Amazon CloudWatch Events

ユーザーガイド

Amazon CloudWatch Events: ユーザーガイド

Table of Contents

Amazon CloudWatch Events とは

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

Amazon CloudWatch Events は、Amazon Web Services (AWS) リソースの変更を示すシステムイベントのほぼリアルタイムのストリームを提供します。すぐに設定できる簡単なルールを使用して、ルールに一致したイベントを 1 つ以上のターゲット関数またはストリームに振り分けることができます。オペレーションの変更が発生すると、CloudWatch イベント はその変更を認識します。CloudWatch イベント は、オペレーションの変更に応答し、必要に応じて、応答メッセージを環境に送り、機能をアクティブ化し、変更を行い、状態情報を収集することによって、修正アクションを実行します。

CloudWatch イベント を使用して、cron 式や rate 式により特定の時間に自己トリガーする自動化されたアクションをスケジュールすることもできます。詳細については、「ルールのスケジュール式 (p. 34)」を参照してください。

以下の AWS のサービスを、CloudWatch イベント のターゲットとして設定できます。

- Amazon EC2 インスタンス
- AWS Lambda 関数
- Amazon Kinesis Data Streams のストリーム
- Amazon Kinesis Data Firehose の配信ストリーム
- Amazon CloudWatch Logs のロググループ
- Amazon ECS タスク
- Systems Manager コマンドを実行
- Systems Manager オートメーション
- AWS Batch ジョブ
- Step Functions ステートマシン
- CodePipeline のパイプライン
- CodeBuild プロジェクト
- Amazon Inspector の評価テンプレート
- Amazon SNS のトピック
- Amazon SQS キュー
- 組み込みターゲット: `EC2 CreateSnapshot API call`、`EC2 RebootInstances API call`、`EC2 StopInstances API call`、および `EC2 TerminateInstances API call`。
- 別の AWS アカウントのデフォルトのイベントバス

概念

CloudWatch イベント の使用を開始する前に、以下の概念を理解する必要があります。

- Events – イベント、 は、AWS 環境の変化を示します。AWS リソースは、その状態が変わったときにイベントを生成できます。たとえば、Amazon EC2 は EC2 インスタンスの状態が保留中から実行中に

変わったときにイベントを生成し、Amazon EC2 Auto Scaling はインスタンスを起動または終了したときにイベントを生成します。AWS CloudTrail は、API コールを行ったときにイベントを発行します。カスタムアプリケーションレベルのイベントを生成し、CloudWatch イベント に発行することができます。定期的に生成される予定されたイベントをセットアップすることもできます。イベントを生成するサービスの一覧や、各サービスのサンプルイベントについては、「サポートされている各サービスからの CloudWatch イベント イベントの例 (p. 43)」を参照してください。

- ルール – ルールは、一致した受信イベントを検出し、処理のためにターゲットに振り分けます。1 つのルールで、複数のターゲットを振り分けることができ、それらのすべてが並列に処理されます。ルールは特定の順序で処理されません。これにより、組織のさまざまな部署が目的のイベントを検索して処理できます。ルールは、特定の部分のみ渡したり、定数で上書きしたりすることにより、ターゲットに送信される JSON をカスタマイズできます。
- ターゲット – ターゲットはイベントを処理します。ターゲットには、Amazon EC2 インスタンス、AWS Lambda 関数、Kinesis ストリーム、Amazon ECS タスク、Step Functions ステートマシン、Amazon SNS トピック、Amazon SQS キュー、および組み込みターゲットを含めることができます。ターゲットは、JSON 形式のイベントを受け取ります。

  ルールのターゲットは、ルールと同じリージョンに存在する必要があります。

関連 AWS サービス

CloudWatch イベント と併せて使用されるサービスは次のとおりです。

- AWS CloudTrail では、アカウントの CloudWatch イベント API 宛ての呼び出し (AWS マネジメントコンソール、AWS CLI、その他のサービスによって行われる呼び出しを含む) をモニタリングすることができます。CloudTrail ログ記録がオンの場合、CloudWatch イベント はログファイルを S3 バケットに書き込みます。リクエストを満たすためにアクションをいくつ実行する必要があったかに応じて、各ログファイルには 1 個以上のレコードが含まれる可能性があります。詳細については、「AWS CloudTrail を使用した Amazon CloudWatch Events API コールのログ記録 (p. 107)」を参照してください。
- AWS CloudFormation では、AWS リソースのモデリングと設定を行うことができます。必要なすべての AWS リソースを説明するテンプレートを作成すれば、AWS CloudFormation がお客さまに代わって、これらのリソースのプロビジョニングや設定を処理します。CloudWatch イベント ルールは、AWS CloudFormation テンプレートで使用できます。詳細については、『AWS CloudFormation ユーザーガイド』の「AWS::Events::Rule」を参照してください。
- AWS Config を使用すると、AWS リソースへの設定変更を記録できます。これには、リソース間の関係と設定の履歴が含まれるため、時間の経過と共に設定と関係がどのように変わるかを確認できます。AWS Config ルールを作成して、リソースが組織のポリシーに準拠しているかどうか確認できます。詳細については、「AWS Config 開発者ガイド」を参照してください。
- AWS Identity and Access Management (IAM) は、ユーザーのために AWS リソースへのアクセスを安全にコントロールする際に役立ちます。IAM により、どのユーザーがお客様の AWS リソースを使用できるか (認証)、それらのユーザーがどのリソースをどのような方法で使用できるか (承認) を制御できます。詳細については.
- Amazon Kinesis Data Streams により、高速かつほぼ継続的にデータの取り込みと集約を行うことができます。使用されるデータのタイプには、IT インフラストラクチャのログデータ、アプリケーションのログ、ソーシャルメディア、マーケットデータフィード、ウェブのクリックストリームデータなどがあります。データの取り込みと処理の応答時間はリアルタイムであるため、処理は一般的に軽量です。詳細については、「Amazon Kinesis Data Streams 開発者ガイド」を参照してください。
- AWS Lambda により、新規情報に迅速に対応するアプリケーションを構築できます。アプリケーションコードを Lambda 関数としてアップロードします。Lambda は可用性の高いコンピューティングインフラストラクチャでお客様のコードを実行します。Lambda はコンピューティングリソースの管理をすべて担当します。これにはサーバーおよびオペレーティングシステムの管理、キャパシティーのプロビジョニングおよび自動スケーリング、コードおよびセキュリティパッチのデプロイ、モニタリングおよびログ記録などが含まれます。詳細については、『AWS Lambda Developer Guide』を参照してください。

Amazon CloudWatch Events のセットアップ

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

Amazon CloudWatch Events を使用するには、AWS アカウントが必要です。AWS アカウントがあれば、Amazon EC2 などのサービスを利用して、CloudWatch コンソール (ウェブベースのインターフェース) で表示できるイベントを生成できます。加えて、AWS Command Line Interface (AWS CLI) をインストールおよび設定してコマンドラインインターフェイスを使用することができます。

アマゾン ウェブ サービス (AWS) にサインアップ

AWS アカウントを作成すると、すべての AWS サービスに自動的にサインアップされます。料金が発生するのは、お客様が使用したサービスの分のみです。

既に AWS アカウントをお持ちの場合は次の手順に進んでください。AWS アカウントをお持ちでない場合は、次に説明する手順にしたがってアカウントを作成してください。

サインアップして AWS アカウントを作成するには

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

   サインアップ手順の一環として、通話呼び出しを受け取り、電話のキーパッドを用いて確認コードを入力することが求められます。

Amazon CloudWatch コンソールにサインインする

Amazon CloudWatch コンソールにサインインするには

1. AWS マネジメントコンソール にサインインした後、https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. 必要に応じてリージョンを変更します。ナビゲーションバーから、AWS リソースがあるリージョンを選択します。
3. ナビゲーションペインの [Events] を選択します。

アカウント認証情報

ルートユーザー認証情報を使用して CloudWatch イベント にアクセスできますが、AWS Identity and Access Management (IAM) アカウントを使用することをお勧めします。IAM アカウントを使用して CloudWatch にアクセスする場合、次のアクセス権限が必要です。

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

コマンドラインインターフェイスをセットアップする

AWS CLI を使用して、CloudWatch イベント オペレーションを実行できます。

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

リージョンのエンドポイント

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

Amazon CloudWatch Events を使用するためのサンプルの使用シナリオを提供する

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

このセクションの手順を使用して、CloudWatch イベント ルールを作成および削除します。これらは、あらゆるイベントソースまたはターゲットに使用できる一般的な手順です。特定のシナリオおよびターゲット用に記述されたチュートリアルについては、チュートリアルを参照してください。

各ルール

目次

制限

- ルールに関連付けるターゲットは、ルールと同じリージョンに存在する必要があります。
- ターゲットタイプによっては、一部のリージョンで利用できません。詳細については、アマゾン ウェブ サービス全般のリファレンス の「リージョンとエンドポイント」を参照してください。
- 組み込みターゲットでのルール作成は、AWS マネジメントコンソール でのみサポートされています。
- 暗号化された Amazon SQS キューをターゲットとするルールを作成する場合は、次のセクションを KMS キーポリシーに含める必要があります。これにより、イベントは、暗号化されたキューに正常に配信されます。

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

```
}
```

イベントでトリガーする CloudWatch イベント ルールの作成

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

以下のステップを使用して、AWS のサービスによって生成されたイベントでトリガーする CloudWatch イベント ルールを作成します。

イベントでトリガーするルールを作成するには:

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[イベント]、[ルールの作成] の順に選択します。
3. [Event source] で、以下の操作を実行します。
   a. [サービス別のイベントに一致するイベントパターンの構築] で、[イベントパターン] を選択します。
   b. [サービス名] で、ルールをトリガーするイベントを生成するサービスを選択します。
   c. [イベントタイプ] で、ルールをトリガーする特定のイベントを選択します。唯一のオプションが [AWS API Call via CloudTrail] である場合、選択されたサービスはイベントを生成せず、このサービスに対して行われた API コールのみをルールのベースにできます。この種のルールの作成については、「AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch イベント ルールの作成 (p. 7)」を参照してください。
   d. イベントを生成するサービスに応じて、[Any...] および [Specific...] のオプションが表示される場合があります。選択された任意の種類のイベントでイベントトリガーを発生させるには [Any...] を選択し、1 つ以上の特定のイベントタイプを選択するには [Specific...] を選択します。
4. [ターゲット] で [ターゲットの追加] を選択し、選択した種類のイベントが検出されたときに対応する AWS のサービスを選択します。
5. このセクションの他のフィールドに、このターゲットタイプに固有の情報を入力します (必要な場合)。
6. 多くのターゲットタイプで、CloudWatch イベント はターゲットにイベントを送信するためのアクセス許可が必要です。これらの場合、CloudWatch イベント は、イベントの実行に必要な IAM ロールを作成できます。
   - 自動的に IAM ロールを作成するには、[この特定のリソースに対して新しいロールを作成する] を選択します。
   - 以前に作成した IAM ロールを使用するには、[既存のルールの使用] を選択します。
7. オプションで、このルールに別のターゲットを追加するには、ステップ 4～6 を繰り返します。
8. [設定の詳細] を選択します。[Rule definition] で、ルールの名前と説明を入力します。

   ルール名はこのリージョン内で一意である必要があります。
9. [Create rule] を選択します。

AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch イベント ルールの作成

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

イベントを生成しない AWS のサービスによるアクションでトリガーするルールを作成するには、そのサービスによる API コールをルールのトリガー元にすることができます。API コールは AWS CloudTrail によって記録されます。ルールのトリガーとして使用できる API コールの詳細については、「CloudTrail イベント履歴でサポートされるサービス」を参照してください。

CloudWatch イベント のルールは、作成したリージョンでのみ機能します。複数のリージョンで API コールを追跡するように CloudTrail を設定し、これらの各リージョンで CloudTrail に基づくルールをトリガーする場合は、追跡するリージョンごとに別個のルールを作成する必要があります。

CloudTrail 経由で送信されたイベントはすべて、`detail-type` の値が `AWS API Call via CloudTrail` になっています。

> **Note**
>
> CloudWatch イベント では、ルールが繰り返し開始される無限ループにつながるルールを作成する可能性があります。たとえば、S3 バケットで ACL が変更されたことを検出し、ソフトウェアをトリガーして ACL を目的の状態に変更するルールがあるとします。このルールが慎重に記述されていない場合は、その後 ACL を変更するとルールが再び開始され、無限ループが作成されます。
> これを防ぐには、トリガーされたアクションが同じルールを再び開始しないようにルールを記述します。たとえば、変更された後ではなく、エラー状態にある ACL が見つかった場合にのみ、ルールが開始されるようにします。
> 無限ループにより、予想よりも高い料金がすぐに発生する可能性があります。指定した制限を料金が超えるとアラートで知らせる予算設定を使用することをお勧めします。詳細については、「予算によるコストの管理」を参照してください。

CloudTrail を介して API コールでトリガーするルールを作成するには:

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[イベント]、[ルールの作成] の順に選択します。
3. [Event source] で、以下の操作を実行します。
   a. [サービス別のイベントに一致するイベントパターンの構築] で、[イベントパターン] を選択します。
   b. [サービス名] で、トリガーとして使用する API オペレーションを使用するサービスを選択します。
   c. [イベントタイプ] で、[AWS API Call via CloudTrail] を選択します。
   d. このサービスの API オペレーションを呼び出すときにルールをトリガーするには、[任意のオペレーション] を選択します。特定の API オペレーションを呼び出した場合にのみルールをトリガーするには、[特定のオペレーション] を選択し、横のボックスにオペレーション名を入力して Enter キーを押します。さらにオペレーションを追加するには、[+] を選択します。
4. [ターゲット] で [ターゲットの追加] を選択し、選択した種類のイベントが検出されたときに対応する AWS のサービスを選択します。

5. このセクションの他のフィールドに、このターゲットタイプに固有の情報を入力します (必要な場合)。
6. 多くのターゲットタイプで、CloudWatch イベント はターゲットにイベントを送信するためのアクセス許可が必要です。これらの場合、CloudWatch イベント は、イベントの実行に必要な IAM ロールを作成できます。

   - 自動的に IAM ロールを作成するには、[この特定のリソースに対して新しいロールを作成する] を選択します。
   - 以前に作成した IAM ロールを使用するには、[既存のルールの使用] を選択します。
7. オプションで、このルールに別のターゲットを追加するには、ステップ 4～6 を繰り返します。
8. [設定の詳細] を選択します。[Rule definition] で、ルールの名前と説明を入力します。

   ルール名はこのリージョン内で一意である必要があります。
9. [Create rule] を選択します。

スケジュールに従ってトリガーする CloudWatch イベント ルールの作成

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

次のステップを使用して、定期的なスケジュールに従ってトリガーする CloudWatch イベント ルールを作成します。

定期的なスケジュールに従ってトリガーするルールを作成するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[イベント]、[ルールの作成] の順に選択します。
3. [イベントソース] で、[スケジュール] を選択します。
4. [一定の速度] を選択してタスクを実行する頻度を指定するか、[Cron 式] を選択してタスクをいつトリガーするか定義する cron 式を指定します。cron 式の構文の詳細については、「ルールのスケジュール式 (p. 34)」を参照してください。
5. [ターゲット] で [ターゲットの追加] を選択し、選択した種類のイベントが検出されたときに対応する AWS のサービスを選択します。
6. このセクションの他のフィールドに、このターゲットタイプに固有の情報を入力します (必要な場合)。
7. 多くのターゲットタイプで、CloudWatch イベント はターゲットにイベントを送信するためのアクセス許可が必要です。これらの場合、CloudWatch イベント は、イベントの実行に必要な IAM ロールを作成できます。

   - 自動的に IAM ロールを作成するには、[この特定のリソースに対して新しいロールを作成する] を選択します。
   - 以前に作成した IAM ロールを使用するには、[既存のルールの使用] を選択します。
8. オプションで、このルールに別のターゲットを追加するには、ステップ 5～7 を繰り返します。
9. [設定の詳細] を選択します。[Rule definition] で、ルールの名前と説明を入力します。

ルール名はこのリージョン内で一意である必要があります。

10. [Create rule] を選択します。

CloudWatch イベント ルールの削除または無効化

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

次のステップを使用して CloudWatch イベント ルールを削除または無効化します。

ルールを削除または無効化するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで [ルール] を選択します。

   マネージドルールは、その名前の横にボックスアイコンがあります。詳細については、「Amazon CloudWatch Events マネージドルール (p. 103)」を参照してください。
3. 以下のいずれかを行います。

   a. ルールを削除するには、ルールの横にあるボタンを選択し、[アクション]、[削除]、[削除] の順に選択します。

      ルールがマネージドルールである場合は、ルールの名前を入力してそれがマネージドルールであること、およびルールを削除するとルールの作成元のサービスが機能しなくなる場合があることを了承します。続行するには、ルール名を入力し、[Force 削除 (強制削除)] を選択します。

   b. ルールを一時的に無効化するには、ルールの横にあるボタンを選択し、[アクション]、[無効化]、[無効化] の順に選択します。

      マネージドルールを無効化することはできません。

CloudWatch イベントのチュートリアル

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

以下のチュートリアルでは、特定のタスクとターゲット用の CloudWatch イベント ルールを作成する方法を示します。

チュートリアル:

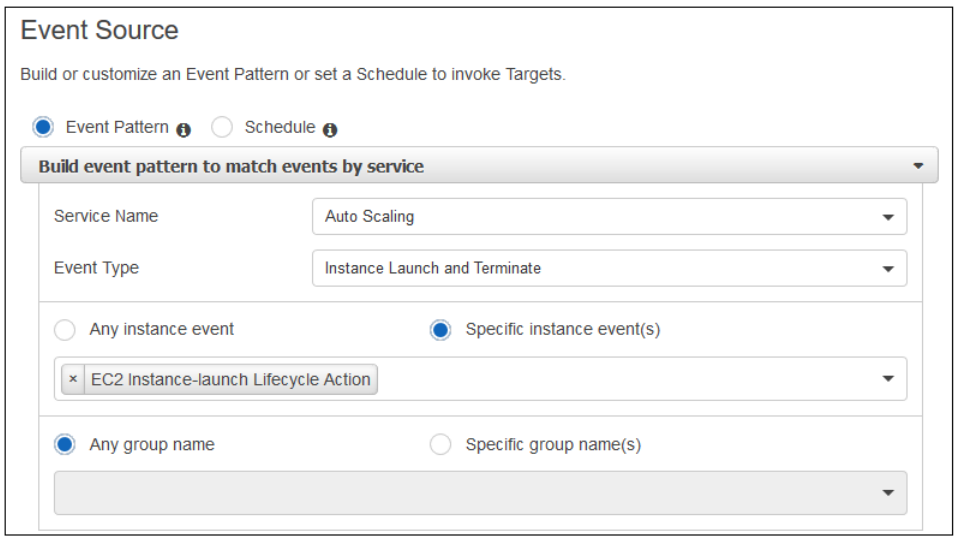
チュートリアル: CloudWatch イベント を使用して AWS Systems Manager 実行コマンドにイベントを中継する

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

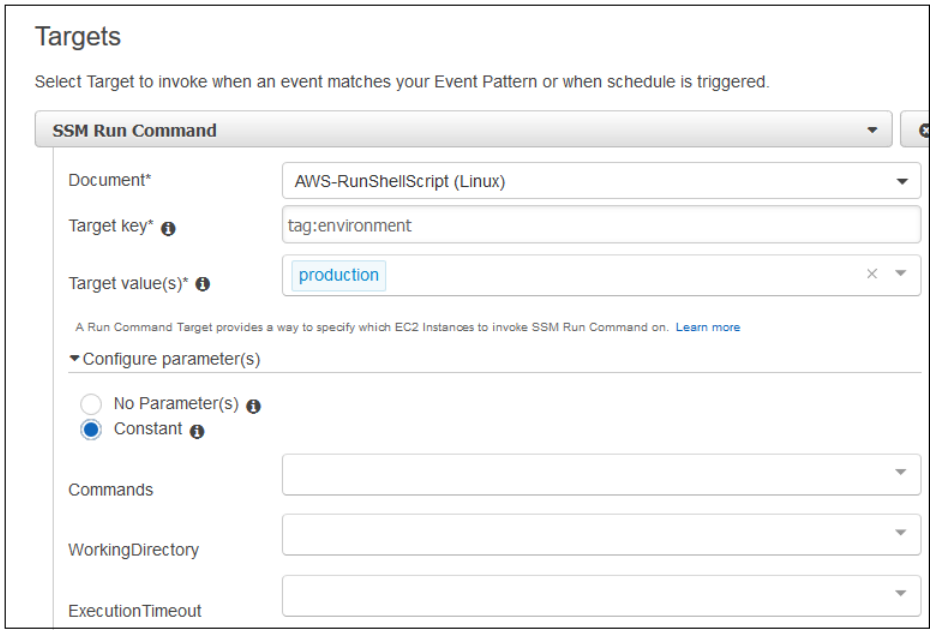
特定のイベントが発生したときに Amazon CloudWatch Events を使用して AWS Systems Manager 実行コマンドを呼び出し、Amazon EC2 インスタンスでアクションを実行できます。このチュートリアルでは、実行コマンドを設定してシェルコマンドを実行し、Amazon EC2 Auto Scaling グループで起動される新しいインスタンスをそれぞれ設定します。このチュートリアルでは、すでに `environment` をキーに、`production` を値として Amazon EC2 Auto Scaling グループにタグを割り当てているものと仮定しています。

CloudWatch イベント ルールを作成するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Events]、[Create rule] の順に選択します。
3. [Event source] で、以下の操作を実行します。
   a. [Build event pattern to match events by service] で、[Event Pattern] を選択します。
   b. [Service Name] で、[Auto Scaling] を選択します[Event Type] で、[Instance Launch and Terminate] を選択します。
   c. [Specific instance event(s)]、[EC2 Instance-launch Lifecycle Action] を選択します。
   d. デフォルトでは、このルールはリージョン内のすべての Amazon EC2 Auto Scaling グループと一致します。ルールを特定のグループに一致させるには、[Specific group name(s)] を選択して 1 つ以上のグループを選択します。



4. [Targets] で、[Add Target]、[SSM Run Command] を選択します。
5. [ドキュメント] で、[AWS-RunShellScript (Linux)] を選択します。Linux インスタンスと Windows インスタンスの両方を扱う他の多くのドキュメントのオプションがあります。[ターゲットキー] に「`tag:environment`」と入力します。[ターゲット値] で、「`production`」と入力し、[追加] を選択します。
6. [Configure parameter(s)] で、[Constant] を選択します。
7. [Commands] で、シェルコマンドを入力し、[Add] を選択します。インスタンスの起動時にすべてのコマンドを実行するには、このステップを繰り返します。
8. 必要に応じて、[WorkingDirectory] および [ExecutionTimeout] に適切な情報を入力します。
9. CloudWatch イベント は、以下のイベントの実行に必要な IAM ロールを作成できます。
   - 自動的に IAM ロールを作成するには、[この特定のリソースに対して新しいロールを作成する] を選択します。
   - 以前に作成した IAM ロールを使用するには、[既存のルールの使用] を選択します。



10. [Configure details] を選択します。[Rule definition] で、ルールの名前と説明を入力します。
11. [Create rule] を選択します。

チュートリアル: CloudWatch イベント を使用して Amazon EC2 インスタンスの状態をログに記録する

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features.

> Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

Amazon EC2 インスタンスの状態の変化をログに記録する AWS Lambda 関数を作成できます。状態が遷移すると実行されるルールや、関心のある 1 つ以上の状態が遷移すると実行されるルールを作成することができます。このチュートリアルでは、新しいインスタンスが起動されるたびにログに記録します。

ステップ 1: AWS Lambda 関数を作成する

状態変更イベントのログを記録する Lambda 関数を作成します。ルールを作成するときに、この関数を指定します。

Lambda 関数を作成するには

1. AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. Lambda を初めて使用する場合は、ウェルカムページを参照してください。[Get Started Now] を選択します。そうでない場合、[Lambda 関数の作成] を選択します。
3. [設計図の選択] ページで、フィルターに `hello`を入力し、[hello-world] 設計図を選択します。
4. [Configure triggers] ページで、[Next] を選択します。
5. [: Configure function] ページで、以下の作業を行います。
   a. Lambda 関数の名前と説明を入力します。たとえば、関数名を「LogEC2InstanceStateChange」とします。
   b. Lambda 関数のサンプルコードを編集します。例:

```
'use strict';

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

   c. [Role] で、[Choose an existing role] を選択します。[既存のロール] で基本的な実行ロールを選択します。それ以外の場合は、新しい基本的な実行ロールを作成します。
   d. [次へ] を選択します。
6. [Review] ページで、[Create function] を選択します。

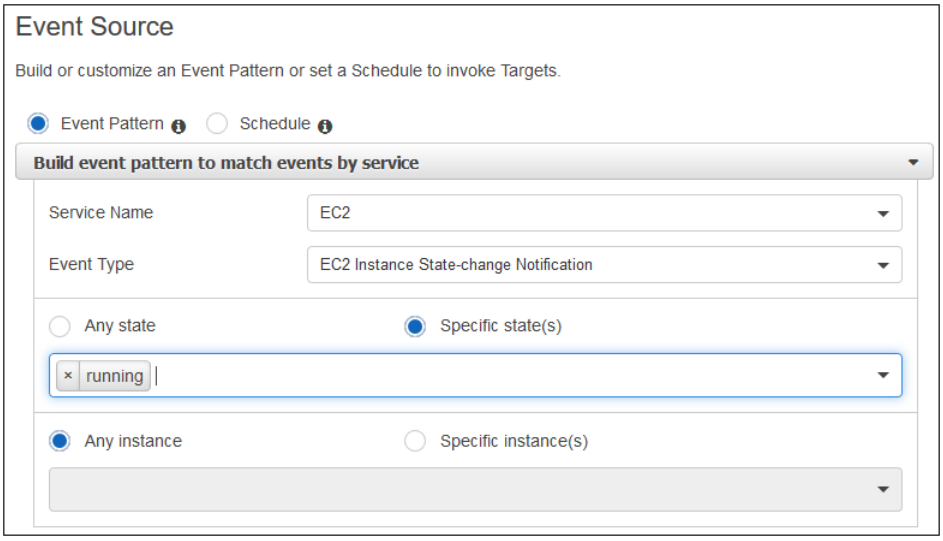
ステップ 2: ルールを作成する

Amazon EC2 インスタンスを起動するたびに Lambda 関数を実行するルールを作成します。

CloudWatch イベント ルールを作成するには:

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Events]、[Create rule] の順に選択します。
3. [Event source] で、以下の操作を実行します。
   a. [Event Pattern] を選択します。
   b. [Build event pattern to match events by service] を選択します。
   c. [EC2]、[EC2 インスタンスの状態変更通知] の順に選択します。
   d. [特定の状態] を選択し、[実行中] を選択します。

e. デフォルトでは、ルールはリージョン内のすべてのインスタンスに一致します。ルールを特定のインスタンスに一致させるには、[特定のインスタンス] を選択し、1 つ以上のインスタンスを選択します。



4. [ターゲット] で、[ターゲットの追加]、[Lambda 関数] の順に選択します。
5. [関数] で、作成した Lambda 関数を選択します。
6. [Configure details] を選択します。
7. [Rule definition] で、ルールの名前と説明を入力します。
8. [Create rule] を選択します。

ステップ 3 : ルールをテストする

ルールをテストするには、Amazon EC2 インスタンスを起動します。インスタンスの起動と初期化の数分後に、Lambda 関数が呼び出されたことが確認できます。

インスタンスを起動してルールをテストするには

1. https://console.aws.amazon.com/ec2/ で Amazon EC2 コンソールを開きます。
2. インスタンスを起動します。詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「インスタンスの起動」を参照してください。
3. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
4. ナビゲーションペインで、[Events]、[Rules] を選択し、作成したルールの名前を選択して、[Show metrics for the rule] を選択します。
5. Lambda 関数からの出力を表示するには、以下の操作を実行します。
   a. ナビゲーションペインで [Logs] を選択します。
   b. Lambda 関数 (/aws/lambda/function-name) のロググループの名前を選択します。
   c. 起動したインスタンスの関数によって提供されるデータを表示するログのストリーム名を選択します。
6. (オプション) 終了したら、Amazon EC2 コンソールを開き、起動したインスタンスを停止または削除できます。詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「インスタンスの終了」を参照してください。

チュートリアル: CloudWatch イベント を使用して Auto Scaling グループの状態を口グに記録する

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

Auto Scaling グループが Amazon EC2 インスタンスを起動または終了するたびにイベントを口グに記録し、その起動または終了イベントが成功したかどうかを口グに記録する AWS Lambda 関数を実行できます。

Amazon EC2 Auto Scaling イベントを使用した他の CloudWatch イベント シナリオの詳細については、『Amazon EC2 Auto Scaling ユーザーガイド』の「Auto Scaling グループスケーリング時の CloudWatch イベント の取得」を参照してください。

ステップ 1: AWS Lambda 関数を作成する

Lambda 関数を作成して、Auto Scaling グループのスケールアウトおよびスケールインイベントの口グを記録します。ルールを作成するときに、この関数を指定します。

Lambda 関数を作成するには

1. AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. Lambda を初めて使用する場合は、ウェルカムページを参照してください。[Get Started Now] を選択します。そうでない場合、[Lambda 関数の作成] を選択します。
3. [設計図の選択] ページで、フィルターに `hello`を入力し、[hello-world] 設計図を選択します。
4. [Configure triggers] ページで、[Next] を選択します。
5. [: Configure function] ページで、以下の作業を行います。
   a. Lambda 関数の名前と説明を入力します。たとえば、関数名を「LogAutoScalingEvent」とします。
   b. Lambda 関数のサンプルコードを編集します。例:

```
'use strict';

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

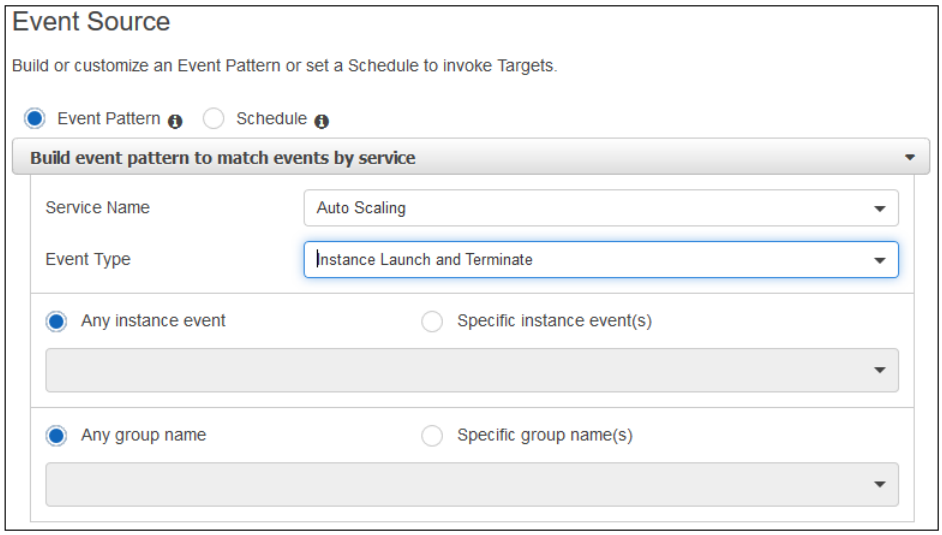
   c. [Role] で、[Choose an existing role] を選択します。[既存の口ール] で基本的な実行口ールを選択します。それ以外の場合は、新しい基本的な実行口ールを作成します。
   d. [次へ] を選択します。
6. [Create function] を選択します。

ステップ 2: ルールを作成する

Auto Scaling グループがインスタンスを起動または終了するたびに、Lambda 関数を実行するルールを作成します。

ルールを作成するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Events]、[Create rule] の順に選択します。
3. [Event source] で、以下の操作を実行します。
   a. [Event Pattern] を選択します。
   b. [Build event pattern to match events by service] を選択します。
   c. [Auto Scaling]、[Instance Launch and Terminate (インスタンスの起動と削除)] を選択します。
   d. すべての成功と失敗のインスタンスの起動と終了イベントを収集するには、[任意のインスタンスイベント] を選択します。



4. デフォルトでは、このルールはリージョン内のすべての Auto Scaling グループと一致します。ルールを特定の Auto Scaling グループに一致させるには、[特定のグループ名] を選択して 1 つ以上の Auto Scaling グループを選択します。
5. [ターゲット] で、[ターゲットの追加]、[Lambda 関数] の順に選択します。
6. [関数] で、作成した Lambda 関数を選択します。
7. [Configure details] を選択します。
8. [Rule definition] で、ルールの名前と説明を入力します。たとえば、ルールを「Auto Scaling グループがスケールアウトまたはインするたびにログを記録する」としてルールを記述します。
9. [Create rule] を選択します。

ステップ 3 : ルールをテストする

ルールをテストするには、インスタンスを起動するように Auto Scaling グループを手動でスケーリングします。スケールアウトイベント発生の数分後に、Lambda 関数が呼び出されたことを確認します。

Auto Scaling グループを使用してルールをテストするには

1. Auto Scaling グループのサイズを増やすには、以下の操作を実行します。
   a. https://console.aws.amazon.com/ec2/ で Amazon EC2 コンソールを開きます。
   b. ナビゲーションペインで、[Auto Scaling]、[Auto Scaling Groups] を選択します。
   c. Auto Scaling グループのチェックボックスを選択します。
   d. [Details] タブで、[Edit] を選択します。[Desired] で、希望する容量を 1 つ増やします。たとえば、現在の値が 2 の場合は 3 と入力します。希望する容量はグループの最大サイズと同じかそれ以下である必要があります。[Desired] の新しい値が、[Max] よりも大きい場合、[Max] を更新する必要があります。完了したら、[Save] を選択します。
2. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
3. ナビゲーションペインで、[Events]、[Rules] を選択し、作成したルールの名前を選択して、[Show metrics for the rule] を選択します。
4. Lambda 関数からの出力を表示するには、以下の操作を実行します。
   a. ナビゲーションペインで [Logs] を選択します。
   b. Lambda 関数 (/aws/lambda/function-name) のロググループの名前を選択します。
   c. 起動したインスタンスの関数によって提供されるデータを表示するログのストリーム名を選択します。
5. (オプション) 終了すると、Auto Scaling グループが以前のサイズに戻るように、必要な容量を 1 減らすことができます。

チュートリアル: CloudWatch イベント を使用して Amazon S3 オブジェクトレベル操作のログを記録する

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

S3 バケットにオブジェクトレベルの API 操作のログを記録することができます。Amazon CloudWatch Events がこれらのイベントと一致する前に、AWS CloudTrail を使用してこれらのイベントを受信するように設定された証跡を設定する必要があります。

ステップ 1: AWS CloudTrail 証跡を設定する

S3 バケットのデータイベントを AWS CloudTrail および CloudWatch イベント に記録するには、証跡を作成します。証跡は、アカウントでの API コールと関連イベントをキャプチャし、指定した S3 バケットにログファイルを提供します。既存の証跡を更新するか、新しい証跡を作成できます。

証跡を作成するには

1. https://console.aws.amazon.com/cloudtrail/ にある CloudTrail コンソールを開きます。

2. ナビゲーションペインで、[Trails (証跡)]、[Add new trail (新しい証跡の追加)] を選択します。
3. [Trail name] に、証跡の名前を入力します。
4. [Data events] に、バケット名とプレフィックスを入力します (オプション)。証跡ごとに、最大 250 個の Amazon S3 オブジェクトを追加できます。

   - バケットのすべての Amazon S3 オブジェクトのデータイベントを記録するには、S3 バケットと空のプレフィックスを指定します。その バケットのオブジェクトでイベントが発生すると、証跡がイベントを処理して記録します。
   - 特定の Amazon S3 オブジェクトのデータイベントを記録するには、[S3 バケットの追加] を選択し、S3 バケットおよびオプションとしてオブジェクトのプレフィックスを指定します。そのバケットのオブジェクトでイベントが発生し、オブジェクトが指定したプレフィックスで始まっていると、証跡がイベントを処理して記録します。
5. 各リソースについて、ログ記録の対象を [読み取り] イベントにするか、[書き込み] イベントにするか、または両方のタイプのイベントにするかを指定します。
6. [Storage location] で、ログファイルストレージに指定する S3 バケットを作成するか既存のバケットを選択します。
7. [Create] を選択します。

詳細については、『AWS CloudTrail User Guide』の「データイベント」を参照してください。

ステップ 2: AWS Lambda 関数を作成する

S3 バケットのデータイベントのログを記録する Lambda 関数を作成します。ルールを作成するときに、この関数を指定します。

Lambda 関数を作成するには

1. AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. Lambda を初めて使用する場合は、ウェルカムページを参照してください。[Create a function] を選択します。それ以外の場合は、[関数の作成] を選択します。
3. [Author from scratch] を選択します。
4. [一から作成] で、次の操作を行います。

   a. Lambda 関数の名前を入力します。たとえば、関数名を「LogS3DataEvents」とします。
   b. [Role] で、[Create a custom role] を選択します。

      新しいウィンドウが開きます。必要に応じて [ロール名] を変更し、[許可] を選択します。
   c. Lambda コンソールに戻り、[関数の作成] を選択します。
5. Lambda 関数のコードを次のように編集し、[保存] を選択します。

```
'use strict';

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

ステップ 3: ルールを作成する

Amazon S3 データイベントに応答して Lambda 関数を実行するルールを作成します。

ルールを作成するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Rules]、[Create rule] の順に選択します。
3. [Event source] で、以下の操作を実行します。
   a. [Event Pattern] を選択します。
   b. [Build event pattern to match events by service] を選択します。
   c. [Simple Storage Service (S3)] を選択し、[オブジェクトレベルのオペレーション] を選択します。
   d. [特定のオペレーション]、[PutObject] の順に選択します。
   e. デフォルトでは、このルールはリージョン内のすべてのバケットのデータイベントと一致します。特定のバケットのデータイベントに一致させるには、[Specify bucket(s) by name] で指定し、1 つ以上のバケットを指定します。
4. [ターゲット] で、[ターゲットの追加]、[Lambda 関数] の順に選択します。
5. [関数] で、作成した Lambda 関数を選択します。
6. [Configure details] を選択します。
7. [Rule definition] で、ルールの名前と説明を入力します。
8. [Create rule] を選択します。

ステップ 4 : ルールをテストする

ルールをテストするには、オブジェクトを S3 バケットに配置します。Lambda 関数が呼び出されたことを確認できます。

Lambda 関数のログを表示するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで [Logs] を選択します。
3. Lambda 関数 (/aws/lambda/function-name) のロググループの名前を選択します。
4. 起動したインスタンスの関数によって提供されるデータを表示するログのストリーム名を選択します。

また、証跡に指定した S3 バケット内の CloudTrail ログの内容を確認することもできます。詳細については、AWS CloudTrail User Guideの「CloudTrail ログファイルの取得と表示」を参照してください。

チュートリアル: イベントターゲットに渡されるものを Input Transformer を使用してカスタマイズする

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

CloudWatch イベント の入力変換機能を使用すると、イベントから取得したテキストを、ルールのターゲットに入力する前にカスタマイズできます。

イベントからの JSON パスを複数定義し、その出力をさまざまな変数に割り当てることができます。その後、こうした変数は、<*variable-name*> 形式の入力テンプレートで使用できます。文字 < および > は エスケープできません。

指定した変数と一致する JSON パスがイベントに存在しない場合、その変数は作成されず、出力には表示されません

このチュートリアルでは、インスタンスの状態変更イベントから Amazon EC2 インスタンスの instance-id と状態を抽出します。入力変換を使用して、そのデータを、Amazon SNS トピックに送信される読みやすいメッセージに入力します。ルールは、任意のインスタンスが任意の状態に変わった時点でトリガーされます。たとえば、このルールでは、次の Amazon EC2 インスタンスの状態変更通知イベントによって、"EC2 インスタンス i-1234567890abcdef0 の状態が "停止" に変更されました" という Amazon SNS メッセージが生成されます。

```
{
   "id":"7bf73129-1428-4cd3-a780-95db273d1602",
   "detail-type":"EC2 Instance State-change Notification",
   "source":"aws.ec2",
   "account":"123456789012",
   "time":"2015-11-11T21:29:54Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:ec2:us-east-1:123456789012:instance/ i-1234567890abcdef0"
   ],
   "detail":{
      "instance-id":" i-1234567890abcdef0",
      "state":"stopped"
   }
}
```

これを実現するには、*instance* 変数を、イベントからの `$.detail.instance-id` JSON パスにマッピングし、*state* 変数を `$.detail.state` JSON パスにマッピングします。次に、入力テンプレートを "EC2 インスタンス <instance> の状態が <state> に変更されました" と設定します。

ルールを作成する

入力変換を使用して、ターゲットに送信されるインスタンスの状態変更情報をカスタマイズするには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Events]、[Create rule] の順に選択します。
3. [Event source] で、以下の操作を実行します。
   a. [Event Pattern] を選択します。
   b. [Build event pattern to match events by service] を選択します。
   c. [EC2]、[EC2 インスタンスの状態変更通知] の順に選択します。
   d. [任意の状態]、[任意のインスタンス] を選択します。
4. [ターゲット] で、[Add target (ターゲットの追加)]、[SNS トピック] を選択します。
5. [トピック] で、Amazon EC2 インスタンスの状態が変更されたときに通知を受け取る Amazon SNS トピックを選択します。
6. [Configure input]、[Input Transformer] の順に選択します。
7. 次のボックスに、「{"state" : "$.detail.state", "instance" : "$.detail.instance-id"}」と入力します

8. 次のボックスに、「EC2 インスタンス <instance> の状態が <state>." に変更されました」と入力します
9. [Configure details] を選択します。
10. ルールの名前と説明を入力し、[Create rule] を選択します。

チュートリアル: CloudWatch イベント 使用して、AWS API コールのログを記録する

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

各 AWS API コールをログに記録する AWS Lambda 関数を使用できます。たとえば、Amazon EC2 内のオペレーションをすべてログに記録するルールを作成したり、特定の API 呼び出しのみをログに記録するようにこのルールを制限したりできます。このチュートリアルでは、Amazon EC2 インスタンスが停止されるたびにログに記録します。

前提条件

これらのイベントに一致する前に、AWS CloudTrail を使用して証跡を設定する必要があります。証跡がない場合は、以下の操作を実行します。

証跡を作成するには

1. https://console.aws.amazon.com/cloudtrail/ にある CloudTrail コンソールを開きます。
2. [Trails (証跡)]、[Create trail (証跡の作成)] の順に選択します。
3. [Trail name] に、証跡の名前を入力します。
4. [Storage location (ストレージの場所)] の [Create a new S3 bucket (新しい S3 バケットの作成)] で、CloudTrail によるログの配信先となる新しいバケットの名前を入力します。
5. [Create] を選択します。

ステップ 1: AWS Lambda 関数を作成する

Lambda 関数を作成して、API 呼び出しイベントのログを記録します。ルールを作成するときに、この関数を指定します。

Lambda 関数を作成するには

1. AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. Lambda を初めて使用する場合は、ウェルカムページを参照してください。[Get Started Now] を選択します。そうでない場合、[Lambda 関数の作成] を選択します。
3. [設計図の選択] ページで、フィルターに `hello`を入力し、[hello-world] 設計図を選択します。
4. [Configure triggers] ページで、[Next] を選択します。
5. [: Configure function] ページで、以下の作業を行います。

a. Lambda 関数の名前と説明を入力します。(たとえば、関数名を「LogEC2StopInstance」とします)。
b. Lambda 関数のサンプルコードを編集します。例:

```
'use strict';

exports.handler = (event, context, callback) => {
    console.log('LogEC2StopInstance');
    console.log('Received event:', JSON.stringify(event, null, 2));
    callback(null, 'Finished');
};
```

c. [Role] で、[Choose an existing role] を選択します。[既存のロール] で基本的な実行ロールを選択します。それ以外の場合は、新しい基本的な実行ロールを作成します。
d. [次へ] を選択します。

6. [Review] ページで、[Create function] を選択します。

ステップ 2: ルールを作成する

Amazon EC2 インスタンスを停止するたびに Lambda 関数を実行するルールを作成します。

ルールを作成するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Events]、[Create rule] の順に選択します。
3. [Event source] で、以下の操作を実行します。
   a. [Event Pattern] を選択します。
   b. [Build event pattern to match events by service] を選択します。
   c. [EC2]、[AWS API Call via CloudTrail (CloudTrail 経由の AWS API コール)] の順に選択します。
   d. [Specific operation(s)] を選択し、下のボックスに「`StopInstances`」と入力します。
4. [ターゲット] で、[ターゲットの追加]、[Lambda 関数] の順に選択します。
5. [関数] で、作成した Lambda 関数を選択します。
6. [Configure details] を選択します。
7. [Rule definition] で、ルールの名前と説明を入力します。
8. [Create rule] を選択します。

ステップ 3 : ルールをテストする

Amazon EC2 コンソールを使用して Amazon EC2 インスタンスを停止することで、ルールをテストできます。インスタンスが停止されるまで数分間待ったら、CloudWatch コンソールで AWS Lambda メトリクスを調べて、関数が呼び出されたことを確認します。

インスタンスを停止してルールをテストするには

1. https://console.aws.amazon.com/ec2/ で Amazon EC2 コンソールを開きます。
2. インスタンスを起動します。詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「インスタンスの起動」を参照してください。
3. インスタンスを停止します。詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「インスタンスの停止と起動」を参照してください。

4. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
5. ナビゲーションペインで、[Events] を選択し、作成したルールの名前を選択して、[Show metrics for the rule] を選択します。
6. Lambda 関数からの出力を表示するには、以下の操作を実行します。

   a. ナビゲーションペインで [Logs] を選択します。
   b. Lambda 関数 (/aws/lambda/function-name) のロググループの名前を選択します。
   c. 停止したインスタンスの関数によって提供されるデータを表示するログのストリーム名を選択します。
7. (オプション) 終了したら、停止したインスタンスを終了できます。詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「インスタンスの終了」を参照してください。

チュートリアル: CloudWatch イベント を使用した、自動化された Amazon EBS スナップショットのスケジュール

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

スケジュールに従って CloudWatch イベント ルールを実行できます。このチュートリアルでは、スケジュールに基づいて既存の Amazon Elastic Block Store (Amazon EBS) ボリュームの自動化されたスナップショットを作成します。スナップショットは、一定の速度 (数分ごと) で作成することも、特定の時間帯を指定して作成することもできます。

> **Important**
>
> 組み込みターゲットでのルール作成は、AWS マネジメントコンソール でのみサポートされています。

ステップ 1: ルールを作成する

スケジュールに従ってスナップショットを作成するルールを作成します。レート式または cron 式を使用してスケジュールを指定できます。詳細については、「ルールのスケジュール式 (p. 34)」を参照してください。

ルールを作成するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Events]、[Create rule] の順に選択します。
3. [Event Source] で、以下の操作を実行します。

   a. [Schedule] を選択します。
   b. [Fixed rate of] を選択し、スケジュール間隔 (たとえば 5 分) を指定します。また、[Cron expression] を選択し、cron 式を指定します (たとえば、現在の時刻から月曜日から金曜日まで 15 分ごとに指定します)。

4. [ターゲット] で、[ターゲットの追加] を選択し、[EC2 CreateSnapshot API 呼び出し] を選択します。EC2 CreateSnapshot API コールを見つけるには、必要に応じてターゲット候補のリストを上にスクロールします。
5. [ボリューム ID] には、ターゲットの Amazon EBS ボリュームのボリューム ID を入力します。
6. [Create a new role for this specific resource] を選択します。新しいロールはターゲットに、お客様の代わりにリソースにアクセスする権限を与えます。
7. [Configure details] を選択します。
8. [Rule definition] で、ルールの名前と説明を入力します。
9. [Create rule] を選択します。

ステップ 2 : ルールをテストする

最初のスナップショットを作成した後、そのスナップショットを表示することでルールを検証できます。

ルールをテストするには

1. https://console.aws.amazon.com/ec2/ で Amazon EC2 コンソールを開きます。
2. ナビゲーションペインで [Elastic Block Store]、[Snapshots] の順に選択します。
3. 最初のスナップショットがリストに表示されることを確認します。
4. (オプション) 終了すると、ルールを無効にして、追加のスナップショットが作成されないようにできます。

    a. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
    b. ナビゲーションペインで、[Events]、[Rules] を選択します。
    c. ルールを選択し、[アクション]、[無効化] を選択します。
    d. 確認を求められたら、[Disable] を選択します。

チュートリアル: CloudWatch イベント を使用して AWS Lambda 関数をスケジュールする

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

スケジュールに基づいて AWS Lambda 関数を実行するルールを設定できます。このチュートリアルでは、AWS マネジメントコンソール または AWS CLI を使用してルールを作成する方法について説明します。AWS CLI を使用したいがインストールしていない場合は、AWS Command Line Interface ユーザーガイド を参照してください。

CloudWatch イベント は、スケジュール式で第 2 レベルの精度を提供しません。Cron 式を使用した最小の解決は分です。CloudWatch イベント とターゲットサービスが持つ分散性の特質により、スケジュールされたルールがトリガーされてから、ターゲットサービスがターゲットリソースの実行を優先するまでの遅延は、数秒となる可能性があります。スケジュールされたルールは、その分のうちにトリガーされますが、正確に 0 秒にトリガーされません。

ステップ 1: AWS Lambda 関数を作成する

スケジュールされたイベントのログを記録するラムダ関数を作成します。ルールを作成するときに、この関数を指定します。

Lambda 関数を作成するには

1. AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。
2. Lambda を初めて使用する場合は、ウェルカムページを参照してください。[Get Started Now] を選択します。そうでない場合、[Lambda 関数の作成] を選択します。
3. [設計図の選択] ページで、フィルターに `hello`を入力し、[hello-world] 設計図を選択します。
4. [Configure triggers] ページで、[Next] を選択します。
5. [: Configure function] ページで、以下の作業を行います。
   a. Lambda 関数の名前と説明を入力します。たとえば、関数名を「LogScheduledEvent」とします。
   b. Lambda 関数のサンプルコードを編集します。例:

```
'use strict';

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

   c. [Role] で、[Choose an existing role] を選択します。[既存のロール] で基本的な実行ロールを選択します。それ以外の場合は、新しい基本的な実行ロールを作成します。
   d. [次へ] を選択します。
6. [Review] ページで、[Create function] を選択します。

ステップ 2: ルールを作成する

Lambda 関数をスケジュールに従って実行するルールを作成します。

コンソールを使用してルールを作成するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Events]、[Create rule] の順に選択します。
3. [Event Source] で、以下の操作を実行します。
   a. [Schedule] を選択します。
   b. [Fixed rate of] を選択し、スケジュール間隔 (たとえば 5 分) を指定します。
4. [ターゲット] で、[ターゲットの追加]、[Lambda 関数] の順に選択します。
5. [関数] で、作成した Lambda 関数を選択します。
6. [Configure details] を選択します。
7. [Rule definition] で、ルールの名前と説明を入力します。
8. [Create rule] を選択します。

必要に応じて、AWS CLI を使用してルールを作成できます。まず、Lambda 関数を呼び出すためのアクセス権限をルールに付与する必要があります。次にルールを作成し、Lambda 関数をターゲットとして追加できます。

AWS CLI を使用してルールを作成するには

1. 次の [put-rule] コマンドを使用して、スケジュールに従ってトリガーするルールを作成します。

```
aws events put-rule \
--name my-scheduled-rule \
--schedule-expression 'rate(5 minutes)'
```

このルールがトリガーされると、このルールのターゲットへの入力として機能するイベントを生成します。以下に示しているのは、イベントの例です。

```
{
    "version": "0",
    "id": "53dc4d37-cffa-4f76-80c9-8b7d4a4d2eaa",
    "detail-type": "Scheduled Event",
    "source": "aws.events",
    "account": "123456789012",
    "time": "2015-10-08T16:53:06Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:events:us-east-1:123456789012:rule/my-scheduled-rule"
    ],
    "detail": {}
}
```

2. 次の add-permission コマンドを使用して、信頼する CloudWatch イベント サービスプリンシパル (events.amazonaws.com) や指定された Amazon リソースネーム (ARN) を持つルールに対するスコープ権限 :

```
aws lambda add-permission \
--function-name LogScheduledEvent \
--statement-id my-scheduled-event \
--action 'lambda:InvokeFunction' \
--principal events.amazonaws.com \
--source-arn arn:aws:events:us-east-1:123456789012:rule/my-scheduled-rule
```

3. 次の put-targets コマンドを使用して、このルールに作成した Lambda 関数を 5 分ごとに実行するように追加します。

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

次の内容で、targets.json ファイルを作成します。

```
[
  {
    "Id": "1",
    "Arn": "arn:aws:lambda:us-east-1:123456789012:function:LogScheduledEvent"
  }
]
```

ステップ 3: ルールの確認

ステップ 2 を完了してから少なくとも 5 分後に、Lambda 関数が呼び出されたことを確認できます。

ルールをテストするには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。

2. ナビゲーションペインで、[Events]、[Rules] を選択し、作成したルールの名前を選択して、[Show metrics for the rule] を選択します。
3. Lambda 関数からの出力を表示するには、以下の操作を実行します。

   a. ナビゲーションペインで [Logs] を選択します。
   b. Lambda 関数 (/aws/lambda/function-name) のロググループの名前を選択します。
   c. 起動したインスタンスの関数によって提供されるデータを表示するログのストリーム名を選択します。
4. (オプション) 終了したら、ルールを無効にすることができます。

   a. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
   b. ナビゲーションペインで、[Events]、[Rules] を選択します。
   c. ルールを選択し、[アクション]、[無効化] を選択します。
   d. 確認を求められたら、[Disable] を選択します。

チュートリアル: AWS Systems Manager Automation を CloudWatch イベント ターゲットとして設定する

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

CloudWatch イベント を使用して、定期なスケジュールで、または指定されたイベントが検出されたときに、AWS Systems Manager Automation を呼び出すことができます。このチュートリアルでは、特定のイベントに基づいて Systems Manager Automation を呼び出すことを前提としています。

CloudWatch イベント ルールを作成するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Events]、[Create rule] の順に選択します。
3. [Event source] で、以下の操作を実行します。

   a. [Event Pattern] および [Build event pattern to match events by service] を選択します。
   b. [Service Name] と [Event Type] で、トリガーとして使用するサービスとイベントタイプを選択します。

      選択したサービスとイベントタイプに応じて、[Event Source] で追加のオプションの指定が必要になる場合があります。
4. [Targets] で、[Add Target]、[SSM Automation] を選択します。
5. [ドキュメント] で、ターゲットがトリガーされたときに実行する Systems Manager ドキュメントを選択します。
6. (オプション) ドキュメントの特定のバージョンを指定するには、[Configure document version] を選択します。
7. [Configure parameter(s)] を選択し、[No Parameter(s)] または [Constant] を選択します。

[Constant] を選択した場合は、ドキュメントの実行に渡す定数を指定します。

8. CloudWatch イベント は、以下のイベントの実行に必要な IAM ロールを作成できます。
   - 自動的に IAM ロールを作成するには、[この特定のリソースに対して新しいロールを作成する] を選択します。
   - 以前に作成した IAM ロールを使用するには、[既存のルールの使用] を選択します。
9. [Configure details] を選択します。[Rule definition] で、ルールの名前と説明を入力します。
10. [Create rule] を選択します。

チュートリアル: CloudWatch イベント を使用して Amazon Kinesis ストリームにイベントを中継する

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

このシナリオでは、Amazon Kinesis のストリームに CloudWatch イベント の AWS API コールイベントを中継します。

前提条件

AWS CLI をインストールします。詳細については、『AWS Command Line Interface ユーザーガイド』を参照してください。

ステップ 1: Amazon Kinesis ストリームを作成する

ストリームを作成するには、以下の create-stream コマンドを使用します。

```
aws kinesis create-stream --stream-name test --shard-count 1
```

ストリームのステータスが ACTIVE の場合、ストリームは準備完了です。ストリームのステータスを確認するには、以下の describe-stream コマンドを使用します。

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

ステップ 2: ルールを作成する

たとえば、Amazon EC2 インスタンスを停止したときにイベントをストリームに送信するルールを作成します。

ルールを作成するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Events]、[Create rule] の順に選択します。
3. [Event source] で、以下の操作を実行します。

a. [Event Pattern] を選択します。
b. [Build event pattern to match events by service] を選択します。
c. [EC2]、[インスタンスの状態変更通知] の順に選択します。
d. [特定の状態] を選択し、[実行中] を選択します。

4. [ターゲット] で、[Add target (ターゲットの追加)] を追加し、[Kinesis ストリーム] を選択します。
5. [Stream] で、作成したストリームを選択します。
6. [Create a new role for this specific resource] を選択します。
7. [Configure details] を選択します。
8. [Rule definition] で、ルールの名前と説明を入力します。
9. [Create rule] を選択します。

ステップ 3 : ルールをテストする

ルールをテストするには、Amazon EC2 インスタンスを停止します。インスタンスが停止されるまで数分間待ったら、CloudWatch メトリクスを調べて、関数が呼び出されたことを確認します。

インスタンスを停止してルールをテストするには

1. https://console.aws.amazon.com/ec2/ で Amazon EC2 コンソールを開きます。
2. インスタンスを起動します。詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「インスタンスの起動」を参照してください。
3. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
4. ナビゲーションペインで、[Events]、[Rules] を選択し、作成したルールの名前を選択して、[Show metrics for the rule] を選択します。
5. (オプション) 終了したら、インスタンスを終了できます。詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「インスタンスの終了」を参照してください。

ステップ 4: イベントを中継されることを確認する

ストリームからレコードを取得して、イベントがリレーされたことを確認できます。

レコードを取得するには

1. 次の get-shard-iterator コマンドを使用して、Kinesis ストリームからの読み取りを開始します。

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

出力例を次に示します。

```
{
    "ShardIterator": "AAAAAAAAAAHSywljv0zEgPX4NyKdZ5wryMzP9yALs8NeKbUjp1IxtZs1Sp
+KEd9I6AJ9ZG4lNR1EMi+9Md/nHvtLyxpfhEzYvkTZ4D9DQVz/mBYWRO6OTZRKnW9gd
+efGN2aHFdkH1rJl4BL9Wyrk+ghYG22D2T1Da2EyNSH1+LAbK33gQweTJADBdyMwlo5r6PqcP2dzhg="
}
```

2. レコードを取得するには、次の get-records コマンドを使用します。シャードイテレータは、前のステップで取得したものです。

```
aws kinesis get-records --shard-
iterator AAAAAAAAAAHSywljv0zEgPX4NyKdZ5wryMzP9yALs8NeKbUjp1IxtZs1Sp+KEd9I6AJ9ZG4lNR1EMi
```

```
+9Md/nHvtLyxpfhEzYvkTZ4D9DQVz/mBYWRO6OTZRKnW9gd+efGN2aHFdkH1rJl4BL9Wyrk
+ghYG22D2T1Da2EyNSH1+LAbK33gQweTJADBdyMwlo5r6PqcP2dzhg=
```

コマンドが成功すると、指定シャードのストリームからレコードをリクエストします。0 以上のレコードを受け取ることができます。返されるレコードは、ストリーム内のすべてのレコードを表すとは限りません。ご希望のデータを受け取っていない場合は、`get-records` を継続して呼び出します。

Kinesis のレコードは Base64 エンコードされています。ただし、AWS CLI のストリームのサポートでは、base64 のデコードは用意されていません。base64 デコーダーを使用して、データを手動で復号する場合、それは JSON 形式でストリームに中継されたイベントであることがわかります。

チュートリアル: ファイルが Amazon S3 バケットにアップロードされたときに Amazon ECS タスクを実行する

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

特定の AWS イベントが発生したときに Amazon ECS タスクを実行するには、CloudWatch イベント を使用できます。このチュートリアルでは、Amazon S3 PUT オペレーションを使用して、ファイルが特定の Amazon S3 バケットにアップロードされたときに Amazon ECS タスクを実行するように CloudWatch イベント ルールを設定します。

このチュートリアルでは、「Amazon ECS」でタスクを作成済みであることを前提としています。

PUT オペレーションを使用して S3 バケットにファイルがアップロードされるたびに Amazon ECS タスクを実行するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Events]、[Create rule] の順に選択します。
3. [Event source] で、以下の操作を実行します。
   a. [Event Pattern] を選択します。
   b. [Build event pattern to match events by service] を選択します。
   c. [サービス名] で、[Simple Storage Service (S3)] を選択します。
   d. [イベントタイプ] で、[オブジェクトレベルのオペレーション] を選択します。
   e. [特定のオペレーション]、[Put Object] の順に選択します。
   f. [特定のバケット (名前別)] を選択し、バケットの名前を入力します。
4. [ターゲット] で、以下の作業を行います。
   a. [Add target (ターゲットの追加)]、[ECS task (ECS タスク)] の順に選択します。
   b. [クラスター] および [タスク定義] で、作成したリソースを選択します。
   c. [起動タイプ] で、[FARGATE] または [EC2] を選択します。[FARGATE] は、AWS Fargate がサポートされているリージョンのみで表示されます。

d. (オプション) [タスクグループ] の値を指定します。[起動タイプ] が [FARGATE] の場合、オプションで [プラットフォームバージョン] を指定します。1.1.0 など、プラットフォームバージョンの数値部分のみを指定します。

e. (オプション) タスク定義のリビジョンとタスク数を指定します。タスク定義のリビジョンを指定しない場合、最新のものが使用されます。

f. タスク定義が awsvpc ネットワークモードを使用する場合は、サブネットおよびセキュリティグループを指定する必要があります。すべてのサブネットとセキュリティグループは、同じ VPC 内にある必要があります。

   複数のセキュリティグループまたはサブネットを指定する場合、スペースではなくカンマで区切ります。

   [サブネット] で、以下の例のように、各サブネットの `subnet-id` の値全体を指定します。

   ```
   subnet-123abcd,subnet-789abcd
   ```

g. パブリック IP アドレスの自動割り当てを許可するかどうか選択します。

h. CloudWatch イベント は、以下のタスクの実行に必要な IAM ロールを作成できます。

   - 自動的に IAM ロールを作成するには、[この特定のリソースに対して新しいロールを作成する] を選択します。
   - 以前に作成した IAM ロールを使用するには、[既存のルールの使用] を選択します。これは、ビルドを呼び出すのに十分なアクセス権限をすでに持つロールでなければなりません。CloudWatch イベント は、選択したロールに追加のアクセス権限を付与しません。

5. [Configure details] を選択します。
6. [Rule definition] で、ルールの名前と説明を入力します。
7. [Create rule] を選択します。

チュートリアル: CodeBuild を使用した、自動化されたビルドのスケジュール

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

このチュートリアルの例では、平日の毎晩午後 8 時 (GMT) にビルドを実行するように CodeBuild をスケジュールします。このスケジュールされたビルドで使用する定数も CodeBuild に渡します。

毎晩午後 8 時に CodeBuild プロジェクトのビルドをスケジュールするルールを作成するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Events]、[Create rule] の順に選択します。
3. [Event Source] で、以下の操作を実行します。

   a. [Schedule] を選択します。

   b. [Cron 式] を選択し、式として次の式を指定します: 0 20 ? * MON-FRI *cron 式の詳細については、「ルールのスケジュール式 (p. 34)」を参照してください。

4. [ターゲット] で、[Add target (ターゲットの選択)] を選択し、[CodeBuild プロジェクト] を選択します。
5. [Project ARN] で、ビルドプロジェクトの ARN を入力します。
6. このチュートリアルでは、パラメータを CodeBuild に渡してデフォルトを上書きするオプションステップを追加します。これは、ターゲットとして CodeBuild を設定するときは必要ありません。パラメータを渡すには、[Configure input]、[Constant (JSON text)] を選択します。

   [Constant (JSON text)] のボックスに、{ "timeoutInMinutesOverride": 30 } と入力して、それらのスケジュールされたビルドのタイムアウト上書きを 30 分に設定します。

   渡すことができるパラメータの詳細については、「StartBuild」を参照してください。このフィールドでは、`projectName` パラメータを渡すことはできません。代わりに、[Project ARN] で ARN を使用してプロジェクトを指定します。
7. CloudWatch イベント は、以下のビルドプロジェクトの実行に必要な IAM ロールを作成できます。

   - 自動的に IAM ロールを作成するには、[この特定のリソースに対して新しいロールを作成する] を選択します。
   - 以前に作成した IAM ロールを使用するには、[既存のルールの使用] を選択します。これは、ビルドを呼び出すのに十分なアクセス権限をすでに持つロールでなければなりません。CloudWatch イベント は、選択したロールに追加のアクセス権限を付与しません。
8. [Configure details] を選択します。
9. [Rule definition] で、ルールの名前と説明を入力します。
10. [Create rule] を選択します。

チュートリアル: Amazon EC2 インスタンスの状態の変化をログに記録する

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

このチュートリアルの例では、Amazon EC2 での状態変化通知を CloudWatch Logs に記録させるルールを作成します。

Amazon EC2 の状態変化通知を CloudWatch Logs に記録するルールを作成するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[イベント]、[ルールの作成] の順に選択します。
3. [Event Source] で、以下の操作を実行します。

   a. [Event Pattern] を選択します。
   b. [サービス名] には [EC2] を選びます。
   c. [イベントタイプ] に、[インスタンスの状態変更通知] を選択します。
4. [Targets] で、[Add target] を選択します。サービスのリストで、[CloudWatch log group (CloudWatch ロググループ)] を選択します。
5. [ロググループ] に、状態変化通知を受け取るロググループの名前を入力します。

6. [Configure details] を選択します。
7. [Rule definition (ルール定義)] に、ルールの名前と説明を入力します。
8. [Create rule] を選択します。

ルールのスケジュール式

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

cron または rate 式を使用して、CloudWatch イベント で自動化されたスケジュールに基づいて自己トリガーするルールを作成できます。すべてのスケジュールされたイベントは UTC タイムゾーンを使用し、スケジュールの最小精度は 1 分です。

CloudWatch イベント は、cron 式や rate 式をサポートしています。rate 式は定義が簡単ですが、cron 式がサポートするきめ細かいスケジュール制御を提供しません。たとえば、cron 式を使用すると、毎週または毎月の特定の日の指定した時間にトリガーされるルールを定義できます。それに対して、rate 式では、1 時間に 1 回または 1 日 1 回など、通常のレートでルールを起動します。

> **Note**
>
> CloudWatch イベント は、スケジュール式で第 2 レベルの精度を提供しません。Cron 式を使用した最小の解決は分です。CloudWatch イベント とターゲットサービスが持つ分散性の特質により、スケジュールされたルールがトリガーされてから、ターゲットサービスがターゲットリソースの実行を優先するまでの遅延は、数秒となる可能性があります。スケジュールされたルールは、その分のうちにトリガーされますが、正確に 0 秒にトリガーされません。

形式

- cron 式 (p. 34)
- rate 式 (p. 36)

cron 式

Cron 式には 6 つの必須フィールドがあり、それらは空白で区切られます。

構文

```
cron(fields)
```

フィールド	値	[ワイルドカード]
分	0-59	, - * /
時間	0-23	, - * /
日	1-31	, - * ? / L W
月	1-12 または JAN-DEC	, - * /
曜日	1-7 または SUN-SAT	, - * ? L #
年	1970-2199	, - * /

ワイルドカード

- ワイルドカード , (カンマ) 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。
- ワイルドカード - (ダッシュ) 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。
- ワイルドカード * (アスタリスク) 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ています。[*] 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。
- ワイルドカード / (スラッシュ) 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) ワイルドカードはいずれかを意味します。[日] フィールドに 7 と入力し、7 日が何曜日であってもかまわない場合、[曜日] フィールドに ? を入力できます。
- Day-of-month フィールドまたは Day-of-week フィールドの、ワイルドカード L は月または週の最終日を指定します。
- Day-of-month フィールドのワイルドカード **W** は、平日を指定します。Day-of-month フィールドで、**3W** は月の 3 日目に最も近い平日を指定します。
- Day-of-week フィールドの # ワイルドカードは、月の指定された曜日の特定のインスタンスを指定します。たとえば、3#2 は、月の第 2 火曜日を示します。3 は週の 3 番目の日 (火曜日) を示し、2 は月のそのタイプの 2 番目の日を示します。

制限

- cron 式の日フィールドと曜日フィールドを同時に指定することはできません。一方のフィールドに値 (または *) を指定する場合、もう一方のフィールドで ? (疑問符) を使用する必要があります。
- 1 分より短い間隔を導き出す cron 式はサポートされていません。

例

スケジュールに基づいたルールを作成するときは、以下のサンプルの cron 文字列を使用できます。

分	時間	日	月	曜日	年	意味
0	10	*	*	?	*	毎日午前 10:00 (UTC) に実行
15	12	*	*	?	*	毎日午後 12:15 (UTC) に実行
0	18	?	*	MON-FRI	*	毎週月曜日から金曜日まで午後 6:00 (UTC) に実行
0	8	1	*	?	*	毎月 1 日の午前 8:00 (UTC) に実行
0/15	*	*	*	?	*	15 分ごとに実行

分	時間	日	月	曜日	年	意味
0/10	*	?	*	MON-FRI	*	月曜日から金曜日まで 10 分ごとに実行
0/5	8-17	?	*	MON-FRI	*	月曜日から金曜日まで午前 8:00 から午後 5:55 (UTC) の間に 5 分ごとに実行

以下の例に示しているのは、AWS CLI の put-rule コマンドで Cron 式を使用する方法です。最初の例では、毎日午後 12:00 (UTC) にトリガーされるルールを作成します。

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

次の例では、毎日午後 2:05 と 2:35 (UTC) にトリガーされるルールを作成します。

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

次の例では、2002 ~ 2005 年の毎月最後の金曜日の午前 10:15 (UTC) にトリガーされるルールを作成します。

```
aws events put-rule --schedule-expression "cron(15 10 ? * 6L 2002-2005)" --name MyRule3
```

rate 式

rate 式は、予定されたイベントルールを作成すると開始され、その定義済されたスケジュールに基づいて実行されます。

rate 式は 2 つの必須フィールドがあります。フィールドは空白で区切ります。

構文

```
rate(value unit)
```

value

正数。

単位

時刻の単位。値 1 には、`minute` などさまざまな単位が必要です。また、1 を超える値には `minutes` などの単位が必要です。

有効な値: minute | minutes | hour | hours | day | days

制限

値が 1 に等しい場合、単位は単数形であることが必要です。同様に、1 より大きい値の場合、単位は複数であることが必要です。たとえば、rate(1 hours) と rate(5 hour) は有効ではありませんが、rate(1 hour) と rate(5 hours) は有効です。

例

以下の例に示しているのは、AWS CLI の put-rule コマンドで rate 式を使用する方法です。最初の例では、1 分ごとにルールを起動し、次の例は 5 分ごとにルールを起動し、3 番目は 1 時間に 1 回ルールを起動し、最後の例は 1 日に 1 回ルールを起動します。

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

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

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

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

CloudWatch イベント のイベントパターン

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

Amazon CloudWatch Events でのイベントは、JSON オブジェクトとして表されます。JSON オブジェクトの詳細については、「RFC 7159」を参照してください。以下に示しているのは、イベントの例です。

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

イベントについて以下の詳細を覚えておくことが重要です。

- 上記の例に示しているように、同じ最上位のフィールドがあり、それらは必須です。
- [detail] の最上位のフィールドの内容は、どのサービスがイベントを生成したか、そのイベントが何であるかによって異なります。[source] フィールドと [detail-type] フィールドの組み合わせは、[detail] フィールドで見つかるフィールドと値を識別するために役立ちます。AWS のサービスによって生成されるイベントの例については、「CloudWatch イベントのイベントタイプ」を参照してください。

以下に説明しているのは、各イベントフィールドです。

バージョン

デフォルトでは、これはすべてのイベントで 0 (ゼロ) に設定されます。

id

一意の値はすべてのイベントに対して生成されます。これは、イベントがルールからターゲットに移動して処理されるとき、それらのイベントを追跡するために役立ちます。

detail-type

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

CloudTrail 経由で送信されたイベントはすべて、`detail-type` の値が `AWS API Call via CloudTrail` になっています。詳細については、「CloudTrail 経由で配信されたイベント (p. 85)」を参照してください。

送信元

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

AWS のサービスの `source` の正しい値を見つけるには、「AWS サービスの名前空間」を参照してください。たとえば、Amazon CloudFront の `source` の値は、`aws.cloudfront` となります。

アカウント

AWS アカウントを識別する 12 桁の数字。

time

イベントを発生したサービスによって指定できるイベントのタイムスタンプ。イベントが時間間隔にまたがる場合、サービスは開始時間をレポートするように選択可能であるため、この値は、イベントが実際に受け取られるより大幅に前の時間になることがあります。

リージョン

イベントが発生した AWS リージョンを識別します。

リソース

この JSON 配列には、イベントにかかわるリソースを識別する ARN が格納されます。これらの ARN を格納するかどうかは、サービスによって異なります。たとえば、Amazon EC2 インスタンスの状態変更では、Amazon EC2 インスタンス ARN が格納され、Auto Scaling イベントでは、インスタンスと Auto Scaling グループの両方の ARN が格納されますが、AWS CloudTrail での API 呼び出しでは、リソース ARN は格納されません。

detail

JSON オブジェクトであり、その内容はイベントを発生したサービスによって異なります。上記の例の detail の内容は、非常に単純な 2 つのフィールドのみです。AWS API 呼び出しイベントには、約 50 個のフィールドがいく層もの入れ子になった detail オブジェクトあります。

イベントパターン

ルールでは、イベントパターンを使用してイベントを選択し、ターゲットに振り分けます。パターンは、イベントに一致するか、一致しないかのいずれかになります。イベントパターンは、イベントと同様の構造になった JSON オブジェクトとして表されます。たとえば、以下のようになります。

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

イベントパターンマッチングについて以下を覚えておくことが重要です。

- パターンがイベントに一致するには、イベントには、パターンに指定されているすべてのフィールド名が含まれている必要があります。フィールド名は、同じ入れ子構造になったイベントに表示されます。
- パターンに登録されていないイベントの他のフィールドは無視されます。実際には、登録されていないフィールドには "*": "*" ワイルドカードがあります。

- マッチングは厳密 (文字単位) であり、大文字の小文字化など文字列の正規化は行われません。
- 値、つまり、引用符で囲まれた文字列、数字、引用符で囲まれていないキーワード (`true`、`false`、`null`) は JSON 形式のルールに従ってマッチングが調べられます。
- 数字のマッチングは文字列表現レベルで調べられます。たとえば、300、300.0、3.0e2 は等しいとはみなされません。

イベントと一致するパターンを記述するときは、`TestEventPattern` API または `test-event-pattern`CLI コマンドを使用して、パターンが目的のイベントと一致することを確認します。詳細については、「TestEventPattern」または「test-event-pattern」を参照してください。

以下のイベントパターンは、このページ上部のイベントと一致します。1 つめのパターンは、パターンで指定されているインスタンス値のいずれかがイベントと一致していることから、一致します (イベントに含まれていないその他のフィールドをパターンで指定することはできません)。2 つめのパターンは、「終了」状態がイベントに含まれていることから、一致します。

```
{
  "resources": [
    "arn:aws:ec2:us-east-1:123456789012:instance/i-12345678",
    "arn:aws:ec2:us-east-1:123456789012:instance/i-abcdefgh"
  ]
}
```

```
{
  "detail": {
    "state": [ "terminated" ]
  }
}
```

以下のイベントパターンは、このページの上部のイベントと一致しません。1 つめのパターンは、パターンで「保留」状態を表す値が指定されており、イベントにこの値が表示されていないことから、一致しません。2 つめのパターンは、パターンで指定されているリソース値がイベントに表示されていないことから、一致しません。

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

```
{
  "source": [ "aws.ec2" ],
  "detail-type": [ "EC2 Instance State-change Notification" ],
  "resources": [ "arn:aws:ec2:us-east-1::image/ami-12345678" ]
}
```

イベントパターンでの Null 値や空の文字列との一致

Null 値や空の文字列を持つイベントフィールドと一致するパターンを作成できます。この仕組みを確認するには、次のイベント例を検討してください。

```
{
  "version": "0",
  "id": "3e3c153a-8339-4e30-8c35-687ebef853fe",
  "detail-type": "EC2 Instance Launch Successful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2015-11-11T21:31:47Z",
  "region": "us-east-1",
  "resources": [
   ],
  "detail": {
    "eventVersion": "",
    "responseElements": null
   }
}
```

`eventVersion` の値が空の文字列であるイベントと一致させるには、イベント例と一致する次のパターンを使用します。

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

`responseElements` の値が Null であるイベントと一致させるには、イベント例と一致する次のパターンを使用します。

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

Null 値および空の文字列は、パターンマッチングで交換可能ではありません。空の文字列を検出するために記述されたパターンは、`null` の値をキャッチしません。

CloudWatch イベント パターンの配列

パターン内の各フィールドの値は 1 つ以上の値を格納する配列であり、配列の値のいずれかがイベントの値と一致すれば、パターンが一致したとみなされます。イベントの値が配列の場合、パターン配列とイベント配列の交差部分が空でないなら、パターンが一致したとみなされます。

たとえば、イベントパターンの例には次のテキストが含まれています。

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

パターンの例は、次のテキストが含まれているイベントに一致します。パターン配列の最初の項目が、イベント配列の 2 番目の項目と一致するためです。

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

サポートされている各サービスからの CloudWatch イベント イベントの例

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

以下のリストの AWS のサービスでは、CloudWatch イベント によって検出できるイベントを出力します。

また、CloudWatch イベント は、イベントを出力しないが、このページに記載されていないサービスで使用することもできます。そのためには、CloudTrail を使用して送信されるイベントを監視します。詳細については、「CloudTrail 経由で配信されたイベント (p. 85)」を参照してください。

イベントタイプ

Amazon Augmented AI イベント

Amazon Augmented AI によって生成されるイベントの例については、「Amazon Augmented AI でイベントを使用する」を参照してください。

Application Auto Scaling イベント

Application Auto Scaling によって生成されるイベントの例については、「アプリケーションの Auto Scaling イベントと EventBridge」を参照してください。

AWS Batch イベント

AWS Batch により生成されたイベントの例については、「AWS Batch イベント」を参照してください。

Amazon CloudWatch Events のスケジュールイベント

予定されているイベントの例を次に示します。

```
{
  "id": "53dc4d37-cffa-4f76-80c9-8b7d4a4d2eaa",
  "detail-type": "Scheduled Event",
  "source": "aws.events",
  "account": "123456789012",
  "time": "2019-10-08T16:53:06Z",
  "region": "us-east-1",
  "resources": [ "arn:aws:events:us-east-1:123456789012:rule/MyScheduledRule" ],
```

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

Amazon Chime イベント

Amazon Chime によって生成されるイベントの例については、「EventBridge を使用した Amazon Chime の自動化」を参照してください。

CloudWatch からのイベント

CloudWatch からのイベントの例については、AWS CodeBuild ユーザーガイドの「アラームイベントと EventBridge」を参照してください。

CodeBuild イベント

CodeBuild のサンプルイベントについては、『AWS CodeBuild ユーザーガイド』の「ビルド通知入力形式リファレンス」を参照してください。

CodeCommit イベント

CodeCommit のイベント例については、AWS CodeCommit ユーザーガイドの「EventBridge および CloudWatch イベント の CodeCommit イベントのモニタリング」を参照してください。

AWS CodeDeploy イベント

以下は、CodeDeploy のイベントの例です。詳細については、『AWS CodeDeploy User Guide』の「CloudWatch イベント によるデプロイのモニタリング」を参照してください。

CodeDeploy のデプロイ状態変更通知

デプロイの状態に変化がありました。

```
{
  "account": "123456789012",
  "region": "us-east-1",
  "detail-type": "CodeDeploy Deployment State-change Notification",
  "source": "aws.codedeploy",
  "version": "0",
  "time": "2016-06-30T22:06:31Z",
  "id": "c071bfbf-83c4-49ca-a6ff-3df053957145",
  "resources": [
    "arn:aws:codedeploy:us-east-1:123456789012:application:myApplication",
    "arn:aws:codedeploy:us-east-1:123456789012:deploymentgroup:myApplication/
myDeploymentGroup"
  ],
  "detail": {
    "instanceGroupId": "9fd2fbef-2157-40d8-91e7-6845af69e2d2",
    "region": "us-east-1",
    "application": "myApplication",
    "deploymentId": "d-123456789",
```

```
    "state": "SUCCESS",
    "deploymentGroup": "myDeploymentGroup"
  }
}
```

CodeDeploy インスタンスの状態変更通知

デプロイグループに属するインスタンスの状態に変化がありました。

```
{
  "account": "123456789012",
  "region": "us-east-1",
  "detail-type": "CodeDeploy Instance State-change Notification",
  "source": "aws.codedeploy",
  "version": "0",
  "time": "2016-06-30T23:18:50Z",
  "id": "fb1d3015-c091-4bf9-95e2-d98521ab2ecb",
  "resources": [
    "arn:aws:ec2:us-east-1:123456789012:instance/i-0000000aaaaaaaaaa",
    "arn:aws:codedeploy:us-east-1:123456789012:deploymentgroup:myApplication/
myDeploymentGroup",
    "arn:aws:codedeploy:us-east-1:123456789012:application:myApplication"
  ],
  "detail": {
    "instanceId": "i-0000000aaaaaaaaaa",
    "region": "us-east-1",
    "state": "SUCCESS",
    "application": "myApplication",
    "deploymentId": "d-123456789",
    "instanceGroupId": "8cd3bfa8-9e72-4cbe-a1e5-da4efc7efd49",
    "deploymentGroup": "myDeploymentGroup"
  }
}
```

CodePipeline イベント

以下は、CodePipeline のイベントの例です。

パイプライン実行の状態変更

```
{
  "version": "0",
  "id": "CWE-event-id",
  "detail-type": "CodePipeline Pipeline Execution State Change",
  "source": "aws.codepipeline",
  "account": "123456789012",
  "time": "2017-04-22T03:31:47Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:codepipeline:us-east-1:123456789012:pipeline:myPipeline"
  ],
  "detail": {
    "pipeline": "myPipeline",
    "version": "1",
    "state": "STARTED",
    "execution-id": "01234567-0123-0123-0123-012345678901"
  }
}
```

ステージ実行の状態変更

```
{
  "version": "0",
  "id": "CWE-event-id",
  "detail-type": "CodePipeline Stage Execution State Change",
  "source": "aws.codepipeline",
  "account": "123456789012",
  "time": "2017-04-22T03:31:47Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:codepipeline:us-east-1:123456789012:pipeline:myPipeline"
  ],
  "detail": {
    "pipeline": "myPipeline",
    "version": "1",
    "execution-id": "01234567-0123-0123-0123-012345678901",
    "stage": "Prod",
    "state": "STARTED"
  }
}
```

アクション実行の状態変更

このサンプルでは、2 つの `region` フィールドがあります。一番上のものは、ターゲットパイプラインのアクションが実行される AWS リージョンの名前です。この例では、`us-east-1` です。`detail` セクションの `region` は、イベントが作成された AWS リージョンです。これは、パイプラインが作成されたリージョンと同じです。この例では、`us-west-2` です。

```
{
  "version": "0",
  "id": "CWE-event-id",
  "detail-type": "CodePipeline Action Execution State Change",
  "source": "aws.codepipeline",
  "account": "123456789012",
  "time": "2017-04-22T03:31:47Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:codepipeline:us-east-1:123456789012:pipeline:myPipeline"
  ],
  "detail": {
    "pipeline": "myPipeline",
    "version": 1,
    "execution-id": "01234567-0123-0123-0123-012345678901",
    "stage": "Prod",
    "action": "myAction",
    "state": "STARTED",
    "region":"us-west-2",
    "type": {
      "owner": "AWS",
      "category": "Deploy",
      "provider": "CodeDeploy",
      "version": 1
    }
  }
}
```

AWS Config イベント

AWS Config イベントの詳細については、『AWS Config Developer Guide』の「Amazon CloudWatch Events を使用した AWS Config イベントのモニタリング」を参照してください。

Amazon EBS イベント

Amazon EBS イベントの詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「Amazon EBS の Amazon CloudWatch Events」を参照してください。

Amazon EC2 Auto Scaling イベント

Auto Scaling イベントの詳細については、『Amazon EC2 Auto Scaling ユーザーガイド』の「Auto Scaling グループスケーリング時の CloudWatch イベントの取得」を参照してください。

Amazon EC2 スポットインスタンスの中断イベント

スポットインスタンスの中断イベントの詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「スポットインスタンスの中断の通知」を参照してください。

Amazon EC2 状態変更イベント

以下は、インスタンスの状態が変化した場合の Amazon EC2 インスタンスのイベントの例です。

EC2 インスタンスの状態変更通知

この例は、`pending` 状態のインスタンス用です。`state` の他の有効な値には、`running`、`shutting-down`、`stopped`、`stopping`、`terminated` があります。

```
{
   "id":"7bf73129-1428-4cd3-a780-95db273d1602",
   "detail-type":"EC2 Instance State-change Notification",
   "source":"aws.ec2",
   "account":"123456789012",
   "time":"2019-11-11T21:29:54Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1111"
   ],
   "detail":{
      "instance-id":"i-abcd1111",
      "state":"pending"
   }
}
```

Amazon Elastic Container Registry イベント

Amazon ECR は、イメージアクションイベントを EventBridge に送信します。イベントは、イメージがプッシュ、スキャン、または削除されたときに送信されます。

Amazon ECS サンプルイベントについては、Amazon Elastic Container Registry ユーザーガイドの「Amazon ECR イベント」を参照してください。

Amazon Elastic Container Service イベント

Amazon ECS から EventBridge に送信されるイベントには、コンテナインスタンスイベントとタスクイベントの 2 種類があります。コンテナインスタンスイベントが送信されるのは、タスクに EC2 起動タイプを使用している場合のみです。Fargate 起動タイプを使用しているタスクの場合、タスクの状態イベントのみ受信します。Amazon ECS は、コンテナインスタンスとタスクの状態を追跡します。いずれかのリソースが変更されると、イベントがトリガーされます。イベントは、コンテナインスタンス状態変更イベントまたはタスク状態変更イベントのいずれかに分類されます。

Amazon ECS サンプルイベントについては、Amazon Elastic Container Service Developer Guideの「Amazon ECS イベント」を参照してください。

AWS Elemental MediaConvert イベント

MediaConvert サンプルイベントについては、『AWS Elemental MediaConvert ユーザーガイド』の「CloudWatch イベント を使用して AWS Elemental MediaConvert ジョブをモニタリングする」を参照してください。

AWS Elemental MediaPackage イベント

MediaPackage サンプルイベントについては、『AWS Elemental MediaPackage ユーザーガイド』の「Amazon CloudWatch Events を使用した AWS Elemental MediaPackage のモニタリング」を参照してください。

AWS Elemental MediaStore イベント

MediaStore サンプルイベントについては、『AWS Elemental MediaStore ユーザーガイド』の「CloudWatch イベント を使用した AWS Elemental MediaStore の自動化」を参照してください。

Amazon EMR イベント

Amazon EMR によって報告されるイベントには、`Source` の値として `aws.emr` があり、CloudTrail によって報告される Amazon EMR API イベントには、`Source` の値として `aws.elasticmapreduce` があります。

以下は、Amazon EMR で報告されるイベントの例です。

Amazon EMR Auto Scaling のポリシー状態の変更

```
{
   "version":"0",
   "id":"2f8147ab-8c48-47c6-b0b6-3ee23ec8d300",
   "detail-type":"EMR Auto Scaling Policy State Change",
   "source":"aws.emr",
   "account":"123456789012",
   "time":"2016-12-16T20:42:44Z",
   "region":"us-east-1",
   "resources":[],
   "detail":{
      "resourceId":"ig-X2LBMHTGPCBU",
      "clusterId":"j-1YONHTCP3YZKC",
```

```
      "state":"PENDING",
      "message":"AutoScaling policy modified by user request",
      "scalingResourceType":"INSTANCE_GROUP"
   }
}
```

Amazon EMR クラスター状態の変更 – 開始中

```
{
  "version": "0",
  "id": "999cccaa-eaaa-0000-1111-123456789012",
  "detail-type": "EMR Cluster State Change",
  "source": "aws.emr",
  "account": "123456789012",
  "time": "2016-12-16T20:43:05Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "severity": "INFO",
    "stateChangeReason": "{\"code\":\"\"}",
    "name": "Development Cluster",
    "clusterId": "j-123456789ABCD",
    "state": "STARTING",
    "message": "Amazon EMR cluster j-123456789ABCD (Development Cluster) was requested at
 2016-12-16 20:42 UTC and  is being created."
  }
}
```

Amazon EMR クラスター状態の変更 – 終了

```
{
  "version": "0",
  "id": "1234abb0-f87e-1234-b7b6-000000123456",
  "detail-type": "EMR Cluster State Change",
  "source": "aws.emr",
  "account": "123456789012",
  "time": "2016-12-16T21:00:23Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "severity": "INFO",
    "stateChangeReason": "{\"code\":\"USER_REQUEST\",\"message\":\"Terminated by user
 request\"}",
    "name": "Development Cluster",
    "clusterId": "j-123456789ABCD",
    "state": "TERMINATED",
    "message": "Amazon EMR Cluster jj-123456789ABCD (Development Cluster) has terminated at
 2016-12-16 21:00 UTC with a reason of USER_REQUEST."
  }
}
```

Amazon EMR インスタンスグループの状態の変更

```
{
  "version": "0",
  "id": "999cccaa-eaaa-0000-1111-123456789012",
  "detail-type": "EMR Instance Group State Change",
  "source": "aws.emr",
  "account": "123456789012",
  "time": "2016-12-16T20:57:47Z",
  "region": "us-east-1",
  "resources": [],
```

```
  "detail": {
    "market": "ON_DEMAND",
    "severity": "INFO",
    "requestedInstanceCount": "2",
    "instanceType": "m3.xlarge",
    "instanceGroupType": "CORE",
    "instanceGroupId": "ig-ABCDEFGHIJKL",
    "clusterId": "j-123456789ABCD",
    "runningInstanceCount": "2",
    "state": "RUNNING",
    "message": "The resizing operation for instance group ig-ABCDEFGHIJKL in Amazon EMR
 cluster j-123456789ABCD (Development Cluster) is complete. It now has an instance count of
 2. The resize started at 2016-12-16 20:57 UTC and took 0 minutes to complete."
  }
}
```

Amazon EMR ステップ状態の変更

```
{
  "version": "0",
  "id": "999cccaa-eaaa-0000-1111-123456789012",
  "detail-type": "EMR Step Status Change",
  "source": "aws.emr",
  "account": "123456789012",
  "time": "2016-12-16T20:53:09Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "severity": "ERROR",
    "actionOnFailure": "CONTINUE",
    "stepId": "s-ZYXWVUTSRQPON",
    "name": "CustomJAR",
    "clusterId": "j-123456789ABCD",
    "state": "FAILED",
    "message": "Step s-ZYXWVUTSRQPON (CustomJAR) in Amazon EMR cluster j-123456789ABCD
 (Development Cluster) failed at 2016-12-16 20:53 UTC."
  }
}
```

Amazon GameLift イベント

以下は、Amazon GameLift のイベントの例です。詳細については、『Amazon GameLift 開発者ガイド』の「FlexMatch イベントリリファレンス」を参照してください。

マッチメーキング検索

```
{
  "version": "0",
  "id": "cc3d3ebe-1d90-48f8-b268-c96655b8f013",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-08T21:15:36.421Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/SampleConfiguration"
  ],
  "detail": {
    "tickets": [
      {
```

```
        "ticketId": "ticket-1",
        "startTime": "2017-08-08T21:15:35.676Z",
        "players": [
          {
            "playerId": "player-1"
          }
        ]
      }
    ],
    "estimatedWaitMillis": "NOT_AVAILABLE",
    "type": "MatchmakingSearching",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1"
        }
      ]
    }
  }
}
```

作成されたマッチング候補

```
{
  "version": "0",
  "id": "fce8633f-aea3-45bc-aeba-99d639cad2d4",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-08T21:17:41.178Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/SampleConfiguration"
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-08T21:15:35.676Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      },
      {
        "ticketId": "ticket-2",
        "startTime": "2017-08-08T21:17:40.657Z",
        "players": [
          {
            "playerId": "player-2",
            "team": "blue"
          }
        ]
      }
    ],
    "acceptanceTimeout": 600,
    "ruleEvaluationMetrics": [
      {
        "ruleName": "EvenSkill",
        "passedCount": 3,
        "failedCount": 0
      },
      {
```

```
        "ruleName": "EvenTeams",
        "passedCount": 3,
        "failedCount": 0
      },
      {
        "ruleName": "FastConnection",
        "passedCount": 3,
        "failedCount": 0
      },
      {
        "ruleName": "NoobSegregation",
        "passedCount": 3,
        "failedCount": 0
      }
    ],
    "acceptanceRequired": true,
    "type": "PotentialMatchCreated",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        },
        {
          "playerId": "player-2",
          "team": "blue"
        }
      ]
    },
    "matchId": "3faf26ac-f06e-43e5-8d86-08feff26f692"
  }
}
```

一致の受け入れ

```
{
  "version": "0",
  "id": "b3f76d66-c8e5-416a-aa4c-aa1278153edc",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-09T20:04:42.660Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/SampleConfiguration"
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-09T20:01:35.305Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      },
      {
        "ticketId": "ticket-2",
        "startTime": "2017-08-09T20:04:16.637Z",
        "players": [
          {
            "playerId": "player-2",
            "team": "blue",
```

```
            "accepted": false
          }
        ]
      }
    ],
    "type": "AcceptMatch",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        },
        {
          "playerId": "player-2",
          "team": "blue",
          "accepted": false
        }
      ]
    },
    "matchId": "848b5f1f-0460-488e-8631-2960934d13e5"
  }
}
```

マッチの受け入れ完了

```
{
  "version": "0",
  "id": "b1990d3d-f737-4d6c-b150-af5ace8c35d3",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-08T20:43:14.621Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/SampleConfiguration"
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-08T20:30:40.972Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      },
      {
        "ticketId": "ticket-2",
        "startTime": "2017-08-08T20:33:14.111Z",
        "players": [
          {
            "playerId": "player-2",
            "team": "blue"
          }
        ]
      }
    ],
    "acceptance": "TimedOut",
    "type": "AcceptMatchCompleted",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
```

```
          "team": "red"
        },
        {
          "playerId": "player-2",
          "team": "blue"
        }
      ]
    },
    "matchId": "a0d9bd24-4695-4f12-876f-ea6386dd6dce"
  }
}
```

マッチメーキングの成功

```
{
  "version": "0",
  "id": "5ccb6523-0566-412d-b63c-1569e00d023d",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-09T19:59:09.159Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/SampleConfiguration"
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-09T19:58:59.277Z",
        "players": [
          {
            "playerId": "player-1",
            "playerSessionId": "psess-6e7c13cf-10d6-4756-a53f-db7de782ed67",
            "team": "red"
          }
        ]
      },
      {
        "ticketId": "ticket-2",
        "startTime": "2017-08-09T19:59:08.663Z",
        "players": [
          {
            "playerId": "player-2",
            "playerSessionId": "psess-786b342f-9c94-44eb-bb9e-c1de46c472ce",
            "team": "blue"
          }
        ]
      }
    ],
    "type": "MatchmakingSucceeded",
    "gameSessionInfo": {
      "gameSessionArn": "arn:aws:gamelift:us-west-2:123456789012:gamesession/836cf48d-
bcb0-4a2c-bec1-9c456541352a",
      "ipAddress": "192.168.1.1",
      "port": 10777,
      "players": [
        {
          "playerId": "player-1",
          "playerSessionId": "psess-6e7c13cf-10d6-4756-a53f-db7de782ed67",
          "team": "red"
        },
        {
          "playerId": "player-2",
          "playerSessionId": "psess-786b342f-9c94-44eb-bb9e-c1de46c472ce",
```

```
          "team": "blue"
        }
      ]
    },
    "matchId": "c0ec1a54-7fec-4b55-8583-76d67adb7754"
  }
}
```

マッチメーキングのタイムアウト

```
{
  "version": "0",
  "id": "fe528a7d-46ad-4bdc-96cb-b094b5f6bf56",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-09T20:11:35.598Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/SampleConfiguration"
  ],
  "detail": {
    "reason": "TimedOut",
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-09T20:01:35.305Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      }
    ],
    "ruleEvaluationMetrics": [
      {
        "ruleName": "EvenSkill",
        "passedCount": 3,
        "failedCount": 0
      },
      {
        "ruleName": "EvenTeams",
        "passedCount": 3,
        "failedCount": 0
      },
      {
        "ruleName": "FastConnection",
        "passedCount": 3,
        "failedCount": 0
      },
      {
        "ruleName": "NoobSegregation",
        "passedCount": 3,
        "failedCount": 0
      }
    ],
    "type": "MatchmakingTimedOut",
    "message": "Removed from matchmaking due to timing out.",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        }
```

```
      ]
    }
  }
}
```

マッチメーキングのキャンセル

```
{
  "version": "0",
  "id": "8d6f84da-5e15-4741-8d5c-5ac99091c27f",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-09T20:00:07.843Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/SampleConfiguration"
  ],
  "detail": {
    "reason": "Cancelled",
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-09T19:59:26.118Z",
        "players": [
          {
            "playerId": "player-1"
          }
        ]
      }
    ],
    "ruleEvaluationMetrics": [
      {
        "ruleName": "EvenSkill",
        "passedCount": 0,
        "failedCount": 0
      },
      {
        "ruleName": "EvenTeams",
        "passedCount": 0,
        "failedCount": 0
      },
      {
        "ruleName": "FastConnection",
        "passedCount": 0,
        "failedCount": 0
      },
      {
        "ruleName": "NoobSegregation",
        "passedCount": 0,
        "failedCount": 0
      }
    ],
    "type": "MatchmakingCancelled",
    "message": "Cancelled by request.",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1"
        }
      ]
    }
  }
}
```

マッチメーキングの失敗

```
{
  "version": "0",
  "id": "025b55a4-41ac-4cf4-89d1-f2b3c6fd8f9d",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-16T18:41:09.970Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/SampleConfiguration"
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-16T18:41:02.631Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      }
    ],
    "customEventData": "foo",
    "type": "MatchmakingFailed",
    "reason": "UNEXPECTED_ERROR",
    "message": "An unexpected error was encountered during match placing.",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        }
      ]
    },
    "matchId": "3ea83c13-218b-43a3-936e-135cc570cba7"
  }
}
```

AWS Glue イベント

AWS Glue イベントの形式は次のとおりです。

ジョブ実行に成功

```
{
    "version":"0",
    "id":"abcdef00-1234-5678-9abc-def012345678",
    "detail-type":"Glue Job State Change",
    "source":"aws.glue",
    "account":"123456789012",
    "time":"2017-09-07T18:57:21Z",
    "region":"us-west-2",
    "resources":[],
    "detail":{
        "jobName":"MyJob",
        "severity":"INFO",
        "state":"SUCCEEDED",
```

```
        "jobRunId":"jr_abcdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789",
        "message":"Job run succeeded"
    }
}
```

ジョブ実行に失敗

```
{
    "version":"0",
    "id":"abcdef01-1234-5678-9abc-def012345678",
    "detail-type":"Glue Job State Change",
    "source":"aws.glue",
    "account":"123456789012",
    "time":"2017-09-07T06:02:03Z",
    "region":"us-west-2",
    "resources":[],
    "detail":{
        "jobName":"MyJob",
        "severity":"ERROR",
        "state":"FAILED",
        "jobRunId":"jr_0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef",
        "message":"JobName:MyJob and
 JobRunId:jr_0123456789abcdef0123456789abcdef0123456789abcdef failed to
 execute with exception Role arn:aws:iam::123456789012:role/Glue_Role should be given
 assume role permissions for Glue Service."
    }
}
```

Timeout

```
{
    "version":"0",
    "id":"abcdef00-1234-5678-9abc-def012345678",
    "detail-type":"Glue Job State Change",
    "source":"aws.glue",
    "account":"123456789012",
    "time":"2017-11-20T20:22:06Z",
    "region":"us-east-1",
    "resources":[],
    "detail":{
        "jobName":"MyJob",
        "severity":"WARN",
        "state":"TIMEOUT",
        "jobRunId":"jr_abc0123456789abcdef0123456789abcdef0123456789abcdef0123456789def",
        "message":"Job run timed out"
    }
}
```

ジョブ実行の停止

```
{
   "version":"0",
   "id":"abcdef00-1234-5678-9abc-def012345678",
   "detail-type":"Glue Job State Change",
   "source":"aws.glue",
   "account":"123456789012",
   "time":"2017-11-20T20:22:06Z",
   "region":"us-east-1",
   "resources":[],
   "detail":{
       "jobName":"MyJob",
       "severity":"INFO",
```

```
       "state":"STOPPED",
       "jobRunId":"jr_abc0123456789abcdef0123456789abcdef0123456789abcdef0123456789def",
       "message":"Job run stopped"
    }
}
```

クローラ開始

```
{
   "version":"0",
   "id":"05efe8a2-c309-6884-a41b-3508bcdc9695",
   "detail-type":"Glue Crawler State Change",
   "source":"aws.glue",
   "account":"561226563745",
   "time":"2017-11-11T01:09:46Z",
   "region":"us-east-1",
   "resources":[

   ],
   "detail":{
      "accountId":"561226563745",
      "crawlerName":"S3toS3AcceptanceTestCrawlera470bd94-9e00-4518-8942-e80c8431c322",
      "startTime":"2017-11-11T01:09:46Z",
      "state":"Started",
      "message":"Crawler Started"
   }
}
```

クローラ成功

```
{
   "version":"0",
   "id":"3d675db5-59b9-6388-b8e8-e0a9b6d567a9",
   "detail-type":"Glue Crawler State Change",
   "source":"aws.glue",
   "account":"561226563745",
   "time":"2017-11-11T01:25:00Z",
   "region":"us-east-1",
   "resources":[

   ],
   "detail":{
      "tablesCreated":"0",
      "warningMessage":"N/A",
      "partitionsUpdated":"0",
      "tablesUpdated":"0",
      "message":"Crawler Succeeded",
      "partitionsDeleted":"0",
      "accountId":"561226563745",
      "runningTime (sec)":"7",
      "tablesDeleted":"0",
      "crawlerName":"SchedulerTestCrawler51fb3a8b-1015-49f0-a969-ca126680b94b",
      "completionDate":"2017-11-11T01:25:00Z",
      "state":"Succeeded",
      "partitionsCreated":"0",
      "cloudWatchLogLink":"https://console.aws.amazon.com/
cloudwatch/home?region=us-east-1#logEventViewer:group=/aws-glue/
crawlers;stream=SchedulerTestCrawler51fb3a8b-1015-49f0-a969-ca126680b94b"
   }
}
```

クローラ失敗

```
{
   "version":"0",
   "id":"f7965b59-470f-2e06-bb89-a8cebaabefac",
   "detail-type":"Glue Crawler State Change",
   "source":"aws.glue",
   "account":"782104008917",
   "time":"2017-10-20T05:10:08Z",
   "region":"us-east-1",
   "resources":[

   ],
   "detail":{
      "crawlerName":"test-crawler-notification",
      "errorMessage":"Internal Service Exception",
      "accountId":"1234",
      "cloudWatchLogLink":"https://console.aws.amazon.com/cloudwatch/home?region=us-
east-1#logEventViewer:group=/aws-glue/crawlers;stream=test-crawler-notification",
      "state":"Failed",
      "message":"Crawler Failed"
   }
}
```

ジョブ実行は開始中状態

```
{
   "version":"0",
   "id":"66fbc5e1-aac3-5e85-63d0-856ec669a050",
   "detail-type":"Glue Job Run Status",
   "source":"aws.glue",
   "account":"123456789012",
   "time":"2018-04-24T20:57:34Z",
   "region":"us-east-1",
   "resources":[],
   "detail":{
      "jobName":"MyJob",
      "severity":"INFO",
      "notificationCondition":{
         "NotifyDelayAfter":1.0
      },
      "state":"STARTING",
          "jobRunId":"jr_6aa58e7a3aa44e2e4c7db2c50e2f7396cb57901729e4b702dcb2cfbbeb3f7a86",
      "message":"Job is in STARTING state",
      "startedOn":"2018-04-24T20:55:47.941Z"
   }
}
```

ジョブ実行は実行中状態

```
{
   "version":"0",
   "id":"66fbc5e1-aac3-5e85-63d0-856ec669a050",
   "detail-type":"Glue Job Run Status",
   "source":"aws.glue",
   "account":"123456789012",
   "time":"2018-04-24T20:57:34Z",
   "region":"us-east-1",
   "resources":[],
   "detail":{
      "jobName":"MyJob",
      "severity":"INFO",
      "notificationCondition":{
         "NotifyDelayAfter":1.0
      },
```

```
      "state":"RUNNING",
    "jobRunId":"jr_6aa58e7a3aa44e2e4c7db2c50e2f7396cb57901729e4b702dcb2cfbbeb3f7a86",
      "message":"Job is in RUNNING state",
      "startedOn":"2018-04-24T20:55:47.941Z"
   }
}
```

ジョブ実行は停止中状態

```
{
   "version":"0",
   "id":"66fbc5e1-aac3-5e85-63d0-856ec669a050",
   "detail-type":"Glue Job Run Status",
   "source":"aws.glue",
   "account":"123456789012",
   "time":"2018-04-24T20:57:34Z",
   "region":"us-east-1",
   "resources":[],
   "detail":{
      "jobName":"MyJob",
      "severity":"INFO",
      "notificationCondition":{
         "NotifyDelayAfter":1.0
      },
      "state":"STOPPING",
    "jobRunId":"jr_6aa58e7a3aa44e2e4c7db2c50e2f7396cb57901729e4b702dcb2cfbbeb3f7a86",
      "message":"Job is in STOPPING state",
      "startedOn":"2018-04-24T20:55:47.941Z"
   }
}
```

AWS Glueデータカタログのテーブルの状態変更

```
{
    "version": "0",
    "id": "2617428d-715f-edef-70b8-d210da0317a0",
    "detail-type": "Glue Data Catalog Table State Change",
    "source": "aws.glue",
    "account": "123456789012",
    "time": "2019-01-16T18:16:01Z",
    "region": "eu-west-1",
    "resources": [
        "arn:aws:glue:eu-west-1:123456789012:table/d1/t1"
    ],
    "detail": {
        "databaseName": "d1",
        "changedPartitions": [
            "[C.pdf, dir3]",
            "[D.doc, dir4]"
        ],
        "typeOfChange": "BatchCreatePartition",
        "tableName": "t1"
    }
}
```

AWS Glueデータカタログのデータベースの状態変更

次の例で、typeofChange は CreateTable です。このフィールドで可能なその他の値は、CreateDatabase と UpdateTable です。

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

```
    "id": "60e7ddc2-a588-5328-220a-21c060f6c3f4",
    "detail-type": "Glue Data Catalog Database State Change",
    "source": "aws.glue",
    "account": "123456789012",
    "time": "2019-01-16T18:08:48Z",
    "region": "eu-west-1",
    "resources": [
        "arn:aws:glue:eu-west-1:123456789012:table/d1/t1"
    ],
    "detail": {
        "databaseName": "d1",
        "typeOfChange": "CreateTable",
        "changedTables": [
            "t1"
        ]
    }
}
```

AWS Ground Station イベント

AWS Ground Station イベントの例については、「CloudWatch イベント による AWS Ground Station の自動化」 (AWS Ground Station ユーザーガイド) を参照してください。

Amazon GuardDuty イベント

Amazon GuardDuty サンプルイベントの詳細については、Amazon GuardDuty ユーザーガイドの「Amazon CloudWatch Events で Amazon GuardDuty をモニタリングする」を参照してください。

AWS Health イベント

AWS Personal Health Dashboard (AWS Health) イベントの形式です。詳細については、『AWS Health のユーザーガイド』の「Amazon CloudWatch Events を使用した AWS Health イベントの管理」を参照してください。

AWS Health イベントの形式

```
{
  "version": "0",
  "id": "7bf73129-1428-4cd3-a780-95db273d1602",
  "detail-type": "AWS Health Event",
  "source": "aws.health",
  "account": "123456789012",
  "time": "2016-06-05T06:27:57Z",
  "region": "region",
  "resources": [],
  "detail": {
    "eventArn": "arn:aws:health:region::event/id",
    "service": "service",
    "eventTypeCode": "AWS_service_code",
    "eventTypeCategory": "category",
    "startTime": "Sun, 05 Jun 2016 05:01:10 GMT",
    "endTime": "Sun, 05 Jun 2016 05:30:57 GMT",
    "eventDescription": [{
      "language": "lang-code",
      "latestDescription": "description"
    }]
```

```
    ...
  }
}
```

eventTypeCategory

イベントのカテゴリコード指定できる値は、issue、accountNotification、および scheduledChange です。

eventTypeCode

イベントタイプの一意の識別子。例には、AWS_EC2_INSTANCE_NETWORK_MAINTENANCE_SCHEDULED や AWS_EC2_INSTANCE_REBOOT_MAINTENANCE_SCHEDULED が含まれます。MAINTENANCE_SCHEDULED を含むイベントは、通常、startTime の約 2 週間前にプッシュアウトされます。

id

イベントの一意の識別子。

サービス

イベントに影響を受ける AWS サービス。EC2、S3、REDSHIFT、RDS などが該当します。

Elastic Load BalancingAPI の問題

```
{
  "version": "0",
  "id": "121345678-1234-1234-1234-123456789012",
  "detail-type": "AWS Health Event",
  "source": "aws.health",
  "account": "123456789012",
  "time": "2016-06-05T06:27:57Z",
  "region": "ap-southeast-2",
  "resources": [],
  "detail": {
    "eventArn": "arn:aws:health:ap-southeast-2::event/
AWS_ELASTICLOADBALANCING_API_ISSUE_90353408594353980",
    "service": "ELASTICLOADBALANCING",
    "eventTypeCode": "AWS_ELASTICLOADBALANCING_API_ISSUE",
    "eventTypeCategory": "issue",
    "startTime": "Sat, 11 Jun 2016 05:01:10 GMT",
    "endTime": "Sat, 11 Jun 2016 05:30:57 GMT",
    "eventDescription": [{
      "language": "en_US",
      "latestDescription": "A description of the event will be provided here"
  }
}
```

Amazon EC2インスタンスストアドライブのパフォーマンス低下

```
{
  "version": "0",
  "id": "121345678-1234-1234-1234-123456789012",
  "detail-type": "AWS Health Event",
  "source": "aws.health",
  "account": "123456789012",
  "time": "2016-06-05T06:27:57Z",
  "region": "us-west-2",
  "resources": [
    "i-abcd1111"
  ],
```

```
  "detail": {
    "eventArn": "arn:aws:health:us-west-2::event/
AWS_EC2_INSTANCE_STORE_DRIVE_PERFORMANCE_DEGRADED_90353408594353980",
    "service": "EC2",
    "eventTypeCode": "AWS_EC2_INSTANCE_STORE_DRIVE_PERFORMANCE_DEGRADED",
    "eventTypeCategory": "issue",
    "startTime": "Sat, 05 Jun 2016 15:10:09 GMT",
    "eventDescription": [{
      "language": "en_US",
      "latestDescription": "A description of the event will be provided here"
    }],
    "affectedEntities": [{
      "entityValue": "i-abcd1111",
      "tags": {
        "stage": "prod",
        "app": "my-app"
  }
}
```

AWS KMS イベント

以下は、AWS Key Management Service (AWS KMS) イベントの例です。詳細については、『AWS Key Management Service Developer Guide』の「AWS KMS イベント」を参照してください。

KMS CMK ローテーション

AWS KMS は、CMK のキーデータを自動的にローテーションしました。

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "KMS CMK Rotation",
  "source": "aws.kms",
  "account": "111122223333",
  "time": "2016-08-25T21:05:33Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  ],
  "detail": {
    "key-id": "1234abcd-12ab-34cd-56ef-1234567890ab"
  }
}
```

KMS でインポートされたキーマテリアルの有効期限

AWS KMS は、CMK の期限切れキーマテリアルを削除しました。

```
{
  "version": "0",
  "id": "9da9af57-9253-4406-87cb-7cc400e43465",
  "detail-type": "KMS Imported Key Material Expiration",
  "source": "aws.kms",
  "account": "111122223333",
  "time": "2016-08-22T20:12:19Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  ],
  "detail": {
```

```
    "key-id": "1234abcd-12ab-34cd-56ef-1234567890ab"
  }
}
```

KMS CMK 削除

AWS KMS は、スケジュールされた CMK を完全に削除しました。

```
{
  "version": "0",
  "id": "e9ce3425-7d22-412a-a699-e7a5fc3fbc9a",
  "detail-type": "KMS CMK Deletion",
  "source": "aws.kms",
  "account": "111122223333",
  "time": "2016-08-19T03:23:45Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  ],
  "detail": {
    "key-id": "1234abcd-12ab-34cd-56ef-1234567890ab"
  }
}
```

Amazon Macie Classic イベント

Amazon Macie Classic イベントの例は、次のとおりです。

アラートの作成

```
{
  "version": "0",
  "id": "CWE-event-id",
  "detail-type": "Macie Alert",
  "source": "aws.macie",
  "account": "123456789012",
  "time": "2017-04-24T22:28:49Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:macie:us-east-1:123456789012:trigger/trigger_id/alert/alert_id",
    "arn:aws:macie:us-east-1:123456789012:trigger/trigger_id"
  ],
  "detail": {
    "notification-type": "ALERT_CREATED",
    "name": "Scanning bucket policies",
    "tags": [
      "Custom_Alert",
      "Insider"
    ],
    "url": "https://lb00.us-east-1.macie.aws.amazon.com/111122223333/posts/alert_id",
    "alert-arn": "arn:aws:macie:us-east-1:123456789012:trigger/trigger_id/alert/alert_id",
    "risk-score": 80,
    "trigger": {
      "rule-arn": "arn:aws:macie:us-east-1:123456789012:trigger/trigger_id",
      "alert-type": "basic",
      "created-at": "2017-01-02 19:54:00.644000",
      "description": "Alerting on failed enumeration of large number of bucket policies",
      "risk": 8
    },
    "created-at": "2017-04-18T00:21:12.059000",
```

```
    "actor": "555566667777:assumed-role:superawesome:aroaidpldc7nsesfnheji",
    "summary": {
      "Description": "Alerting on failed enumeration of large number of bucket policies",
      "IP": {
        "34.199.185.34": 121,
        "34.205.153.2": 2,
        "72.21.196.70": 2
      },
      "Time Range": [
        {
          "count": 125,
          "start": "2017-04-24T20:23:49Z",
          "end": "2017-04-24T20:25:54Z"
        }
      ],
      "Source ARN": "arn:aws:sts::123456789012:assumed-role/RoleName",
      "Record Count": 1,
      "Location": {
        "us-east-1": 125
      },
      "Event Count": 125,
      "Events": {
        "GetBucketLocation": {
          "count": 48,
          "ISP": {
            "Amazon": 48
          }
        },
        "ListRoles": {
          "count": 2,
          "ISP": {
            "Amazon": 2
          }
        },
        "GetBucketPolicy": {
          "count": 37,
          "ISP": {
            "Amazon": 37
          },
          "Error Code": {
            "NoSuchBucketPolicy": 22
          }
        },
        "GetBucketAcl": {
          "count": 37,
          "ISP": {
            "Amazon": 37
          }
        },
        "ListBuckets": {
          "count": 1,
          "ISP": {
            "Amazon": 1
          }
        }
      },
      "recipientAccountId": {
        "123456789012": 125
      }
    }
  }
}
```

```
{
```

```
  "version": "0",
  "id": "CWE-event-id",
  "detail-type": "Macie Alert",
  "source": "aws.macie",
  "account": "123456789012",
  "time": "2017-04-18T18:15:41Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:macie:us-east-1:123456789012:trigger/trigger_id/alert/alert_id",
    "arn:aws:macie:us-east-1:123456789012:trigger/trigger_id"
  ],
  "detail": {
    "notification-type": "ALERT_CREATED",
    "name": "Bucket is writable by all authenticated users",
    "tags": [
      "Custom_Alert",
      "Audit"
    ],
    "url": "https://lb00.us-east-1.macie.aws.amazon.com/111122223333/posts/alert_id",
    "alert-arn": "arn:aws:macie:us-east-1:123456789012:trigger/trigger_id/alert/alert_id",
    "risk-score": 70,
    "trigger": {
      "rule-arn": "arn:aws:macie:us-east-1:123456789012:trigger/trigger_id",
      "alert-type": "basic",
      "created-at": "2017-04-08 00:21:30.749000",
      "description": "Bucket is writable by all authenticated users",
      "risk": 7
    },
    "created-at": "2017-04-18T18:16:17.046454",
    "actor": "444455556666",
    "summary": {
      "Description": "Bucket is writable by all authenticated users",
      "Bucket": {
        "secret-bucket-name": 1
      },
      "Record Count": 1,
      "ACL": {
        "secret-bucket-name": [
          {
            "Owner": {
              "DisplayName": "bucket_owner",
              "ID": "089d2842f4b392f5c5c61f073bd2e4a37b3bb2e62659318c6960e8981648a17e"
            },
            "Grants": [
              {
                "Grantee": {
                  "Type": "Group",
                  "URI": "http://acs.amazonaws.com/groups/global/AuthenticatedUsers"
                },
                "Permission": "WRITE"
              }
            ]
          }
        ]
      },
      "Event Count": 1,
      "Timestamps": {
        "2017-01-10T22:48:06.784937": 1
      }
    }
  }
}
```


アラートの更新

```
{
  "version": "0",
  "id": "CWE-event-id",
  "detail-type": "Macie Alert",
  "source": "aws.macie",
  "account": "123456789012",
  "time": "2017-04-18T17:47:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:macie:us-east-1:123456789012:trigger/trigger_id/alert/alert_id",
    "arn:aws:macie:us-east-1:123456789012:trigger/trigger_id"
  ],
  "detail": {
    "notification-type": "ALERT_UPDATED",
    "name": "Public bucket contains high risk object",
    "tags": [
      "Custom_Alert",
      "Audit"
    ],
    "url": "https://lb00.us-east-1.macie.aws.amazon.com/111122223333/posts/alert_id",
    "alert-arn": "arn:aws:macie:us-east-1:123456789012:trigger/trigger_id/alert/alert_id",
    "risk-score": 100,
    "trigger": {
     "rule-arn": "arn:aws:macie:us-east-1:123456789012:trigger/trigger_id",
      "alert-type": "basic",
      "created-at": "2017-04-08 00:23:39.138000",
      "description": "Public bucket contains high risk object",
      "risk": 10
    },
    "created-at": "2017-04-08T00:36:26.270000",
    "actor": "public_bucket",
    "summary": {
      "Description": "Public bucket contains high risk object",
      "Object": {
        "public_bucket/secret_key.txt": 1,
        "public_bucket/financial_summary.txt": 1
      },
      "Record Count": 2,
      "Themes": {
        "Secret Markings": 1,
        "Corporate Proposals": 1,
        "Confidential Markings": 1
      },
      "Event Count": 2,
      "DLP risk": {
        "7": 2
      },
      "Owner": {
        "bucket_owner": 2
      },
      "Timestamps": {
        "2017-04-03T16:12:53+00:00": 2
      }
    }
  }
}
```



```
{
  "version": "0",
  "id": "CWE-event-id",
  "detail-type": "Macie Alert",
  "source": "aws.macie",
  "account": "123456789012",
  "time": "2017-04-22T03:31:47Z",
```

```
 "region": "us-east-1",
 "resources": [
   "arn:aws:macie:us-east-1:123456789012:trigger/macie/alert/alert_id",
   "arn:aws:macie:us-east-1:123456789012:trigger/macie"
 ],
 "detail": {
   "notification-type": "ALERT_UPDATED",
   "name": "Lists the instance profiles that have the specified associated IAM role, Lists
the names of the inline policies that are embedded in the specified IAM role",
   "tags": [
     "Predictive",
     "Behavioral_Anomaly"
   ],
   "url": "https://lb00.us-east-1.macie.aws.amazon.com/111122223333/posts/alert_id",
   "alert-arn": "arn:aws:macie:us-east-1:123456789012:trigger/macie/alert/alert_id",
   "risk-score": 20,
   "created-at": "2017-04-22T03:08:35.256000",
   "actor": "123456789012:assumed-role:rolename",
   "trigger": {
     "alert-type": "predictive",
     "features": {
       "distinctEventName": {
         "name": "distinctEventName",
         "description": "Event Names executed during a user session",
         "narrative": "A sudden increase in event names utilized by a user can be an
indicator of a change in user behavior or account risk",
         "risk": 3
       },
       "ListInstanceProfilesForRole": {
         "name": "ListInstanceProfilesForRole",
         "description": "Lists the instance profiles that have the specified associated
IAM role",
         "narrative": "Information collection activity suggesting the start of a
reconnaissance or exfiltration campaign",
         "anomalous": true,
         "multiplier": 8.420560747663552,
         "excession_times": [
           "2017-04-21T18:00:00Z"
         ],
         "risk": 1
       },
       "ListRolePolicies": {
         "name": "ListRolePolicies",
         "description": "Lists the names of the inline policies that are embedded in the
specified IAM role",
         "narrative": "Information collection activity suggesting the start of a
reconnaisance or exfiltration campaign",
         "anomalous": true,
         "multiplier": 12.017441860465116,
         "excession_times": [
           "2017-04-21T18:00:00Z"
         ],
         "risk": 2
       }
     }
   }
 }
}
```

Amazon Macie イベント

Amazon Macie によって生成されたイベントの例については、「Amazon Macie の結果のイベントスキーマ」を参照してください。

AWS マネジメントコンソール サインインイベント

AWS マネジメントコンソール サインインイベントは、米国東部 (バージニア北部) リージョンの CloudWatch イベント でのみ検出できます。

以下は、コンソールサインインイベントの例です。

```
{
  "id": "6f87d04b-9f74-4f04-a780-7acf4b0a9b38",
  "detail-type": "AWS Console Sign In via CloudTrail",
  "source": "aws.signin",
  "account": "123456789012",
  "time": "2016-01-05T18:21:27Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
      "eventVersion": "1.02",
      "userIdentity": {
            "type": "Root",
            "principalId": "123456789012",
            "arn": "arn:aws:iam::123456789012:root",
            "accountId": "123456789012"
            },
      "eventTime": "2016-01-05T18:21:27Z",
      "eventSource": "signin.amazonaws.com",
      "eventName": "ConsoleLogin",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "0.0.0.0",
      "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36",
      "requestParameters": null,
      "responseElements": {
            "ConsoleLogin": "Success"
            },
      "additionalEventData": {
            "LoginTo": "https://console.aws.amazon.com/console/home?state=hashArgs
%23&isauthcode=true",
            "MobileVersion": "No",
            "MFAUsed": "No" },
      "eventID": "324731c0-64b3-4421-b552-dfc3c27df4f6",
      "eventType": "AwsConsoleSignIn"
      }
}
```

AWS OpsWorks スタックイベント

以下は、AWS OpsWorks のスタックイベントの例です。

AWS OpsWorks スタックインスタンス状態の変更

AWS OpsWorks スタックインスタンス状態の変更を示します。インスタンスの状態は次のようになります。

- booting
- connection_lost
- online
- pending
- rebooting
- requested
- running_setup
- setup_failed
- shutting_down
- start_failed
- stopping
- stop_failed
- stopped
- terminating
- terminated

```
{
  "version": "0",
  "id": "dc5fa8df-48f1-2108-b1b9-1fe5ebcf2296",
  "detail-type": "OpsWorks Instance State Change",
  "source": "aws.opsworks",
  "account": "123456789012",
  "time": "2018-01-25T11:12:23Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:opsworks:us-east-1:123456789012:instance/a648d98f-fdd8-4323-952a-a50z3e4z500z"
  ],
  "detail": {
    "initiated_by": "user",
    "hostname": "testing1",
    "stack-id": "acd3df16-e859-4598-8414-377b12a902da",
    "layer-ids": [
      "d1a0cb7f-c7e9-4a63-811c-976f0267b2c8"
    ],
    "instance-id": "a648d98f-fdd8-4323-952a-a50z3e4z500z",
    "ec2-instance-id": "i-08b1c2b67aa292276",
    "status": "requested"
  }
}
```

initiated_by フィールドは、インスタンスが requested、terminating あるいは stopping 状態にある場合にのみ、値を表示します。initiated_by フィールドには次のいずれかの値が含まれます。

- user - ユーザーが API または AWS マネジメントコンソール を使用して、インスタンス状態の変更をリクエストしました。
- auto-scaling - AWS OpsWorks スタック自動スケーリング機能によって、インスタンス状態の変更が開始されました。
- auto-healing - AWS OpsWorks スタック自動ヒーリング機能によって、インスタンス状態の変更が開始されました。

AWS OpsWorks スタックコマンド状態の変更

AWS OpsWorks スタックコマンド状態に変更が生じました。コマンド状態は次のようになります。

- `expired` - コマンドタイムアウト。
- `failed` - 一般的なコマンドエラーが発生しました。
- `skipped`- AWS OpsWorks スタックと Amazon EC2 でインスタンスが異なる状態にあるため、コマンドはスキップされました。
- `successful` - コマンドは成功しました。
- `superseded` - すでに適用された設定変更を適用しようと試みたため、コマンドはスキップされました。

```
{
  "version": "0",
  "id": "96c778b6-a40e-c8c1-aafc-c9852a3a7b52",
  "detail-type": "OpsWorks Command State Change",
  "source": "aws.opsworks",
  "account": "123456789012",
  "time": "2018-01-26T08:54:40Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:opsworks:us-east-1:123456789012:instance/a648d98f-fdd8-4323-952a-a50a3e4e500f"
  ],
  "detail": {
    "command-id": "acc9f4f3-a3ec-4fab-b70f-c7d04e71e3ec",
    "instance-id": "a648d98f-fdd8-4323-952a-a50a3e4e500f",
    "type": "setup",
    "status": "successful"
  }
}
```

AWS OpsWorks スタックデプロイ状態の変更

AWS OpsWorks スタックデプロイ状態に変更が生じました。デプロイ状態は次のようになります。

- `running`
- `successful`
- `failed`

```
{
  "version": "0",
  "id": "b8230afa-60c7-f43f-b632-841c1cfb22ff",
  "detail-type": "OpsWorks Deployment State Change",
  "source": "aws.opsworks",
  "account": "123456789012",
  "time": "2018-01-25T11:15:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:opsworks:us-east-1:123456789012:instance/a648d98f-fdd8-4323-952a-a50a3e4e500f"
  ],
  "detail": {
    "duration": 16,
    "stack-id": "acd3df16-e859-4598-8414-377b12a902da",
    "instance-ids": [
      "a648d98f-fdd8-4323-952a-a50a3e4e500f"
    ],
    "deployment-id": "606419dc-418e-489c-8531-bff9770fc346",
    "command": "configure",
    "status": "successful"
  }
}
```

`duration` フィールドは、デプロイが終了したときにのみ秒数で時間を表示します。

AWS OpsWorks のスタックアラート

AWS OpsWorks スタックサービスエラーが発生しました。

```
{
  "version": "0",
  "id": "f99faa6f-0e27-e398-95bb-8f190806d275",
  "detail-type": "OpsWorks Alert",
  "source": "aws.opsworks",
  "account": "123456789012",
  "time": "2018-01-20T16:51:29Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "stack-id": "2f48f2be-ac7d-4dd5-80bb-88375f94db7b",
    "instance-id": "986efb74-69e8-4c6d-878e-5b77c054cbb0",
    "type": "InstanceStop",
    "message": "The shutdown of the instance timed out. Please try stopping it again."
  }
}
```

SageMaker イベント

SageMaker のイベントの例については、SageMaker 開発者ガイドの「Amazon EventBridge を使用した SageMaker の自動化」を参照してください。

AWS Security Hub イベント

セキュリティハブ サンプルイベントの詳細については、AWS Security Hub ユーザーガイドの「Amazon CloudWatch Events で AWS Security Hub をモニタリングする」を参照してください。

AWS Server Migration Service イベント

以下は、AWS Server Migration Service のイベントの例です。

[Deleted replication job notification] (レプリケーションジョブ通知の削除)

```
{
    "version": "0",
    "id": "5630992d-92cd-439f-f2a8-92c8212aee24",
    "detail-type": "Server Migration Job State Change",
    "source": "aws.sms",
    "account": "123456789012",
    "time": "2018-02-07T22:30:11Z",
    "region": "us-west-1",
    "resources": [
        "arn:aws:sms:us-west-1:123456789012:sms-job-21a64348"
    ],
    "detail": {
        "state": "Deleted",
        "replication-run-id": "N/A",
        "replication-job-id": "sms-job-21a64348",
        "version": "1.0"
```

```
    }
}
```

[Completed replication job notification] (レプリケーションジョブ通知の完了)

```
{
    "version": "0",
    "id": "3f9c59cc-f941-522a-be6d-f08e44ff1715",
    "detail-type": "Server Migration Job State Change",
    "source": "aws.sms",
    "account": "123456789012",
    "time": "2018-02-07T22:54:00Z",
    "region": "us-west-1",
    "resources": [
        "arn:aws:sms:us-west-1:123456789012:sms-job-2ea64347",
        "arn:aws:sms:us-west-1:123456789012:sms-job-2ea64347/sms-run-e1a64388"
    ],
    "detail": {
        "state": "Completed",
        "replication-run-id": "sms-run-e1a64388",
        "replication-job-id": "sms-job-2ea64347",
        "ami-id": "ami-746d6314",
        "version": "1.0"
    }
}
```

AWS Systems Manager イベント

以下は、AWS Systems Manager のイベントの例です。詳細については、AWS Systems Manager ユーザーガイド の「Amazon EventBridge を使用した Systems Manager のイベントのモニタリング」を参照してください。

Systems Manager のイベントタイプ

AWS Systems Manager Automation のイベント

Automation ステップステータス変更の通知

```
{
  "version": "0",
  "id": "eeca120b-a321-433e-9635-dab369006a6b",
  "detail-type": "EC2 Automation Step Status-change Notification",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2016-11-29T19:43:35Z",
  "region": "us-east-1",
  "resources": ["arn:aws:ssm:us-east-1:123456789012:automation-
execution/333ba70b-2333-48db-b17e-a5e69c6f4d1c",
    "arn:aws:ssm:us-east-1:123456789012:automation-definition/runcommand1:1"],
```

```
  "detail": {
    "ExecutionId": "333ba70b-2333-48db-b17e-a5e69c6f4d1c",
    "Definition": "runcommand1",
    "DefinitionVersion": 1.0,
    "Status": "Success",
    "EndTime": "Nov 29, 2016 7:43:25 PM",
    "StartTime": "Nov 29, 2016 7:43:23 PM",
    "Time": 2630.0,
    "StepName": "runFixedCmds",
    "Action": "aws:runCommand"
  }
}
```

Automation 実行ステータス変更の通知

```
{
  "version": "0",
  "id": "d290ece9-1088-4383-9df6-cd5b4ac42b99",
  "detail-type": "EC2 Automation Execution Status-change Notification",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2016-11-29T19:43:35Z",
  "region": "us-east-1",
  "resources": ["arn:aws:ssm:us-east-1:123456789012:automation-
execution/333ba70b-2333-48db-b17e-a5e69c6f4d1c",
    "arn:aws:ssm:us-east-1:123456789012:automation-definition/runcommand1:1"],
  "detail": {
    "ExecutionId": "333ba70b-2333-48db-b17e-a5e69c6f4d1c",
    "Definition": "runcommand1",
    "DefinitionVersion": 1.0,
    "Status": "Success",
    "StartTime": "Nov 29, 2016 7:43:20 PM",
    "EndTime": "Nov 29, 2016 7:43:26 PM",
    "Time": 5753.0,
    "ExecutedBy": "arn:aws:iam::123456789012:user/userName"
  }
}
```

AWS Systems Manager Compliance のイベント

以下は、AWS Systems Manager Compliance のイベントの例です。

関連付けの準拠

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Configuration Compliance State Change",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2017-07-17T19:03:26Z",
  "region": "us-west-1",
  "resources": [
    "arn:aws:ssm:us-west-1:461348341421:managed-instance/i-01234567890abcdef"
  ],
  "detail": {
    "last-runtime": "2017-01-01T10:10:10Z",
    "compliance-status": "compliant",
    "resource-type": "managed-instance",
    "resource-id": "i-01234567890abcdef",
    "compliance-type": "Association"
  }
```

```
}
```

関連付けの非準拠

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Configuration Compliance State Change",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2017-07-17T19:02:31Z",
  "region": "us-west-1",
  "resources": [
    "arn:aws:ssm:us-west-1:461348341421:managed-instance/i-01234567890abcdef"
  ],
  "detail": {
    "last-runtime": "2017-01-01T10:10:10Z",
    "compliance-status": "non_compliant",
    "resource-type": "managed-instance",
    "resource-id": "i-01234567890abcdef",
    "compliance-type": "Association"
  }
}
```

パッチの準拠

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Configuration Compliance State Change",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2017-07-17T19:03:26Z",
  "region": "us-west-1",
  "resources": [
    "arn:aws:ssm:us-west-1:461348341421:managed-instance/i-01234567890abcdef"
  ],
  "detail": {
    "resource-type": "managed-instance",
    "resource-id": "i-01234567890abcdef",
    "compliance-status": "compliant",
    "compliance-type": "Patch",
    "patch-baseline-id": "PB789",
    "severity": "critical"
  }
}
```

パッチの非準拠

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "Configuration Compliance State Change",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2017-07-17T19:02:31Z",
  "region": "us-west-1",
  "resources": [
    "arn:aws:ssm:us-west-1:461348341421:managed-instance/i-01234567890abcdef"
  ],
  "detail": {
    "resource-type": "managed-instance",
```

```
    "resource-id": "i-01234567890abcdef",
    "compliance-status": "non_compliant",
    "compliance-type": "Patch",
    "patch-baseline-id": "PB789",
    "severity": "critical"
  }
}
```

AWS Systems Manager メンテナンスウィンドウのイベント

以下は、Systems Manager メンテナンスウィンドウのイベントの例です。

ターゲットの登録

その他の有効なステータス値は `DEREGISTERED` です。

```
{
   "version":"0",
   "id":"01234567-0123-0123-0123-0123456789ab",
   "detail-type":"Maintenance Window Target Registration Notification",
   "source":"aws.ssm",
   "account":"012345678901",
   "time":"2016-11-16T00:58:37Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:ssm:us-west-2:001312665065:maintenancewindow/mw-0ed7251d3fcf6e0c2",
      "arn:aws:ssm:us-west-2:001312665065:windowtarget/
e7265f13-3cc5-4f2f-97a9-7d3ca86c32a6"
   ],
   "detail":{
      "window-target-id":"e7265f13-3cc5-4f2f-97a9-7d3ca86c32a6",
      "window-id":"mw-0ed7251d3fcf6e0c2",
      "status":"REGISTERED"
   }
}
```

ウィンドウの実行タイプ

その他の有効なステータス値は
`PENDING`、`IN_PROGRESS`、`SUCCESS`、`FAILED`、`TIMED_OUT`、`SKIPPED_OVERLAPPING` です。

```
{
   "version":"0",
   "id":"01234567-0123-0123-0123-0123456789ab",
   "detail-type":"Maintenance Window Execution State-change Notification",
   "source":"aws.ssm",
   "account":"012345678901",
   "time":"2016-11-16T01:00:57Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:ssm:us-west-2:0123456789ab:maintenancewindow/mw-123456789012345678"
   ],
   "detail":{
      "start-time":"2016-11-16T01:00:56.427Z",
      "end-time":"2016-11-16T01:00:57.070Z",
      "window-id":"mw-0ed7251d3fcf6e0c2",
      "window-execution-id":"b60fb56e-776c-4e5c-84ee-123456789012",
      "status":"TIMED_OUT"
   }
```

```
}
```

タスクの実行タイプ

その他の有効なステータス値は IN_PROGRESS、SUCCESS、FAILED、TIMED_OUT です。

```
{
   "version":"0",
   "id":"01234567-0123-0123-0123-0123456789ab",
   "detail-type":"Maintenance Window Task Execution State-change Notification",
   "source":"aws.ssm",
   "account":"012345678901",
   "time":"2016-11-16T01:00:56Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:ssm:us-west-2:0123456789ab:maintenancewindow/mw-123456789012345678"
   ],
   "detail":{
      "start-time":"2016-11-16T01:00:56.759Z",
      "task-execution-id":"6417e808-7f35-4d1a-843f-123456789012",
      "end-time":"2016-11-16T01:00:56.847Z",
      "window-id":"mw-0ed7251d3fcf6e0c2",
      "window-execution-id":"b60fb56e-776c-4e5c-84ee-123456789012",
      "status":"TIMED_OUT"
   }
}
```

処理されるタスクのターゲット

その他の有効なステータス値は IN_PROGRESS、SUCCESS、FAILED、TIMED_OUT です。

```
{
   "version":"0",
   "id":"01234567-0123-0123-0123-0123456789ab",
   "detail-type":"Maintenance Window Task Target Invocation State-change Notification",
   "source":"aws.ssm",
   "account":"012345678901",
   "time":"2016-11-16T01:00:57Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:ssm:us-west-2:0123456789ab:maintenancewindow/mw-123456789012345678"
   ],
   "detail":{
      "start-time":"2016-11-16T01:00:56.427Z",
      "end-time":"2016-11-16T01:00:57.070Z",
      "window-id":"mw-0ed7251d3fcf6e0c2",
      "window-execution-id":"b60fb56e-776c-4e5c-84ee-123456789012",
      "task-execution-id":"6417e808-7f35-4d1a-843f-123456789012",
      "window-target-id":"e7265f13-3cc5-4f2f-97a9-123456789012",
      "status":"TIMED_OUT",
      "owner-information":"Owner"
   }
}
```

ウィンドウの状態の変更

有効なステータス値は ENABLED および DISABLED です。

```
{
   "version":"0",
   "id":"01234567-0123-0123-0123-0123456789ab",
   "detail-type":"Maintenance Window State-change Notification",
```

```
   "source":"aws.ssm",
   "account":"012345678901",
   "time":"2016-11-16T00:58:37Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:ssm:us-west-2:0123456789ab:maintenancewindow/mw-123456789012345678"
   ],
   "detail":{
      "window-id":"mw-123456789012",
      "status":"DISABLED"
   }
}
```

AWS Systems Manager パラメータストアのイベント

以下は、Systems Manager パラメータストアのイベントの例です。

パラメータの作成

```
{
  "version": "0",
  "id": "6a7e4feb-b491-4cf7-a9f1-bf3703497718",
  "detail-type": "Parameter Store Change",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2017-05-22T16:43:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ssm:us-east-1:123456789012:parameter/foo"
  ],
  "detail": {
    "operation": "Create",
    "name": "foo",
    "type": "String",
    "description": "Sample Parameter"
  }
}
```

パラメータの更新

```
{
  "version": "0",
  "id": "9547ef2d-3b7e-4057-b6cb-5fdf09ee7c8f",
  "detail-type": "Parameter Store Change",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2017-05-22T16:44:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ssm:us-east-1:123456789012:parameter/foo"
  ],
  "detail": {
    "operation": "Update",
    "name": "foo",
    "type": "String",
    "description": "Sample Parameter"
  }
}
```

パラメータの削除

```
{
  "version": "0",
  "id": "80e9b391-6a9b-413c-839a-453b528053af",
  "detail-type": "Parameter Store Change",
  "source": "aws.ssm",
  "account": "123456789012",
  "time": "2017-05-22T16:45:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ssm:us-east-1:123456789012:parameter/foo"
  ],
  "detail": {
    "operation": "Delete",
    "name": "foo",
    "type": "String",
    "description": "Sample Parameter"
  }
}
```

AWS Systems Manager Run Command のイベント

Run Command ステータス変更の通知

```
{
    "version": "0",
    "id": "51c0891d-0e34-45b1-83d6-95db273d1602",
    "detail-type": "EC2 Command Status-change Notification",
    "source": "aws.ssm",
    "account": "123456789012",
    "time": "2016-07-10T21:51:32Z",
    "region": "us-east-1",
    "resources": ["arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1111"],
    "detail": {
        "command-id": "e8d3c0e4-71f7-4491-898f-c9b35bee5f3b",
        "document-name": "AWS-RunPowerShellScript",
        "expire-after": "2016-07-14T22:01:30.049Z",
        "parameters": {
            "executionTimeout": ["3600"],
            "commands": ["date"]
        },
        "requested-date-time": "2016-07-10T21:51:30.049Z",
        "status": "Success"
    }
}
```

Run Command 呼び出しステータス変更の通知

```
{
    "version": "0",
    "id": "4780e1b8-f56b-4de5-95f2-95db273d1602",
    "detail-type": "EC2 Command Invocation Status-change Notification",
    "source": "aws.ssm",
    "account": "123456789012",
    "time": "2016-07-10T21:51:32Z",
    "region": "us-east-1",
    "resources": ["arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1111"],
    "detail": {
        "command-id": "e8d3c0e4-71f7-4491-898f-c9b35bee5f3b",
        "document-name": "AWS-RunPowerShellScript",
        "instance-id": "i-9bb89e2b",
        "requested-date-time": "2016-07-10T21:51:30.049Z",
        "status": "Success"
```

```
    }
}
```

AWS Systems Manager State Manager のイベント

ステートマネージャーの関連付け状態の変更

```
{
   "version":"0",
   "id":"db839caf-6f6c-40af-9a48-25b2ae2b7774",
   "detail-type":"EC2 State Manager Association State Change",
   "source":"aws.ssm",
   "account":"123456789012",
   "time":"2017-05-16T23:01:10Z",
   "region":"us-west-1",
   "resources":[
      "arn:aws:ssm:us-west-1::document/AWS-RunPowerShellScript"
   ],
   "detail":{
      "association-id":"6e37940a-23ba-4ab0-9b96-5d0a1a05464f",
      "document-name":"AWS-RunPowerShellScript",
      "association-version":"1",
      "document-version":"Optional.empty",
      "targets":"[{\"key\":\"InstanceIds\",\"values\":[\"i-12345678\"]}]",
      "creation-date":"2017-02-13T17:22:54.458Z",
      "last-successful-execution-date":"2017-05-16T23:00:01Z",
      "last-execution-date":"2017-05-16T23:00:01Z",
      "last-updated-date":"2017-02-13T17:22:54.458Z",
      "status":"Success",
      "association-status-aggregated-count":"{\"Success\":1}",
      "schedule-expression":"cron(0 */30 * * * ? *)",
      "association-cwe-version":"1.0"
   }
}
```

ステートマネージャーインスタンスの関連付け状態の変更

```
{
   "version":"0",
   "id":"6a7e8feb-b491-4cf7-a9f1-bf3703467718",
   "detail-type":"EC2 State Manager Instance Association State Change",
   "source":"aws.ssm",
   "account":"123456789012",
   "time":"2017-02-23T15:23:48Z",
   "region":"us-east-1",
   "resources":[
      "arn:aws:ec2:us-east-1:123456789012:instance/i-12345678",
      "arn:aws:ssm:us-east-1:123456789012:document/my-custom-document"
   ],
   "detail":{
      "association-id":"34fcb7e0-9a14-4984-9989-0e04e3f60bd8",
      "instance-id":"i-12345678",
      "document-name":"my-custom-document",
      "document-version":"1",
      "targets":"[{\"key\":\"instanceids\",\"values\":[\"i-12345678\"]}]",
      "creation-date":"2017-02-23T15:23:48Z",
      "last-successful-execution-date":"2017-02-23T16:23:48Z",
      "last-execution-date":"2017-02-23T16:23:48Z",
      "status":"Success",
      "detailed-status":"",
      "error-code":"testErrorCode",
      "execution-summary":"testExecutionSummary",
```

```
      "output-url":"sampleurl",
      "instance-association-cwe-version":"1"
   }
}
```

AWS Step Functions イベント

Step Functions のサンプルイベントについては、AWS Step Functions 開発者ガイドの「Step Functions イベントの例」を参照してください。

AWS リソースのタグ変更イベント

タグイベントの例を次に示します。

```
{
    "version": "0",
    "id": "ffd8a6fe-32f8-ef66-c85c-111111111111",
    "detail-type": "Tag Change on Resource",
    "source": "aws.tag",
    "account": "123456789012",
    "time": "2018-09-18T20:41:06Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:ec2:us-east-1:123456789012:instance/i-0000000aaaaaaaaaa"
    ],
    "detail": {
        "changed-tag-keys": [
            "key2",
            "key3"
        ],
        "service": "ec2",
        "resource-type": "instance",
        "version": 5,
        "tags": {
            "key4": "value4",
            "key1": "value1",
            "key2": "value2"
        }
    }
}
```

AWS Trusted Advisor イベント

以下は、AWS Trusted Advisor のイベントの例です。詳細については、『AWS サポート User Guide』の「Amazon CloudWatch Events を使用した Trusted Advisor のチェック結果のモニタリング」を参照してください。

低稼働率の Amazon EC2 インスタンス

```
{
  "version": "0",
  "id": "1234abcd-ab12-123a-123a-1234567890ab",
  "detail-type": "Trusted Advisor Check Item Refresh Notification",
  "source": "aws.trustedadvisor",
  "account": "123456789012",
```

```
  "time": "2018-01-12T20:07:49Z",
  "region": "us-east-2",
  "resources": [],
  "detail": {
    "check-name": "Low Utilization Amazon EC2 Instances",
    "check-item-detail": {
      "Day 1": "0.1%  0.00MB",
      "Day 2": "0.1%  0.00MB",
      "Day 3": "0.1%  0.00MB",
      "Region/AZ": "ca-central-1a",
      "Estimated Monthly Savings": "$9.22",
      "14-Day Average CPU Utilization": "0.1%",
      "Day 14": "0.1%  0.00MB",
      "Day 13": "0.1%  0.00MB",
      "Day 12": "0.1%  0.00MB",
      "Day 11": "0.1%  0.00MB",
      "Day 10": "0.1%  0.00MB",
      "14-Day Average Network I/O": "0.00MB",
      "Number of Days Low Utilization": "14 days",
      "Instance Type": "t2.micro",
      "Instance ID": "i-01234567890abcdef",
      "Day 8": "0.1%  0.00MB",
      "Instance Name": null,
      "Day 9": "0.1%  0.00MB",
      "Day 4": "0.1%  0.00MB",
      "Day 5": "0.1%  0.00MB",
      "Day 6": "0.1%  0.00MB",
      "Day 7": "0.1%  0.00MB"
    },
    "status": "WARN",
    "resource_id": "arn:aws:ec2:ca-central-1:123456789012:instance/i-01234567890abcdef",
    "uuid": "aa12345f-55c7-498e-b7ac-123456789012"
  }
}
```

ロードバランサーの最適化

```
{
  "version": "0",
  "id": "1234abcd-ab12-123a-123a-1234567890ab",
  "detail-type": "Trusted Advisor Check Item Refresh Notification",
  "source": "aws.trustedadvisor",
  "account": "123456789012",
  "time": "2018-01-12T20:07:03Z",
  "region": "us-east-2",
  "resources": [],
  "detail": {
    "check-name": "Load Balancer Optimization ",
    "check-item-detail": {
      "Instances in Zone a": "1",
      "Status": "Yellow",
      "Instances in Zone b": "0",
      "# of Zones": "2",
      "Region": "eu-central-1",
      "Load Balancer Name": "my-load-balance",
      "Instances in Zone e": null,
      "Instances in Zone c": null,
      "Reason": "Single AZ",
      "Instances in Zone d": null
    },
    "status": "WARN",
    "resource_id": "arn:aws:elasticloadbalancing:eu-central-1:123456789012:loadbalancer/my-
load-balancer",
    "uuid": "aa12345f-55c7-498e-b7ac-123456789012"
  }
```

```
}
```

公開されたアクセスキー

```
{
  "version": "0",
  "id": "1234abcd-ab12-123a-123a-1234567890ab",
  "detail-type": "Trusted Advisor Check Item Refresh Notification",
  "source": "aws.trustedadvisor",
  "account": "123456789012",
  "time": "2018-01-12T19:38:24Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "check-name": "Exposed Access Keys",
    "check-item-detail": {
      "Case ID": "12345678-1234-1234-abcd-1234567890ab",
      "Usage (USD per Day)": "0",
      "User Name (IAM or Root)": "my-username",
      "Deadline": "1440453299248",
      "Access Key ID": "AKIAIOSFODNN7EXAMPLE",
      "Time Updated": "1440021299248",
      "Fraud Type": "Exposed",
      "Location": "www.example.com"
    },
    "status": "ERROR",
    "resource_id": "",
    "uuid": "aa12345f-55c7-498e-b7ac-123456789012"
  }
}
```

Amazon WorkSpaces イベント

Amazon WorkSpaces イベントの詳細については、『Amazon WorkSpaces Administration Guide』の「CloudWatch イベントを使用して WorkSpaces をモニタリングする」を参照してください。

CloudTrail 経由で配信されたイベント

イベントを出力せず、このページのリストに含まれていないサービスで CloudWatch イベント を使用することもできます。AWS CloudTrail は、AWS API コールなどのイベントを自動的に記録するサービスです。CloudTrail によってキャプチャされた情報でトリガーする CloudWatch イベント ルールを作成することができます。CloudTrail の詳細については、「AWS CloudTrail とは」を参照してください。CloudTrail を使用する CloudWatch イベント ルールの作成の詳細については、「AWS CloudTrail を使用して AWS API コールでトリガーする CloudWatch イベント ルールの作成 (p. 7)」を参照してください。

CloudTrail 経由で送信されたイベントはすべて、`detail-type` の値が `AWS API Call via CloudTrail` になっています。

AWS で発生した内容は、サービス自体と CloudTrail の両方で CloudWatch イベント にレポートできますが、その方法は異なります。たとえば、インスタンスを起動または終了する Amazon EC2 API コールでは、CloudTrail を通じて CloudWatch イベント で使用できるイベントが生成されます。ただし、Amazon EC2 インスタンスの状態が変更されると (例: 「実行中」から「終了中」)、CloudWatch イベント イベント自体が生成されます。

CloudTrail より送信されるイベントの例を以下に示します。イベントは、バケットを作成するために Amazon S3 への AWS API コールを行うことによって生成されました。

```
{
    "version": "0",
    "id": "36eb8523-97d0-4518-b33d-ee3579ff19f0",
    "detail-type": "AWS API Call via CloudTrail",
    "source": "aws.s3",
    "account": "123456789012",
    "time": "2016-02-20T01:09:13Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "Root",
            "principalId": "123456789012",
            "arn": "arn:aws:iam::123456789012:root",
            "accountId": "123456789012",
            "sessionContext": {
                "attributes": {
                    "mfaAuthenticated": "false",
                    "creationDate": "2016-02-20T01:05:59Z"
                }
            }
        },
        "eventTime": "2016-02-20T01:09:13Z",
        "eventSource": "s3.amazonaws.com",
        "eventName": "CreateBucket",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "100.100.100.100",
        "userAgent": "[S3Console/0.4]",
        "requestParameters": {
            "bucketName": "bucket-test-iad"
        },
        "responseElements": null,
        "requestID": "9D767BCC3B4E7487",
        "eventID": "24ba271e-d595-4e66-a7fd-9c16cbf8abae",
        "eventType": "AwsApiCall"
    }
}
```

256 KB よりも大きい AWS API 呼び出しイベントはサポートされていません。ルールのトリガーとして使用できる API コールの詳細については、「CloudTrail イベント履歴でサポートされるサービス」を参照してください。

AWS アカウント間のイベントの送受信

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

他の AWS アカウントとの間でイベントを送受信するように AWS アカウントを設定できます。これは、アカウントが同じ組織に属している、パートナーである組織に属している、または同様の関係を持っている場合に役立ちます。

イベントを送受信するようにアカウントを設定する場合は、イベントの送信先または受信元の AWS アカウントを指定できます。AWS Organizations 機能を使用する場合は、組織を指定し、その組織内のすべてのアカウントにアクセス許可を付与できます。詳細については、AWS Organizations ユーザーガイド の「AWS Organizations とは」を参照してください。

全体の手順は次のとおりです。

- 受信側アカウントで、指定した AWS アカウント、組織、またはすべての AWS アカウントが受取側アカウントにイベントを送信できるように、デフォルトのイベントバスに対するアクセス許可を編集します。
- 送信側アカウントで、受信側アカウントのデフォルトのイベントバスをターゲットとする 1 つ以上のルールを設定します。

  送信側アカウントが、アクセス許可を持つ AWS 組織に属するため、イベントを送信するアクセス許可を持っている場合、送信側アカウントには、受信側アカウントにイベントを送信するためのポリシーが適用された IAM ロールも必要です。AWS マネジメントコンソール を使用して受信側アカウントをターゲットとするルールを作成する場合、このロールは自動的に作成されます。AWS CLI を使用する場合は、このロールを手動で作成する必要があります。
- 受信側アカウントで、送信側アカウントからのイベントに一致する 1 つ以上のルールを設定します。

受信側アカウントがデフォルトのイベントバスにアクセス権限を追加する AWS リージョンは、送信側アカウントが受信側アカウントにイベントを送信するためのルールを作成するリージョンと同じである必要があります。

1 つのアカウントから別のアカウントに送信されたイベントは、カスタムイベントとして送信側アカウントに課金されます。受信側アカウントには課金されません。詳細については、Amazon CloudWatch 料金表をご覧ください。

受信側アカウントで、送信側アカウントから受信したイベントを第三のアカウントに送信するルールが設定されていても、それらのイベントは第三のアカウントには送信されません。

AWS アカウントで他の AWS アカウントからイベントを受信できるようにする

他のアカウントまたは組織からイベントを受信するには、まずアカウントのデフォルトのイベントバスに対するアクセス許可を編集する必要があります。デフォルトのイベントバスは、AWS のサービス、他の承認された AWS アカウント、および `PutEvents` コールからイベントを受け入れることができます。

他の AWS アカウントにアクセス許可を付与するように、デフォルトのイベントバスに対するアクセス許可を編集するときは、アカウント ID または組織 ID でアカウントを指定できます。または、すべての AWS アカウントからイベントを受信する選択ができます。

> Warning
>
> すべての AWS アカウントからイベントを受信することを選択した場合、他のアカウントから受信するイベントにのみ一致するルールを慎重に作成してください。より安全なルールを作成するには、各ルールのイベントパターンに、`Account` フィールドと、イベントの受信元の 1 つ以上のアカウントのアカウント ID が必ず含まれるようにします。アカウントフィールドを含むイベントパターンを持つルールは、`Account` フィールドにリストされていないアカウントから送信されたイベントと一致しません。詳細については、「CloudWatch イベント のイベントパターン (p. 38)」を参照してください。

コンソールを使用して、アカウントで他の AWS アカウントからイベントを受信できるようにするには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[イベントバス]、[アクセス許可の追加] を選択します。
3. [AWS アカウント] または [Organization (組織)] を選択します。

   [AWS アカウント] を選択した場合は、イベントの受信元 AWS アカウントの 12 桁のアカウント ID を入力します。他のすべての AWS アカウントからイベントを受信するには、[全員(*)] を選択します。

   [Organization (組織)] を選択した場合は、[自分の組織] を選択して、現在のアカウントが属する組織内のすべてのアカウントにアクセス許可を付与します。または、[Another organization (別の組織)] を選択し、その組織の組織 ID を入力します。組織 ID を入力するときは、o- プレフィックスを含める必要があります。
4. [追加] を選択します。
5. これらの手順を繰り返して、他のアカウントや組織を追加できます。

AWS CLI を使用して、アカウントで他の AWS アカウントからイベントを受信するには

1. 1 つの特定の AWS アカウントでイベントの送信を有効にするには、次のコマンドを実行します。

```
aws events put-permission --action events:PutEvents --statement-id MySid --
principal SenderAccountID
```

   AWS 組織がイベントを送信できるようにするには、以下のコマンドを実行します。

```
aws events put-permission --action events:PutEvents --statement-id MySid
                  --principal \* --condition '{"Type" : "StringEquals", "Key":
 "aws:PrincipalOrgID", "Value": "SenderOrganizationID"}'
```

   他のすべての AWS アカウントでイベントを送信を有効にするには、次のコマンドを実行します。

```
aws events put-permission --action events:PutEvents --statement-id MySid --principal \*
```

`aws events put-permission` を複数回実行して、個々の AWS アカウントと組織の両方にアクセス許可を付与できますが、1 つのコマンドで個々のアカウントと組織の両方を指定することはできません。

2. デフォルトのイベントバスに対するアクセス許可を設定したら、オプションで `describe-event-bus` コマンドを使用してそれらのアクセス許可を確認できます。

```
aws events describe-event-bus
```

別の AWS アカウントへのイベントの送信

別のアカウントにイベントを送信するには、別の AWS アカウントのデフォルトのイベントバスをターゲットとする CloudWatch イベント ルールを設定できます。その受信側アカウントのデフォルトのイベントバスも、アカウントからイベントを受信するように設定する必要があります。

コンソールを使用してアカウントから別の AWS アカウントにイベントを送信するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[イベント]、[ルールの作成] の順に選択します。
3. [イベントソース] で、[イベントパターン] を選択し、他のアカウントに送信するサービス名とイベントタイプを選択します。
4. [ターゲットの追加] を選択します。
5. [ターゲット] で、[別の AWS アカウントのイベントバス] を選択します。[アカウント ID] に、イベントの送信先 AWS アカウントの 12 桁のアカウント ID を入力します。
6. 受信側アカウントが組織全体にアクセス許可を付与しているため、この送信側アカウントがイベントを送信するアクセス許可を持っている場合は、IAM ロールが必要です。

   - 自動的に IAM ロールを作成するには、[この特定のリソースに対して新しいロールを作成する] を選択します。
   - または、[既存のロールの使用] を選択します。ビルドを呼び出すのに十分なアクセス許可をすでに持つロールを選択します。CloudWatch イベント は、選択したロールに追加のアクセス許可を付与しません。
7. ページの下部で、[設定の詳細] を選択します。
8. ルールの名前と説明を入力し、[ルールの作成] を選択します。

AWS CLI を使用して別の AWS アカウントにイベントを送信するには

1. 送信側アカウントが、受信側アカウントによってアクセス許可を付与される先の AWS 組織に属するため、イベントを送信するアクセス許可を持っている場合、送信側アカウントには、受信側アカウントにイベントを送信するためのポリシーが適用されたロールも必要です。この手順では、そのロールを作成する方法について説明します。

   送信者アカウントに、組織ではなく AWS アカウント ID を通じてイベントを送信するアクセス許可が付与されている場合、この手順はオプションです。ステップ 2 に進めます。

   a. 送信側アカウントに組織を通じてアクセス許可が付与されている場合は、必要な IAM ロールを作成します。まず、以下の内容で `assume-role-policy-document.json` というファイルを作成します。

```
{
```

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

b. ロールを作成するには、以下のコマンドを入力します。

```
$ aws iam create-role \
--profile sender \
--role-name event-delivery-role \
--assume-role-policy-document file://assume-role-policy-document.json
```

c. 以下の内容で permission-policy.json という名前のファイルを作成します。

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

d. このポリシーをロールにアタッチするには、以下のコマンドを入力します。

```
$ aws iam put-role-policy \
--profile sender \
--role-name event-delivery-role \
--policy-name EventBusDeliveryRolePolicy
--policy-document file://permission-policy.json
```

2. put-rule コマンドを使用して、他のアカウントに送信するイベントタイプに一致するルールを作成します。
3. 他のアカウントのデフォルトのイベントバスを、ルールのターゲットとして追加します。

   送信側アカウントに、アカウント ID を通じてイベントを送信するアクセス許可が付与されている場合、ロールを指定する必要はありません。次のコマンドを実行します。

```
aws events put-targets --rule NameOfRuleMatchingEventsToSend --targets
 "Id"="MyId","Arn"="arn:aws:events:region:$ReceiverAccountID:event-bus/default"
```

   送信側アカウントに組織を通じてイベントを送信するアクセス許可が付与されている場合は、以下の例のようにロールを指定します。

```
aws events put-targets --rule NameOfRuleMatchingEventsToSend --targets
 "Id"="MyId","Arn"="arn:aws:events:region:$ReceiverAccountID:event-bus/
default","RoleArn"="arn:aws:iam:${sender_account_id}:role/event-delivery-role"
```

別の AWS アカウントからのイベントに一致するルールの作成

アカウントが他の AWS アカウントからイベントを受信するように設定されている場合、それらのイベントに一致するルールを作成できます。他のアカウントから受信するイベントに一致するルールのイベントパターンを設定します。

ルールのイベントパターンで `account` を指定した場合を除き、アカウントのルール (新規と既存の両方) のうち、他のアカウントから受信したイベントに一致するすべてのルールがトリガーされます。別のアカウントからイベントを受信し、自分のアカウントから生成されたときにそのイベントパターンのみでルールがトリガーされるようにするには、`account` を追加し、自分のアカウント ID をルールのイベントパターンに指定する必要があります。

すべての AWS アカウントからイベントを受け入れるように AWS アカウントを設定する場合、アカウントの各 CloudWatch イベント ルールに `account` を追加することを強くお勧めします。これにより、アカウントのルールで、不明な AWS アカウントからのイベントによるトリガーの発生を防止できます。ルールで `account` フィールドを指定するときは、1 つ以上の AWS アカウントのアカウント ID をフィールドで指定できます。

アクセス許可を付与した AWS アカウントの一致するイベントに対してルールをトリガーするには、ルールの [account] フィールドに * を指定しないでください。これにより、* はイベントの `account` フィールドに表示されないため、どのイベントとも一致しません。代わりに、ルールから `account` フィールドを省略します。

コンソールを使用して、別のアカウントからのイベントに一致するルールを作成するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[イベント]、[ルールの作成] の順に選択します。
3. [イベントソース] で、[イベントパターン] を選択し、ルールで一致するサービス名とイベントタイプを選択します。
4. [イベントパターンのプレビュー] の近くにある [編集] を選択します。
5. 編集ウィンドウで、このイベントを送信するどの AWS アカウントがルールと一致するかを指定する `Account` 行を追加します。たとえば、編集ウィンドウの元の表示が次のようであるとします。

```
{
  "source": [
    "aws.ec2"
  ],
  "detail-type": [
    "EBS Volume Notification"
  ]
}
```

EBS ボリューム通知を送信する AWS アカウント 123456789012 および 111122223333 がルールと一致するようにするには、以下の行を追加します。

```
{
  "account": [
    "123456789012","111122223333"
  ],
  "source": [
    "aws.ec2"
  ],
  "detail-type": [
    "EBS Volume Notification"
  ]
```

```
}
```

6. イベントパターンを編集し、[保存] を選択します。
7. 通常どおりルールの作成を終了し、アカウントで 1 つ以上のターゲットを設定します。

AWS CLI を使用して、別の AWS アカウントからのイベントに一致するルールを作成するには

- `put-rule` コマンドを実行します。ルールのイベントパターンの `Account` フィールドに、ルールに一致するようにする他の AWS アカウントを指定します。次の例では、ルールは AWS アカウント 123456789012 および 111122223333 の Amazon EC2 インスタンスの状態変更に一致します。

```
aws events put-rule --name "EC2InstanceStateChanges" --event-pattern "{\"account\":
["123456789012", "111122223333"],\"source\":[\"aws.ec2\"],\"detail-type\":[\"EC2
 Instance State-change Notification\"]}"  --role-arn "arn:aws:iam::123456789012:role/
MyRoleForThisRule"
```

送信側と受信側の関係を移行して AWS Organizations を使用する

送信側アカウントのアカウント ID にアクセス許可が直接付与されている場合、これらのアクセス許可を取り消して (組織にアクセス許可を付与することで) 送信側アカウントにアクセス権を与える場合は、追加のステップを実行する必要があります。これらのステップでは、送信側アカウントからのイベントが依然として受信側アカウントに到達できることを確認します。これは、組織を介してイベントを送信するアクセス許可を持つアカウントは、イベントを送信するために IAM ロールも使用する必要があるためです。

送信側と受信側の関係を移行するために必要なアクセス許可を追加するには

1. 送信側アカウントで、IAM ロールを作成します。このロールには、受信側アカウントにイベントを送信することを許可するポリシーを割り当てます。

   a. 以下の内容で `assume-role-policy-document.json` という名前のファイルを作成します。

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

   b. IAM ロールを作成するには、次のコマンドを入力します。

```
$ aws iam create-role \
--profile sender \
--role-name event-delivery-role \
--assume-role-policy-document file://assume-role-policy-document.json
```

   c. 以下の内容で `permission-policy.json` という名前のファイルを作成します。

```
{
```

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

d. このポリシーをロールにアタッチするには、以下のコマンドを入力します。

```
$ aws iam put-role-policy \
--profile sender \
--role-name event-delivery-role \
--policy-name EventBusDeliveryRolePolicy
--policy-document file://permission-policy.json
```

2. 送信側アカウントの既存のルールのうち、受信側アカウントのデフォルトのイベントバスをターゲットとする各ルールを編集します。ステップ 1 で作成したロールをターゲット情報に追加することで、ルールを編集します。次のコマンドを使用します。

```
aws events put-targets --rule Rulename --targets
 "Id"="MyID","Arn"="arn:aws:events:region:$ReceiverAccountID:event-bus/
default","RoleArn"="arn:aws:iam:${sender_account_id}:role/event-delivery-role"
```

3. 受信側アカウントで、次のコマンドを実行します。これにより、受信側アカウントにイベントを送信するアクセス許可を組織のアカウントに付与します。

```
aws events put-permission --action events:PutEvents --statement-id Sid-For-Organization
 --principal \* --condition '{"Type" : "StringEquals", "Key": "aws:PrincipalOrgID",
 "Value": "SenderOrganizationID"}'
```

必要に応じて、最初に送信側アカウントに直接付与されたアクセス許可を取り消すこともできます。

```
aws events remove-permission --statement-id Sid-for-SenderAccount
```

PutEvents を使用したイベントの追加

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

`PutEvents` アクションは、1 つのリクエストで CloudWatch イベント に複数のイベントを送ります。詳細については、『Amazon CloudWatch Events API リファレンス』の「PutEvents」および『AWS CLI Command Reference』の「put-events」を参照してください。

各 `PutEvents` リクエストでサポートされているエントリの数は制限されています。詳細については、「CloudWatch イベント のクォータ (p. 110)」を参照してください。`PutEvents` オペレーションでは、リクエストの自然な順序ですべてのエントリを処理するように試みます。各イベントには、`PutEvents` を呼び出した後に CloudWatch イベント によって固有の ID が割り当てられます。

以下の Java コード例は CloudWatch イベント に 2 つの同一のイベントを送っています。

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

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

PutEventsResult result = awsEventsClient.putEvents(request);

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

`PutEvents` の結果には、応答配列のエントリが含まれます。応答配列の各エントリは、リクエスト配列のエントリと自然な順序 (リクエストや応答の上から下へ) で直接相互に関連付けられます。応答 `Entries` 配列には、常にリクエスト配列と同じ数のエントリが含まれます。

PutEvents を使用するときのエラーの処理

デフォルトでは、リクエスト内の個々のエントリでエラーが発生しても、リクエスト内のそれ以降のエントリの処理は停止されません。つまり、応答 Entries 配列には、正常に処理されたレコードと、正常に処

理されなかったエントリの両方が含まれます。正常に処理されなかったエントリを検出し、それ以降の呼び出しに含める必要があります。

成功した結果エントリには ID 値が含まれ、失敗した結果エントリには `ErrorCode` および `ErrorMessage` 値が含まれます。`ErrorCode` パラメータはエラーのタイプを反映しています。`ErrorMessage` は、エラーに関する詳細情報を提供します。以下の例では、`PutEvents` リクエストに対する 3 つの結果エントリがあります。2 番目のエントリは失敗し、応答に反映されています。

例: PutEvents 応答シンタックス

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

正常に処理されなかったエントリは、以降の `PutEvents` リクエストに含めることができます。最初に、`PutEventsResult` の `FailedRecordCount` パラメータを調べて、リクエスト内にエラーとなったレコードがあるかどうかを確認します。このようなレコードがある場合は、`Entry` が null 以外である各 `ErrorCode` を、以降のリクエストに追加してください。このタイプのハンドラーの例については、次のコードを参照してください。

例: PutEvents 失敗ハンドラー

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

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

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

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

```
    }
```

AWS CLI を使用したイベントの送信

AWS CLI を使用してカスタムイベントを送ることができます。以下の例では、CloudWatch イベント に 1 つのカスタムイベントを送っています。

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

また、以下のようにファイル (entries.json など) を作成できます。

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

AWS CLI を使用して、このファイルからエントリを読み込み、イベントを送ることができます。コマンドプロンプトで、次のように入力します。

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

PutEvents イベントエントリのサイズの計算

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

`PutEvents` アクションを使用して CloudWatch イベント にカスタムイベントを挿入できます。エントリの合計サイズが 256 KB 未満である限り、`PutEvents` アクションを使用して複数のイベントを挿入できます。以下の手順に従ってイベントエントリのサイズをあらかじめ計算できます。その後、効率化のために複数のイベントエントリを 1 つのリクエストにまとめることができます。

> **Note**
>
> サイズ制限はエントリに適用されます。エントリがサイズ制限より小さくても、CloudWatch イベント でのイベントが同じくこのサイズより小さいわけではありません。逆に、イベントのサイズは常に、JSON 形式のイベントの必要な文字やキーのため、エントリのサイズよりも大きくな

ります。詳細については、「CloudWatch イベント のイベントパターン (p. 38)」を参照してください。

以下のように PutEventsRequestEntry サイズが計算されます。

- Time パラメータが指定されている場合は、14 バイトとして測定されます。
- Source および DetailType パラメータは、UTF-8 エンコード形式のバイト数として測定されます。
- Detail パラメータが指定されている場合は、UTF-8 エンコード形式のバイト数として測定されます。
- Resources パラメータが指定されている場合は、各エントリは UTF-8 エンコード形式のバイト数として測定されます。

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

```
int getSize(PutEventsRequestEntry entry) {
    int size = 0;
    if (entry.getTime() != null) {
        size += 14;
    }
    size += entry.getSource().getBytes(StandardCharsets.UTF_8).length;
    size += entry.getDetailType().getBytes(StandardCharsets.UTF_8).length;
    if (entry.getDetail() != null) {
        size += entry.getDetail().getBytes(StandardCharsets.UTF_8).length;
    }
    if (entry.getResources() != null) {
        for (String resource : entry.getResources()) {
            if (resource != null) {
                size += resource.getBytes(StandardCharsets.UTF_8).length;
            }
        }
    }
    return size;
}
```

CloudWatch イベント とインターフェイス VPC エンドポイントの使用

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

Amazon Virtual Private Cloud (Amazon VPC) を使用して AWS リソースをホストする場合は、VPC と CloudWatch イベント の間にプライベート接続を確立できます。この接続を使用すると、CloudWatch イベント はパブリックインターネットを経由せずに、VPC のリソースと通信できます。

Amazon VPC は、お客様の定義する仮想ネットワークで AWS リソースを起動するために使用できる AWS サービスです。VPC を使用すると、IP アドレス範囲、サブネット、ルートテーブル、ネットワークゲートウェイなどのネットワーク設定を制御できます。VPC を CloudWatch イベント に接続するには、CloudWatch イベント の インターフェイス VPC エンドポイントを定義します。このタイプのエンドポイントにより、VPC を AWS サービスに接続できるようになります。このエンドポイントは、インターネットゲートウェイ、ネットワークアドレス変換 (NAT) インスタンス、または VPN 接続を必要とせず、信頼性が高くスケーラブルな CloudWatch イベント への接続を提供します。詳細については、Amazon VPC ユーザーガイド の「Amazon VPC とは」を参照してください。

インターフェイス VPC エンドポイントは AWS PrivateLink を利用しています。これは、Elastic Network Interface とプライベート IP アドレスを使用して AWS のサービス間のプライベート通信を可能にする AWS のテクノロジーです。詳細については、「AWS サービスの新しい AWS PrivateLink」を参照してください。

次のステップは Amazon VPC のユーザー向けです。詳細については、Amazon VPC ユーザーガイド の「開始方法」を参照してください。

現在利用できるリージョン

現在、CloudWatch イベント は、次のリージョンで VPC エンドポイントをサポートしています。

- 米国東部 (オハイオ)
- 米国東部 (バージニア北部)
- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (東京)

- カナダ (中部)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (パリ)
- 南米 (サンパウロ)

CloudWatch イベント 用の VPC エンドポイントの作成

VPC で CloudWatch イベント の使用を開始するには、CloudWatch イベント のインターフェイス VPC エンドポイントを作成します。選択するサービス名は、[com.amazonaws.***Region***.events] です。詳細については、Amazon VPC ユーザーガイド の「インターフェイスエンドポイントの作成」を参照してください。

CloudWatch イベント の設定を変更する必要はありません。CloudWatch イベント は、パブリックエンドポイントまたはプライベートインターフェイス VPC エンドポイントのうち使用中のいずれかを使用して、他の AWS サービスを呼び出します。たとえば、CloudWatch イベント のインターフェイス VPC エンドポイントを作成し、トリガーされたときに Amazon SNS に通知を送信する CloudWatch イベント ルールが既にある場合、通知はインターフェイス VPC エンドポイントを通じて流れ始めます。

CloudWatch イベント VPC エンドポイントへのアクセスのコントロール

VPC エンドポイントポリシーは、エンドポイントを作成または変更するときにエンドポイントにアタッチする IAM リソースポリシーです。エンドポイントの作成時にポリシーをアタッチしない場合、サービスへのフルアクセスを許可するデフォルトのポリシーがアタッチされます。エンドポイントポリシーは、IAM ユーザーポリシーやサービス固有のポリシーを上書き、または置き換えません。これは、エンドポイントから指定されたサービスへのアクセスを制御するための別のポリシーです。

エンドポイントのポリシーは、JSON 形式で記述される必要があります。

詳細については、Amazon VPC ユーザーガイド の「VPC エンドポイントによるサービスのアクセスコントロール」を参照してください。

CloudWatch イベント のエンドポイントポリシーの例を次に示します。このポリシーでは、VPC を介して CloudWatch イベント に接続するユーザーは CloudWatch イベント にイベントを送信できますが、他の CloudWatch イベント アクションを実行することはできません。

```
{
  "Statement": [
    {
      "Sid": "PutOnly",
      "Principal": "*",
      "Action": [
        "events:PutEvents"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

CloudWatch イベント の VPC エンドポイントポリシーを変更するには

1. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。
2. ナビゲーションペインで、[エンドポイント] を選択します。
3. CloudWatch イベント のエンドポイントをまだ作成していない場合は、[エンドポイントの作成] を選択します。次に、[com.amazonaws.***Region***.events] を選択し、[エンドポイントの作成] を選択します。
4. [com.amazonaws.***Region***.events] エンドポイントを選択し、画面の下部で [ポリシー] タブを選択します。
5. [ポリシーの編集] を選択してポリシーを変更します。

CloudWatch メトリクスの使用状況のモニタリング

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

CloudWatch イベント は 1 分ごとにメトリクスを Amazon CloudWatch に送信します。

CloudWatch イベント のメトリクス

`AWS/Events` 名前空間には、次のメトリクスが含まれます。

これらのメトリックはすべてカウントを単位として使用するため、合計およびサンプル数が最も有用な統計となります。

メトリクス	説明
`DeadLetterInvocations`	イベントに反応してルールのターゲットが呼び出されなかった回数を測定します。これには、呼び出しによって同じルールが再度トリガーされ、無限ループが発生したものが含まれます。 有効なディメンション: RuleName 単位: 個
`Invocations`	イベントに反応してルールのターゲットが呼び出された回数を測定します。これには、成功した呼び出しと失敗した呼び出しが含まれますが、スロットルされた試行と再実行された試行は完全に失敗するまで含められません。これには DeadLetterInvocations は含まれません。 **Note** 値が 0 以外の場合、CloudWatch イベント はこのメトリックスを CloudWatch にのみ送信します。 有効なディメンション: RuleName 単位: 個
`FailedInvocations`	完全に失敗した呼び出しの数を測定します。これには、再試行された呼び出しや、再試行の後に成功した呼び出しは含まれません。また、DeadLetterInvocations にカウントされる失敗した呼び出しはカウントされません。 有効なディメンション: RuleName 単位: 個

メトリクス	説明
`TriggeredRules`	いずれかのイベントに一致した、トリガーされたルールの数を測定します。 有効なディメンション: RuleName 単位: 個
`MatchedEvents`	いずれかのルールと一致したイベントの数を測定します。 有効なディメンション: なし 単位: 個
`ThrottledRules`	スロットルされているトリガーされたルールの数を測定します。 有効なディメンション: RuleName 単位: 個

CloudWatch イベント メトリクスのディメンション

CloudWatch イベント メトリクスには、以下のようなディメンションがあります。

ディメンション	説明
`RuleName`	使用可能なメトリクスをルール名でフィルタ処理します。

Amazon CloudWatch Events マネージドルール

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

AWS の他のサービスは、各サービスの特定の関数で必要な CloudWatch イベント ルールを AWS アカウント内に作成して管理できます。これらは、マネージドルールと呼ばれます。

サービスでは、マネージドルールを作成する際に、IAM ポリシーも作成する場合があります。このポリシーにより、ルールを作成するアクセス許可をサービスに付与します。この方法で作成された IAM ポリシーの適用範囲はリソースレベルのアクセス許可に限定され、許可されるのは必要なルールの作成のみです。

マネージドルールは、[Force delete (強制削除)] オプションを使用して削除できます。このオプションは、このルールが他のサービスで不要であることが確かな場合にのみ使用してください。それ以外の場合、マネージドルールを削除すると、このルールに依存する機能が動作しなくなります。

Amazon CloudWatch Events のセキュリティ

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

CloudWatch イベント のセキュリティ情報については、「Amazon EventBridge のセキュリティ」を参照してください。

Amazon CloudWatch Events リソースのタグ付け

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

タグとは、ユーザーまたは AWS が AWS リソースに割り当てるカスタム属性ラベルです。各タグは 2 つの部分で構成されます。

- タグキー (例: `CostCenter`、`Environment`、または `Project`)。タグキーでは、大文字と小文字が区別されます。
- タグ値として知られるオプションのフィールド (例: `111122223333` または `Production`)。タグ値を省略すると、空の文字列を使用した場合と同じになります。タグキーとタグ値は大文字と小文字が区別されます。

タグは、以下のことに役立ちます。

- AWS リソースの特定と整理。多くの AWS のサービスではタグ付けがサポートされるため、さまざまなサービスからリソースに同じタグを割り当てて、リソースの関連を示すことができます。たとえば、同じタグを、EC2 インスタンスに割り当てる CloudWatch イベント ルールに割り当てることができます。
- AWS のコストの追跡。これらのタグは、AWS Billing and Cost Management ダッシュボードにアクティブベースします。AWS では、タグを使用してコストを分類し、毎月のコスト割り当てレポートを設定することができます。詳細については、『AWS Billing and Cost Management ユーザーガイド』の「コスト配分タグの使用」を参照してください。

以下のセクションでは、CloudWatch イベント のタグの詳細について説明しています。

CloudWatch イベント でサポートされているリソース

CloudWatch イベント の次のリソースがタグ付けをサポートしています。

- ルール

タグを追加および管理する方法については、「タグを管理する (p. 105)」を参照してください。

タグを管理する

タグは、リソースの `Key` および `Value` プロパティで構成されています。このようなプロパティの値を追加、編集、削除するには、CloudWatch コンソール、AWS CLI、CloudWatch イベント API を使用できます。タグの使用については、以下を参照してください。

- Amazon CloudWatch Events API リファレンス の「TagResource」、「UntagResource」、「ListTagsForResource」
- Amazon CloudWatch CLI Reference の「tag-resource」、「untag-resource」、「list-tags-for-resource」
- リソースグループ ユーザーガイドの「タグエディターの使用」

タグの名前付けと使用状況の規則

CloudWatch イベント リソースでのタグの使用には、次の基本的な命名規則と使用規則が適用されます。

- 各リソースには、最大 50 個のタグを設定できます。
- タグキーは、リソースごとにそれぞれ一意である必要があります。また、各タグキーに設定できる値は 1 つのみです。
- タグキーの最大長は UTF-8 で 128 Unicode 文字です。
- タグ値の最大長は UTF-8 で 256 Unicode 文字です。
- 使用できる文字は、UTF-8 対応の文字、数字、スペースと、文字 (. : + = @ _ / -) (ハイフン) です。
- タグのキーと値は大文字と小文字が区別されます。ベストプラクティスとして、タグを大文字にするための戦略を決定し、その戦略をすべてのリソースタイプにわたって一貫して実装します。たとえば、`Costcenter`、`costcenter`、`CostCenter` を使用するかどうかを決定し、すべてのタグに同じ規則を使用します。整合性がないケースで同様のタグを使用することは避けてください。
- `aws:` プレフィックスは AWS の使用に予約されているため、タグで使用することはできません。このプレフィックスが含まれるタグのキーや値を編集したり削除することはできません。このプレフィックスを持つタグは、リソースあたりのタグ数のクォータにはカウントされません。

AWS CloudTrail を使用した Amazon CloudWatch Events API コールのログ記録

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

Amazon CloudWatch Events は、CloudWatch イベント のユーザーやロール、または AWS 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、AWS アカウントによって、または AWS アカウントの代わりに行われた API コールをキャプチャします。キャプチャされた呼び出しには、CloudWatch コンソールの呼び出しと、CloudWatch イベント 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、CloudWatch イベント のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信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 [Event history (イベント履歴)] で最新のイベントを表示できます。CloudTrail によって収集された情報を使用して、リクエストの作成元の IP アドレス、リクエストの実行者、リクエストの実行日時などの詳細を調べて、CloudWatch イベント に対してどのようなリクエストが行われたかを判断できます。

CloudTrail の詳細 (設定して有効にする方法など) については、『AWS CloudTrail User Guide』を参照してください。

トピック

CloudTrail 内の CloudWatch イベント 情報

CloudTrail は、アカウント作成時に AWS アカウントで有効になります。CloudWatch イベント でサポートされるイベントアクティビティが発生すると、そのアクティビティは CloudTrail イベントとして AWS のサービスの他のイベントとともに [Event history (イベント履歴)] 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。

CloudWatch イベント 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については、証跡を作成します。証跡により、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、コンソールで作成した証跡がすべての AWS リージョンに適用されます。証跡では、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がログに記録され、指定した Amazon S3 バケットにログファイルが配信されます。さらに、より詳細な分析と AWS ログで収集されたデータに基づいた行動のためにその他の CloudTrail サービスを設定できます。詳細については、以下を参照してください。

- 証跡を作成するための概要

- CloudTrail でサポートされるサービスと統合
- 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」と「複数のアカウントから CloudTrail ログファイルを受け取る」

CloudWatch イベント では、以下のアクションをイベントとして CloudTrail ログファイルに記録することをサポートしています。

- DeleteRule
- DescribeEventBus
- DescribeRule
- DisableRule
- EnableRule
- ListRuleNamesByTarget
- ListRules
- ListTargetsByRule
- PutPermission
- PutRule
- PutTargets
- RemoveTargets
- TestEventPattern

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。この ID 情報は以下のことを確認するのに役立ちます。

- リクエストが、ルートまたは 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どうか。
- リクエストが、ロールとフェデレーティッドユーザーのどちらの一時的なセキュリティ認証情報を使用して送信されたか.
- リクエストが、別の AWS サービスによって送信されたかどうか。

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

例: CloudWatch イベント ログファイルエントリ

証跡は、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できる設定です。CloudTrail ログファイルには、1 つ以上のログエントリが含まれます。イベントは任意の送信元からの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストのパラメータなどに関する情報が含まれます。CloudTrail ログファイルは、パブリック API 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。

以下の CloudTrail ログファイルエントリは、ユーザーが CloudWatch イベント PutRule アクションを呼び出したことを示します。

```
{
        "eventVersion":"1.03",
        "userIdentity":{
           "type":"Root",
           "principalId":"123456789012",
           "arn":"arn:aws:iam::123456789012:root",
```

```
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "sessionContext":{
               "attributes":{
                  "mfaAuthenticated":"false",
                  "creationDate":"2015-11-17T23:56:15Z"
               }
            }
         },
         "eventTime":"2015-11-18T00:11:28Z",
         "eventSource":"events.amazonaws.com",
         "eventName":"PutRule",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"AWS Internal",
         "userAgent":"AWS CloudWatch Console",
         "requestParameters":{
            "description":"",
            "name":"cttest2",
            "state":"ENABLED",
            "eventPattern":"{\"source\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance
 State-change Notification\"]}",
            "scheduleExpression":""
         },
         "responseElements":{
            "ruleArn":"arn:aws:events:us-east-1:123456789012:rule/cttest2"
         },
         "requestID":"e9caf887-8d88-11e5-a331-3332aa445952",
         "eventID":"49d14f36-6450-44a5-a501-b0fdcdfaeb98",
         "eventType":"AwsApiCall",
         "apiVersion":"2015-10-07",
         "recipientAccountId":"123456789012"
}
```

CloudWatch イベント のクォータ

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

CloudWatch イベント および EventBridge のサービスクォータの詳細については、「Amazon EventBridge のクォータ」を参照してください。

詳細については、以下を参照してください。

- Amazon EventBridge
- EventBridge サービスクォータ
- Amazon EventBridge API リファレンス

CloudWatch イベント のトラブルシューティング

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

このセクションの手順を使用して、CloudWatch イベント のトラブルシューティングを行えます。

トピック

ルールはトリガーされたが、Lambda 関数が呼び出されなかった

Lambda 関数に対する適切なアクセス権限が設定されていることを確認します。AWS CLI を使用して以下のコマンドを実行します (関数名を実際の関数に置き換え、関数がある AWS リージョンを使用します)。

```
aws lambda get-policy --function-name MyFunction --region us-east-1
```

以下のような出力が表示されます。

```
{
    "Policy": "{\"Version\":\"2012-10-17\",
    \"Statement\":[
        {\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":\"arn:aws:events:us-
east-1:123456789012:rule/MyRule\"}},
        \"Action\":\"lambda:InvokeFunction\",
        \"Resource\":\"arn:aws:lambda:us-east-1:123456789012:function:MyFunction\",
        \"Effect\":\"Allow\",
        \"Principal\":{\"Service\":\"events.amazonaws.com\"},
        \"Sid\":\"MyId\"}
    ],
    \"Id\":\"default\"}"
}
```

以下のように表示された場合:

```
A client error (ResourceNotFoundException) occurred when calling the GetPolicy operation:
 The resource you requested does not exist.
```

または、出力が表示されたが、信頼できるエンティティとして events.amazonaws.com がポリシーにない場合は、以下のコマンドを実行します。

```
aws lambda add-permission \
--function-name MyFunction \
--statement-id MyId \
--action 'lambda:InvokeFunction' \
--principal events.amazonaws.com \
--source-arn arn:aws:events:us-east-1:123456789012:rule/MyRule
```

Note

ポリシーが正しくない場合は、そのポリシーを削除してからルールに追加し直すことで、CloudWatch イベント コンソールでルールを編集することもできます。CloudWatch イベント コンソールで、ターゲットに対する適切なアクセス権限を設定します。
特定の Lambda エイリアスまたはバージョンを使用する場合は、`aws lambda get-policy` および `aws lambda add-permission` コマンドに `--qualifier` パラメータを追加する必要があります。

```
aws lambda add-permission \
--function-name MyFunction \
--statement-id MyId \
--action 'lambda:InvokeFunction' \
--principal events.amazonaws.com \
--source-arn arn:aws:events:us-east-1:123456789012:rule/MyRule
--qualifier alias or version
```

Lambda 関数がトリガーに失敗するもう 1 つの理由は、`get-policy` の実行中に表示されるポリシーに、`SourceAccount` フィールドが含まれている場合です。`SourceAccount` 設定により、CloudWatch イベントは関数を呼び出すことができなくなります。

ルールを修正/作成したが、テストイベントと一致しなかった

ルールまたはそのターゲットを変更すると、受信イベントはすぐに、新しいか更新されたルールへのマッチングを開始/停止しないことがあります。変更が有効になるまで、しばらくお待ちくだ

さい。しばらく待っても、イベントがまだ一致しない場合は、ルールの CloudWatch メトリクス (`TriggeredRules`、`Invocations`、`FailedInvocations` など) を確認して、さらなるデバッグを行うことができます。カスタムメトリクスの詳細については、『Amazon CloudWatch ユーザーガイド』の「Amazon CloudWatch Eventsメトリクスとディメンション」を参照してください。

ルールが AWS サービスからのイベントによってトリガーされる場合は、`TestEventPattern` アクションを使用して、テストイベントでルールのイベントパターンをテストすることで、ルールのイベントパターンが正しく設定されていることを確認することもできます。詳細については、『Amazon CloudWatch Events API リファレンス』の「TestEventPattern」を参照してください。

ScheduleExpression に指定されている時間にルールが自己トリガーされなかった

ScheduleExpressions は UTC です。UTC タイムゾーンで自己トリガーするルールのスケジュールが設定されていることを確認します。ScheduleExpression が正しい場合は、「ルールを修正/作成したが、テストイベントと一致しなかった (p. 112)」の手順に従います。

予期した時間にルールがトリガーされなかった

CloudWatch イベント では、期間ごとに実行するルールを作成するときに、正確な開始時間の設定がサポートされません。実行時間へのカウントダウンは、ルールを作成するとすぐに開始されます。

Cron 式を使用して、指定した時間にターゲットを起動できます。たとえば、Cron 式を使用して、4 時間おきのちょうど 0 分にトリガーされるルールを作成することができます。CloudWatch コンソールでは Cron 式 `0 0/4 * * ? *` を使用し、AWS CLI では Cron 式 `cron(0 0/4 * * ? *)` を使用できます。たとえば、AWS CLI を使用して、4 時間ごとにトリガーされる TestRule という名前のルールを作成するには、コマンドプロンプトで次のように入力します。

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

`0/5 * * * ? *` Cron 式を使用して、5 分ごとにルールをトリガーできます。以下に例を示します。

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

CloudWatch イベント は、スケジュール式で第 2 レベルの精度を提供しません。Cron 式を使用した最小の解決は分です。CloudWatch イベント とターゲットサービスが持つ分散性の特質により、スケジュールされたルールがトリガーされてから、ターゲットサービスがターゲットリソースの実行を優先するまでの遅延は、数秒となる可能性があります。スケジュールされたルールは、その分のうちにトリガーされますが、正確に 0 秒にトリガーされません。

ルールは IAM API 呼び出しに一致するが、トリガーされなかった

IAM サービスは 米国東部 (バージニア北部) リージョン でのみ使用できるため、IAM からの AWS API 呼び出しイベントはそのリージョンでのみ使用できます。詳細については、「サポートされている各サービスからの CloudWatch イベント イベントの例 (p. 43)」を参照してください。

ルールがトリガーされるときに、ルールに関連付けられている IAM ロールが無視されるため、ルールが機能しない

ルールの IAM ロールは、Kinesis ストリームにのみイベントを関連付けるために使用します。Lambda 関数と Amazon SNS トピックの場合、リソースベースのアクセス権限を付与する必要があります。

リージョンの AWS STS エンドポイントが有効になっていることを確認します。CloudWatch イベントは、指定された IAM ロールを引き受けるときに、リージョンの AWS STS エンドポイントに問い合わせます。詳細については、『IAM ユーザーガイド』の「AWS リージョンでの AWS STS のアクティブ化と非アクティブ化」を参照してください。

リソースに一致することを条件とする EventPattern を使用してルールを作成したが、このルールに一致するいずれのイベントも表示されない

AWS のほとんどのサービスでは、Amazon リソースネーム (ARN) 内のコロン (:) またはスラッシュ (/) は同じ文字として扱われます。ただし、CloudWatch イベント では、イベントパターンとルールで完全一致が使用されます。イベントパターンの作成時に正しい ARN 文字を使用して、一致させるイベント内の ARN 構文とそれらの文字が一致するようにしてください。

また、必ずしもすべてのイベントでリソースフィールドが入力されているわけではありません (CloudTrail からの AWS API コールイベントなど)。

ターゲットへのイベントの配信で遅延が発生した

ターゲットリソースが制約されているシナリオを除き、CloudWatch イベント は、最大 24 時間にわたりターゲットにイベントの配信をしようとします。最初の試行は、イベントがイベントストリームに到達するとすぐに行われます。ただし、ターゲットサービスに問題がある場合、CloudWatch イベント は自動的に別の配信を再スケジュールします。イベントの到着から 24 時間が経過すると、それ以上の試行はスケジュールされず、`FailedInvocations` メトリクスが CloudWatch で発行されます。`FailedInvocations` メトリクスに基づいて CloudWatch アラームを作成することをお勧めします。

一部のイベントがターゲットに配信されない

CloudWatch イベント ルールのターゲットが長時間制約されている場合、CloudWatch イベント は配信を再試行しない可能性があります。たとえば、ターゲットが受信イベントのトラフィックを処理するようにプロビジョニングされておらず、ターゲットサービスが、CloudWatch イベント がユーザーに代わって行うリクエストをスロットリングしている場合、CloudWatch イベント は配信を再試行しない可能性があります。

1 つのイベントに応じてルールが複数回トリガーされました。CloudWatch イベント で、ルールのトリガーまたはターゲットへのイベントの提供で何が保証されますか。

まれに、単一のイベントまたはスケジュールされた期間に対して同じルールを複数回トリガーしたり、特定のトリガーされたルールに対して同じターゲットを複数回起動したりする場合があります。

無限ループの防止

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

これを防ぐには、トリガーされたアクションが同じルールを再び開始しないようにルールを記述します。たとえば、変更された後ではなく、エラー状態にある ACL が見つかった場合にのみ、ルールが開始されるようにします。

無限ループにより、予想よりも高い料金がすぐに発生する可能性があります。指定したクォータを料金が超えるとアラートで知らせる予算設定を使用することをお勧めします。詳細については、「予算によるコストの管理」を参照してください。

マイイベントがターゲットの Amazon SQS キューに配信されない

Amazon SQS キューが暗号化されている可能性があります。暗号化された Amazon SQS キューをターゲットとして使用してルールを作成する場合、その暗号化されたキューにイベントを正常に配信するには、KMS キーポリシーに次のセクションが含まれている必要があります。

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

ルールがトリガーされているが、Amazon SNS トピックにいずれのメッセージもパブリッシュされません。

Amazon SNS トピックに対する適切なアクセス権限が設定されていることを確認します。AWS CLI を使用して以下のコマンドを実行します (トピック ARN を実際のトピックに置き換え、トピックがある AWS リージョンを使用します)。

```
aws sns get-topic-attributes --region us-east-1 --topic-arn "arn:aws:sns:us-
east-1:123456789012:MyTopic"
```

ポリシーには以下のような属性があります。

```
"{\"Version\":\"2012-10-17\",
\"Id\":\"__default_policy_ID\",
\"Statement\":[{\"Sid\":\"__default_statement_ID\",
\"Effect\":\"Allow\",
\"Principal\":{\"AWS\":\"*\"},
\"Action\":[\"SNS:Subscribe\",
\"SNS:ListSubscriptionsByTopic\",
\"SNS:DeleteTopic\",
\"SNS:GetTopicAttributes\",
\"SNS:Publish\",
\"SNS:RemovePermission\",
\"SNS:AddPermission\",
\"SNS:Receive\",
\"SNS:SetTopicAttributes\"],
\"Resource\":\"arn:aws:sns:us-east-1:123456789012:MyTopic\",
\"Condition\":{\"StringEquals\":{\"AWS:SourceOwner\":\"123456789012\"}}},{\"Sid\":
\"Allow_Publish_Events\",
\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"events.amazonaws.com\"},
\"Action\":\"sns:Publish\",
\"Resource\":\"arn:aws:sns:us-east-1:123456789012:MyTopic\"}]}"
```

以下のようなポリシーがある場合は、デフォルトのポリシーのみが設定されています。

```
"{\"Version\":\"2008-10-17\",
\"Id\":\"__default_policy_ID\",
\"Statement\":[{\"Sid\":\"__default_statement_ID\",
\"Effect\":\"Allow\",
\"Principal\":{\"AWS\":\"*\"},
\"Action\":[\"SNS:Subscribe\",
\"SNS:ListSubscriptionsByTopic\",
\"SNS:DeleteTopic\",
\"SNS:GetTopicAttributes\",
\"SNS:Publish\",
\"SNS:RemovePermission\",
\"SNS:AddPermission\",
\"SNS:Receive\",
\"SNS:SetTopicAttributes\"],
\"Resource\":\"arn:aws:sns:us-east-1:123456789012:MyTopic\",
\"Condition\":{\"StringEquals\":{\"AWS:SourceOwner\":\"123456789012\"}}}]}"
```

ポリシーに `events.amazonaws.com` に対するパブリッシュアクセス権限がない場合は、AWS CLI を使用してトピックポリシー属性を設定します。

現在のポリシーをコピーし、ステートメントのリストに以下のステートメントを追加します。

```
{\"Sid\":\"Allow_Publish_Events\",
\"Effect\":\"Allow\",\"Principal\":{\"Service\":\"events.amazonaws.com\"},
\"Action\":\"sns:Publish\",
\"Resource\":\"arn:aws:sns:us-east-1:123456789012:MyTopic\"}
```

新しいポリシーは前に説明したようになります。

AWS CLI で以下のようにトピック属性を設定します。

```
aws sns set-topic-attributes --region us-east-1 --topic-arn "arn:aws:sns:us-
east-1:123456789012:MyTopic" --attribute-name Policy --attribute-value NEW_POLICY_STRING
```

Note

ポリシーが正しくない場合は、そのポリシーを削除してからルールに追加し直すことで、CloudWatch イベント コンソールでルールを編集することもできます。CloudWatch イベント は、ターゲットで適切なアクセス権限を設定します。

Amazon SNS トピックに関連付けられたルールを削除した後でも、Amazon SNS トピックに CloudWatch イベント のアクセス権限がある

ターゲットとして Amazon SNS でルールを作成すると、CloudWatch イベント はユーザーに代わって Amazon SNS トピックにアクセス権限を追加します。作成後すぐにルールを削除すると、CloudWatch イベント は Amazon SNS トピックからアクセス権限を削除できなくなる場合があります。その場合は、aws sns set-topic-attributes コマンドを使用してトピックからアクセス権限を削除できます。

CloudWatch イベント で使用できる IAM 条件キー

CloudWatch イベント では、AWS 全体の条件キー (IAM ユーザーガイドの「使用可能なキー」を参照) に加え、以下のサービス固有の条件キーがサポートされています。

CloudWatch イベント ルールが壊れているときに通知するアラームを作成する方法

以下のアラームを使用して、CloudWatch イベント ルールが壊れているときに通知されるようにできます。

ルールが壊れているときに警告するアラームを作成するには

1. https://console.aws.amazon.com/cloudwatch/ にある CloudWatch コンソールを開きます。
2. [Create Alarm] を選択します。[CloudWatch Metrics by Category] ペインで [Events Metrics] を選択します。
3. メトリクスのリストで、[FailedInvocations] を選択します。
4. グラフの上で、[Statistic]、[Sum] を選択します。

5. [Period] で、値 (例: 5 分) を選択します。[次へ] を選択します。
6. [Alarm Threshold] の下の [Name] に一意のアラーム名 (例: myFailedRules) を入力します。[Description] に、アラームの説明として、たとえば「Rules are not delivering events to targets」と入力します。
7. [is] で [>=] および [1] を選択します。[for] に「10」と入力します。
8. [Actions] の [Whenever this alarm] で、[State is ALARM] を選択します。
9. [通知の送信先] で、既存の Amazon SNS トピックを選択するか、新しいトピックを作成します。新しいトピックを作成するには、[新しいリスト] を選択します。新しい Amazon SNS トピックの名前 (例: myFailedRules) を入力します。
10. [Email list] に、アラームが ALARM 状態に変わったら通知する E メールアドレスを、カンマ区切りのリストに入力します。
11. [Create Alarm] を選択します。

ドキュメント履歴

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. CloudWatch イベント and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

次の表に、2018 年 6 月以降の CloudWatch イベント ユーザーガイドの各リリースにおける重要な変更点を示します。このドキュメントの更新に関する通知については、RSS フィードにサブスクライブできます。

update-history-change	update-history-description	update-history-date
タグ指定のサポート (p. 119)	一部の CloudWatch イベント リソースにタグ付けできるようになりました。詳細については、『Amazon CloudWatch Events ユーザーガイド』の「Amazon CloudWatch Events リソースにタグを付ける」を参照してください。	March 21, 2019
Amazon VPC エンドポイントのサポート (p. 119)	これで、VPC と CloudWatch イベント との間でプライベート接続を確立できます。詳細については、『Amazon CloudWatch Events ユーザーガイド』の「VPC エンドポイントでの CloudWatch イベント の使用」を参照してください。	June 28, 2018

次の表は Amazon CloudWatch Events ユーザーガイド の重要な変更点をまとめたものです。

変更	説明	リリース日
ターゲットとしての CodeBuild	イベントルールのターゲットとして CodeBuild が追加されました。詳細については、「チュートリアル: CodeBuild を使用した、自動化されたビルドのスケジュール (p. 31)」を参照してください。	2017 年 12 月 13 日
ターゲットとしての AWS Batch	イベントルールのターゲットとして AWS Batch が追加されました。詳細については、「AWS Batchイベント」を参照してください。	2017 年 9 月 8 日
CodePipeline と AWS Glue イベント	CodePipeline および AWS Glue からのイベントのサポートが追加されました。詳細については、「CodePipeline イベント (p. 46)」および「AWS Glue イベント (p. 58)」を参照してください。	2017 年 9 月 8 日

変更	説明	リリース日
CodeBuild イベントと CodeCommit イベント	CodeBuild および CodeCommit からのイベントのサポートが追加されました。詳細については、「CodeBuild イベント (p. 45)」を参照してください。	2017 年 8 月 3 日
追加ターゲットのサポート	CodePipeline と Amazon Inspector をイベントのターゲットにできます。	2017 年 6 月 29 日
AWS アカウント間でのイベントの送受信のサポート	AWS アカウントは、イベントを別の AWS アカウントに送信できます。詳細については、「AWS アカウント間のイベントの送受信 (p. 87)」を参照してください。	2017 年 6 月 29 日
追加ターゲットのサポート	これで、Amazon EC2 インスタンス (Run Command 経由) と AWS ステートマシンの 2 つの Step Functions サービスをさらにイベントアクションとして設定できるようになりました。詳細については、「Amazon CloudWatch Events を使用するためのサンプルの使用シナリオを提供する (p. 5)」を参照してください。	2017 年 3 月 7 日
Amazon EMR イベント	Amazon EMR 用のイベントのサポートが追加されました。詳細については、「Amazon EMR イベント (p. 49)」を参照してください。	2017 年 3 月 7 日
AWS Health イベント	AWS Health のイベントのサポートが追加されました。詳細については、「AWS Health イベント (p. 63)」を参照してください。	2016 年 12 月 1 日
Amazon Elastic Container Service イベント	Amazon ECS 用のイベントのサポートが追加されました。詳細については、「Amazon Elastic Container Service イベント (p. 49)」を参照してください。	2016 年 11 月 21 日
AWS Trusted Advisor イベント	Trusted Advisor 用のイベントのサポートが追加されました。詳細については、「AWS Trusted Advisor イベント (p. 83)」を参照してください。	2016 年 11 月 18 日
Amazon Elastic Block Store イベント	Amazon EBS 用のイベントのサポートが追加されました。詳細については、「Amazon EBS イベント (p. 48)」を参照してください。	2016 年 11 月 14 日
AWS CodeDeploy イベント	CodeDeploy 用のイベントのサポートが追加されました。詳細については、「AWS CodeDeploy イベント (p. 45)」を参照してください。	2016 年 9 月 9 日
1 分単位で予定されたイベント	1 分単位で予定されたイベントのサポートの追加詳細については、「cron 式 (p. 34)」および「rate 式 (p. 36)」を参照してください。	2016 年 4 月 19 日
ターゲットとして Amazon Simple Queue Service キュー	ターゲットとして、Amazon SQS キューのサポートが追加されました。詳細については、「Amazon CloudWatch Events とは (p. 1)」を参照してください。	2016 年 3 月 30 日
Auto Scaling イベント	Auto Scaling ライフサイクルフックのイベントのサポートが追加されました。詳細については、「Amazon EC2 Auto Scaling イベント (p. 48)」を参照してください。	2016 年 2 月 24 日

変更	説明	リリース日
新しいサービス	CloudWatch イベント の初回リリース	2016 年 1 月 14 日

AWS の用語集

最新の AWS の用語については、『AWS General Reference』の「AWS の用語集」を参照してください。