Amazon S3 バケットへの CloudTrail イベントの CD を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールの [イベント履歴] で最新のイベントを表示できます。CloudTrail で収集された情報を使用して、EventBridge スケジューラに対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

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

CloudTrail における EventBridge スケジューラの情報

CloudTrail は、AWS アカウントを作成すると、その中で有効になります。アクティビティが EventBridge スケジューラで発生すると、そのアクティビティは [イベント履歴] の他の AWS サービスのイベントとともに、CloudTrail イベントに記録されます。最近のイベントは、AWS アカウントで表示、検索、ダウンロードできます。詳細については、 CloudTrail イベント履歴でのイベントの表示を参照してください。

EventBridge スケジューラのイベントなど、AWS アカウント のイベントの継続的な記録を残すには、証跡を作成します。追跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョン に適用されます。証跡は、AWS パーティションのすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集したイベントデータをより詳細に分析し、それに基づいて対応するため、他の AWS サービスを構成できます。詳細については、次を参照してください:

- 追跡を作成するための概要
- 「CloudTrail がサポートされているサービスと統合」
- CloudTrail の Amazon SNS 通知の設定
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

EventBridge スケジューラのすべての API アクションは CloudTrail によってログに記録されます。これらのアクションについては、「Amazon EventBridge スケジューラ API リファレンス」で説明しています。例えば、`CreateSchedule`、`UpdateSchedule`、`DeleteSchedule` の各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

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、以下を判別するのに役立ちます:

- リクエストが、ルート認証情報と AWS Identity and Access Management (IAM) ユーザー認証情報のどちらを使用して送信されたか。
- 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 サービスによって行われたかどうか。

詳細については、「CloudTrail userIdentity エレメント」を参照してください。

EventBridge スケジューラのログファイルエントリについて

「トレイル」は、指定した Amazon S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail のログファイルは、単一か複数のログエントリを含みます。イベントは任意ソースからの単一リクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどの情報を含みます。CloudTrail ログファイルは、パブリック API 呼び出しの順序付けられたスタックトレースではないため、特定の順序では表示されません。

Amazon EventBridge スケジューラのクォータ

AWS アカウントには、 AWS サービスごとに、以前は制限と呼ばれていたデフォルトのクォータがあります。特に明記していない限り、クォータはリージョン固有です。大部分のクォータは引き上げをリクエストできますが、できないものもあります。

EventBridge スケジューラのクォータを表示するには、Service Quotas コンソールを開きます。ナビゲーションペインで [AWS サービス] を選択し、次に [EventBridge スケジューラ] を選択します。

クォータの引き上げをリクエストするには、「Service Quotas ユーザーガイド」の「Requesting a quota increase」(クォータ引き上げリクエスト) を参照してください。Service Quotas でクォータがまだ利用できない場合は、[上限引き上げ] フォームを使用してください。

AWS アカウントには、EventBridge スケジューラに関連する次のクォータがあります。

名前	デフォルト	引き上げ可能	説明
CreateSchedule リクエストレート	us-east-1: 5,000 us-east-2: 5,000 us-west-2: 5,000 ap-northeast-1: 5,000 ap-south-1: 5,000 ap-southeast-1: 5,000 ap-southeast-2: 5,000	あり	1 秒あたりの CreateSchedule リクエストの最大数。このクォータに達すると、EventBridge スケジューラは残りの間、この操作のリクエストを拒否します。これは、1 秒あたり数万件のリクエストに調整できます。

名前	デフォルト	引き上げ可能	説明
	eu-central-1: 5,000 eu-west-1: 5,000 eu-west-2: 5,000 sa-east-1: 5,000 他のサポートされている各リージョン: 250		
CreateScheduleGroup リクエストレート	サポートされている各リージョン: 10	あり	1 秒あたりの CreateScheduleGroup リクエストの最大数。このクォータに達すると、EventBridge スケジューラは残りの間、この操作のリクエストを拒否します。

名前	デフォルト	引き上げ可能	説明
DeleteSchedule リクエストレート	us-east-1: 1,000 us-east-2: 1,000 us-west-2: 1,000 ap-northeast-1: 1,000 ap-south-1: 1,000 ap-southeast-1: 1,000 ap-southeast-2: 1,000 eu-central-1: 1,000 eu-west-1: 1,000 eu-west-2: 1,000 sa-east-1: 1,000 他のサポートされている各リージョン: 250	あり	1 秒あたりの DeleteSchedule リクエストの最大数。このクォータに達すると、EventBridge スケジューラは残りの間、この操作のリクエストを拒否します。これは、1 秒あたり数万件のリクエストに調整できます。

名前	デフォルト	引き上げ可能	説明
DeleteScheduleGroup リクエストレート	サポートされている各リージョン: 10	あり	1 秒あたりの DeleteScheduleGroup リクエストの最大数。このクォータに達すると、EventBridge スケジューラは残りの間、この操作のリクエストを拒否します。

名前	デフォルト	引き上げ可能	説明
GetSchedule リクエストレート	us-east-1: 1,000 us-east-2: 1,000 us-west-2: 1,000 ap-northeast-1: 1,000 ap-south-1: 1,000 ap-southeast-1: 1,000 ap-southeast-2: 1,000 eu-central-1: 1,000 eu-west-1: 1,000 eu-west-2: 1,000 sa-east-1: 1,000 他のサポートされている各リージョン: 250	あり	1 秒あたりの GetSchedule リクエストの最大数。このクォータに達すると、EventBridge スケジューラは残りの間、この操作のリクエストを拒否します。これは、1 秒あたり数万件のリクエストに調整できます。

名前	デフォルト	引き上げ可能	説明
GetScheduleGroup リクエストの最大数。	サポートされている各リージョン: 10	あり	1 秒あたりの GetScheduleGroup リクエストの最大数。このクォータに達すると、EventBridge スケジューラは残りの間、この操作のリクエストを拒否します。

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: 1,000 us-east-2: 1,000 us-west-2: 1,000 ap-northeast-1: 1,000 ap-south-1: 1,000 ap-southeast-1: 1,000 ap-southeast-2: 1,000 eu-central-1: 1,000 eu-west-1: 1,000 eu-west-2: 1,000 sa-east-1: 1,000 他のサポートされている各リージョン:500	あり	呼び出しは、定義されたターゲットに配信されるスケジュールペイロードです。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。これは 1 秒あたり数万件のトランザクションに調整できます。

名前	デフォルト	引き上げ可能	説明
ListScheduleGroups リクエストレート	サポートされている各リージョン: 10	あり	1 秒あたりの ListScheduleGroups リクエストの最大数 このクォータに達すると、EventBridge スケジューラは残りの間、この操作のリクエストを拒否します。
ListSchedules リクエストレート	サポートされている各リージョン: 50	可能	1 秒あたりの ListSchedules リクエストの最大数。このクォータに達すると、EventBridge スケジューラは残りの間、この操作のリクエストを拒否します。
ListTagsForResource のリクエストレート	サポートされている各リージョン: 10	あり	スケジューラ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します。
スケジュールグループの数	サポートされている各リージョン: 500	あり	リージョンあたりのスケジュールグループの最大数。

名前	デフォルト	引き上げ可能	説明
スケジュールの数	サポートされている各リージョン: 10,000,000	あり	リージョンあたりのスケジュールの最大数。このクォータには、実行が完了した 1 回限りのスケジュールが含まれます。ActionAfterCompletion 機能を使用して、完了後にスケジュールが自動的に削除されるよう設定することをお勧めします。これは数十億件のスケジュールに調整できます。
TagResource リクエストレート	サポートされている各リージョン: 1	あり	指定されたスケジューラリソースに 1 つ以上のタグ (キーと値のペア) を割り当てます。
UntagResource リクエストレート	サポートされている各リージョン: 1	あり	指定したスケジューラリソースから 1 つまたは複数のタグを削除します。

名前	デフォルト	引き上げ可能	説明
UpdateSchedule リクエストレート	us-east-1: 1,000 us-east-2: 1,000 us-west-2: 1,000 ap-northeast-1: 1,000 ap-south-1: 1,000 ap-southeast-1: 1,000 ap-southeast-2: 1,000 eu-central-1: 1,000 eu-west-1: 1,000 eu-west-2: 1,000 sa-east-1: 1,000 他のサポートされている各リージョン: 250	あり	1 秒あたりの UpdateSchedule の最大数。このクォータに達すると、EventBridge スケジューラは残りの間、この操作のリクエストを拒否します。これは、1 秒あたり数万件のリクエストに調整できます。

EventBridge スケジューラのクォータとサービスエンドポイントの詳細については、「AWS ジェネラルリファレンスガイド」の「Amazon EventBridge スケジューラのエンドポイントとクォータ」を参照してください。

EventBridge スケジューラのクォータのトラブルシューティング

次の情報は、EventBridge スケジューラのクォータに関して発生する可能性がある一般的な問題の診断や修復に役立ちます。

ServiceQuotaExceededException

デフォルトのレート制限を下回っていても、`CreateSchedule`、`DeleteSchedule`、`GetSchedule`、または `UpdateSchedule` リクエストレートでスロットリングエラーが発生しています。

一般的な原因

2023 年 9 月 7 日、EventBridge スケジューラは、実行ロールの信頼ポリシーで Schedule ARN の代わりに ScheduleGroup ARN (Amazon リソースネーム) のサポートを開始しました。信頼ポリシーで Schedule ARN を引き続き使用するように許可リストに登録されているお客様は、デフォルトの制限である 250 ~ 1000 TPS (リージョンによって異なります) ではなく、50 TPS の制限が適用される場合があります。

解決方法

サポートに連絡して、上限の引き上げをリクエストします。

防止

次のいずれかの方法で、既存の信頼ポリシーを変更します。

- ロールからすべてのスコープを削除します。
- Schedule ARN または ScheduleGroup ARN を使用して引き受けられるようにロールをスコープします。

  例えば、次の既存の信頼ポリシーがあるとします。

```
{
    "Effect": "Allow",
```

```
    "Principal": {
        "Service": "scheduler.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
        "StringEquals": {
            "aws:SourceArn":
 "arn:aws:scheduler:region:account:schedule/schedule_group/schedule"
        }
    }
}
```

信頼ポリシーを次のように更新できます。

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "scheduler.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
        "ForAnyValue:StringEquals": {
            "aws:SourceArn": [
                "arn:aws:scheduler:region:account:schedule/schedule_group/schedule",
                "arn:aws:scheduler:region:account:schedule-group/schedule_group"
            ]
        }
    }
}
```

Amazon EventBridge スケジューラのトラブルシューティング

このセクションのトピックを使用して、Amazon EventBridge スケジューラの一般的な問題のトラブルシューティングを行えます。

トピック

スケジュールがターゲットエラーで失敗する

ターゲット呼び出しの失敗は、EventBridge スケジューラの最も一般的な問題の 1 つです。これらの障害は、以下のいくつかの理由で発生する可能性があります。

一般的な原因:

- ターゲットパラメータがないか、正しくない。
- ネットワーク接続の問題。
- API スロットリング。
- ターゲット設定が正しくない。

トラブルシューティングのステップ

1. デッドレターキュー (DLQ) を設定する

- DLQ は、失敗した呼び出しをキャプチャして分析するのに役立ちます。
- 失敗した呼び出しは、詳細なエラーメッセージとともに DLQ に送信されます。
- DLQ を設定するには、スケジュール設定に追加します。

```
{
    "DeadLetterConfig": {
        "Arn": "arn:aws:sqs:region:account-id:MyDLQ"
    }
}
```

注: DLQ が KMS キーで暗号化されている場合は、キーポリシーで EventBridge スケジューラによる使用が許可されていることを確認してください。

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

2. API パラメータを確認する
   - ターゲット API コールに必要なすべてのパラメータが存在し、正しくフォーマットされていることを確認します。
   - パラメータ値が許容範囲内であることを確認します。
   - VPC エンドポイントを使用している場合は、VPC から API エンドポイントにアクセスできることを確認します。
3. ネットワーク設定を確認する
   - 一時的なネットワークの問題が原因で呼び出しが失敗する場合は、再試行ロジックを実装します。

- 再試行ポリシーの例:

```
{
    "RetryPolicy": {
        "MaximumRetryAttempts": 3,
        "MaximumEventAgeInSeconds": 3600
    }
}
```

4. ターゲット固有の設定を確認する
   - テンプレート化されたターゲット (ECS タスクなど) の場合は、スケジュール作成 API の `Target.Input` パラメータを使用してオーバーライドを指定してください。
   - ターゲットサービスがサポートされており、正しく設定されていることを確認します。

スケジュール実行ロールのアクセス許可の問題

IAM ロールのアクセス許可の問題は、スケジュールの実行が失敗する一般的な理由です。これらの問題のトラブルシューティングと解決方法は次のとおりです。

一般的な原因

- ターゲットサービスに必要なアクセス許可がない
- スケジュールのロール設定が正しくない
- EventBridge スケジューラサービスとの信頼関係がない
- 暗号化されたリソースにアクセスするためのアクセス許可が不十分

症状

- CloudWatch での `TargetErrorCount` メトリクスの増加
- スケジュール設定で明らかな問題なくスケジュールを実行できない

トラブルシューティングのステップ

1. CloudWatch メトリクスのモニタリング

- CloudWatch で TargetErrorCount メトリクスをチェックします。

2. デッドレターキュー (DLQ) を使用してアクセス許可の問題を確認する
   - スケジュールに合わせて DLQ を設定します。
   - ターゲットにアクセス許可の問題があり、DLQ が正しく設定されている場合、DLQ に失敗した呼び出しとアクセス許可関連のエラーメッセージが表示されます。
   - CloudWatch メトリクスに失敗した実行にもかかわらず DLQ が空のままである場合、EventBridge スケジューラが DLQ 自体に書き込むことを妨げるアクセス許可の問題を示している可能性があります。

> **Note**
>
> DLQ 自体に正しいアクセス許可があるかどうかを確認します。暗号化されている場合は、EventBridge スケジューラに KMS キーを使用するアクセス許可があることを確認してください。

3. 信頼関係を検証する
   - IAM ロールに EventBridge スケジューラとの正しい信頼関係があることを確認します。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "scheduler.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }]
}
```

4. スケジュール実行ロールのアクセス許可を確認する
   - スケジュールの実行ロールには、さまざまなターゲットタイプを呼び出すための特定のアクセス許可が必要です。
   - スケジュールの実行ロールポリシーに含めるアクセス許可の例:

```
// For Lambda function targets - add to schedule execution role
```

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "lambda:InvokeFunction"
        ],
        "Resource": "arn:aws:lambda:region:account-id:function:function-name"
    }]
}

// For SQS queue targets - add to schedule execution role
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "sqs:SendMessage"
        ],
        "Resource": "arn:aws:sqs:region:account-id:queue-name"
    }]
}
```

5. 暗号化されたリソースアクセスを確認する

- ターゲットが暗号化されたリソース (KMS で暗号化された SQS キューなど) を使用している場合は、ロールに KMS キーを使用するアクセス許可があることを確認します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:region:account-id:key/key-id"
        }
    ]
}
```

6. ロール ARN 設定を確認する
   - スケジュール設定のロール ARN が正しいことを確認します。
   - スケジュールと同じ AWS アカウント およびリージョンにロールが存在することを確認します。

Service Quotas の理解と管理

スケジュールの作成や呼び出しのスロットリングに問題がある場合は、Service Quotas の制限に達している可能性があります。EventBridge スケジューラには、スケジュール数、スケジュールグループ、呼び出しレートのクォータがあり、リージョンによって異なる場合があります。

クォータの問題の特定

クォータ制限に達しているかどうかを判断するには:

1. CloudWatch メトリクスのモニタリング
   - `InvocationThrottleCount` メトリクスを確認します。このメトリクスの増加は、呼び出しレートの制限を超えていることを示します。
   - `InvocationAttemptCount` メトリクスを確認して、現在の使用状況を把握します。
2. 特定のエラーメッセージを監視する
   - スケジュールを作成または変更する場合、`LimitExceededException` はスケジュールまたはスケジュールグループの最大数に達したことを示します。
   - スロットリングエラーを返す API コールは、API リクエストのクォータを超えていることを示します。

クォータの問題の解決

クォータ制限に達していると判断した場合:

1. 現在のスケジュールを確認して最適化します。同様のスケジュールを統合するか、未使用のスケジュールを削除することを検討してください。
2. API スロットリングの場合は、API コールにバックオフによる再試行を実装します。

3. より高いクォータが必要な場合は、Service Quotas コンソールから引き上げをリクエストしてください。EventBridge スケジューラを選択し、引き上げるクォータを選択し、ビジネス上の理由とともにリクエストを送信します。

スケジュールパターンとトリガータイミングの問題

ユーザーは、予定された時間にスケジュールがトリガーされない問題に遭遇することがあります。これは最も一般的に、スケジュールパターン、夏時間の変更、または柔軟な時間枠に関する誤解が原因である可能性があります。

一般的な原因

- cron 式の誤解。
- 夏時間の変更中の予期しない動作。
- 柔軟な時間枠に関する混乱。
- rate 式の誤解。

トラブルシューティングのステップ

1. cron 式を検証する
   - cron 式の形式が正しいことを確認します。
   - cron 式では日フィールドと曜日フィールドの両方を同時に指定することはできないことに注意してください。
2. タイムゾーンに関する考慮事項
   - スケジュールの作成時に希望するタイムゾーンを選択します。
   - この調整は UTC に基づいているため、夏時間がスケジュールにどのように影響するかを理解します。

   夏時間への影響の例: GMT 午前 7 時に実行するようにスケジュールを設定した場合:
   - 冬季: スケジュールは GMT 午前 7 時に実行されます (GMT = UTC)
   - 夏季: スケジュールは引き続き UTC 午前 7:00 に実行されます。現在は GMT/BST 午前 6:00 です。

スケジュールを一年中同じ現地時間で実行する必要がある場合は、スケジュールの作成時に適切なタイムゾーンを選択し、夏時間がどのようにそのタイムゾーンに影響するかを確認してください。

3. 柔軟な時間枠を理解する
   - 柔軟な時間枠により、EventBridge スケジューラは呼び出しを最適化できます。
   - スケジュールは、時間枠の開始時に正確にトリガーされない場合があります。
   - 実際の呼び出し時間をモニタリングして、動作を理解します。
4. rate 式と cron 式を確認する
   - rate 式の形式が正しいことを確認します (例: `rate(5 minutes)`、`rate(1 hour)`)。
   - rate 式と cron 式の両方でスケジュール呼び出しは 1 分間の 0 秒にクランプされないことに注意してください。
   - スケジュールは、指定された 1 分以内にトリガーされる場合がありますが、必ずしも 1 分の正確な開始時にトリガーされるとは限りません。

   例えば、次のようになります。
   - `rate(1 hour)` を使用するスケジュールは、午後 2:00:45、午後 3:00:32、午後 4:00:18 などに実行される場合があります。
   - `0 * * * ? *` (1 時間ごと) に設定された cron スケジュールは、午後 2 時 00 分 15 秒、午後 3 時 00 分 7 秒、午後 4 時 00 分 52 秒などに実行される場合があります。
5. CloudWatch メトリクスのモニタリング
   - `InvocationAttemptCount` メトリクスを使用して、スケジュールがトリガーされているかどうかを確認します。
   - 呼び出しが失敗しているかどうか、`TargetErrorCount` を確認します。
   - デッドレターキューを設定している場合は、`InvocationsSentToDeadLetterCount` をモニタリングして、失敗した呼び出しを追跡します。

スケジュールパターンと cron 式の作成

ユーザーが、特に cron 式でスケジュールパターンを作成するときに、問題が発生することがよくあります。一般的な問題とその対処方法は次のとおりです。

一般的な問題

- cron 構文が正しくない
- サポートされていない cron 機能の使用を試みる
- 一緒に使用できるフィールドに関する混乱

トラブルシューティングのステップ

1. cron 式の構文を確認する
   - cron 式が次のような正しい形式であることを確認してください。`Minutes Hours Day-of-month Month Day-of-week Year`
   - EventBridge スケジューラは、追加の Year フィールドで cron 標準を使用することに注意してください。
2. 制限を理解する
   - ここで説明するように、日フィールドと曜日フィールドの両方を同時に指定することはできません。
   - 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。
3. スケジュールプレビュー機能を使用する
   - スケジュールを作成または編集すると、EventBridge スケジューラは次の 10 の実行時間のプレビューを提供します。
   - このプレビューを使用して、スケジュールが意図した時間に実行されることを確認します。
   - プレビューが期待と一致しない場合は、cron 式を確認して調整します。

ターゲットがトリガーされているか。

ターゲットがトリガーされているかどうかを確認するには:

1. CloudWatch メトリクスを確認します。
   - `InvocationAttemptCount` は、試行された呼び出しの数を示します
   - `TargetErrorCount` は、呼び出しが失敗したかどうかを示します
   - `TargetErrorThrottledCount` は、ターゲットがスロットリングされているかどうかを示します
   - `InvocationDroppedCount` は、呼び出しが削除されたかどうかを示します

2. 失敗した呼び出しをキャプチャして分析するようデッドレターキュー (DLQ) を設定します。

テンプレート化されたターゲットとユニバーサルターゲット

「提供されたリクエストが無効: [サービス] がターゲットでサポートされていないサービス」などのエラーが表示された場合は、サポートされていないサービスをテンプレート化されたターゲットとして使用しようとしている可能性があります。

これを解決するには:

1. 目的のサービスがテンプレート化されたターゲットとしてサポートされているかどうかを確認します。
2. サポートされていない場合は、代わりにユニバーサルターゲットを使用し、サービスに対して適切な API コールを行うように設定します。

無効なユニバーサルターゲット入力設定

ユニバーサルターゲットを使用してスケジュールを作成すると、EventBridge スケジューラはターゲット ARN 形式を検証しますが、ダウンストリームサービスの API に対して `Input`フィールドの内容を検証しません。つまり、 にターゲットサービスが呼び出し時に拒否する値`Input`が含まれている場合でも、スケジュールは正常に作成できます。

無効なターゲット入力設定のスケジュールは、設定された式でトリガーされますが、呼び出しのたびに失敗します。スケジュールが呼び出されるまで、設定ミスを検出できない場合があります。これは、作成後数時間または数日になる可能性があります。

症状

- スケジュールはエラーなしで作成されましたが、`TargetErrorCount`CloudWatch メトリクスは呼び出しごとに増加します。
- DLQ メッセージには、 ではなく、ターゲットサービス (`InvalidParameterValueException`や などValidationException) からのエラーコードが含まれています`AWS.Scheduler.InternalServerError`。
- DLQ メッセージの `ERROR_MESSAGE`は、特定の入力パラメータの検証の失敗を参照します。

例

次の例は、ユニバーサルターゲット () の AWS Lambda 一般的な無効な入力設定を示しています`arn:aws:scheduler:::aws-sdk:lambda:invoke`。

修飾子の不一致

次の入力のスケジュールでは、 2でバージョンを指定`FunctionName`し、 `Qualifier`フィールド1でバージョンを指定します。

```
{
    "FunctionName": "MyFunction:2",
    "Qualifier": "1"
}
```

このスケジュールは正常に作成されますが、すべての呼び出しは失敗します。DLQ メッセージには以下が含まれます。

- `ERROR_CODE`: `InvalidParameterValueException`
- `ERROR_MESSAGE`: `The derived qualifier from the function name does not match the specified qualifier.`

無効な関数名

次の入力のスケジュールは、 の空白のみの値を指定します`FunctionName`。

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

DLQ メッセージには以下が含まれます。

- `ERROR_CODE`: `ValidationException`
- `ERROR_MESSAGE`: 関数名が必須パターンと一致していないことを示す検証エラー。

解決方法

1. DLQ を設定します。ユニバーサルターゲットを使用するスケジュールには、必ずデッドレターキューを設定します。DLQ メッセージ属性 (`ERROR_CODE` および `ERROR_MESSAGE`) には、無効

な入力パラメータを識別するターゲットサービスによって返される特定のエラーが含まれています。

2. ターゲットサービス API に対して入力パラメータを検証します。スケジュールを作成する前に、ターゲット API を直接呼び出して、 Inputフィールドの JSON に有効な値が含まれていることを確認します。たとえば、 API を使用して同じパラメータで AWS Lambda 関数を AWS Lambda Invoke呼び出し、リクエストが成功したことを確認します。
3. 1 回限りのスケジュールでテストします。1 回限りのスケジュールを作成して、定期的なスケジュールを設定する前にターゲット呼び出しが成功することを確認します。
4. ターゲットサービス API リファレンスを確認します。ターゲットとするサービスの API リファレンスをチェックして、必要なパラメータ、有効な値の範囲、制約を確認します。については AWS Lambda Invoke、「 AWS Lambda デベロッパーガイド」の「呼び出し」を参照してください。

予期しない呼び出しをトリガーする更新をスケジュールする

スケジュールを変更すると、呼び出しに更新されたスケジュールがすぐには反映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。例えば、元のトリガー時間に近いスケジュールを更新すると、元のスケジュール設定に基づいて呼び出しが表示されることがあります。

1 回限りのスケジュールの無効化または有効化

元のスケジュールされた時刻が経過した後に 1 回限りのスケジュールを再度有効にすると、スケジュールはすぐにターゲットを呼び出すことがあります。これは、スケジュールが元の実行時間より前に無効になっていても発生する可能性があります。

例えば、次のようになります。

- 現在の時刻: 13:15 UTC
- 1 回限りのスケジュールの作成: 13:30 UTC
- スケジュールは 13:30 UTC より前に無効になりました
- スケジュールが 14:00 UTC に再度有効になりました
- 結果: ターゲットは、再度有効にするとすぐに呼び出される可能性があります

EventBridge スケジューラユーザーガイドのドキュメント履歴

以下の表は、EventBridge スケジューラのリリースドキュメント内容を示します。

変更	説明	日付
実行ロールと混乱した代理の防止の変更	今回の更新では、ロールの権限ポリシーで混乱した代理の防止を実装した場合に、スケジュールグループリソースに実行ロールが適用される方法の変更点について説明しています。 • the section called “混乱した代理の防止”	2023 年 9 月 7 日
完了後のスケジュールの自動削除	EventBridge スケジューラは自動削除に対応しています。自動削除を設定すると、EventBridge スケジューラは最後に予定されていた呼び出しの後にスケジュールを削除します。 • the section called “スケジュール完了後の削除”	2023 年 8 月 2 日
ユニバーサルターゲットの使用に関するトピックを更新	EventBridge スケジューラがターゲットにして統合できるサポート対象サービスのリストを更新しました。この更新には、サポートされていない GET API オペレーションのリストも含まれています。また、ユニバーサルターゲット	2023 年 3 月 17 日

	の例の改善のほか、ガイド全体に対するその他の軽微な改善も含まれています。 • the section called “ユニバーサルターゲットの使用”	
開始日が設定されていないレートベースのスケジュールに関する情報を更新	`StartDate` を指定しない場合に、EventBridge スケジューラがレートベースのスケジュールをどのように管理するかに関する情報を追加しました。 • the section called “レートベースのスケジュール”	2023 年 3 月 17 日
スケジューラグループの管理に関する新しいトピック	EventBridge スケジューラでスケジューラグループを作成する方法に関する新しい章を追加しました。この章では、グループの作成、グループへのスケジュールの追加、EventBridge スケジューラのリソースの管理と監視をより簡単にするためのタグの適用、そしてグループの削除の方法を説明しています。 • スケジュールグループの管理	2023 年 3 月 17 日

夏時間とタイムゾーンに関する新しいトピック	EventBridge スケジューラが夏時間を管理する方法、および異なるタイムゾーンでスケジュールを作成する方法を説明する新しいセクションを追加しました。 • the section called “夏時間” • the section called “タイムゾーン”	2022 年 11 月 17 日
メトリクスに関する新しいトピック	EventBridge スケジューラが CloudWatch に公開するメトリクスを説明する新しいトピックを追加しました。これらのメトリクスを使用して、呼び出しの失敗を監視し、スケジュールの問題を解決する方法を把握できます。 • the section called “CloudWatch によるモニタリング”	2022 年 11 月 15 日
初回リリース	EventBridge スケジューラユーザーガイドの初回リリース。	2022 年 11 月 10 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。