Amazon CloudWatch Logs

ユーザーガイド

Amazon CloudWatch Logs: ユーザーガイド

Table of Contents

Amazon CloudWatch Logs とは

Amazon CloudWatch Logs を使用して、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、AWS CloudTrail、Route 53、およびその他のソースのログファイルの監視、保存、アクセスができます。その後、CloudWatch Logs から、関連するログデータを取得できます。

機能

- Amazon EC2 インスタンスのログのモニタリング — CloudWatch Logs で、ログデータを使用してアプリケーションとシステムをモニタリングできます。たとえば、CloudWatch Logs はアプリケーションログに存在するエラーの数をトラッキングし、エラー率が指定のしきい値を超えたときに管理者に通知を送ることができます。お客様のログが CloudWatch Logs によるモニタリングに使用されるので、コードの変更は不要です。たとえば、アプリケーションログの特定のリテラルターム (例: 「NullReferenceException」) をモニタリングしたり、ログデータの特定の場所でリテラルターム (例: Apache アクセスログの「404」ステータスコード) の発生数をカウントしたりできます。検索した語句が見つかると、CloudWatch Logs は指定された CloudWatch メトリクスにデータをレポートします。ログデータは、転送時や保管時に暗号化されます。開始するには、CloudWatch Logs の使用開始 (p. 5) を参照してください。
- AWS CloudTrail のログに記録されたイベントのモニタリング— CloudWatch にアラームを作成して、CloudTrail がキャプチャした特定の API アクティビティの通知を受け取り、通知をトラブルシューティングの実行に使用できます。使用を開始するには、AWS CloudTrail User Guideの「CloudTrail イベントの CloudWatch Logs への送信」を参照してください。
- ログの保持期間—デフォルトでは、ログは無制限に保持され、失効しません。ロググループごとに保持ポリシーを調整し、無制限の保持期間を維持するか、1 日間 ~ 10 年間の保持期間を選択することができます。
- ログデータをアーカイブする—CloudWatch Logs を使用して耐久性が高いストレージにログデータを保存できます。CloudWatch Logs エージェントにより、ローテーションするログデータもローテーションしないログデータも、ホストからログサービスに簡単にすばやく送信できます。その後は、必要なときに生のログデータにアクセスできます。
- Route 53 DNS クエリのログ—CloudWatch Logs を使用して、Route 53 が受け取る DNS クエリに関するログ情報を使用できます。詳細については、『Amazon Route 53 開発者ガイド』の「DNS クエリのログ記録」を参照してください。

AWS の関連サービス

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

- AWS CloudTrail は、お客様のアカウントの CloudWatch Logs API に対する呼び出し (AWS マネジメントコンソール、コマンドラインインターフェイス (CLI)、およびその他のサービスからの呼び出しを含む) をモニタリングするために使用できるウェブサービスです。CloudTrail ログ記録がオンになると、CloudTrail は、API コールをアカウントにキャプチャし、指定する Amazon S3 バケットにログファイルを送信します。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。AWS CloudTrail の詳細については、『AWS CloudTrail User Guide』の「AWS CloudTrail とは」を参照してください。CloudWatch が CloudTrail のログファイルに書き込むデータのタイプの例については、「Amazon CloudWatch Logs での AWS CloudTrail API コールのログ記録 (p. 119)」を参照してください。
- AWS Identity and Access Management (IAM) は、ユーザーに対して AWS リソースへのアクセスを安全に制御するためのウェブサービスです。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、IAM ユーザーガイド の「IAM とは」を参照してください。

- Amazon Kinesis Data Streams は、高速かつ継続的にデータの取り込みと集約を行うためのウェブサービスです。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 開発者ガイドの「Amazon Kinesis Data Streams とは」を参照してください。
- AWS Lambda は、新しい情報にすばやく対応するアプリケーションを簡単に構築するためのウェブサービスです。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、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。必要な操作は、Lambda がサポートするいずれかの言語でコードを指定するだけです。詳細については、AWS Lambda Developer Guide の「AWS Lambda とは」を参照してください。

料金表

AWS にサインアップすると、CloudWatch Logs 無料利用枠を利用して、AWS を無料で使い始めることができます。

標準料金は、CloudWatch Logs を使用した他のサービスによって保存されたログ (たとえば、Amazon VPC フローログおよび Lambda ログ) に適用されます。

詳細については、「Amazon CloudWatch 料金表」を参照してください。

Amazon CloudWatch Logs の概念

CloudWatch Logs を理解し使用するために重要な用語と概念を、以下に示します。

ログイベント

ログイベントは、モニタリングされているアプリケーションまたはリソースによって記録されたアクティビティのレコードです。CloudWatch Logs が理解するログイベントレコードには 2 つのプロパティがあります。イベント発生時のタイムスタンプおよび生のイベントメッセージです。イベントメッセージは UTF-8 でエンコードされている必要があります。

ログストリーム

ログストリームは、同じソースを共有する一連のログイベントです。より具体的には、ログストリームは一般的に、モニタリングされているアプリケーションインスタンスやリソースから送信された順序でイベントを表すものです。たとえば、ログストリームは特定のホストの Apache アクセスログと関連付けられる場合があります。ログストリームを必要としなくなった場合、aws logs delete-log-stream コマンドを使用して、削除できます。さらに、AWS は 2 か月を超える空のログストリームを削除する場合もあります。

ロググループ

ロググループは、保持、監視、アクセス制御について同じ設定を共有するログストリームのグループを定義します。各ログストリームは、1 つのロググループに属している必要があります。たとえば、各ホストから Apache アクセスログの別のログストリームがある場合は、それらのログストリームを `MyWebsite.com/Apache/access_log` という名前の 1 つのロググループにグループ化できます。

1 つのロググループに属することができるログストリームの数に制限はありません。

メトリクスフィルタ

メトリクスフィルタを使用して、取り込まれたイベントからメトリクスの監視データを抽出し、CloudWatch メトリクスのデータポイントに変換できます。メトリクスフィルタはロググループに割り当てられ、ロググループに割り当てられたすべてのフィルタはそのログストリームに適用されます。

保持設定

保持設定は、CloudWatch Logs にログイベントを保持する期間を指定するために使用できます。期限切れのログイベントは自動的に削除されます。メトリクスフィルタと同様に、保持設定はロググループに割り当てられ、ロググループに割り当てられた保持期間はその ログストリームに適用されます。

準備作業

Amazon CloudWatch Logs 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) で表示できるログを生成できます。さらに、AWS Command Line Interface (AWS CLI) をインストールして設定できます。

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://aws.amazon.com/ を開き、[AWS アカウントの作成] を選択します。

   Note

   AWS アカウントのルートユーザー 認証情報を使用して、すでに AWS マネジメントコンソール にサインインしている場合は、[Sign in to a different account (別のアカウントにサインインする)] を選択します。IAM 認証情報を使用して、すでにコンソールにサインインしている場合は、[Sign-in using root account credentials (ルートアカウントの資格情報を使ってサインイン)] を選択します。[新しい AWS アカウントの作成] を選択します。
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で [Logs] を選択します。

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 Logs 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 コマンドラインインターフェースのセットアップ」を参照してください。

CloudWatch Logs の使用開始

Amazon EC2 インスタンスおよびオンプレミスサーバーから CloudWatch Logs へのログを収集する方法として、AWS では 2 つのオプションを用意しています。

- 推奨 – 統合 CloudWatch エージェント。ログと高度なメトリクスの両方を 1 つのエージェントで収集できます。Windows Server を実行しているサーバーなど、オペレーティングシステム全体にわたるサポートが提供されています。このエージェントでも優れたパフォーマンスを提供します。

  統合エージェントを使用して CloudWatch メトリクスを収集する場合、ゲスト内の可視性のための追加のシステムメトリクスも収集できます。また、`StatsD` または `collectd` を使用して、カスタムメトリクスを収集することもできます。

  詳細については、『Amazon CloudWatch ユーザーガイド』の「CloudWatch エージェントを使用して Amazon EC2 インスタンスとオンプレミスサーバーからメトリクスとログを収集する」を参照してください。
- サポートされているが、廃止予定 – 以前の CloudWatch Logs エージェント。Linux を実行しているサーバーのみからのログを収集することができます。既にそのエージェントを使用している場合は、引き続きそのエージェントを使用することができます。

  CloudWatch Logs エージェントから統合 CloudWatch エージェントに移行する場合、統合エージェントの設定ウィザードでは、現在の CloudWatch Logs エージェント設定ファイルを読み込み、同じログを収集するように新しいエージェントを設定することができます。ウィザードの詳細については、『Amazon CloudWatch ユーザーガイド』の「ウィザードで CloudWatch エージェント設定ファイルを作成する」を参照してください。

コンテンツ

統合 CloudWatch エージェントを使用して CloudWatch Logs の使用を開始する

統合 CloudWatch エージェントを使用して CloudWatch Logs の使用を開始する方法の詳細については、『Amazon CloudWatch ユーザーガイド』の「CloudWatch エージェントを使用して Amazon EC2 インスタンスとオンプレミスサーバーからメトリクスとログを収集する」を参照してください。このセクションに記載されている手順を実行してエージェントのインストールと設定を行い、開始します。エージェントを使用して CloudWatch メトリクスを収集していない場合、メトリクスを参照するセクションはすべて無視できます。

現在古い CloudWatch Logs エージェントを使用していて、新しい統合エージェントの使用に移行する場合は、新しいエージェントパッケージに含まれているウィザードを使用することをお勧めします。このウィザードは、現在の CloudWatch Logs エージェント設定ファイルを読み込み、CloudWatch エージェントを設定して同じログを収集することができます。ウィザードの詳細については、『Amazon CloudWatch ユーザーガイド』の「ウィザードで CloudWatch エージェント設定ファイルを作成する」を参照してください。

以前の CloudWatch Logs エージェントを使用して CloudWatch Logs の使用を開始する

CloudWatch Logs エージェントを使用して、Linux または Windows Server を実行する Amazon EC2 インスタンスのログデータおよび AWS CloudTrail から記録されたイベントを発行できます。代わりに CloudWatch 統合エージェントを使用してログデータを発行することをお勧めします。新しいエージェントの詳細については、『Amazon CloudWatch ユーザーガイド』の「CloudWatch エージェントを使用して Amazon EC2 インスタンスとオンプレミスサーバーからメトリクスとログを収集する」を参照してください。あるいは、以前の CloudWatch Logs エージェントを引き続き使用することもできます。

コンテンツ

CloudWatch Logs エージェントの前提条件

CloudWatch Logs エージェントでは、Python バージョン 2.7、3.0、または 3.3、および次のいずれかのバージョンの Linux が必要です。

- Amazon Linuxバージョン 2014.03.02 以降
- Ubuntu Server バージョン 12.04、14.04、または 16.04
- CentOS バージョン 6、6.3、6.4、6.5、または 7.0
- Red Hat Enterprise Linux (RHEL) バージョン 6.5 または 7.0
- Debian 8.0

クイックスタート: 実行中の EC2 Linux インスタンスに CloudWatch Logs エージェントをインストールして設定する

> **Tip**
>
> CloudWatch には、EC2 インスタンスとオンプレミスサーバーからログとメトリクスの両方を収集できる新しい統合エージェントが含まれています。古い CloudWatch Logs エージェントをすでに使用しなくなった場合は、新しい統合 CloudWatch エージェントを使用することをお勧めします。詳細については、「CloudWatch Logs の使用開始 (p. 5)」を参照してください。

このセクションの残りの部分では、古い CloudWatch Logs エージェントの使用について説明します。

実行中の EC2 Linux インスタンスで古い CloudWatch Logs エージェントを設定する

既存の EC2 インスタンスで CloudWatch Logs エージェントインストーラを使用して、CloudWatch Logs エージェントをインストールして設定できます。インストールが完了したら、エージェントのインストール中に、インスタンスから、作成したログストリームにログが自動的に流れます。エージェントは開始を確認し無効にされるまで実行し続けます。

エージェントに加えて、AWS CLI、CloudWatch Logs SDK、または CloudWatch Logs API を使用してログデータを発行できます。AWS CLI は、コマンドラインまたはスクリプトでのデータの発行に適しています。CloudWatch Logs SDK は、アプリケーションから直接、または独自のログ発行アプリケーションを構築してログデータを発行する場合に適しています。

ステップ 1: IAM ロールまたは CloudWatch Logs ユーザーを設定する

CloudWatch Logs エージェントは IAM ロールとユーザーをサポートしています。インスタンスに関連付けられた IAM ロールがすでに存在する場合、その下に IAM ポリシーが含まれていることを確認してください。インスタンスに関連付けられた IAM ロールがまだ存在しない場合は、次のステップで IAM 認証情報を使用するか、IAM ロールインスタンスに割り当てることができます。詳細については、「IAM ロールをインスタンスにアタッチする」を参照してください。

IAM ロールまたは CloudWatch Logs ユーザーを設定するには

1. https://console.aws.amazon.com/iam/ にある IAM コンソールを開きます。
2. ナビゲーションペインで [Roles (ロール)] を選択します。
3. ロール名を選択してロールを選択します (名前の横にあるチェックボックスを選択しないでください)。
4. [Attach Policies (ポリシーのアタッチ)] を選択して、[ポリシーの作成] を選択します。

   新しいブラウザタブまたはウィンドウが開きます。
5. [JSON] タブを選択して、次の JSON ポリシードキュメントを入力します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
    ],
      "Resource": [
        "arn:aws:logs:*:*:*"
    ]
  }
 ]
}
```

6. 完了したら、[ポリシーの確認] を選択します。構文エラーがある場合、Policy Validator によって報告されます。
7. [ポリシーの確認] ページで、作成するポリシーの [名前] と [説明] (オプション) を入力します。ポリシーの [概要] を確認して、ポリシーで許可されている権限を確認します。次に [ポリシーの作成] を選択して作業を保存します。

8. ブラウザタブまたはウィンドウを閉じ、ロールの [アクセス権限の追加] を選択します。[更新] を選択し、新しいポリシーを選択してロールに追加します。
9. [Attach Policy] を選択します。

ステップ 2: 既存の Amazon EC2 インスタンスに CloudWatch Logs をインストールして設定する

CloudWatch Logs エージェントのインストール手順は、Amazon EC2 インスタンスが Amazon Linux、Ubuntu、CentOS、Red Hat のどれを実行しているかによって異なります。インスタンスの Linux のバージョンに適切な手順を使用してください。

既存の CloudWatch Logs インスタンスに Amazon Linux をインストールして設定するには

Amazon Linux AMI 2014.09 から、awslogs パッケージを使用した RPM のインストールに CloudWatch Logs エージェントが使用可能です。それ以前のバージョンの Amazon Linux は、`sudo yum update -y` コマンドを使用してインスタンスを更新することで awslogs パッケージにアクセスできます。CloudWatch Logs インストーラを使用する代わりに awslogs パッケージを RPM としてインストールすると、インスタンスは AWS から通常の更新とパッチのパッケージを受信します。CloudWatch Logs エージェントを手動で再インストールする必要はありません。

> Warning
>
> 以前に Python スクリプトを使用してエージェントをインストールした場合は、RPM インストール方法を使用して CloudWatch Logs エージェントを更新しないでください。設定に問題が発生して CloudWatch Logs エージェントが CloudWatch にログを送信できなくなる恐れがあります。

1. Amazon Linux インスタンスに接続します。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「インスタンスへの接続」を参照してください。

   問題に接続する方法の詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「インスタンスへの接続に関するトラブルシューティング」を参照してください。
2. Amazon Linux インスタンスを更新してパッケージリポジトリの最新の変更を取得します。

   ```
   sudo yum update -y
   ```
3. `awslogs` パッケージをインストールします。これは Amazon Linux インスタンスで awslogs をインストールする推奨手段です。

   ```
   sudo yum install -y awslogs
   ```
4. `/etc/awslogs/awslogs.conf` ファイルを編集して追跡するログを設定します。このファイルの編集方法については、「CloudWatch Logs エージェントのリファレンス (p. 122)」を参照してください。
5. デフォルトでは、`/etc/awslogs/awscli.conf` は us-east-1 リージョンを指します。ログを別の領域にプッシュするには、`awscli.conf` ファイルを編集し、その領域を指定します。
6. `awslogs` サービスを開始します。

   ```
   sudo service awslogs start
   ```

   Amazon Linux 2 を実行している場合は、次のコマンドを使用して `awslogs` サービスを開始します。

   ```
   sudo systemctl start awslogsd
   ```
7. (オプション) `/var/log/awslogs.log` ファイルでサービス開始時に記録されたエラーがあるかどうか確認します。

8. （オプション）システム起動時に毎回 awslogs サービスを起動する場合は、次のコマンドを実行します。

```
sudo chkconfig awslogs on
```

Amazon Linux 2 を実行している場合は、次のコマンドを使用してシステムブートのたびにサービスを開始します。

```
sudo systemctl enable awslogsd.service
```

9. エージェントが実行されてしばらくしたら、CloudWatch コンソールに新しく作成されたロググループとログストリームを確認してください。

詳細については、「CloudWatch Logs に送信されたログデータの表示 (p. 50)」を参照してください。

既存の Ubuntu Server、CentOS、Red Hat のインスタンスに CloudWatch Logs をインストールして設定するには

Ubuntu Server、CentOS、または Red Hat を実行する AMI を使用している場合、次の手順でインスタンスに CloudWatch Logs エージェントを手動インストールします。

1. EC2 インスタンスに接続します。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「インスタンスへの接続」を参照してください。

問題に接続する方法の詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「インスタンスへの接続に関するトラブルシューティング」を参照してください。

2. 次の 2 つのオプションのいずれかを使用して CloudWatch Logs エージェントインストーラを実行します。インターネットから直接実行するか、ファイルをダウンロードしてスタンドアロンで実行できます。

> **Note**
>
> CentOS 6.x、Red Hat 6.x、または Ubuntu 12.04 を実行している場合は、インストーラをダウンロードしてスタンドアロンで実行する手順を使用してください。これらのシステムでは、インターネットから直接 CloudWatch Logs エージェントをインストールすることはサポートされていません。

> **Note**
>
> Ubuntu では、次のコマンドを実行する前に `apt-get update` を実行してください。

インターネットから直接実行するには、次のコマンドを使用してプロンプトに従います。

```
curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-setup.py -O
```

```
sudo python ./awslogs-agent-setup.py --region us-east-1
```

前のコマンドが機能しない場合は、以下を試してください。

```
sudo python3 ./awslogs-agent-setup.py --region us-east-1
```

スタンドアロンをダウンロードして実行するには、次のコマンドを使用してプロンプトに従います。

```
curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-setup.py -O
```

```
curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/AgentDependencies.tar.gz
 -O
```

```
tar xvf AgentDependencies.tar.gz -C /tmp/
```

```
sudo python ./awslogs-agent-setup.py --region us-east-1 --dependency-path /tmp/
AgentDependencies
```

CloudWatch Logs リージョンを指定することで us-east-1, us-west-1, us-west-2, ap-south-1, ap-northeast-2, ap-southeast-1, ap-southeast-2, ap-northeast-1, eu-central-1, eu-west-1, or sa-east-1 エージェントをインストールできます。

Note

`awslogs-agent-setup` の現行バージョンとバージョン履歴の詳細については、CHANGELOG.txt を参照してください。

CloudWatch Logs エージェントのインストーラはセットアップ時に特定の情報が必要です。開始する前に、モニタリングするログファイルとそのタイムスタンプ形式を知っておく必要があります。また、次の情報を準備する必要があります。

項目	説明
AWS アクセスキー ID	IAM ロールを使用する場合は Enter キーを押します。それ以外の場合は、AWS アクセスキー ID を入力します。
AWS シークレットアクセスキー	IAM ロールを使用する場合は Enter キーを押します。それ以外の場合は、AWS シークレットアクセスキーを入力します。
デフォルトリージョン名	Enter キーを押します。デフォルト: us-east-2。us-east-1, us-west-1, us-west-2, ap-south-1, ap-northeast-2, ap-southeast-1, ap-southeast-2, ap-northeast-1, eu-central-1, eu-west-1, or sa-east-1 に設定できます。
デフォルト出力形式	空白にしたまま Enter キーを押します。
アップロードするログファイルのパス	送信するログデータを含むファイルの場所です。インストーラはパスの候補を表示します。
送信先ロググループ名	ロググループの名前です。インストーラはロググループ名の候補を表示します。
送信先ログストリーム名	デフォルトでは、ホストの名前です。インストーラはホスト名の候補を表示します。
タイムスタンプ形式	指定ログファイル内のタイムスタンプ形式を指定します。独自の形式を指定するには、[custom] を選択します。
初期位置	データをどのようにアップロードできますかデータファイルのすべてをアップロードする場合は [start_of_file] に設定します。新しく付け加えられたデータのみをアップロードする場合は [end_of_file] に設定します。

これらの手順が完了すると、インストーラは別のログファイルを設定するかどうか尋ねてきます。各ログファイルについて何回でもプロセスを実行できます。他にモニタリングするログファイルがない場合、別のログをセットアップするようにインストーラに求められた時に [N] を選択します。

エージェント設定ファイルの設定の詳細については、「CloudWatch Logs エージェントのリファレンス (p. 122)」を参照してください。

Note

複数のログソースから単一のログストリームにデータを送信する設定はサポートされていません。

3. エージェントが実行されてしばらくしたら、CloudWatch コンソールに新しく作成されたロググループとログストリームを確認してください。

詳細については、「CloudWatch Logs に送信されたログデータの表示 (p. 50)」を参照してください。

クイックスタート: EC2 Linux インスタンスの起動時に CloudWatch Logs エージェントをインストールして設定する

Tip

CloudWatch には、EC2 インスタンスとオンプレミスサーバーからログとメトリクスの両方を収集できる新しい統合エージェントが含まれています。古い CloudWatch Logs エージェントをすでに使用しなくなった場合は、新しい統合 CloudWatch エージェントを使用することをお勧めします。詳細については、「CloudWatch Logs の使用開始 (p. 5)」を参照してください。
このセクションの残りの部分では、古い CloudWatch Logs エージェントの使用について説明します。

起動時に EC2 Linux インスタンスに古い CloudWatch Logs エージェントをインストールする

Amazon EC2 ユーザーデータは、インスタンスの起動時にパラメータ情報を渡す Amazon EC2 の機能です。これを使用してインスタンスに CloudWatch Logs エージェントをインストールして設定します。CloudWatch Logs エージェントのインストール情報と設定情報を Amazon EC2 に渡すには、Amazon S3 バケットのようなネットワークの場所に設定ファイルを用意します。

複数のログソースから単一のログストリームにデータを送信する設定はサポートされていません。

前提条件

ロググループとログストリームをすべて記述したエージェントの設定ファイルを作成します。これは、モニタリングするログファイルとそのアップロード先のロググループおよびログストリームが記述されているテキストファイルです。エージェントはこの設定ファイルを使用して記述されたすべてのログファイルのモニタリングおよびアップロードを開始します。エージェント設定ファイルの設定の詳細については、「CloudWatch Logs エージェントのリファレンス (p. 122)」を参照してください。

以下は、Amazon Linux のサンプルエージェント設定ファイルです。

```
[general]
state_file = /var/awslogs/state/agent-state

[/var/log/messages]
file = /var/log/messages
log_group_name = /var/log/messages
log_stream_name = {instance_id}
datetime_format = %b %d %H:%M:%S
```

以下は、Ubuntu のサンプルエージェント設定ファイルです。

```
[general]
state_file = /var/awslogs/state/agent-state

[/var/log/syslog]
file = /var/log/syslog
log_group_name = /var/log/syslog
log_stream_name = {instance_id}
datetime_format = %b %d %H:%M:%S
```

IAM ロールを設定するには

1. https://console.aws.amazon.com/iam/ にある IAM コンソールを開きます。
2. ナビゲーションペインで、[Policies]、[Create Policy] の順に選択します。
3. [Create Policy] ページの [Create Your Own Policy] で、[Select] を選択します。カスタムポリシーの作成の詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「Amazon EC2 の IAM ポリシー」を参照してください。
4. [Review Policy] ページで、[Policy Name] にポリシーの名前を入力します。
5. [Policy Document] に、次のポリシーをコピーして貼り付けます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::myawsbucket/*"
            ]
        }
    ]
}
```

6. [Create Policy] を選択します。
7. ナビゲーションペインで [Roles]、[Create New Role] の順に選択します。
8. [Set Role Name] ページで、ロールの名前を入力し、[Next Step] を選択します。
9. [Select Role Type] ページで、[Amazon EC2] の隣にある [Select] を選択します。
10. [Attach Policy] ページのテーブルのヘッダーで、[Policy Type]、[Customer Managed] の順に選択します。
11. 作成した IAM ポリシーを選択し、[次のステップ] を選択します。
12. [Create Role] を選択します。

    IAM のユーザーとポリシーの詳細については、『IAM ユーザーガイド』の「IAM ユーザーとグループ」と「IAM ポリシーを管理する」を参照してください。

新しいインスタンスを起動して CloudWatch Logs を有効にするには

1. https://console.aws.amazon.com/ec2/) にある Amazon EC2 コンソールを開きます。
2. [インスタンスの作成] を選択します。

   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. [Step 1: Choose an Amazon Machine Image (AMI)] ページで、起動する Linux インスタンスタイプを選択し、[Step 2: Choose an Instance Type] ページで [Next: Configure Instance Details] を選択します。

   cloud-init が Amazon Machine Image (AMI) に含まれていることを確認します。Amazon Linux AMI や Ubuntu および RHEL の AMI にはすでに cloud-init が含まれていますが、CentOS および AWS Marketplace のその他の AMI には含まれていない場合があります。
4. [ステップ 3: Configure Instance Details (インスタンスの詳細の設定)] ページの [IAM ロール] で、作成した IAM ロールを選択します。
5. [Advanced Details] の [User data] で、以下のスクリプトをボックス内に貼り付けます。その後、スクリプトを更新するには、[-c] オプションの値を、エージェント設定ファイルの位置に変更します。

   ```
   #!/bin/bash
   curl https://s3.amazonaws.com//aws-cloudwatch/downloads/latest/awslogs-agent-setup.py -
   O
   chmod +x ./awslogs-agent-setup.py
   ./awslogs-agent-setup.py -n -r us-east-1 -c s3://myawsbucket/my-config-file
   ```
6. そのほかインスタンスへの必要な変更を行い、起動設定を確認して [Launch] を選択します。
7. エージェントが実行されてしばらくしたら、CloudWatch コンソールに新しく作成されたロググループとログストリームを確認してください。

   詳細については、「CloudWatch Logs に送信されたログデータの表示 (p. 50)」を参照してください。

クイックスタート: CloudWatch Logs エージェントを使用して CloudWatch Logs にログを送信するために Windows Server 2016 を実行する Amazon EC2 インスタンスを有効にする

Tip

CloudWatch には、EC2 インスタンスとオンプレミスサーバーからログとメトリクスの両方を収集できる新しい統合エージェントが含まれています。新しい統合 CloudWatch エージェントを使用することをお勧めします。詳細については、「CloudWatch Logs の使用開始 (p. 5)」を参照してください。
このセクションの残りの部分では、古い CloudWatch Logs エージェントの使用について説明します。

Windows Server 2016 を実行している Amazon EC2 インスタンスで古い CloudWatch Logs エージェントを使用した CloudWatch Logs へのログの送信を有効にする

Windows Server 2016 を実行しているインスタンスで CloudWatch Logs にログを送信するには、複数の方法を使用できます。このセクションのステップでは、Systems Manager Run Command を使用します。

他の可能な方法の詳細については、「Amazon CloudWatch へのログ、イベント、パフォーマンスカウンターの送信」を参照してください。

ステップ

サンプル設定ファイルのダウンロード

以下のサンプルファイルをコンピュータにダウンロードします。AWS.EC2.Windows.CloudWatch.json。

CloudWatch の JSON ファイルの設定

CloudWatch に送信するログを決めるには、設定ファイルで選択して指定します。このファイルを作成し、項目を選択して指定するプロセスは、完了までに 30 分以上かかる場合があります。このタスクを 1 回完了したら、すべてのインスタンスで設定ファイルを再利用できます。

ステップ

ステップ 1: CloudWatch Logs を有効化する

JSON ファイルの先頭で、`IsEnabled` の「false」を「true」に変更します。

```
"IsEnabled": true,
```

ステップ 2: CloudWatch 設定の構成

認証情報、リージョン、ロググループ名、およびログストリーム名前空間を指定します。これにより、インスタンスがログデータを CloudWatch Logs に送信できます。同じログデータを複数の異なる宛先に送信する場合は、一意の ID (たとえば「CloudWatchLogs2」および「CloudWatchLogs3」) および各 ID に異なるリージョンを付加してセクションを追加できます。

ログデータを CloudWatch Logs に送信するように設定するには

1. JSON ファイルで、`CloudWatchLogs` セクションを見つけます。

```
{
    "Id": "CloudWatchLogs",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "AccessKey": "",
        "SecretKey": "",
        "Region": "us-east-1",
        "LogGroup": "Default-Log-Group",
```

```
        "LogStream": "{instance_id}"
    }
},
```

2. [AccessKey] および [SecretKey] フィールドは空白のままにしておきます。IAM ロールを使用して認証情報を設定します。
3. Region には、ログデータを送信するリージョンを入力します (たとえば、us-east-2)。
4. LogGroup には、ロググループの名前を入力します。この名前は、CloudWatch コンソールの [ロググループ] 画面に表示されます。
5. LogStream には、送信先のログストリームを入力します。この名前は、CloudWatch コンソールの [ロググループ] > [ストリーム] 画面に表示されます。

   デフォルトの {instance_id} を使用する場合、ログストリーム名はこのインスタンスのインスタンス ID です。

   存在しないログストリーム名を特定すると、CloudWatch Logs によってログストリームが自動的に作成されます。リテラル文字列、事前定義された変数 {instance_id}、{hostname}、{ip_address}、またはこれらの組み合わせを使用してログストリーム名を定義できます。

ステップ 3: 送信するデータを設定する

イベントログデータ、Event Tracing for Windows (ETW) データ、および他のログデータを CloudWatch Logs に送信できます。

Windows アプリケーションイベントログデータを CloudWatch Logs に送信するには

1. JSON ファイルで、ApplicationEventLog セクションを見つけます。

```
{
    "Id": "ApplicationEventLog",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "Application",
        "Levels": "1"
    }
},
```

2. Levels には、アップロードするメッセージのタイプを指定します。次のいずれかの値を指定できます。

   - **1** - エラーメッセージだけをアップロードします。
   - **2** - 警告メッセージだけをアップロードします。
   - **4** - 情報メッセージだけをアップロードします。

   値を組み合わせて複数のタイプのメッセージを含めることができます。たとえば、値 **3** はエラーメッセージ (**1**) と警告メッセージ (**2**) をアップロードします。値 **7** は、エラーメッセージ (**1**)、警告メッセージ (**2**)、情報メッセージ (**4**) をアップロードします。

セキュリティログデータを CloudWatch Logs に送信するには

1. JSON ファイルで、SecurityEventLog セクションを見つけます。

```
{
    "Id": "SecurityEventLog",
```

```
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "Security",
        "Levels": "7"
    }
},
```

2. `Levels` には、**7** と入力してすべてのメッセージをアップロードします。

システムイベントログデータを CloudWatch Logs に送信するには

1. JSON ファイルで、`SystemEventLog` セクションを見つけます。

```
{
    "Id": "SystemEventLog",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "System",
        "Levels": "7"
    }
},
```

2. `Levels` には、アップロードするメッセージのタイプを指定します。次のいずれかの値を指定できます。

   - **1** - エラーメッセージだけをアップロードします。
   - **2** - 警告メッセージだけをアップロードします。
   - **4** - 情報メッセージだけをアップロードします。

   値を組み合わせて複数のタイプのメッセージを含めることができます。たとえば、値 **3** はエラーメッセージ (**1**) と警告メッセージ (**2**) をアップロードします。値 **7** は、エラーメッセージ (**1**)、警告メッセージ (**2**)、情報メッセージ (**4**) をアップロードします。

他の種類のイベントログデータを CloudWatch Logs に送信するには

1. JSON ファイルに、新しいセクションを追加します。各セクションには固有の `Id` が必要です。

```
{
    "Id": "Id-name",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "Log-name",
        "Levels": "7"
    }
},
```

2. `Id` には、アップロードするログの名前を入力します (たとえば、**`WindowsBackup`**)。
3. `LogName` には、アップロードするログの名前を入力します。ログの名前を次のように確認できます。

   a. イベントビューワーを開きます。
   b. ナビゲーションペインで、[Applications and Services Logs] を選択します。
   c. ログに移動し、[Actions]、[Properties] を選択します。

4. `Levels` には、アップロードするメッセージのタイプを指定します。次のいずれかの値を指定できます。

- **1** - エラーメッセージだけをアップロードします。
- **2** - 警告メッセージだけをアップロードします。
- **4** - 情報メッセージだけをアップロードします。

値を組み合わせて複数のタイプのメッセージを含めることができます。たとえば、値 **3** はエラーメッセージ (**1**) と警告メッセージ (**2**) をアップロードします。値 **7** は、エラーメッセージ (**1**)、警告メッセージ (**2**)、情報メッセージ (**4**) をアップロードします。

イベントトレース (Windows) データを CloudWatch Logs に送信するには

ETW (Event Tracing for Windows) には、アプリケーションがログを書き込むことができる効率的できめ細かいログ記録メカニズムが用意されています。各 ETW は、ログ記録セッションを開始および停止できるセッションマネージャにより制御されます。各セッションには、プロバイダーと 1 つ以上のコンシューマーが存在します。

1. JSON ファイルで、`ETW` セクションを見つけます。

```
{
    "Id": "ETW",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "Microsoft-Windows-WinINet/Analytic",
        "Levels": "7"
    }
},
```

2. `LogName` には、アップロードするログの名前を入力します。
3. `Levels` には、アップロードするメッセージのタイプを指定します。次のいずれかの値を指定できます。

- **1** - エラーメッセージだけをアップロードします。
- **2** - 警告メッセージだけをアップロードします。
- **4** - 情報メッセージだけをアップロードします。

値を組み合わせて複数のタイプのメッセージを含めることができます。たとえば、値 **3** はエラーメッセージ (**1**) と警告メッセージ (**2**) をアップロードします。値 **7** は、エラーメッセージ (**1**)、警告メッセージ (**2**)、情報メッセージ (**4**) をアップロードします。

カスタムログ (テキストベースのログファイル) を CloudWatch Logs に送信するには

1. JSON ファイルで、`CustomLogs` セクションを見つけます。

```
{
    "Id": "CustomLogs",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogDirectoryPath": "C:\\CustomLogs\\",
        "TimestampFormat": "MM/dd/yyyy HH:mm:ss",
        "Encoding": "UTF-8",
        "Filter": "",
        "CultureName": "en-US",
        "TimeZoneKind": "Local",
        "LineCount": "5"
```

```
    }
},
```

2. `LogDirectoryPath` には、ログがインスタンスに格納されるパスを入力します。
3. `TimestampFormat` には、使用するタイムスタンプ形式を入力します。サポートされる値の詳細については、MSDN の「カスタムの日付と時刻の書式指定文字列」を参照してください。

   Important

   ソースログファイルには、各ログ行の先頭にタイムスタンプがあり、タイムスタンプの後にスペースがある必要があります。
4. `Encoding` には、使用するファイルエンコード (たとえば、UTF-8) を入力します。サポートされる値の一覧については、MSDN の「Encoding クラス」を参照してください。

   Note

   表示名ではなく、エンコード名を使用します。
5. (オプション) `Filter` には、ログファイル名のプレフィックスを入力します。すべてのファイルをモニタリングするには、このパラメータを空白のままにします。サポートされる値の詳細については、MSDN の「FileSystemWatcherFilter プロパティ」を参照してください。
6. (オプション) `CultureName` には、タイムスタンプが記録されているロケールを入力します。`CultureName` が空の場合、Windows インスタンスにより現在使用されているのと同じロケールがデフォルトになります。詳細については、MSDN の「Product Behavior」トピックの表で、`Language tag` 列を参照してください。

   Note

   `div`、`div-MV`、`hu`、および `hu-HU` 値は、サポートされていません。
7. (オプション) `TimeZoneKind` には、`Local` または `UTC` を入力します。これを設定すると、ログのタイムスタンプにタイムゾーン情報が含まれていない場合にタイムゾーン情報を提供できます。このパラメータが空になっていて、タイムスタンプにタイムゾーン情報が含まれていない場合、CloudWatch Logs ではデフォルトでローカルタイムゾーンが使用されます。タイムスタンプに既にタイムゾーン情報が含まれている場合、このパラメータは無視されます。
8. (オプション) `LineCount` には、ログファイルを識別するためのヘッダーの行数を入力します。たとえば、IIS のログファイルのヘッダーはほぼ同じです。「`5`」と入力すると、ログファイルのヘッダーの最初の 3 行が読み取られ、ログファイルを識別できます。IIS ログファイルで、3 番目の行は日時のスタンプですが、タイムスタンプは常にログファイル間で異なるという保証はありません。そのため、ログファイルの一意のフィンガープリントを作成するには、実際のログデータの少なくとも 1 行を含めることをお勧めします。

IIS ログデータを CloudWatch Logs に送信するには

1. JSON ファイルで、`IISLog` セクションを見つけます。

```
{
    "Id": "IISLogs",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogDirectoryPath": "C:\\inetpub\\logs\\LogFiles\\W3SVC1",
        "TimestampFormat": "yyyy-MM-dd HH:mm:ss",
        "Encoding": "UTF-8",
        "Filter": "",
        "CultureName": "en-US",
        "TimeZoneKind": "UTC",
        "LineCount": "5"
    }
},
```

2. `LogDirectoryPath` には、個々のサイト (たとえば、`C:\inetpub\logs\LogFiles\W3SVC`*n*) の IIS ログが格納されているフォルダを入力します。

   Note

   W3C ログ形式のみサポートされます。IIS、NCSA、カスタム形式はサポートされません。

3. `TimestampFormat` には、使用するタイムスタンプ形式を入力します。サポートされる値の詳細については、MSDN の「カスタムの日付と時刻の書式指定文字列」を参照してください。
4. `Encoding` には、使用するファイルエンコード (たとえば、UTF-8) を入力します。サポートされる値の詳細については、MSDN の「Encoding クラス」を参照してください。

   Note

   表示名ではなく、エンコード名を使用します。

5. (オプション) `Filter` には、ログファイル名のプレフィックスを入力します。すべてのファイルをモニタリングするには、このパラメータを空白のままにします。サポートされる値の詳細については、MSDN の「FileSystemWatcherFilter プロパティ」を参照してください。
6. (オプション) `CultureName` には、タイムスタンプが記録されているロケールを入力します。`CultureName` が空の場合、Windows インスタンスにより現在使用されているのと同じロケールがデフォルトになります。サポートされる値の詳細については、MSDN の「Product Behavior」トピックの表で、`Language tag` 列を参照してください。

   Note

   `div`、`div-MV`、`hu`、および `hu-HU` 値は、サポートされていません。

7. (オプション) `TimeZoneKind` には、`Local` または `UTC` を入力します。これを設定すると、ログのタイムスタンプにタイムゾーン情報が含まれていない場合にタイムゾーン情報を提供できます。このパラメータが空になっていて、タイムスタンプにタイムゾーン情報が含まれていない場合、CloudWatch Logs ではデフォルトでローカルタイムゾーンが使用されます。タイムスタンプに既にタイムゾーン情報が含まれている場合、このパラメータは無視されます。
8. (オプション) `LineCount` には、ログファイルを識別するためのヘッダーの行数を入力します。たとえば、IIS のログファイルのヘッダーはほぼ同じです。「`5`」と入力すると、ログファイルのヘッダーの最初の 5 行が読み取られ、ログファイルを識別できます。IIS ログファイルで、3 番目の行は日時のスタンプですが、タイムスタンプは常にログファイル間で異なるという保証はありません。そのため、ログファイルの一意のフィンガープリントを作成するには、実際のログデータの少なくとも 1 行を含めることをお勧めします。

ステップ 4: フロー制御を設定する

各データ型は、`Flows` セクションに対応する送信先を持っている必要があります。たとえば、カスタムログ、ETW ログ、およびシステムログを CloudWatch Logs に送信するには、`(CustomLogs,ETW,SystemEventLog),CloudWatchLogs` を `Flows` セクションに追加します。

Warning

無効なブロックを追加すると、フローがブロックされます。たとえば、ディスクメトリクスのステップを追加したが、インスタンスにディスクがない場合は、フローのすべてのステップがブロックされます。

同じログファイルを複数の宛先に送信できます。たとえば、アプリケーションログを `CloudWatchLogs` セクションで定義付けた 2 つの送信先に送信するには、`ApplicationEventLog,(CloudWatchLogs,CloudWatchLogs2)` を `Flows` セクションに追加します。

フロー制御を設定するには

1. `AWS.EC2.Windows.CloudWatch.json` ファイルで、「`Flows`」セクションを見つけます。

```
"Flows": {
```

```
    "Flows": [
      "PerformanceCounter,CloudWatch",
      "(PerformanceCounter,PerformanceCounter2), CloudWatch2",
      "(CustomLogs, ETW, SystemEventLog),CloudWatchLogs",
      "CustomLogs, CloudWatchLogs2",
      "ApplicationEventLog,(CloudWatchLogs, CloudWatchLogs2)"
    ]
}
```

2. `Flows` には、アップロードされる各データ型 (たとえば、`ApplicationEventLog`) とその送信先 (たとえば、`CloudWatchLogs`) を追加します。

ステップ 5: JSON コンテンツを保存する

JSON ファイルの編集はこれで完了です。ファイルを保存し、ファイルコンテンツをテキストエディタ内の別のウィンドウに貼り付けます。ファイルコンテンツは、この手順の後のステップで必要になります。

Systems Manager の IAM ユーザーおよびロールを作成する

インスタンスの認証情報の IAM ロールは、Systems Manager Run Command の実行時に必要になります。このロールにより、Systems Manager はインスタンス上でアクションを実行できます。必要に応じて、Systems Manager を設定して実行するための一意な IAM ユーザーアカウントを作成できます。詳細については、『AWS Systems Manager ユーザーガイド』の「Systems Manager のセキュリティロールの設定」を参照してください。既存のインスタンスに IAM ロールをアタッチする方法については、『Windows インスタンスの Amazon EC2 ユーザーガイド』の「IAM ロールをインスタンスにアタッチする」を参照してください。

Systems Manager の前提条件の確認

Systems Manager Run Command を使用して CloudWatch Logs との統合を設定する前に、インスタンスが最低要件を満たしていることを確認してください。詳細については、『AWS Systems Manager ユーザーガイド』の「Systems Manager の前提条件」を参照してください。

インターネットアクセスを確認する

CloudWatch にログとイベントのデータを送信するには、Amazon EC2 Windows Server のインスタンスとマネージドインスタンスにアウトバウンドのインターネットアクセスが必要です。インターネットアクセスの詳しい設定方法については、『Amazon VPC ユーザーガイド』の「インターネットゲートウェイ」を参照してください。

Systems Manager Run Command を使用した CloudWatch Logs の有効化

Run Command では、インスタンスの設定をオンデマンドで管理できます。Systems Manager ドキュメントを指定してパラメータを指定し、1 つ以上のインスタンスでコマンドを実行します。インスタンスの SSM エージェントは、コマンドを処理し、指定されたとおりにインスタンスを設定します。

Run Command を使用して CloudWatch Logs との統合を設定するには

1. https://console.aws.amazon.com/ec2/) にある Amazon EC2 コンソールを開きます。
2. ナビゲーションペインで、[Systems Managerサービス]、[Run Command] を選択します。
3. [Run a command] を選択します。
4. [Command document] で、[AWS-ConfigureCloudWatch] を選択します。
5. [Target instances (ターゲットインスタンス)] で、CloudWatch Logs と統合するインスタンスを選択します。このリストに表示されていないインスタンスは、Run Command 用に設定されていない場合があります。詳細については、『Windows インスタンスの Amazon EC2 ユーザーガイド』の「Systems Manager の前提条件」を参照してください。

6. [Status] で、[Enabled] を選択します。
7. [Properties] で、前のタスクで作成した JSON の内容をコピーして貼り付けます。
8. 残りのオプションフィールドを入力し、[Run] を選択します。

次の手順を使用して、Amazon EC2 コンソールでコマンドの実行結果を表示します。

コンソールでコマンド出力を表示するには

1. コマンドを選択します。
2. [Output] タブを選択します。
3. [View Output] を選択します。コマンド出力ページには、コマンドの実行結果が表示されます。

クイックスタート: Windows Server 2012 および Windows Server 2008 を実行している Amazon EC2 インスタンスで、CloudWatch Logs へのログの送信を有効にする

> Tip
>
> CloudWatch には、EC2 インスタンスとオンプレミスサーバーからログとメトリクスの両方を収集できる新しい統合エージェントが含まれています。新しい統合 CloudWatch エージェントを使用することをお勧めします。詳細については、「CloudWatch Logs の使用開始 (p. 5)」を参照してください。
> このセクションの残りの部分では、古い CloudWatch Logs エージェントの使用について説明します。

Windows Server 2012 および Windows Server 2008 を実行している Amazon EC2 インスタンスで、CloudWatch Logs へのログの送信を有効にする

Windows Server 2012 および Windows Server 2008 を実行しているインスタンスで、CloudWatch Logs へのログの送信を有効にするには、以下のステップを実行します。

サンプル設定ファイルのダウンロード

以下のサンプル JSON ファイルをコンピュータにダウンロードします。AWS.EC2.Windows.CloudWatch.jsonこのファイルは以降のステップで編集します。

CloudWatch の JSON ファイルの設定

CloudWatch に送信するログを決めるには、JSON 設定ファイルで選択して指定します。このファイルを作成し、項目を選択して指定するプロセスは、完了までに 30 分以上かかる場合があります。このタスクを 1 回完了したら、すべてのインスタンスで設定ファイルを再利用できます。

ステップ

ステップ 1: CloudWatch Logs を有効化する

JSON ファイルの先頭で、`IsEnabled` の「false」を「true」に変更します。

```
"IsEnabled": true,
```

ステップ 2: CloudWatch 設定の構成

認証情報、リージョン、ロググループ名、およびログストリーム名前空間を指定します。これにより、インスタンスがログデータを CloudWatch Logs に送信できます。同じログデータを複数の異なる宛先に送信する場合は、一意の ID (たとえば「CloudWatchLogs2」および「CloudWatchLogs3」) および各 ID に異なるリージョンを付加してセクションを追加できます。

ログデータを CloudWatch Logs に送信するように設定するには

1. JSON ファイルで、`CloudWatchLogs` セクションを見つけます。

```
{
    "Id": "CloudWatchLogs",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "AccessKey": "",
        "SecretKey": "",
        "Region": "us-east-1",
        "LogGroup": "Default-Log-Group",
        "LogStream": "{instance_id}"
    }
},
```

2. [`AccessKey`] および [`SecretKey`] フィールドは空白のままにしておきます。IAM ロールを使用して認証情報を設定します。
3. `Region` には、ログデータを送信するリージョンを入力します (たとえば、`us-east-2`)。
4. `LogGroup` には、ロググループの名前を入力します。この名前は、CloudWatch コンソールの [ロググループ] 画面に表示されます。
5. `LogStream` には、送信先のログストリームを入力します。この名前は、CloudWatch コンソールの [ロググループ] > [ストリーム] 画面に表示されます。

   デフォルトの `{instance_id}` を使用する場合、ログストリーム名はこのインスタンスのインスタンス ID です。

   存在しないログストリーム名を特定すると、CloudWatch Logs によってログストリームが自動的に作成されます。リテラル文字列、事前定義された変数 `{instance_id}`、`{hostname}`、`{ip_address}`、またはこれらの組み合わせを使用してログストリーム名を定義できます。

ステップ 3: 送信するデータを設定する

イベントログデータ、Event Tracing for Windows (ETW) データ、および他のログデータを CloudWatch Logs に送信できます。

Windows アプリケーションイベントログデータを CloudWatch Logs に送信するには

1. JSON ファイルで、`ApplicationEventLog` セクションを見つけます。

```
{
    "Id": "ApplicationEventLog",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
```

```
    "Parameters": {
        "LogName": "Application",
        "Levels": "1"
    }
},
```

2. Levels には、アップロードするメッセージのタイプを指定します。次のいずれかの値を指定できます。

   - **1** - エラーメッセージだけをアップロードします。
   - **2** - 警告メッセージだけをアップロードします。
   - **4** - 情報メッセージだけをアップロードします。

   値を組み合わせて複数のタイプのメッセージを含めることができます。たとえば、値 **3** はエラーメッセージ (**1**) と警告メッセージ (**2**) をアップロードします。値 **7** は、エラーメッセージ (**1**)、警告メッセージ (**2**)、情報メッセージ (**4**) をアップロードします。

セキュリティログデータを CloudWatch Logs に送信するには

1. JSON ファイルで、SecurityEventLog セクションを見つけます。

```
{
    "Id": "SecurityEventLog",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "Security",
        "Levels": "7"
    }
},
```

2. Levels には、**7** と入力してすべてのメッセージをアップロードします。

システムイベントログデータを CloudWatch Logs に送信するには

1. JSON ファイルで、SystemEventLog セクションを見つけます。

```
{
    "Id": "SystemEventLog",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "System",
        "Levels": "7"
    }
},
```

2. Levels には、アップロードするメッセージのタイプを指定します。次のいずれかの値を指定できます。

   - **1** - エラーメッセージだけをアップロードします。
   - **2** - 警告メッセージだけをアップロードします。
   - **4** - 情報メッセージだけをアップロードします。

   値を組み合わせて複数のタイプのメッセージを含めることができます。たとえば、値 **3** はエラーメッセージ (**1**) と警告メッセージ (**2**) をアップロードします。値 **7** は、エラーメッセージ (**1**)、警告メッセージ (**2**)、情報メッセージ (**4**) をアップロードします。

他の種類のイベントログデータを CloudWatch Logs に送信するには

1. JSON ファイルに、新しいセクションを追加します。各セクションには固有の `Id` が必要です。

```
{
    "Id": "Id-name",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "Log-name",
        "Levels": "7"
    }
},
```

2. `Id` には、アップロードするログの名前を入力します (たとえば、**`WindowsBackup`**)。
3. `LogName` には、アップロードするログの名前を入力します。ログの名前を次のように確認できます。
   a. イベントビューワーを開きます。
   b. ナビゲーションペインで、[Applications and Services Logs] を選択します。
   c. ログに移動し、[Actions]、[Properties] を選択します。
4. `Levels` には、アップロードするメッセージのタイプを指定します。次のいずれかの値を指定できます。

   - **`1`** - エラーメッセージだけをアップロードします。
   - **`2`** - 警告メッセージだけをアップロードします。
   - **`4`** - 情報メッセージだけをアップロードします。

   値を組み合わせて複数のタイプのメッセージを含めることができます。たとえば、値 **`3`** はエラーメッセージ (**`1`**) と警告メッセージ (**`2`**) をアップロードします。値 **`7`** は、エラーメッセージ (**`1`**)、警告メッセージ (**`2`**)、情報メッセージ (**`4`**) をアップロードします。

イベントトレース (Windows) データを CloudWatch Logs に送信するには

ETW (Event Tracing for Windows) には、アプリケーションがログを書き込むことができる効率的できめ細かいログ記録メカニズムが用意されています。各 ETW は、ログ記録セッションを開始および停止できるセッションマネージャにより制御されます。各セッションには、プロバイダーと 1 つ以上のコンシューマーが存在します。

1. JSON ファイルで、`ETW` セクションを見つけます。

```
{
    "Id": "ETW",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogName": "Microsoft-Windows-WinINet/Analytic",
        "Levels": "7"
    }
},
```

2. `LogName` には、アップロードするログの名前を入力します。
3. `Levels` には、アップロードするメッセージのタイプを指定します。次のいずれかの値を指定できます。

   - **`1`** - エラーメッセージだけをアップロードします。
   - **`2`** - 警告メッセージだけをアップロードします。

- **4** - 情報メッセージだけをアップロードします。

値を組み合わせて複数のタイプのメッセージを含めることができます。たとえば、値 **3** はエラーメッセージ (**1**) と警告メッセージ (**2**) をアップロードします。値 **7** は、エラーメッセージ (**1**)、警告メッセージ (**2**)、情報メッセージ (**4**) をアップロードします。

カスタムログ (テキストベースのログファイル) を CloudWatch Logs に送信するには

1. JSON ファイルで、`CustomLogs` セクションを見つけます。

```
{
    "Id": "CustomLogs",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogDirectoryPath": "C:\\CustomLogs\\",
        "TimestampFormat": "MM/dd/yyyy HH:mm:ss",
        "Encoding": "UTF-8",
        "Filter": "",
        "CultureName": "en-US",
        "TimeZoneKind": "Local",
        "LineCount": "5"
    }
},
```

2. `LogDirectoryPath` には、ログがインスタンスに格納されるパスを入力します。
3. `TimestampFormat` には、使用するタイムスタンプ形式を入力します。サポートされる値の詳細については、MSDN の「カスタムの日付と時刻の書式指定文字列」を参照してください。

   Important

   ソースログファイルには、各ログ行の先頭にタイムスタンプがあり、タイムスタンプの後にスペースがある必要があります。
4. `Encoding` には、使用するファイルエンコード (たとえば、UTF-8) を入力します。サポートされる値の詳細については、MSDN の「Encoding クラス」を参照してください。

   Note

   表示名ではなく、エンコード名を使用します。
5. (オプション) `Filter` には、ログファイル名のプレフィックスを入力します。すべてのファイルをモニタリングするには、このパラメータを空白のままにします。サポートされる値の詳細については、MSDN の「FileSystemWatcherFilter プロパティ」を参照してください。
6. (オプション) `CultureName` には、タイムスタンプが記録されているロケールを入力します。`CultureName` が空の場合、Windows インスタンスにより現在使用されているのと同じロケールがデフォルトになります。サポートされる値の詳細については、MSDN の「Product Behavior」トピックの表で、`Language tag` 列を参照してください。

   Note

   `div`、`div-MV`、`hu`、および `hu-HU` 値は、サポートされていません。
7. (オプション) `TimeZoneKind` には、`Local` または `UTC` を入力します。これを設定すると、ログのタイムスタンプにタイムゾーン情報が含まれていない場合にタイムゾーン情報を提供できます。このパラメータが空になっていて、タイムスタンプにタイムゾーン情報が含まれていない場合、CloudWatch Logs ではデフォルトでローカルタイムゾーンが使用されます。タイムスタンプに既にタイムゾーン情報が含まれている場合、このパラメータは無視されます。
8. (オプション) `LineCount` には、ログファイルを識別するためのヘッダーの行数を入力します。たとえば、IIS のログファイルのヘッダーはほぼ同じです。「**5**」と入力すると、ログファイルのヘッダー

の最初の 3 行が読み取られ、ログファイルを識別できます。IIS ログファイルで、3 番目の行は日時のスタンプですが、タイムスタンプは常にログファイル間で異なるという保証はありません。そのため、ログファイルの一意のフィンガープリントを作成するには、実際のログデータの少なくとも 1 行を含めることをお勧めします。

IIS ログデータを CloudWatch Logs に送信するには

1. JSON ファイルで、`IISLog` セクションを見つけます。

```
{
    "Id": "IISLogs",
    "FullName":
 "AWS.EC2.Windows.CloudWatch.CustomLog.CustomLogInputComponent,AWS.EC2.Windows.CloudWatch",
    "Parameters": {
        "LogDirectoryPath": "C:\\inetpub\\logs\\LogFiles\\W3SVC1",
        "TimestampFormat": "yyyy-MM-dd HH:mm:ss",
        "Encoding": "UTF-8",
        "Filter": "",
        "CultureName": "en-US",
        "TimeZoneKind": "UTC",
        "LineCount": "5"
    }
},
```

2. `LogDirectoryPath` には、個々のサイト (たとえば、`C:\inetpub\logs\LogFiles\W3SVC`*n*) の IIS ログが格納されているフォルダを入力します。

   Note

   W3C ログ形式のみサポートされます。IIS、NCSA、カスタム形式はサポートされません。

3. `TimestampFormat` には、使用するタイムスタンプ形式を入力します。サポートされる値の詳細については、MSDN の「カスタムの日付と時刻の書式指定文字列」を参照してください。
4. `Encoding` には、使用するファイルエンコード (たとえば、UTF-8) を入力します。サポートされる値の詳細については、MSDN の「Encoding クラス」を参照してください。

   Note

   表示名ではなく、エンコード名を使用します。

5. (オプション) `Filter` には、ログファイル名のプレフィックスを入力します。すべてのファイルをモニタリングするには、このパラメータを空白のままにします。サポートされる値の詳細については、MSDN の「FileSystemWatcherFilter プロパティ」を参照してください。
6. (オプション) `CultureName` には、タイムスタンプが記録されているロケールを入力します。`CultureName` が空の場合、Windows インスタンスにより現在使用されているのと同じロケールがデフォルトになります。サポートされる値の詳細については、MSDN の「Product Behavior」トピックの表で、`Language tag` 列を参照してください。

   Note

   `div`、`div-MV`、`hu`、および `hu-HU` 値は、サポートされていません。

7. (オプション) `TimeZoneKind` には、`Local` または `UTC` を入力します。これを設定すると、ログのタイムスタンプにタイムゾーン情報が含まれていない場合にタイムゾーン情報を提供できます。このパラメータが空になっていて、タイムスタンプにタイムゾーン情報が含まれていない場合、CloudWatch Logs ではデフォルトでローカルタイムゾーンが使用されます。タイムスタンプに既にタイムゾーン情報が含まれている場合、このパラメータは無視されます。
8. (オプション) `LineCount` には、ログファイルを識別するためのヘッダーの行数を入力します。たとえば、IIS のログファイルのヘッダーはほぼ同じです。「`5`」と入力すると、ログファイルのヘッダーの最初の 5 行が読み取られ、ログファイルを識別できます。IIS ログファイルで、3 番目の行は日時のスタンプですが、タイムスタンプは常にログファイル間で異なるという保証はありません。そのた

め、ログファイルの一意のフィンガープリントを作成するには、実際のログデータの少なくとも 1 行を含めることをお勧めします。

ステップ 4: フロー制御を設定する

各データ型は、`Flows` セクションに対応する送信先を持っている必要があります。たとえば、カスタムログ、ETW ログ、およびシステムログを CloudWatch Logs に送信するには、`(CustomLogs,ETW,SystemEventLog),CloudWatchLogs` を `Flows` セクションに追加します。

> Warning
>
> 無効なブロックを追加すると、フローがブロックされます。たとえば、ディスクメトリクスのステップを追加したが、インスタンスにディスクがない場合は、フローのすべてのステップがブロックされます。

同じログファイルを複数の宛先に送信できます。たとえば、アプリケーションログを `CloudWatchLogs` セクションで定義付けた 2 つの送信先に送信するには、`ApplicationEventLog,(CloudWatchLogs,CloudWatchLogs2)` を `Flows` セクションに追加します。

フロー制御を設定するには

1. `AWS.EC2.Windows.CloudWatch.json` ファイルで、「`Flows`」セクションを見つけます。

```
"Flows": {
    "Flows": [
      "PerformanceCounter,CloudWatch",
      "(PerformanceCounter,PerformanceCounter2), CloudWatch2",
      "(CustomLogs, ETW, SystemEventLog),CloudWatchLogs",
      "CustomLogs, CloudWatchLogs2",
      "ApplicationEventLog,(CloudWatchLogs, CloudWatchLogs2)"
    ]
}
```

2. `Flows` には、アップロードされる各データ型 (たとえば、`ApplicationEventLog`) とその送信先 (たとえば、`CloudWatchLogs`) を追加します。

JSON ファイルの編集はこれで完了です。これは、後のステップで使用します。

エージェントの開始

Windows Server 2012 または Windows Server 2008 を実行している Amazon EC2 インスタンスが、CloudWatch Logs にログを送信できるようにするには、EC2Config サービス (`EC2Config.exe`) を使用します。インスタンスには EC2Config 4.0 以降が必要であり、この手順を使用できます。以前のバージョンの EC2Config を使用する詳細については、『Windows インスタンスの Amazon EC2 ユーザーガイド』の「EC2Config 3.x 以前を使用して CloudWatch を設定する」を参照してください。

EC2Config 4.x を使用して CloudWatch を設定するには

1. この手順で前に編集した AWS.EC2.Windows.CloudWatch.json ファイルのエンコーディングを確認します。BOM のない UTF-8 エンコーディングのみがサポートされています。次に、Windows Server 2008 - 2012 R2 インスタンスで、`C:\Program Files\Amazon\SSM\Plugins\awsCloudWatch\` フォルダにファイルを保存します。
2. Windows サービスのコントロールパネルを使用するか、次の PowerShell コマンドを送信して、SSM エージェント (`AmazonSSMAgent.exe`) を開始または再起動します。

```
PS C:\> Restart-Service AmazonSSMAgent
```

SSM エージェントは、再起動後に設定ファイルを検出し、CloudWatch 統合のためにインスタンスを設定します。ローカル設定ファイルのパラメータと設定を変更する場合は、変更を反映するために SSM エージェントを再起動する必要があります。インスタンスで CloudWatch 統合を無効にする場合は、`IsEnabled` を `false` に変更して、設定ファイルで変更を保存します。

クイックスタート: CloudWatch Logs および Chef を使用して AWS OpsWorks エージェントをインストールする

CloudWatch Logs および Chef を使用して AWS OpsWorks エージェントをインストールしログストリームを作成できます。Chef はサードパーティーのシステムであり、クラウドインフラストラクチャのオートメーションツールです。Chef は、コンピューターにソフトウェアをインストールして設定するために記述する「レシピ」と、レシピのコレクションである「クックブック」を使用して、設定とポリシーの配布タスクを実行します。詳細については、「Chef」を参照してください。

以下の Chef のレシピの例は、各 EC2 インスタンスで 1 個のログファイルをモニタリングする方法を示しています。レシピでは、ロググループとしてスタック名を、ログストリーム名としてインスタンスのホスト名を使用します。複数のログファイルをモニタリングする場合は、複数のロググループとログストリームを作成するようにレシピを拡張する必要があります。

ステップ 1: カスタムレシピを作成する

レシピを保存するリポジトリを作成します。AWS OpsWorks は Git および Subversion をサポートしています。または、Amazon S3 にアーカイブを保存できます。クックブックリポジトリの構造は、『AWS OpsWorks ユーザーガイド』の「クックブックリポジトリ」で説明されています。以下の例では、クックブックの名前を `logs` と仮定します。install.rb レシピは、CloudWatch Logs エージェントをインストールします。クックブックの例をダウンロードすることもできます (CloudWatchLogs-Cookbooks.zip)。

以下のコードを含む metadata.rb というファイルを作成します。

```
#metadata.rb

name             'logs'
version          '0.0.1'
```

CloudWatch Logs 設定ファイルを作成します。

```
#config.rb

template "/tmp/cwlogs.cfg" do
  cookbook "logs"
  source "cwlogs.cfg.erb"
  owner "root"
  group "root"
  mode 0644
end
```

CloudWatch Logs エージェントをダウンロードし、インストールします。

```
# install.rb

directory "/opt/aws/cloudwatch" do
  recursive true
end

remote_file "/opt/aws/cloudwatch/awslogs-agent-setup.py" do
  source "https://s3.amazonaws.com//aws-cloudwatch/downloads/latest/awslogs-agent-setup.py"
```

```
  mode "0755"
end

  execute "Install CloudWatch Logs agent" do
  command "/opt/aws/cloudwatch/awslogs-agent-setup.py -n -r region -c /tmp/cwlogs.cfg"
  not_if { system "pgrep -f aws-logs-agent-setup" }
end
```

Note

上の例で、*region* を次のいずれかに置き換えてください。us-east-1, us-west-1, us-west-2, ap-south-1, ap-northeast-2, ap-southeast-1, ap-southeast-2, ap-northeast-1, eu-central-1, eu-west-1, or sa-east-1。

エージェントのインストールが失敗した場合は、python-dev パッケージがインストールされていることを確認します。そうでない場合は、次のコマンドを使用して、エージェントのインストールを再試行します。

```
sudo apt-get -y install python-dev
```

このレシピでは cwlogs.cfg.erb テンプレートファイルを使用しています。このファイルを変更してどのようなファイルを記録するかなど様々な属性を指定できます。これらの属性の詳細については、「CloudWatch Logs エージェントのリファレンス (p. 122)」を参照してください。

```
[general]
# Path to the AWSLogs agent's state file. Agent uses this file to maintain
# client side state across its executions.
state_file = /var/awslogs/state/agent-state


## Each log file is defined in its own section. The section name doesn't
## matter as long as its unique within this file.
#
#[kern.log]
#
## Path of log file for the agent to monitor and upload.
#
#file = /var/log/kern.log
#
## Name of the destination log group.
#
#log_group_name = kern.log
#
## Name of the destination log stream.
#
#log_stream_name = {instance_id}
#
## Format specifier for timestamp parsing.
#
#datetime_format = %b %d %H:%M:%S
#
#

[<%= node[:opsworks][:stack][:name] %>]
datetime_format = [%Y-%m-%d %H:%M:%S]
log_group_name = <%= node[:opsworks][:stack][:name].gsub(' ','_') %>
file = <%= node[:cwlogs][:logfile] %>
log_stream_name = <%= node[:opsworks][:instance][:hostname] %>
```

テンプレートは、スタック設定およびデプロイメント JSON の対応する属性を参照してスタック名およびホスト名を取得します。記録するファイルを指定する属性は cwlogs クックブックの default.rb 属性ファイル (logs/attributes/default.rb) で定義されます。

```
default[:cwlogs][:logfile] = '/var/log/aws/opsworks/opsworks-agent.statistics.log'
```

ステップ 2: AWS OpsWorks スタックを作成する

1. https://console.aws.amazon.com/opsworks/ にある AWS OpsWorks コンソールを開きます。
2. [OpsWorks Dashboard] で [Add stack (スタックを追加)] を選択して、AWS OpsWorks スタックを作成します。
3. [Add stack] 画面で [Chef 11 stack] を選択します。
4. [Stack name] に、名前を入力します。
5. [Use custom Chef Cookbooks] で、[Yes] を選択します。
6. [Repository type] で、使用するリポジトリのタイプを選択します。上記の例を使用する場合は、[Http Archive] を選択します。
7. [Repository URL] に、前のステップで作成したクックブックを保存したリポジトリを入力します。上記の例を使用する場合は、「**`https://s3.amazonaws.com//aws-cloudwatch/downloads/CloudWatchLogs-Cookbooks.zip`**」と入力します。
8. [Add Stack] を選択し、スタックを作成します。

ステップ 3: IAM ロールを拡張する

CloudWatch Logs インスタンスで AWS OpsWorks を使用するには、インスタンスで使用されている IAM ロールを拡張する必要があります。

1. https://console.aws.amazon.com/iam/ にある IAM コンソールを開きます。
2. ナビゲーションペインで、[Policies]、[Create Policy] の順に選択します。
3. [Create Policy] ページの [Create Your Own Policy] で、[Select] を選択します。カスタムポリシーの作成の詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「Amazon EC2 の IAM ポリシー」を参照してください。
4. [Review Policy] ページで、[Policy Name] にポリシーの名前を入力します。
5. [Policy Document] に、次のポリシーをコピーして貼り付けます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
    ],
    "Resource": [
      "arn:aws:logs:*:*:*"
    ]
  }
 ]
}
```

6. [Create Policy] を選択します。
7. ナビゲーションペインで [ロール] を選択し、コンテンツペインの [ロール名] で、AWS OpsWorks スタックが使用しているインスタンスロールの名前を選択します。スタックの設定でスタックが使用しているロールが見つかります (デフォルトは `aws-opsworks-ec2-role` です) 。

Note

チェックボックスではなく、ロールの名前を選択します。

8. [Permissions] タブを開き、[Managed Policies] で [Attach Policy] を選択します。
9. [Attach Policy] ページのテーブルヘッダー ([Filter] と [Search] の横) で、[Policy Type]、[Customer Managed Policies] を選択します。
10. [カスタマー管理ポリシー] で、上で作成した IAM ポリシーを選択し、[ポリシーのアタッチ] を選択します。

IAM のユーザーとポリシーの詳細については、『IAM ユーザーガイド』の「IAM ユーザーとグループ」と「IAM ポリシーを管理する」を参照してください。

ステップ 4: Layer を追加する

1. https://console.aws.amazon.com/opsworks/ にある AWS OpsWorks コンソールを開きます。
2. ナビゲーションペインで [Layers] を選択します。
3. コンテンツペインでレイヤーを選択し、[Add layer] を選択します。
4. [OpsWorks] タブの [Layer type] で [Custom] を選択します。
5. [Name] および [Short name] フィールドにレイヤーの長い名前と短い名前を入力し、[Add layer] を選択します。
6. [レシピ] タブの [Custom Chef Recipes (カスタムシェフレシピ)] には、AWS OpsWorks ライフサイクルイベントに対応するいくつかの見出し ([セットアップ]、[設定]、[デプロイ]、[デプロイ解除]、および [シャットダウン]) があります。AWS OpsWorks はインスタンスのライフサイクルにおいてこれらのキーポイントでイベントをトリガーし、それにより関連レシピが実行されます。

   Note

   上記のヘッダーが非表示の場合、[Custom Chef Recipes] の [edit] を選択します。
7. [Setup] の隣に「logs::config, logs::install」と入力し、[+] を選択してリストに追加します。次に [Save] を選択します。

   AWS OpsWorks はこの Layer の新しいインスタンスごとに、インスタンスの起動直後にこのレシピを実行します。

ステップ 5: インスタンスを追加する

この Layer はインスタンスの設定方法のみを制御しています。次は、Layer にいくつかインスタンスを追加し、起動する必要があります。

1. https://console.aws.amazon.com/opsworks/ にある AWS OpsWorks コンソールを開きます。
2. ナビゲーションペインで [Instances] を選択し、レイヤーの下にある [+ Instance] を選択します。
3. デフォルト設定を受け入れて [Add Instance] を選択し、レイヤー にインスタンスを追加します。
4. 行の [Actions] 列で [start] をクリックしてインスタンスを起動します。

   AWS OpsWorks が新しい EC2 インスタンスを起動し、CloudWatch Logs を設定します。準備ができると、インスタンスのステータスがオンラインに変更されます。

ステップ 6: ログを表示する

エージェントが実行されてしばらくしたら、CloudWatch コンソールに新しく作成されたロググループとログストリームを確認してください。

詳細については、「CloudWatch Logs に送信されたログデータの表示 (p. 50)」を参照してください。

CloudWatch Logs エージェントのステータスのレポート

EC2 インスタンスでの CloudWatch Logs エージェントのステータスをレポートするには、以下の手順を使用します。

エージェントのステータスをレポートするには

1. EC2 インスタンスに接続します。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「インスタンスへの接続」を参照してください。

   問題に接続する方法の詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「インスタンスへの接続に関するトラブルシューティング」を参照してください。
2. コマンドプロンプトで、次のコマンドを入力します。

   ```
   sudo service awslogs status
   ```

   Amazon Linux 2 を実行している場合は、次のコマンドを入力します。

   ```
   sudo service awslogsd status
   ```
3. /var/log/awslogs.log ファイルで CloudWatch Logs エージェントのエラー、警告、問題を確認してください。

CloudWatch Logs エージェントの開始

EC2 インスタンスの CloudWatch Logs エージェントがインストール後自動的に開始しなかった場合、またはエージェントを停止した場合、以下の手順を使用してエージェントを開始できます。

エージェントを開始するには

1. EC2 インスタンスに接続します。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「インスタンスへの接続」を参照してください。

   問題に接続する方法の詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「インスタンスへの接続に関するトラブルシューティング」を参照してください。
2. コマンドプロンプトで、次のコマンドを入力します。

   ```
   sudo service awslogs start
   ```

   Amazon Linux 2 を実行している場合は、次のコマンドを入力します。

   ```
   sudo service awslogsd start
   ```

CloudWatch Logs エージェントの停止

EC2 インスタンスで CloudWatch Logs エージェントを停止するには、以下の手順を使用します。

エージェントを停止するには

1. EC2 インスタンスに接続します。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「インスタンスへの接続」を参照してください。

   問題に接続する方法の詳細については、『Linux インスタンス用 Amazon EC2 ユーザーガイド』の「インスタンスへの接続に関するトラブルシューティング」を参照してください。
2. コマンドプロンプトで、次のコマンドを入力します。

```
sudo service awslogs stop
```

   Amazon Linux 2 を実行している場合は、次のコマンドを入力します。

```
sudo service awslogsd stop
```

クイックスタート: CloudWatch Logs を使用して AWS CloudFormation を開始する

AWS CloudFormation を使って、JSON 形式で AWS リソースを記述し、プロビジョニングできます。この方法の利点には、AWS リソースのコレクションを単一のユニットとして管理し、AWS リソースをリージョン間で簡単にレプリケートできることなどがあります。

AWS CloudFormation を使用して AWS をプロビジョニングする場合は、使用する AWS リソースを記述するテンプレートを作成します。次の例は、ロググループと、404 の発生数をカウントし、この数をロググループに送信するメトリクスフィルタを作成するテンプレートスニペットです。

```
"WebServerLogGroup": {
    "Type": "AWS::Logs::LogGroup",
    "Properties": {
        "RetentionInDays": 7
    }
},

"404MetricFilter": {
    "Type": "AWS::Logs::MetricFilter",
    "Properties": {
        "LogGroupName": {
            "Ref": "WebServerLogGroup"
        },
        "FilterPattern": "[ip, identity, user_id, timestamp, request, status_code = 404,
 size, ...]",
        "MetricTransformations": [
            {
                "MetricValue": "1",
                "MetricNamespace": "test/404s",
                "MetricName": "test404Count"
            }
        ]
    }
}
```

これは基本的な例です。AWS CloudFormation を使用して、より機能が豊富な CloudWatch Logs デプロイを設定できます。テンプレートの例の詳細については、『AWS CloudFormation ユーザーガイド』の「Amazon CloudWatch Logs テンプレートスニペット」を参照してください。開始方法については、『AWS CloudFormation ユーザーガイド』の「AWS CloudFormation の使用開始」を参照してください。

CloudWatch Logs Insights でログデータを分析する

CloudWatch Logs Insights では、Amazon CloudWatch Logs のログデータをインタラクティブに検索して分析できます。クエリを実行することで、運用上の問題にすばやく効果的に対応できます。問題が発生した場合は、CloudWatch Logs Insights を使用して潜在的原因を特定し、デプロイした修正を検証できます。

CloudWatch Logs Insights には専用のクエリ言語といくつかのシンプルで強力なコマンドが含まれています。また、サンプルクエリ、コマンドの説明、クエリの自動補完、ログフィールドの検出を使用して CloudWatch Logs Insights を簡単に開始できます。サンプルクエリは、AWS のサービスの複数のログタイプ向けに用意されています。

CloudWatch Logs Insights は、AWS のサービス (Amazon Route 53、AWS Lambda、AWS CloudTrail、Amazon VPC など) のログ、およびログイベントを JSON として生成するアプリケーションログやカスタムログのフィールドを自動的に検出します。

CloudWatch Logs Insights では、2018 年 11 月 5 日以降に CloudWatch Logs に送信されたログデータを検索できます。

> **Important**
>
> ネットワークセキュリティチームがウェブソケットの使用を許可しない場合は、現在 CloudWatch Logs Insights コンソールにアクセスすることはできません。CloudWatch Logs Insights のクエリ機能は API を通じて使用できます。詳細については、Amazon CloudWatch Logs API Reference の「StartQuery」を参照してください。

目次

サポートされるログと検出されるフィールド

CloudWatch Logs Insights はすべてのタイプのログをサポートします。CloudWatch Logs に送信されるログごとに、以下の 3 つのシステムフィールドが自動的に生成されます。

- `@message` は、生の未解析のログイベントを示します。
- `@timestamp` は、ログイベントが CloudWatch Logs に追加された時間を示します。
- `@logStream` は、ログイベントの追加先のログストリームの名前を示します。

CloudWatch Logs Insights では、生成されるフィールドの先頭に [@] 記号を挿入します。

CloudWatch Logs は、多くのログタイプで、ログ内のログフィールドも自動的に検出します。これらの自動検出フィールドを以下の表に示します。

CloudWatch Logs Insights が自動的に検出しないフィールドを持つ他のログタイプについては、**parse** コマンドを使用してエフェメラルフィールドを抽出および作成してクエリで使用できます。詳細については、「CloudWatch Logs Insights クエリ構文 (p. 39)」を参照してください。

検出されたログフィールドの名前が @ 文字で始まる場合は、この名前の先頭に @ が追加されて CloudWatch Logs Insights に表示されます。たとえば、ログフィールド名が `@example.com` である場合、このフィールド名は `@@example.com` と表示されます。

ログタイプ	検出されるログフィールド
Amazon VPC フローログ	`@timestamp`、`@logStream`、`@message`、`accountId`、`endTime`、`interfaceId`、`logStatu`
Route 53 ログ	`@timestamp`、`@logStream`、`@message`、`edgeLocation`、`hostZoneId`、`protocol`、`query`
Lambda ログ	`@timestamp`、`@logStream`、`@message`、`@requestId`、`@duration`, 、`@billedDuration`、`@type`、`@maxMemoryUsed`、`@memorySize` CloudWatch Logs Insights は、Lambda ログのログフィールドを自動的に検出します。ただし、各ログイベントに埋め込まれている最初の JSON フラグメントのみが検出されます。Lambda ログイベントに複数の JSON フラグメントが含まれている場合は、**parse** コマンドを使用してログフィールドを解析して抽出できます。詳細については、「JSON ログのフィールド (p. 35)」を参照してください。
CloudTrail ログ JSON 形式のログ	詳細については、「JSON ログのフィールド (p. 35)」を参照してください。
その他のログタイプ	`@timestamp`、`@logStream`、`@message`。

JSON ログのフィールド

CloudWatch Logs Insights は、ネストされた JSON フィールドをドット表記で表します。次の JSON イベントの例で、JSON オブジェクト `userIdentity` のフィールド `type` は、`userIdentity.type` と表されています。

JSON 配列はフィールド名と値のリストにフラット化されます。たとえば、`requestParameters.instancesSet` の最初の項目に対して `instanceId` の値を指定するには、`requestParameters.instancesSet.items.0.instanceId` を使用します。

```
{ "eventVersion": "1.0",
"userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn: aws: iam: : 123456789012: user/Alice",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "accountId": "123456789012",
    "userName": "Alice"
},
"eventTime": "2014-03-06T21: 22: 54Z",
"eventSource": "ec2.amazonaws.com",
"eventName": "StartInstances",
"awsRegion": "us-east-2",
"sourceIPAddress": "205.251.233.176",
"userAgent": "ec2-api-tools1.6.12.2",
"requestParameters": {
    "instancesSet": {
        "items": [
```

```
            {
                "instanceId": "i-abcde123"
            }
        ]
    }
},
"responseElements": {
    "instancesSet": {
        "items": [
            {
                "instanceId": "i-abcde123",
                "currentState": {
                    "code": 0,
                    "name": "pending"
                },
                "previousState": {
                    "code": 80,
                    "name": "stopped"
                }
            }
        ]
    }
}
```

チュートリアル: サンプルクエリを実行および変更する

次のチュートリアルに従って CloudWatch Logs Insights の使用を開始できます。サンプルクエリを実行し、次にこのクエリを変更して再実行する方法を示します。

クエリを実行するには、CloudWatch Logs に保存済みのログが必要です。CloudWatch Logs をすでに使用していて、ロググループとログストリームが設定済みである場合は、開始する準備が整っています。AWS CloudTrail、Amazon Route 53、Amazon VPC などのサービスを使用していて、これらのサービスのログを CloudWatch Logs に送信するように設定済みであれば、すでにログも保存されている可能性があります。CloudWatch Logs にログを送信する方法の詳細については、「CloudWatch Logs の使用開始 (p. 5)」を参照してください。

CloudWatch Logs Insights のクエリは、ログイベントから一連のフィールドを返すか、ログイベントに対して実行された数学的な集約やその他のオペレーションの結果を返します。このチュートリアルでは、ログイベントのリストを返すクエリを示します。

サンプルクエリを実行する

最初にサンプルクエリを実行します。

CloudWatch Logs Insights のサンプルクエリを実行するには

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、[Insights] を選択します。

   画面の上部にクエリエディタがあります。CloudWatch Logs Insights を初めて開くと、このボックスにデフォルトクエリが表示されます。デフォルトでは、最新の 20 件のログイベントが返されます。
3. クエリエディタの上で、クエリを実行する対象のロググループを選択します。

   ロググループを選択すると、CloudWatch Logs Insights はロググループのデータのフィールドを自動的に検出し、右側のペインの [Discovered fields (検出済みフィールド)] に表示します。また、このロ

グループのログイベントを時間の経過に従って棒グラフで表示します。この棒グラフは、表に示されるイベントだけでなく、クエリと時間範囲に一致するロググループ内のイベントの分布を示します。

4. [クエリの実行] を選択します。

   クエリの結果が表示されます。この例では、タイプを問わず、最新の 20 件のログイベントが結果として表示されます。

5. 返されたログイベントのいずれかについて、すべてのフィールドを表示するには、そのログイベントの左側にある矢印を選択します。

サンプルクエリを変更する

このチュートリアルでは、サンプルクエリを変更して、最新のログイベントを 50 件表示します。

前のチュートリアルをまだ実行していない場合は、今すぐ実行してください。このチュートリアルは、前のチュートリアルが終了した箇所から開始します。

> Note
>
> CloudWatch Logs Insights に用意されている一部のサンプルクエリでは、**`limit`** の代わりに `head` コマンドや `tail` コマンドを使用しています。これらのコマンドは廃止済みであり、**`limit`** に置き換えられています。**`limit`** を使用し、`tail in all queries you write.` や `head` は使用しないでください。

CloudWatch Logs Insights のサンプルクエリを変更するには

- クエリエディタで、[20] を [50] に変更します。[クエリの実行] を選択します。

  新しいクエリの結果が表示されます。デフォルトの時間範囲でロググループに十分なデータがあるとして、これで 50 件のログイベントが一覧表示されます。

サンプルクエリにフィルターコマンドを追加する

このチュートリアルでは、クエリエディタを使用してクエリに対してより強力な変更を行う方法を示します。このチュートリアルでは、取得したログイベントのフィールドに基づいて、前のクエリの結果をフィルタリングします。

前の 2 つのチュートリアルをまだ実行していない場合は、今すぐ実行してください。このチュートリアルは、前のチュートリアルが終了した箇所から開始します。

前のクエリにフィルターコマンドを追加するには

1. フィルタリングするフィールドを決定します。特定のログイベントに含まれているフィールドを表示するには、その行の左にある矢印を選択します。[Discovered fields (検出済みフィールド)] 領域に、このロググループが過去 15 分間に受信したログイベントにおいて、CloudWatch Logs が検出した最も一般的なフィールドと、各フィールドが表示されているログイベントの割合 (%) が表示されます。[Discovered fields (検出済みフィールド)] が表示されていない場合は、画面の右上にある左矢印を選択し、右側のパネルを開きます。

   ログ内のイベントに応じて、ログイベントに [awsRegion] フィールドが表示される場合があります。このチュートリアルの残りの部分では、フィルターフィールドとして awsRegion を使用しますが、このフィールドが使用できない場合は、別のフィールドを使用できます。

2. クエリエディタボックスで [50] の後にカーソルを置き、Enter キーを押します。
3. 新しい行で、最初に | (パイプ文字) とスペースを入力します。CloudWatch Logs Insights クエリのコマンドは、パイプ文字で区切る必要があります。

4. 「filter awsRegion="us-east-1"」と入力します。
5. [クエリの実行] を選択します。

   クエリが再度実行されます。今回は、新しいフィルターに一致する 50 件の最新の結果が表示されます。

   別のフィールドにフィルターを適用してエラーが発生した場合は、必要に応じてフィールド名をエスケープします。フィールド名に英数字以外の文字が含まれている場合は、フィールド名の前後にバックティック文字 (`) を挿入します。たとえば、`error-code`="102" のようにします。

   バックティック文字 (`) は、フィールド名に英数字以外の文字が含まれている場合に必要ですが、値には必要ありません。値は、常に二重引用符 (") で囲みます。

CloudWatch Logs Insights には、強力なクエリ機能として、複数の正規表現のコマンドとサポート、算術オペレーション、統計オペレーションなどが含まれています。詳細については、「CloudWatch Logs Insights クエリ構文 (p. 39)」を参照してください。

チュートリアル: 集計関数を使用してクエリを実行する

このチュートリアルでは、ログレコードに対して集計関数を実行した結果を返すクエリを実行します。

集約クエリを実行するには

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、[Insights] を選択します。
3. ロググループを選択します。
4. クエリエディタで、現在表示されているクエリを削除し、以下を入力して、[クエリの実行] を選択します。fieldname は、画面の右側の [Discovered fields (検出済みフィールド)] 領域に表示されているフィールドの名前に置き換えます。

   ```
   stats count(*) by fieldname
   ```

   結果として、CloudWatch Logs が受信したロググループ内のログイベントのうち、選択したフィールド名のそれぞれ異なる値が含まれているものの数が表示されます。

チュートリアル: 視覚化を生成するクエリを実行する

`bin()` 関数を使用するクエリを実行して、返された値を期間別にグループ化すると、結果を折れ線グラフやスタックされたエリアグラフとして表示できます。これにより、時間の経過に伴うログイベントの傾向を簡単に視覚化できます。

視覚化用のクエリを実行するには

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、[Insights] を選択します。
3. ロググループを選択します。
4. クエリエディタで、現在の表示内容を削除し、以下を入力して、[クエリの実行] を選択します。

```
stats count(*) by bin(30s)
```

結果として、CloudWatch Logs が 30 秒間隔で受信したロググループ内のログイベントの数が表示されます。

5. [Visualization (視覚化)] タブを選択します。

   結果が折れ線グラフとして表示されます。スタックされたエリアグラフに切り替えるには、グラフの右上で [スタックされたエリア] を選択します。

CloudWatch Logs Insights クエリ構文

CloudWatch Logs Insights は、ロググループに対するクエリの実行に使用できるクエリ言語をサポートしています。各クエリには、1 つ以上のクエリコマンドを Unix 形式のパイプ文字 (|) で区切って含めることができます。

6 つのクエリコマンドがサポートされています。さらに、正規表現、算術オペレーション、比較オペレーション、数値関数、日時関数、文字列関数、汎用関数など、多数のサポート用の関数やオペレーションが用意されています。

コメントもサポートされています。クエリ内の # 文字で始まる行は無視されます。

CloudWatch Logs Insights のクエリコマンド

次の表は、6 つのサポートされているクエリコマンドおよび基本的な例の一覧です。より強力なサンプルクエリについては、「サンプルクエリ (p. 46)」を参照してください。

コマンド	説明	例
fields	指定したフィールドをログイベントから取得しますfields コマンド内で関数とオペレーションを使用できます。	`fields ` foo-bar `, action, abs(f3-f4)` は、`foo-bar` フィールドと `action` フィールドを取得し、さらにロググループ内のすべてのログイベントについて `f3` と `f4` の差異の絶対値を取得します。 フィールド名に英数字以外の文字が含まれているため、バックティック記号 ` で `foo-bar` を囲む必要があります。クエリ内のログフィールドの名前に @ 記号、ピリオド (.)、および英数字以外の文字が含まれている場合は、ログフィールドをバックティック文字で囲む必要があります。
filter	クエリの結果を 1 つ以上の条件に基づいてフィルタリングします。比較演算子 (=、!=、<、<=、>、>=)、ブール演算子 (and、or、not) および正規表現を使用できます。	`fields f1, f2, f3 | filter (duration>2000)` は、`duration` フィールドの値が 2000 を超えるすべてのログイベントについて、`f1` フィールド、`f2` フィールド、および `f3` フィールドを取得します。 `filter (duration>2000)` も有効なクエリですが、結果には個別のフィールドが表示されません。代わりに、duration が 2000 を超えるすべてのログイベントにつ

コマンド	説明	例
		いて、@message フィールド内のすべてのログデータおよび @timestamp が結果に表示されます。 `fields f1, f2 | filter (f1=10 or f3>25)` は、`f1` が 10 であるか、`f3` が 25 を超えるすべてのログイベントについて、`f1` フィールドと `f2` フィールドを取得します。 `fields f1 | filter statusCode like /2\d\d/` は、`statusCode` フィールドの値が 200 ~ 299 であるログイベントを返します。
stats	ログフィールドの値に基づいて集約統計を計算します。統計演算子として、`avg()`、`sum()`、`count()`、`,min()`、`max()` などがサポートされています。 **stats** と `by` を併用すると、統計の計算時にデータをグループ化するための 1 つ以上の条件を指定できます。	`stats avg (f1) by f2` は、`f2` の一意な値ごとに `f1` の平均値を計算します。
sort	取得したログイベントをソートします。昇順 (`asc`) と降順 (`desc`) の両方がサポートされています。	`fields f1, f2, f3 | sort f1 desc` は、`f1`、`f2`、`f3` の各フィールドを取得し、返されたイベントを `f1` の値に基づいて降順にソートします。
limit	クエリから返されるログイベントの数を制限します。	`fields f1, f2 | sort @timestamp desc | limit 25` は、`f1` フィールドと `f2` フィールドを取得し、`@timestamp` の値に基づいてイベントを降順にソートして、ソート順で最初の 25 件のイベントを返します。この場合、ソート順は最新のタイムスタンプから開始されるため、最新の 25 件のイベントが返されます。 CloudWatch Logs Insights に用意されている一部のサンプルクエリでは、**limit** の代わりに `head` コマンドや `tail` コマンドを使用しています。これらのコマンドは廃止済みであり、**limit** に置き換えられています。**limit** を使用し、`tail in all queries you write.` や `head` は使用しないでください。

コマンド	説明	例

parse	ログフィールドからデータを抽出し、1 つ以上のエフェメラルフィールドを作成してクエリでさらに処理できるようにします。**parse** は、glob 表現と正規表現のいずれも承認します。	`parse @message "user=*, method:*, latency := *" as @user,`
		`@method, @latency | stats avg(@latency) by @method,`
		`@user`
		
		ログフィールド `@message` から、エフェメラルフィールド `@user`、`@method`、`@latency` を抽出し、`@method` と `@user` の一意な組み合わせごとに平均レイテンシーを返します。

フィルターコマンドの正規表現

filter クエリコマンドで `like` または `=~` (等号とチルダ) を使用して部分文字列または正規表現でフィルタリングできます。部分文字列の一致を実行するには、一致文字列を二重引用符または一重引用符で囲みます。正規表現一致を実行するには、一致文字列をスラッシュで囲みます。クエリは、設定した条件に一致するログイベントのみ返します。

例

以下の 3 つの例では、`f1` に単語 `Exception` が含まれているすべてのイベントを返します。最初の 2 つの例では正規表現を使用し、3 番目の例では部分文字列一致を使用します。3 つすべての例で、大文字と小文字が区別されます。

```
fields f1, f2, f3 | filter f1 like /Exception/
```

```
fields f1, f2, f3 | filter f1 =~ /Exception/
```

```
fields f1, f2, f3 | filter f1 like "Exception"
```

次の例では正規表現を使用し、`f1` に単語 `Exception` が含まれているすべてのイベントを返します。クエリの大文字と小文字は区別されません。

```
fields f1, f2, f3 | filter f1 like /(?i)Exception/
```

次の例では正規表現を使用し、`f1` に単語 `Exception` と正確に一致するすべてのイベントを返します。クエリの大文字と小文字は区別されません。

```
fields f1, f2, f3 | filter f1 =~ /^(?i)Exception$/
```

クエリでのエイリアスの使用

`as` を使用してクエリに 1 つ以上のエイリアスを作成できます。エイリアスは、**fields** コマンド、**stats** コマンド、および **sort** コマンドでサポートされています。

ログフィールドのエイリアスと、オペレーションや関数の結果のエイリアスを作成できます。

例

クエリコマンドでのエイリアスの使用例は以下のとおりです。

```
fields abs(myField) as AbsoluteValuemyField, myField2
```

myField の絶対値を AbsoluteValuemyField として返します。また、myField2 フィールドも返します。

```
stats avg(f1) as myAvgF1 | sort myAvgF1 desc
```

f1 の平均値を myAvgF1 として計算し、結果の値を降順で返します。

クエリでのコメントの使用

文字を使用して、クエリの行をコメントアウトすることができます。# 文字で始まる行は無視されます。この機能は、クエリに説明を追加したり、行を削除することなく 1 回の呼び出しで複雑なクエリの一部を一時的に無視したりする場合に役立ちます。

次の例では、クエリの 2 行目は無視されます。

```
fields @timestamp, @message
# | filter @message like /delay/
| limit 20
```

サポートされているオペレーションと関数

クエリ言語は、以下の表に示すように、多数のタイプのオペレーションと関数をサポートしています。

比較オペレーション

比較オペレーションは、**filter** コマンドで使用できます。また、他の関数の引数としても使用できます。比較オペレーションは、すべてのデータ型を引数として受け入れ、ブール値の結果を返します。

```
= != < <= > >=
```

算術オペレーション

算術オペレーションは、**filter** コマンドと **fields** コマンドで使用できます。また、他の関数の引数としても使用できます。算術オペレーションは、数値データ型を引数として受け入れ、数値結果を返します。

オペレーション	説明
a + b	加算
a - b	減算
a * b	乗算
a / b	除算
a ^ b	指数。2 ^ 3 は 8 を返します。
a % b	残余または剰余。10 % 3 は 1 を返します。

数値オペレーション

数値オペレーションは、**filter** コマンドと **fields** コマンドで使用できます。また、他の関数の引数としても使用できます。数値オペレーションは、数値データ型を引数として受け入れ、数値結果を返します。

オペレーション	説明
`abs(a)`	絶対値
`ceil(a)`	上限 (a の値より大きい最小整数) に切り上げられます。
`floor(a)`	下限 (a の値より小さい最大整数) に切り下げられます。
`greatest(a,b,... z)`	最大値を返します。
`least(a, b, ... z)`	最小値を返します。
`log(a)`	自然対数
`sqrt(a)`	平方根

一般関数

一般関数は、**filter** コマンドと **fields** コマンドで使用できます。また、他の関数の引数としても使用できます。

関数	引数	結果タイプ	説明
`ispresent(fieldname)`	ログフィールド	Boolean	フィールドが存在する場合は `true` を返します。
`coalesce(fieldname1, fieldname2, ... fieldnamex)`	ログフィールド	ログフィールド	リストから最初の null でない値を返します。

文字列関数

文字列関数は、**filter** コマンドと **fields** コマンドで使用できます。また、他の関数の引数としても使用できます。

関数	引数	結果タイプ	説明
`isempty(fieldname)`	文字列	Boolean	フィールドが欠落しているか、空の文字列である場合、`true` を返します。
`isblank(fieldname)`	文字列	Boolean	フィールドが欠落しているか、空の文字列であるか、空白が含まれている場合、`true` を返します。
`concat(string1, string2, ... stringz)`	文字列	文字列	複数の文字列を連結します。
`ltrim(string) or ltrim(string1, string2)`	文字列	文字列	文字列の左側から空白を削除します。関数に

関数	引数	結果タイプ	説明
			2 番目の文字列引数がある場合、string1 の左側から string2 の文字を削除します。たとえば、ltrim("xyZfooxyZ","xyZ") は "fooxyZ" を返します。
rtrim(string) or rtrim(string1, string2)	文字列	文字列	文字列の右側から空白を削除します。関数に 2 番目の文字列引数がある場合、string1 の右側から string2 の文字を削除します。たとえば、rtrim("xyZfooxyZ","xyZ") は "xyZfoo" を返します。
trim(string) or trim(string1, string2)	文字列	文字列	文字列の両端から空白を削除します。関数に 2 番目の文字列引数がある場合、string1 の両端から string2 の文字を削除します。たとえば、trim("xyZfooxyZ","xyZ") は "foo" を返します。
strlen(string)	文字列	数値	文字列の長さを Unicode コードポイントで返します。
toupper(string)	文字列	文字列	文字列を大文字に変換します。
tolower(string)	文字列	文字列	文字列を小文字に変換します。
substr(string1, x), or substr(string1, x, y)	文字列、数値または文字列、数値、数値	文字列	数値引数で指定されたインデックスから文字列の末尾までの部分文字列を返します。関数に 2 番目の数値引数がある場合、この引数には取得される部分文字列の長さが含まれます。たとえば、substr("xyZfooxyZ",3, 3) は "foo" を返します。
replace(string1, string2, string3)	文字列、文字列、文字列	文字列	string1 の string2 のすべてのインスタンスを string3 に置き換えます。たとえば、replace("foo","o","0") は "f00" を返します。
strcontains(string1, string2)	文字列	数値	string1 に string2 が含まれている場合は 1 を返し、それ以外の場合は 0 を返します。

日時関数

日時関数は、**filter** コマンドと **fields** コマンドで使用できます。また、他の関数の引数としても使用できます。これらの関数では、集計関数を使用してクエリの時間バケットを作成できます。

日時関数の一部として、数字と `m` (分) または `h` (時間) で構成される期間を使用できます。たとえば、`10m` は 10 分、`1h` は 1 時間です。

関数	引数	結果タイプ	説明
`datefloor(a, period)`	タイムスタンプ、期間	タイムスタンプ	タイムスタンプを特定の期間に切り詰めます。たとえば、`datefloor(@timestamp, 1h)` は `@timestamp` のすべての値を 1 時間の下限に切り詰めます。
`dateceil(a, period)`	タイムスタンプ、期間	タイムスタンプ	タイムスタンプを特定の期間に切り上げ、次に切り詰めます。たとえば、`dateceil(@timestamp, 1h)` は `@timestamp` のすべての値を 1 時間の上限に切り詰めます。
`bin(period)`	期間	タイムスタンプ	`@timestamp` の値を特定の期間に切り上げ、次に切り詰めます。

Stats コマンドの集約関数

集約関数は、**stats** コマンドで使用できます。また、他の関数の引数としても使用できます。

関数	引数	結果タイプ	説明
`avg(NumericFieldname)`	数値ログフィールド	数値	指定したフィールドの値の平均。
`count(fieldname) or count(*)`	ログフィールド	数値	ログレコードをカウントします。count(*) は、ロググループ内のすべてのレコードをカウントします。count (fieldname) は、指定されたフィールド名が含まれるすべてのレコードをカウントします。
`count_distinct(fieldname)`	ログフィールド	数値	フィールドの一意な値の数を返します。このフィールドの濃度が非常に高い場合 (一意な値が多数含まれている場合)、count_distinct から返される値は単なる概算値です。
`max(fieldname)`	ログフィールド	ログフィールド値	クエリを実行したログにおける、このログフィールドの値の最大数。
`min(fieldname)`	ログフィールド	ログフィールド値	クエリを実行したログにおける、このログフィールドの値の最小数。
`pct(fieldname, value)`	ログフィールド値、値	ログフィールド値	パーセンタイルは、データセットにおける値の相対的な位置を示します。たとえば、`pct(@duration, 95)` が `@duration` 値を返した場合、`@duration` の値の 95% がこの値より低く、5% がこの値より高くなります。

関数	引数	結果タイプ	説明
stddev(NumericFieldname)	数値ログフィールド	数値	指定されたフィールドの値の標準偏差。
sum(NumericFieldname)	数値ログフィールド	数値	指定したフィールドの値の合計。

時系列データの視覚化

視覚化を使用すると、ログ内で時間の経過に伴って生じる傾向やパターンを確認できます。CloudWatch Logs Insights では、以下の特性を持つすべてのクエリに対して視覚化を生成します。

- 1 つ以上の集計関数が含まれているクエリ。詳細については、「Stats コマンドの集約関数 (p. 45)」を参照してください。
- bin() 関数を使用して 1 つのフィールドでデータをグループ化するクエリ。

CloudWatch Logs Insights は、折れ線グラフとスタックされたエリアグラフを使用して視覚化を表示します。

視覚化の例

次のクエリでは、myfield1 フィールドの平均値の視覚化を生成します。データポイントは 5 分間隔で作成されます。各データポイントは、それまでの 5 分間隔のログに基づく myfield1 値の平均の集約です。

```
stats avg(myfield1) by bin(5m)
```

次のクエリでは、異なるフィールドに基づく 3 つの値の視覚化を生成します。データポイントは 5 分間隔で作成されます。視覚化が生成されるのは、クエリに集計関数が含まれており、グループ化フィールドとして bin() が使用されているためです。

```
stats avg(myfield1), min(myfield2), max(myfield3) by bin(5m)
```

よくある間違い

次のクエリには複数のグループ化フィールドが含まれているため、視覚化は生成されません。

```
stats avg(myfield1) by bin(5m), myfield4
```

次のクエリでは by bin() グループ化関数が使用されていないため、視覚化は生成されません。

```
stats avg(myfield1) by myfield4
```

サンプルクエリ

このセクションには、CloudWatch Logs Insights のパワーを示すクエリの例が含まれています。

一般的なクエリ

25 件の最近追加されたログイベントを確認する:

```
fields @timestamp, @message | sort @timestamp desc | limit 25
```

1 時間あたりの例外の数のリストを取得する:

```
filter @message like /Exception/
| stats count(*) as exceptionCount by bin(1h)
| sort exceptionCount desc
```

例外ではないログイベントのリストを取得する:

```
fields @message | filter @message not like /Exception/
```

Lambda ログのクエリ

過剰にプロビジョニングされたメモリの量を確認する:

```
filter @type = "REPORT"
| stats max(@memorySize / 1024 / 1024) as provisonedMemoryMB,
    min(@maxMemoryUsed / 1024 / 1024) as smallestMemoryRequestMB,
    avg(@maxMemoryUsed / 1024 / 1024) as avgMemoryUsedMB,
    max(@maxMemoryUsed / 1024 / 1024) as maxMemoryUsedMB,
    provisonedMemoryMB - maxMemoryUsedMB as overProvisionedMB
```

レイテンシーレポートを作成する:

```
filter @type = "REPORT" |
stats avg(@duration), max(@duration), min(@duration) by bin(5m)
```

Amazon VPC フローログのクエリ

ホスト間での上位 15 件のパケット転送を確認する:

```
stats sum(packets) as packetsTransferred by srcAddr, dstAddr
| sort packetsTransferred  desc
| limit 15
```

特定のホストの上位 15 件のバイト転送を確認する:

```
filter srcAddr LIKE "192.0.2."
| stats sum(bytes) as bytesTransferred by dstAddr
| sort bytesTransferred desc
| limit 15
```

データ転送プロトコルとして UDP を使用して IP アドレスを確認する:

```
filter protocol=17 | stats count(*) by srcAddr
```

キャプチャウィンドウでフローレコードがスキップされた IP アドレスを確認する:

```
filter logStatus="SKIPDATA"
| stats count(*) by bin(1h) as t
| sort t
```

Route 53 ログのクエリ

クエリタイプ別に 1 時間あたりのレコードのディストリビューションを確認する:

```
stats count(*) by queryType, bin(1h)
```

リクエスト数が最大である 10 件の DNS リゾルバーを確認する:

```
stats count(*) as numRequests by resolverIp
| sort numRequests desc
| limit 10
```

サーバーが DNS リクエストを完了できなかったレコード数をドメイン別およびサブドメイン別に確認する:

```
filter responseCode="SERVFAIL" | stats count(*) by queryName
```

CloudTrail ログのクエリ

サービス別、イベントタイプ別、およびリージョン別のログエントリ数を確認する:

```
stats count(*) by eventSource, eventName, awsRegion
```

リージョン別の開始または停止された Amazon EC2 ホストを確認する:

```
filter (eventName="StartInstances" or eventName="StopInstances") and region="us-east-2"
```

新しく作成した IAM ユーザーのリージョン、ユーザー名、および ARN を確認する:

```
filter eventName="CreateUser"
| fields awsRegion, requestParameters.userName, responseElements.user.arn
```

API UpdateTrail の呼び出し中に例外が発生したレコードの数を確認する:

```
filter eventName="UpdateTrail" and ispresent(errorCode)
| stats count(*) by errorCode, errorMessage
```

クエリをダッシュボードに追加する、またはクエリ結果をエクスポートする

クエリの実行後に、クエリを CloudWatch ダッシュボードに追加したり、クエリ結果をクリップボードにコピーしたりできます。

ダッシュボードに追加したクエリは、ダッシュボードをロードおよび更新するたびに自動的に再実行されます。これらのクエリは、4 つの同時実行される CloudWatch Logs Insights クエリの制限にカウントされます。

クエリ結果をダッシュボードに追加するには

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、[Insights] を選択します。
3. ロググループを選択してクエリを実行します。

4. [ダッシュボードに追加] を選択します。
5. ダッシュボードを選択するか、[新規作成] を選択して、クエリ結果用の新しいダッシュボードを作成します。
6. [ダッシュボードに追加] を選択します。

クエリ結果をクリップボードにコピーするには

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、[Insights] を選択します。
3. ロググループを選択してクエリを実行します。
4. [アクション]、[Copy query results (クエリ結果のコピー)] の順に選択します。

実行中のクエリまたはクエリ履歴を表示する

現在進行中のクエリや最近のクエリ履歴を表示できます。

現在実行中のクエリには、ダッシュボードに追加したクエリも含まれます。ダッシュボードに追加したクエリも含めて、アカウントあたり 4 つの同時実行されている CloudWatch Logs Insights クエリに制限されます。

現在実行中のクエリを表示するには

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、[Insights] を選択します。
3. [アクション]、[View running queries for this account (このアカウントの実行中のクエリを表示)] の順に選択します。
4. 実行中のクエリのいずれかを停止するには、[キャンセル] を選択します。

   このクエリがダッシュボード追加されている場合は、[キャンセル] を選択すると、現在の実行が停止されるだけです。ダッシュボードを開くか更新すると、クエリは自動的に再実行されます。クエリをダッシュボードから完全に削除するには、ダッシュボードを開いてウィジェットを削除します。

最近のクエリ履歴を表示するには

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、[Insights] を選択します。
3. [アクション]、[View query history for this account (このアカウントのクエリ履歴を表示)] の順に選択します。

   最近のクエリが一覧表示されます。これらのクエリを再実行するには、[クエリの実行] を選択します。

ロググループとログストリームを操作する

ログストリームは、同じソースを共有する一連のログイベントです。CloudWatch Logs へのログの各ソースによって、個別のログストリームが構成されます。

ロググループは、保持、監視、アクセス制御について同じ設定を共有するログストリームのグループです。ロググループを定義して、各グループに入れるストリームを指定することができます。1 つのロググループに属することができるログストリームの数に制限はありません。

このセクションの手順を使用して、ロググループおよびログストリームを処理します。

CloudWatch Logs にロググループを作成します。

Amazon CloudWatch Logs User Guideの前のセクションのステップを使用して、Amazon EC2 インスタンスに CloudWatch Logs エージェントをインストールすると、ロググループは、このプロセスの一環として作成されます。CloudWatch コンソールで、直接ロググループを作成することもできます。

ロググループを作成するには

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 [Logs] を選択します。
3. [Actions]、[Create log group] の順に選択します。
4. ロググループの名前を入力し、[Create log group(ロググループの選択)] を選択します。

CloudWatch Logs に送信されたログデータの表示

CloudWatch Logs エージェントで CloudWatch Logs に送信されるストリームごとにログデータを表示しスクロールできます。表示するログデータの時間範囲を指定できます。

ログデータを表示するには

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. [Log Groups] で、ストリームを表示するロググループを選択します。
4. [Log Streams] で、ログデータを表示するログストリーム名を選択します。
5. ログデータの表示方法を変更するには、次のいずれかを実行します。

   - すべてのログイベントを展開するには、ログイベントのリストの上で、[Expand all] を選択します。
   - すべてのログイベントを展開してプレーンテキストとして表示するには、ログイベントのリストの上で、[Text] を選択します。
   - ログイベントをフィルタするには、検索フィールドに目的の検索フィルタを入力します。詳細については、「ログデータの検索およびフィルタリング (p. 56)」を参照してください。
   - 指定された日時範囲のログデータを表示するには、ログイベントのリストの上で、[custom] を選択します。[Absolute] を選択して日時範囲を指定するか、[Relative] を選択して事前定義された分、時

間、日、週の数を選択することもできます。[UTC] と [Local timezone] を切り替えることもできます。

CloudWatch Logs でのログデータ保管期間の変更

デフォルトでは、ログデータは CloudWatch Logs に永続的に保存されます。ただし、ロググループにログデータを保存する期間を設定できます。現在の保持設定より古いデータはすべて自動的に削除されます。各ロググループのログの保持期間は、いつでも変更できます。

ログの保持設定を変更するには

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. 更新するロググループを見つけます。
4. そのロググループの [Expire Events After] 列で、現在の保持設定 (例: [Never Expire]) を選択します。
5. [Edit Retention] ダイアログボックスの [Retention] で、ログ保管期間の値を選択し、[Ok] を選択します。

Amazon CloudWatch Logs ロググループのタグ付け

Amazon CloudWatch Logs で作成したロググループに、独自のメタデータをタグ形式で割り当てることができます。タグは、ロググループに対して定義するキーと値のペアです。タグの使用は、AWS リソースの管理やデータ (請求データなど) の整理を行うシンプルかつ強力な方法です。

コンテンツ

タグの基本

AWS CLI または CloudWatch Logs API を使用して、次のタスクを実行します。

- ロググループの作成時にタグを追加する
- 既存のロググループにタグを追加する
- ロググループのタグをリストする
- ロググループからタグを削除する

タグを使用すると、ロググループを分類できます。たとえば、目的、所有者、環境などに基づいて分類できます。タグごとにキーと値を定義するため、特定のニーズを満たすためのカテゴリのカスタムセットを作成できます。たとえば、所有者と、関連するアプリケーションに基づいてロググループを追跡するのに役立つタグのセットを定義できます。次にいくつかのタグの例を示します。

- プロジェクト: プロジェクト名
- 所有者: 名前

- 目的: 負荷テスト
- アプリケーション: アプリケーション名
- 環境: 本稼働

タグ付けを使用したコストの追跡

タグを使用して、AWS コストを分類して追跡できます。AWS リソース (ロググループなど) にタグを適用すると、AWS のコスト配分レポートに、タグ別に集計された使用状況とコストが表示されます。自社のカテゴリ (たとえばコストセンター、アプリケーション名、所有者) を表すタグを適用すると、複数のサービスにわたってコストを分類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「コスト配分タグを使用したカスタム請求レポート」を参照してください。

タグの制限

タグには次の制限があります。

基本制限

- タグの最大数はロググループごとに 50 です。
- タグのキーと値は大文字と小文字が区別されます。
- 削除されたロググループのタグを変更または編集することはできません。

タグキーの制限

- 各タグキーは一意である必要があります。既に使用されているキーを含むタグを追加すると、新しいタグで、既存のキーと値のペアが上書きされます。
- `aws:` は AWS が使用するように予約されているため、このプレフィックスを含むタグキーで開始することはできません。AWS ではユーザーの代わりにこのプレフィックスで始まるタグを作成しますが、ユーザーはこれらのタグを編集または削除することはできません。
- タグキーの長さは 1 ~ 128 文字 (Unicode) にする必要があります。
- タグキーは、次の文字で構成する必要があります。Unicode 文字、数字、空白、特殊文字 (`_ . / = + - @`)。

タグ値の制限

- タグ値の長さは 0 ~ 255 文字 (Unicode) にする必要があります。
- タグ値は空白にすることができます。空白にしない場合は、次の文字で構成する必要があります。Unicode 文字、数字、空白、特殊文字 (`_ . / = + - @`)。

AWS CLI を使用したロググループのタグ付け

AWS CLI を使用してタグの追加、一覧表示、および削除を行うことができます。例については、次のドキュメントを参照してください。

create-log-group

ロググループを作成します。ロググループの作成時に、オプションでタグを追加できます。

tag-log-group

指定したロググループのタグを追加または更新します。

list-tags-log-group

指定したロググループのタグを一覧表示します。

untag-log-group

指定したロググループからタグを削除します。

CloudWatch Logs API を使用したロググループのタグ付け

CloudWatch Logs API を使用してタグの追加、一覧表示、および削除を行うことができます。例については、次のドキュメントを参照してください。

CreateLogGroup

ロググループを作成します。ロググループの作成時に、オプションでタグを追加できます。

TagLogGroup

指定したロググループのタグを追加または更新します。

ListTagsLogGroup

指定したロググループのタグを一覧表示します。

UntagLogGroup

指定したロググループからタグを削除します。

AWS KMS を使用した CloudWatch Logs でのログデータの暗号化

AWS Key Management Service (AWS KMS) カスタマーマスターキー (CMK) を使用して CloudWatch Logs でログデータを暗号化できます。暗号化は、ロググループを作成するときか、ロググループが存在しているときに、CMK とロググループを関連付けることによりロググループレベルで有効になります。

CMK をロググループと関連付けると、ロググループの新たに取り込まれたすべてのデータが CMK を使用して暗号化されます。このデータは、保持期間を通じて暗号化形式で保存されます。CloudWatch Logs は、リクエストがあればいつでもこのデータを復号します。暗号化されたデータがリクエストされた場合は必ず、CloudWatch Logs に CMK のアクセス権限が必要です。

ロググループから CMK の関連付けを解除すると、CloudWatch Logs はロググループの新たに取り込まれたデータの暗号化を停止します。それまで取り込まれたすべてのデータは暗号化されたままです。

制限

- CMK をロググループに関連付け、次のステップを実行するには、`kms:CreateKey`、`kms:GetKeyPolicy`、および `kms:PutKeyPolicy` のアクセス権限を持っている必要があります。
- CMK とロググループを関連付けまたは関連付け解除すると、オペレーションが有効になるまで最大 5 分かかることがあります。
- 関連付けられた CMK への CloudWatch Logs のアクセスを取り消したり、関連付けられた CMK を削除した場合、CloudWatch Logs 内の暗号化されたデータを取得できなくなります。
- CloudWatch コンソールを使用して CMK をロググループと関連付けることはできません。

ステップ 1: AWS KMS CMK を作成する

AWS KMS CMK を作成するには、次の create-key コマンドを使用します。

```
aws kms create-key
```

出力には、CMK のキー ID とAmazon リソースネーム (ARN) が含まれます。出力例を次に示します。

```
{
    "KeyMetadata": {
        "KeyId": "6f815f63-e628-448c-8251-e40cb0d29f59",
        "Description": "",
        "Enabled": true,
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "CreationDate": 1478910250.94,
        "Arn": "arn:aws:kms:us-west-2:123456789012:key/6f815f63-e628-448c-8251-
e40cb0d29f59",
        "AWSAccountId": "123456789012"
    }
}
```

ステップ 2: CMK でアクセス権限を設定する

デフォルトでは、すべての AWS KMS CMK がプライベートです。リソース所有者だけが使用してデータを暗号化および復号できます。ただし、リソース所有者は、他のユーザーとリソースに CMK へのアクセス権限を付与することができます。このステップでは、CloudWatch サービスプリンシパルに、CMK を使用するアクセス権限を付与します。このサービスプリンシパルは、CMK が保存されているのと同じリージョンにある必要があります。

まず、get-key-policy コマンドを使用して、CMK のデフォルトポリシーを `policy.json` として保存します。

```
aws kms get-key-policy --key-id key-id --policy-name default --output text > ./policy.json
```

テキストエディタで `policy.json` ファイルを開き、ステートメントを太字で追加します。*region* は、ロググループに使用するリージョンに置き換え、既存のステートメントと新しいステートメントをカンマで区切ります。

```
{
  "Version" : "2012-10-17",
  "Id" : "key-default-1",
  "Statement" : [ {
      "Sid" : "Enable IAM User Permissions",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::Your_account_ID:root"
      },
      "Action" : "kms:*",
      "Resource" : "*"
    },
    {
      "Effect": "Allow",
      "Principal": { "Service": "logs.region.amazonaws.com" },
      "Action": [
        "kms:Encrypt*",
        "kms:Decrypt*",
        "kms:ReEncrypt*",
```

```
        "kms:GenerateDataKey*",
        "kms:Describe*"
      ],
      "Resource": "*"
    }
  ]
}
```

最後に、次の put-key-policy コマンドを使用して更新されたポリシーを追加します。

```
aws kms put-key-policy --key-id key-id --policy-name default --policy file://policy.json
```

ステップ 3: ログ グループと CMK を関連付ける

CMK とロググループは、ロググループの作成時または作成後に関連付けることができます。

ロググループの作成時に CMK をロググループに関連付けるには

次のように、create-log-group コマンドを使用します。

```
aws logs create-log-group --log-group-name my-log-group --kms-key-id "key-arn"
```

CMK を既存のロググループに関連付けるには

次のように、associate-kms-key コマンドを使用します。

```
aws logs associate-kms-key --log-group-name my-log-group --kms-key-id "key-arn"
```

ステップ 4: ロググループを CMK の関連付けから解除する

ロググループに関連付けられた CMK の関連付けを解除するには、次の disassociate-kms-key コマンドを使用します。

```
aws logs disassociate-kms-key --log-group-name my-log-group
```

ログデータの検索およびフィルタリング

CloudWatch Logs エージェントが Amazon CloudWatch へのログデータの発行を開始した後、1 つ以上のメトリクスフィルターを作成して、ログデータの検索およびフィルタリングを開始できます。メトリクスフィルターは CloudWatch Logs に送信されたログデータを検索するための語句とパターンを定義します。CloudWatch Logs はこれらのメトリクスフィルターを使用して、ログデータをグラフの作成やアラームの設定に利用できる数値の CloudWatch メトリクスに変換します。

フィルターは、遡及的にデータをフィルターしません。フィルターは、フィルターが作成された後に発生したイベントのメトリクスのデータポイントをパブリッシュするだけです。フィルターされた結果は最初の 50 行を返しますが、これはフィルターされた結果のタイムスタンプがメトリクスの作成時刻よりも前であれば表示されません。

目次

概念

各メトリクスフィルターは以下のキー要素で構成されています。

フィルターパターン

: 各ログイベントのデータを CloudWatch Logs がどのように解釈するかについての記号による説明です。たとえば、ログエントリにはタイムスタンプ、IP アドレス、文字列などが含まれる可能性があります。パターンを使用して、ログファイルの検索対象を指定します。

メトリクス名

: モニタリングされたログ情報が発行される CloudWatch メトリクスの名前です。たとえば、ErrorCount というメトリクスに発行できます。

メトリクス名前空間

: 新しい CloudWatch メトリクスの送信先名前空間です。

メトリクス値

: 一致するログが見つかるたびにメトリクスに発行する数値。たとえば、「Error」など特定の語句の発生回数をカウントする場合、その値は発生するごとに「1」になります。転送されたバイト数をカウントする場合は、ログイベントに見つかった実際のバイト数で増分できます。

デフォルト値

: 一致するログが見つからなかった期間中にメトリクスフィルターに報告された値。この値を 0 に設定することで、データは毎秒報告され、データがない期間がある「むらがある」メトリクスを回避できます。

フィルターとパターンの構文

メトリクスフィルターで指定されたものに一致する語句や値を、ログイベントの中で検索できます。メトリクスフィルターがログイベントで語句、フレーズ、値を 1 つ見つけたら、CloudWatch メトリクスの値を増分できます。たとえば、ログイベントの中で ERROR という単語を検索して出現回数を数えるメトリクスフィルターを作成します。

メトリクスフィルターには、スペースで区切られたログイベントから数値を抽出する機能もあります。たとえば、ウェブリクエストのレイテンシーです。これらの例では、ログから抽出された実際の数値でメトリクス値を増分できます。

条件演算子やワイルドカードを使用して完全一致を見つけることもできます。メトリクスフィルターを作成する前に、CloudWatch コンソールで検索パターンをテストできます。以下のセクションでは、メトリクスフィルターの構文についてさらに詳しく説明します。

ログイベントの語句の一致

ログイベントで語句を検索するには、語句メトリクスフィルターパターンとして使用します。メトリクスフィルターパターンには複数の語句を指定できますが、一致させるためには語句はすべてログイベントに含まれるものである必要があります。メトリクスフィルターは大文字と小文字を区別します。

アルファベット文字およびアンダースコア以外の文字を含むメトリクスフィルターの語句は二重引用符 ("") で囲む必要があります。

語句を除外するには、語句の前で負符号 (-) を使用します。

例 1: すべてに一致するようにする

フィルターパターン "" はすべてのログイベントに一致します。

例 2: 単一語句

フィルターパターン「ERROR」は、次のように、この語句を含むログイベントメッセージに一致します:

- [ERROR] A fatal exception has occurred
- Exiting with ERRORCODE: -1

例 3: 語句の包含と語句の除外

前の例で、フィルターパターンを「ERROR」 -「Exiting」に変更した場合、ログイベントメッセージ「Exiting with ERRORCODE: -1」が除外されます。

例 4: 複数語句

フィルターパターン「ERROR Exception」は、次のように両方の語句を含むログイベントメッセージに一致します。

- [ERROR] Caught IllegalArgumentException
- [ERROR] Unhandled Exception

フィルターパターン「Failed to process the request」は、次のようにすべての語句を含むログイベントメッセージに一致します。

- [WARN] Failed to process the request

- [ERROR] Unable to continue: Failed to process the request

OR パターンマッチング

OR パターンマッチングを使用して、テキストベースのフィルターで語句を一致させることができます。OR には疑問符を使用します (例: `?term`)。

次の 3 つのログイベント例をご覧ください。`ERROR` は例 1、2 に一致します。`?ERROR ?WARN` は例 1、2、および 3 に一致します。それらのすべてには ERROR または WARN という単語が含まれているためです。`ERROR WARN` は例 1 のみと一致します。これは、これらの両方の単語が含まれている唯一の例であるためです。ERROR -WARN は例 2 と一致します。これは、WARN が含まれている文字列に一致しますが、WARN は含まれないためです。

1. `ERROR WARN message`
2. `ERROR message`
3. `WARN message`

OR パターンマッチングを使用して、スペース区切りフィルターで語句を一致させることができます。スペース区切りフィルターでは、`w1` はログイベントの最初の単語を意味し、`w2` は 2 番目の単語を意味します。以下同様です。以下のサンプルパターンでは、ERROR が最初の単語であるため、[w1=ERROR, w2] はパターン 2 と一致し、[w1=ERROR || w1=WARN, w2] は パターン 2 および 3 と一致します。[w1!=ERROR&&w1!=WARN, w2] は ERROR および WARN の両方を含む行 (パターン 1) と一致します。

1. ERROR WARN メッセージ
2. ERROR メッセージ
3. WARN メッセージ

OR パターンマッチングを使用して、JSON フィルターで語句を一致させることができます。以下のサンプルパターンでは、{$.foo = bar} はパターン 1 と一致し、{$.foo = baz } はパターン 2 と一致し、{$.foo = bar || $.foo = baz } はパターン 1 および 2 と一致します。

1. {"foo": "bar"}
2. {"foo": "baz"}

JSON ログイベントの語句の一致

JSON ログイベントから値を取得できます。JSON ログイベントから値を取得するには、文字列ベースのメトリクスフィルターを作成する必要があります。指数表記を含む文字列はサポートされていません。JSON ログイベントデータの項目は厳密にメトリクスフィルターと一致する必要があります。JSON ログイベントにメトリクスフィルターを作成して表示するのは次のような場合があります。

- 特定のイベントが発生した場合。たとえば eventName が「UpdateTrail」の場合です。
- IP が既知のサブネット以外の場合。たとえば、sourceIPAddress が既知のサブネットの範囲にない場合です。
- 二つ以上の条件の組み合わせが true の場合。たとえば、eventName が「UpdateTrail」で、recipientAccountId が 123456789012 の場合です。

JSON ログイベントから値を取得するメトリクスフィルターの使用

メトリクスフィルターを使用して、JSON ログイベントから値を取得できます。メトリックスフィルターは受信ログをチェックし、ログデータの一致が見つかったときに数値を変更します。メトリクスフィル

ターを作成するときは、ログで一致するテキストが見つかるたびにカウントを増分するか、ログから数値を抽出し、それを使用してメトリクス値を増分することができます。

メトリクスフィルターを使用した JSON 条件の一致

JSON ログイベントのメトリクスフィルター構文は次の形式を使用します。

```
{ SELECTOR EQUALITY_OPERATOR STRING }
```

メトリクスフィルターは、JSON の記述であることを示すために中括弧 { } で囲む必要があります。メトリクスフィルターには次のパーツがあります。

SELECTOR

どの JSON プロパティを確認するかを指定します。プロパティセレクタは常に、JSON のルートを示すドル記号 ($) から始まります。プロパティセレクタは英数字の文字列であり「-」および「_」をサポートします。配列要素は [NUMBER] 構文で示され、プロパティに従う必要があります。例は次のとおりです。$.eventId、$.users[0]、$.users[0].id、$.requestParameters.instanceId。

EQUALITY_OPERATOR

= または != です。

STRING

引用符付きまたは引用符なしの文字列です。アスタリスク「*」ワイルドカード文字を検索語句の中間、前方、後方の文字の一致に使用できます。たとえば、[*Event] は [PutEvent] と [GetEvent] に一致します。[Event*] は [EventId] と [EventName] に一致します。[Ev*ent] は、実際の文字列 [Ev*ent] のみと一致します。英数字のみで構成された文字列を引用符で囲む必要はありません。unicode やそのほかの文字 (たとえば「@」「$」「\」など) を含む文字列は二重引用符で囲んで有効にする必要があります。

JSON メトリクスフィルターの例

JSON の例を次に示します。

```
{
  "eventType": "UpdateTrail",
  "sourceIPAddress": "111.111.111.111",
  "arrayKey": [
        "value",
        "another value"
  ],
  "objectList": [
       {
         "name": "a",
         "id": 1
       },
       {
         "name": "b",
         "id": 2
       }
  ],
  "SomeObject": null,
  "ThisFlag": true
}
```

次のフィルターが一致します。

```
{ $.eventType = "UpdateTrail" }
```

UpdateTrail というイベントタイプのフィルター。

```
{ $.sourceIPAddress != 123.123.* }
```

プレフィックスが 123.123 のサブネットの範囲外にある IP アドレスのフィルター。

```
{ $.arrayKey[0] = "value" }
```

arrayKey の最初のエントリが「value」のフィルター。arrayKey が配列ではない場合、これは false になります。

```
{ $.objectList[1].id = 2 }
```

objectList の 2 番目のエントリが ID = 2 というプロパティを持つフィルター。objectList が配列ではない場合、これは false になります。objectList がオブジェクトではない場合、または ID プロパティがない場合、これは false になります。

```
{ $.SomeObject IS NULL }
```

null に設定されている SomeObject のフィルター。これは、指定したオブジェクトが null に設定されている場合のみ true になります。

```
{ $.SomeOtherObject NOT EXISTS }
```

存在しない SomeOtherObject のフィルター。これは、指定したオブジェクトがログデータに存在しない場合のみ true になります。

```
{ $.ThisFlag IS TRUE }
```

TRUE になっている ThisFlag のフィルター。これは、FALSE 値をチェックするブールフィルターでも機能します。

JSON 複合条件

OR (||) と AND (&&) を使用して複数の条件を組み合わせ、複合式にできます。括弧が使用できます。また、構文は演算子の標準の順序に従い、() > && > || となります。

```
{
    "user": {
        "id": 1,
        "email": "John.Stiles@example.com"
    },
    "users": [
        {
         "id": 2,
         "email": "John.Doe@example.com"
        },
        {
         "id": 3,
         "email": "Jane.Doe@example.com"
        }
    ],
    "actions": [
        "GET",
        "PUT",
```

```
        "DELETE"
    ],
    "coordinates": [
        [0, 1, 2],
        [4, 5, 6],
        [7, 8, 9]
    ]
}
```

例

```
{ ($.user.id = 1) && ($.users[0].email = "John.Doe@example.com") }
```

上記の JSON に一致します。

```
{ ($.user.id = 2 && $.users[0].email = "nonmatch") || $.actions[2] = "GET" }
```

上記の JSON と一致しません。

```
{ $.user.email = "John.Stiles@example.com" || $.coordinates[0][1] = nonmatch &&
 $.actions[2] = nomatch }
```

上記の JSON に一致します。

```
{ ($.user.email = "John.Stiles@example.com" || $.coordinates[0][1] = nonmatch) &&
 $.actions[2] = nomatch }
```

上記の JSON と一致しません。

JSON に関する特別な考慮事項

SELECTOR は JSON の値ノード (文字列または数字) を指定する必要があります。配列またはオブジェクトを指定しても、ログ形式がフィルターと一致しないためフィルターは適用されません。たとえば、{$.users = 1} と {$.users != 1} は両方ともユーザーが配列であるログイベントとは一致しません。

```
{
  "users": [1, 2, 3]
}
```

数値比較

メトリクスフィルター構文は数値比較の正確な一致をサポートします。次の数値比較がサポートされています: <、>、>=、<=、=、!=

数字フィルターは次のような構文です。

```
{ SELECTOR NUMERIC_OPERATOR NUMBER }
```

メトリクスフィルターは、JSON の記述であることを示すために中括弧 { } で囲む必要があります。メトリクスフィルターには次のパーツがあります。

SELECTOR

どの JSON プロパティを確認するかを指定します。プロパティセレクタは常に、JSON のルートを示すドル記号 ($) から始まります。プロパティセレクタは英数字の文字列であり「-」および「_」をサ

ポートします。配列要素は [NUMBER] 構文で示され、プロパティに従う必要があります。例は次のとおりです: $.latency、$.numbers[0]、$.errorCode、$.processes[4].averageRuntime。

NUMERIC_OPERATOR

次のいずれかを指定できます: =、!=、<、>、<=、または >=

NUMBER

オプションで + または - を伴う整数、オプションで + または - を伴う小数、科学的記数法で表した数字 (オプションで + または - を伴う整数または小数に続いて「e」、その後にオプションで + または - を伴う整数または小数) です。

例:

```
{ $.latency >= 500 }
{ $.numbers[0] < 10e3 }
{ $.numbers[0] < 10e-3 }
{ $.processes[4].averageRuntime <= 55.5 }
{ $.errorCode = 400 }
{ $.errorCode != 500 }
{ $.latency > +1000 }
```

スペース区切りログイベントから値を取得するメトリクスフィルターの使用

メトリクスフィルターを使用してスペース区切りログイベントから値を取得できます。角括弧 [] または 2 個の二重引用符 ("") で囲まれた文字は単一のフィールドとして扱われます。例:

```
127.0.0.1 - frank [10/Oct/2000:13:25:15 -0700] "GET /apache_pb.gif HTTP/1.0" 200 1534
127.0.0.1 - frank [10/Oct/2000:13:35:22 -0700] "GET /apache_pb.gif HTTP/1.0" 500 5324
127.0.0.1 - frank [10/Oct/2000:13:50:35 -0700] "GET /apache_pb.gif HTTP/1.0" 200 4355
```

スペース区切りイベントを解析するメトリクスフィルターパターンを指定するには、メトリクスフィルターパターンが、フィールドの名前をカンマで区切り、すべてのパターンを角括弧で囲んで指定する必要があります。たとえば、[ip, user, username, timestamp, request, status_code, bytes] のようになります。

フィールド数が不明な場合は、省略符号 (...) を使用した省略通知を使用できます。例:

```
[..., status_code, bytes]
[ip, user, ..., status_code, bytes]
[ip, user, ...]
```

また、フィールドに条件を追加して、すべての条件に一致するログイベントのみがフィルターに一致するようにできます。例:

```
[ip, user, username, timestamp, request, status_code, bytes > 1000]
[ip, user, username, timestamp, request, status_code = 200, bytes]
[ip, user, username, timestamp, request, status_code = 4*, bytes]
[ip, user, username, timestamp, request = *html*, status_code = 4*, bytes]
```

次の例のように、AND 演算子として && を使用できます。

```
[ip, user, username, timestamp, request, status_code = 4* && bytes > 1000]
[ip, user, username, timestamp, request, status_code = 4* && status_code != 403, bytes]
```

CloudWatch Logs は文字列条件フィールドおよび数値条件フィールドの両方をサポートします。文字列フィールドでは、= または != 演算子をアスタリスク (*) とともに使用できます。

数値フィールドでは、>、<、>=、<=、=、および != 演算子を使用できます。

スペース区切りフィルターを使用している場合、抽出されたフィールドは、フィルターで表示されるスペース区切りフィールド名をこれらの各フィールドの値にマッピングします。スペース区切りフィルターを使用していない場合、これは空となります。

フィルターの例:

```
[..., request=*.html*, status_code=4*,]
```

フィルターのログイベントの例:

```
127.0.0.1 - frank [10/Oct/2000:13:25:15 -0700] \"GET /index.html HTTP/1.0\" 404 1534
```

ログイベントとフィルターパターンに対して抽出されたフィールド:

```
{
   "$status_code": "404",
   "$request": "GET /products/index.html HTTP/1.0",
   "$7": "1534",
   "$4": "10/Oct/2000:13:25:15 -0700",
   "$3": "frank",
   "$2": "-",
   "$1": "127.0.0.1"
}
```

一致が見つかった場合のメトリクス値の変更方法の設定

メトリクスフィルターが、ログイベントで一致するいずれかの用語、語句、または値を見つけた場合、メトリクス値に指定された量だけ CloudWatch メトリクスのカウントを増分します。メトリクス値は 1 分ごとに集計され、報告されます。

1 分の間にログが取り込まれても一致が見つからない場合は、デフォルト値に設定された値 (存在する場合) が報告されます。ただし、1 分の間に取り込まれたログイベントがない場合は、値は報告されません。

デフォルト値を指定すると、その値が 0 の場合でも、データはより頻繁に報告されるので、一致が見つからない場合にむらがあるメトリクスを避けるのに役立ちます。

たとえば、2 つのレコードを公開し、メトリクス値は 1 で、デフォルト値は 0 のロググループがあるとします。最初の 1 分で両方のログレコードで一致が見つかった場合、その分のメトリクス値は 2 になります。次の 1 分で発行されるログレコードに一致する値がない場合、デフォルト値の 0 が両方のログレコードに使用され、その 1 分のメトリクス値は 0 になります。

デフォルト値を指定しない場合、一致するパターンが見つからない期間についてはデータは報告されません。

ログエントリで見つかった数値を発行する

ログで見つかった一致する項目の数をカウントするだけでなく、メトリクスフィルターを使用してログで見つかった数値に基づく値を発行することができます。次の手順は、JSON リクエスト `metricFilter: { $.latency = * } metricValue: $.latency` で見つかったレイテンシーのメトリクスを公開する方法を示しています。

JSON リクエストで見つかったレイテンシーのメトリクスを発行するには

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 [ログ] を選択します。
3. コンテンツペインでロググループを選択し、[メトリクスフィルターの作成] を選択します。
4. [ログメトリクスフィルターの定義] 画面の [フィルターパターン] に、「`{ $.latency = * }`」と入力し、[メトリクスの割り当て] をクリックします。
5. [メトリクスフィルターの作成とメトリクスの割り当て] 画面で、[メトリクスの詳細設定の表示] を選択します。
6. [メトリクス名] に、「myMetric」と入力します。
7. [メトリクス値] に「`$.latency`」と入力します。
8. [デフォルト値] に「0」と入力し、[フィルターの作成] を選択します。デフォルト値を指定すると、ログイベントが発生しない期間でもデータが報告され、データが存在しないことがある、むらのあるメトリクスを回避できます。

次のログイベントは、フィルターの作成後、値 50 をメトリクス myMetric に発行します。

```
{
"latency": 50,
"requestType": "GET"
}
```

メトリクスフィルターの作成

以下の例は、メトリクスフィルターの作成方法を示しています。

例

- 例: ログイベントのカウント (p. 64)
- 例: 語句の出現回数をカウントする (p. 65)
- 例: HTTP 404 コードをカウントする (p. 66)
- 例: HTTP 4xx コードをカウントする (p. 68)
- 例: Apache ログからのフィールドの抽出 (p. 69)

例: ログイベントのカウント

ログイベントのモニタリングで最もシンプルなタイプは、発生したログのイベント数のカウントです。目的はすべてのイベントのカウントや、「ハートビート」形式のモニタリングの作成、あるいは単にメトリクスフィルターの作成練習の場合もあります。

次の CLI の例では、MyAppAccessCount というメトリクスフィルターをロググループ MyApp/access.log に適用して、CloudWatch の名前空間 MyNamespace にメトリクス EventCount を生成します。フィルターは、すべてのログイベントコンテンツに一致し、メトリクスを 1 ずつ増加させるように設定されています。

CloudWatch コンソールを使用してメトリクスフィルターを作成するには

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 [ログ] を選択します。
3. コンテンツペインでロググループを選択し、[メトリクスフィルターの作成] を選択します。

4. [ログメトリクスフィルターの定義] 画面で、[フィルターパターン] を空欄にします。
5. [メトリクスの割り当て] を選択し、[メトリクスフィルターの作成とメトリクスの割り当て] 画面の [フィルター名] フィールドに「**EventCount**」と入力します。
6. [メトリクスの詳細] の [メトリクス名前空間] に、「**MyNameSpace**」と入力します。
7. [メトリクス名] に「**MyAppEventCount**」と入力します。
8. [メトリクスの詳細設定の表示] を選択し、[メトリクス値] が 1 であることを確認します。これにより、各ログイベントのカウントは 1 ずつ増分されます。
9. [デフォルト値] に「0」と入力し、[フィルターの作成] を選択します。デフォルト値を指定すると、ログイベントが発生しない期間でもデータが報告され、データが存在しないことがある、むらのあるメトリクスを回避できます。

AWS CLI を使用してメトリクスにフィルターを作成するには

コマンドプロンプトで、次のコマンドを実行します。

```
aws logs put-metric-filter \
  --log-group-name MyApp/access.log \
  --filter-name EventCount \
  --filter-pattern "" \
  --metric-transformations \
  metricName=MyAppEventCount,metricNamespace=MyNamespace,metricValue=1,defaultValue=0
```

イベントデータを投稿することで、この新しいポリシーをテストできます。メトリクス MyAppAccessEventCount に発行されたデータポイントを参照する必要があります。

AWS CLI を使用してイベントデータを投稿するには

コマンドプロンプトで、次のコマンドを実行します。

```
aws logs put-log-events \
  --log-group-name MyApp/access.log --log-stream-name TestStream1 \
  --log-events \
    timestamp=1394793518000,message="Test event 1" \
    timestamp=1394793518000,message="Test event 2" \
    timestamp=1394793528000,message="This message also contains an Error"
```

例: 語句の出現回数をカウントする

ログイベントにはよく、カウントしておきたい重要なメッセージが含まれています。操作の成功または失敗についてなどです。たとえば、特定の操作に失敗すると、エラーが発生してログファイルに記録される場合があります。エラーの傾向を理解するためのこれらのエントリをモニタリングする場合があります。

次の例では、Error という語句をモニタリングするメトリクスフィルターが作成されます。ポリシーはすでに作成されてロググループ MyApp/message.log に追加されています。CloudWatch Logs は、MyApp/message.log という名前空間の CloudWatch カスタムメトリクス ErrorCount に、Error を含むイベントごとに「1」の値をともなったデータポイントを発行します。Error という単語を含むイベントがない場合、値 0 は発行されません。このデータを CloudWatch コンソールでグラフ化するときは、必ず合計の統計を使用してください。

CloudWatch コンソールを使用してメトリクスフィルターを作成するには

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 [ログ] を選択します。
3. コンテンツペインでロググループを選択し、[メトリクスフィルターの作成] を選択します。
4. [ログメトリクスフィルターの定義] 画面の [フィルターパターン] に「**Error**」と入力します。

Note

[フィルターパターン] のすべての項目は大文字と小文字が区別されます。

5. フィルターパターンをテストするには、[テストするログデータの選択] でメトリクスフィルターをテストするロググループを選択し、[パターンのテスト] を選択します。
6. [結果] に、ログファイルで見つかったフィルターパターンの発生数を示す CloudWatch Logs のメッセージが表示されます。

   詳細結果を表示するには、[テスト結果の表示] を選択します。
7. [メトリクスの割り当て] を選択し、[メトリクスフィルターの作成とメトリクスの割り当て] 画面の [フィルター名] フィールドに「**MyAppErrorCount**」と入力します。
8. [メトリクスの詳細] の [メトリクス名前空間] に、「MyNameSpace」と入力します。
9. [メトリクス名] に「ErrorCount」と入力します。
10. [メトリクスの詳細設定の表示] を選択し、[メトリクス値] が 1 であることを確認します。これにより、「Error」を含む各ログイベントのカウントは 1 ずつ増分されます。
11. [デフォルト値] に「0」と入力し、[フィルターの作成] を選択します。

AWS CLI を使用してメトリクスにフィルターを作成するには

コマンドプロンプトで、次のコマンドを実行します。

```
aws logs put-metric-filter \
  --log-group-name MyApp/message.log \
  --filter-name MyAppErrorCount \
  --filter-pattern 'Error' \
  --metric-transformations \
      metricName=EventCount,metricNamespace=MyNamespace,metricValue=1,defaultValue=0
```

メッセージに「Error」を含むイベントを投稿することで、この新しいポリシーをテストできます。

AWS CLI を使用してイベントを投稿するには

コマンドプロンプトで、次のコマンドを実行します。パターンでは大文字と小文字が区別されます。

```
aws logs put-log-events \
  --log-group-name MyApp/access.log --log-stream-name TestStream1 \
  --log-events \
    timestamp=1394793518000,message="This message contains an Error" \
    timestamp=1394793528000,message="This message also contains an Error"
```

例: HTTP 404 コードをカウントする

CloudWatch Logs を使用して、Apache サーバーが HTTP 404 レスポンス (ページが見つからない場合のレスポンスコード) を返した数をモニタリングできます。サイトの訪問者が目的のリソースを見つけられなかった頻度を理解するためにモニタリングする場合があります。ログレコードが各ログイベント (サイト訪問) に関する次の情報を含むように設定されている前提です。

- 要求者の IP アドレス
- RFC 1413 ID
- Username
- タイムスタンプ
- リクエスト方法およびリクエストされたリソースとプロトコル
- リクエストに対する HTTP レスポンスコード

- リクエストで転送されたバイト数

例は次のようになります。

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 404 2326
```

以下の例に示すように、HTTP 404 エラーの構造にイベントが一致するようにルールを指定できます。

CloudWatch コンソールを使用してメトリクスフィルターを作成するには

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 [ログ] を選択します。
3. コンテンツペインでロググループを選択し、[メトリクスフィルターの作成] を選択します。
4. [ログメトリクスフィルターの定義] 画面の [フィルターパターン] に「**[IP, UserInfo, User, Timestamp, RequestInfo, StatusCode=404, Bytes]**」と入力します。
5. フィルターパターンをテストするには、[テストするログデータの選択] でメトリクスフィルターをテストするロググループを選択し、[パターンのテスト] を選択します。
6. [結果] に、ログファイルで見つかったフィルターパターンの発生数を示す CloudWatch Logs のメッセージが表示されます。

   詳細結果を表示するには、[テスト結果の表示] を選択します。
7. [メトリクスの割り当て] を選択し、[メトリクスフィルターの作成とメトリクスの割り当て] 画面の [フィルター名] フィールドに「HTTP404Errors」と入力します。
8. [メトリクスの詳細] の [メトリクス名前空間] に、「**MyNameSpace**」と入力します。
9. [メトリクス名] に「**ApacheNotFoundErrorCount**」と入力します。
10. [メトリクスの詳細設定の表示] を選択し、[メトリクス値] が 1 であることを確認します。これにより、各 404 エラーイベントのカウントは 1 ずつ増分されます。
11. [デフォルト値] に「0」と入力し、[フィルターの作成] を選択します。

AWS CLI を使用してメトリクスにフィルターを作成するには

コマンドプロンプトで、次のコマンドを実行します。

```
aws logs put-metric-filter \
  --log-group-name MyApp/access.log \
  --filter-name HTTP404Errors \
  --filter-pattern '[ip, id, user, timestamp, request, status_code=404, size]' \
  --metric-transformations \
      metricName=ApacheNotFoundErrorCount,metricNamespace=MyNamespace,metricValue=1
```

この例では、右角括弧や左角括弧、二重引用符、および文字列 404 のようなリテラル文字列が使用されていました。このパターンでは、ログイベントをモニタリングするにはログイベントメッセージ全体が一致する必要があります。

describe-metric-filters コマンドを使用して、メトリクスフィルターの作成を検証できます。このような出力が表示されます。

```
aws logs describe-metric-filters --log-group-name MyApp/access.log

{
    "metricFilters": [
        {
            "filterName": "HTTP404Errors",
            "metricTransformations": [
```

```
                {
                    "metricValue": "1",
                    "metricNamespace": "MyNamespace",
                    "metricName": "ApacheNotFoundErrorCount"
                }
            ],
            "creationTime": 1399277571078,
            "filterPattern": "[ip, id, user, timestamp, request, status_code=404, size]"
        }
    ]
}
```

これでイベントをいくつか手動で投稿できます。

```
aws logs put-log-events \
--log-group-name MyApp/access.log --log-stream-name hostname \
--log-events \
timestamp=1394793518000,message="127.0.0.1 - bob [10/Oct/2000:13:55:36 -0700] \"GET /
apache_pb.gif HTTP/1.0\" 404 2326" \
timestamp=1394793528000,message="127.0.0.1 - bob [10/Oct/2000:13:55:36 -0700] \"GET /
apache_pb2.gif HTTP/1.0\" 200 2326"
```

サンプルログイベントを入力してすぐに、CloudWatch コンソールで ApacheNotFoundErrorCount という名前のメトリクスを取得できます。

例: HTTP 4xx コードをカウントする

前の例と同じように、ウェブサービスアクセスログをモニタリングしたり HTTP 応答コードレベルをモニタリングする場合があります。たとえば、HTTP 400 レベルのエラーをすべてモニタリングする場合です。ただし、それぞれのリターンコードに 1 つずつ新しいメトリクスフィルターを指定したくない場合があります。

以下の例は、「例: HTTP 404 コードをカウントする (p. 66)」の例の Apache アクセスログ形式を使用して、アクセスログから 400 レベルの HTTP コードレスポンスを含むメトリクスを作成する方法を示しています。

CloudWatch コンソールを使用してメトリクスフィルターを作成するには

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 [ログ] を選択します。
3. コンテンツペインでロググループを選択し、[メトリクスフィルターの作成] を選択します。
4. [ログメトリクスフィルターの定義] 画面の [フィルターパターン] に「**[ip, id, user, timestamp, request, status_code=4*, size]**」と入力します。
5. フィルターパターンをテストするには、[テストするログデータの選択] でメトリクスフィルターをテストするロググループを選択し、[パターンのテスト] を選択します。
6. [結果] に、ログファイルで見つかったフィルターパターンの発生数を示す CloudWatch Logs のメッセージが表示されます。

   詳細結果を表示するには、[テスト結果の表示] をクリックします。
7. [メトリクスの割り当て] を選択し、[メトリクスフィルターの作成とメトリクスの割り当て] 画面の [フィルター名] フィールドに「**HTTP4xxErrors**」と入力します。
8. [メトリクスの詳細] の [メトリクス名前空間] に、「**MyNameSpace**」と入力します。
9. [メトリクス名] に「HTTP4xxErrors」と入力します。
10. [メトリクスの詳細設定の表示] を選択し、[メトリクス値] が 1 であることを確認します。これにより、4xx エラーを含む各ログイベントのカウントは 1 ずつ増分されます。
11. [デフォルト値] に「0」と入力し、[フィルターの作成] を選択します。

AWS CLI を使用してメトリクスにフィルターを作成するには

コマンドプロンプトで、次のコマンドを実行します。

```
aws logs put-metric-filter \
  --log-group-name MyApp/access.log \
  --filter-name HTTP4xxErrors \
  --filter-pattern '[ip, id, user, timestamp, request, status_code=4*, size]' \
  --metric-transformations \
  metricName=HTTP4xxErrors,metricNamespace=MyNamespace,metricValue=1,defaultValue=0
```

put-event 呼び出しの次のデータを使用してこのルールをテストできます。前の例のモニタリングのルールを削除していない場合は、2 つの異なるメトリクスを生成します。

```
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /~test/ HTTP/1.1" 200 3
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308
127.0.0.1 - - [24/Sep/2013:11:51:34 -0700] "GET /~test/index.html HTTP/1.1" 200 3
```

例: Apache ログからのフィールドの抽出

カウントの代わりに、個別のログイベント内の値をメトリクス値に使用する方が役に立つ場合があります。この例では、Apache ウェブサーバーが転送したバイト数を計測するメトリクスを作成する抽出ルールの作成方法を示しています。

この抽出ルールは、ログイベントの 7 つのフィールドと一致します。メトリクス値は 7 番目に一致したトークンの値です。抽出ルールの `metricValue` フィールドにある「$7」がトークンの参照です。

CloudWatch コンソールを使用してメトリクスフィルターを作成するには

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 [ログ] を選択します。
3. コンテンツペインでロググループを選択し、[メトリクスフィルターの作成] を選択します。
4. [ログメトリクスフィルターの定義] 画面の [フィルターパターン] に「**`[ip, id, user, timestamp, request, status_code, size]`**」と入力します。
5. フィルターパターンをテストするには、[テストするログデータの選択] でメトリクスフィルターをテストするロググループを選択し、[パターンのテスト] を選択します。
6. [結果] に、ログファイルで見つかったフィルターパターンの発生数を示す CloudWatch Logs のメッセージが表示されます。

   詳細結果を表示するには、[テスト結果の表示] をクリックします。
7. [メトリクスの割り当て] を選択し、[メトリクスフィルターの作成とメトリクスの割り当て] 画面の [フィルター名] フィールドに「size」と入力します。
8. [メトリクスの詳細] の [メトリクス名前空間] に、「**`MyNameSpace`**」と入力します。
9. [メトリクス名] に「**`BytesTransferred`**」と入力します。
10. [メトリクスの詳細設定の表示] を選択し、[メトリクス値] に「**`$size`**」と入力します。
11. [デフォルト値] に「0」と入力し、[フィルターの作成] を選択します。

AWS CLI を使用してメトリクスにフィルターを作成するには

コマンドプロンプトで、次のコマンドを実行します。

```
aws logs put-metric-filter \
--log-group-name MyApp/access.log \
--filter-name BytesTransferred \
--filter-pattern '[ip, id, user, timestamp, request, status_code=4*, size]' \
--metric-transformations \
metricName=BytesTransferred,metricNamespace=MyNamespace,metricValue=$size,defaultValue=0
```

put-log-event 呼び出しの次のデータを使用してこのルールをテストできます。前の例のモニタリングルールを削除していない場合は、2 つの異なるメトリクスを生成します。

```
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /~test/ HTTP/1.1" 200 3
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308
127.0.0.1 - - [24/Sep/2013:11:51:34 -0700] "GET /~test/index.html HTTP/1.1" 200 3
```

メトリクスフィルターの一覧表示

ロググループ内のメトリクスフィルターを一覧表示できます。

CloudWatch コンソールを使用してメトリクスフィルターを一覧表示するには

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 [ログ] を選択します。
3. コンテンツペインのロググループのリストで、[メトリクスフィルター] 列でフィルター数を選択します。

   [ロググループ > フィルター] 画面にそのロググループに関連付けられたすべてのメトリクスフィルターが一覧表示されます。

AWS CLI を使用してメトリクスフィルターを一覧表示するには

コマンドプロンプトで、次のコマンドを実行します。

```
aws logs describe-metric-filters --log-group-name MyApp/access.log
```

出力例を次に示します。

```
{
    "metricFilters": [
        {
            "filterName": "HTTP404Errors",
            "metricTransformations": [
                {
                    "metricValue": "1",
                    "metricNamespace": "MyNamespace",
                    "metricName": "ApacheNotFoundErrorCount"
                }
            ],
            "creationTime": 1399277571078,
            "filterPattern": "[ip, id, user, timestamp, request, status_code=404, size]"
        }
    ]
}
```

メトリクスフィルターの削除

ポリシーは、名前と所属するロググループで識別されます。

CloudWatch コンソールを使用してメトリクスフィルターを削除するには

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 [ログ] を選択します。
3. コンテンツペインの [メトリクスフィルター] 列でメトリクスフィルターを選択します。
4. [ログメトリクスフィルター] 画面のメトリクスフィルターで、[フィルターの削除] を選択します。
5. 確認を求めるメッセージが表示されたら、[はい、削除する] を選択します。

AWS CLI を使用してメトリクスフィルターを削除するには

コマンドプロンプトで、次のコマンドを実行します。

```
aws logs delete-metric-filter --log-group-name MyApp/access.log \
 --filter-name MyFilterName
```

フィルターパターンを使用したログデータ検索

ログデータは、フィルターとパターンの構文 (p. 57) を使用して検索できます。ロググループ内のすべてのログストリームを検索できます。また、AWS CLI を使用して、特定のログストリームを検索することもできます。各検索を実行すると、最大で、見つかったデータの最初のページと、データの次のページを取得するか検索を続行するためのトークンが返されます。結果が返されない場合は、検索を続行できます。

クエリを実行する時間範囲を設定し、検索範囲を制限することができます。広い範囲から開始して関心のあるログ行が収まっている場所を確認した後、時間範囲を短縮し、関心のある時間範囲のログまでビューを絞り込みます。

ログから抽出したメトリクスを直接、対応するログに移動することもできます。

コンソールを使用したログエントリの検索

コンソールを使用して、指定した基準を満たすログエントリを検索することができます。

コンソールを使用してログを検索するには

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 [ログ] を選択します。
3. [ロググループ] で、検索するログストリームを含むロググループの名前を選択します。
4. [ログストリーム] で、検索するログストリームの名前を選択します。
5. [フィルター] で、使用するメトリクスフィルター構文を入力し、Enter キーを押します。

コンソールを使用してすべてのログエントリで時間範囲を検索するには

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 [ログ] を選択します。
3. [ロググループ] で、検索するログストリームを含むロググループの名前を選択します。

4. [Search Events] を選択します。
5. [フィルター] に、使用するメトリクスフィルター構文を入力し、日時範囲を選択して Enter キーを押します。

AWS CLI を使用したログエントリの検索

AWS CLI を使用して、指定した基準を満たすログエントリを検索することができます。

AWS CLI を使用して検索ログエントリを検索するには

コマンドプロンプトで、次の filter-log-events コマンドを実行します。結果を指定したフィルターパターンに限定するには `--filter-pattern` を使用し、結果を指定したロググループに限定するには `--log-stream-names` を使用します。

```
aws logs filter-log-events --log-group-name my-group [--log-stream-
names LIST_OF_STREAMS_TO_SEARCH] --filter-pattern VALID_METRIC_FILTER_PATTERN]
```

AWS CLI を使用して一定の時間範囲におけるログエントリを検索するには

コマンドプロンプトで、次の filter-log-events コマンドを実行します。

```
aws logs filter-log-events --log-group-name my-group [--log-stream-
names LIST_OF_STREAMS_TO_SEARCH] [--start-time 1482197400000] [--end-time 1482217558365]
 [--filter-pattern VALID_METRIC_FILTER_PATTERN]
```

メトリクスからログへのピボット

コンソールの他の部分から、特定のログエントリに移動することができます。

ダッシュボードウィジェットからログに移動するには

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、ダッシュボードを選択します。
3. ダッシュボードを選択します。
4. ウィジェットで [View logs] アイコンを選択し、[View logs in this time range] を選択します。メトリクスフィルターが複数ある場合は、リストから 1 つ選択します。メトリクスフィルターをリストに表示しきれない場合は、[More metric filters] を選択し、メトリクスフィルターを選択するか検索します。

メトリクスからログに移動するには

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 メトリクスを選択します。
3. [All metrics] タブの検索フィールドに、メトリクスの名前を入力して Enter キーを押します。
4. 検索結果から 1 つ以上のメトリクスを選択します。
5. [Actions]、[View logs] の順に選択します。メトリクスフィルターが複数ある場合は、リストから 1 つ選択します。メトリクスフィルターをリストに表示しきれない場合は、[More metric filters] を選択し、メトリクスフィルターを選択するか検索します。

トラブルシューティング

[Search takes too long to complete]

ログデータが多い場合、検索の完了に時間がかかる場合があります。検索の速度を上げるには、次を実行します:

- AWS CLI を使用している場合は、検索対象を関心のあるログストリームのみに制限します。たとえば、ロググループに 1000 のログストリームがある状況で、関連することがわかっているログストリーム 3 つのみを確認する場合は、AWS CLI を使用して、検索対象をロググループ内のその 3 つのログストリームのみに制限できます。
- 時間範囲を短く、細かくして検索対象のデータ量を減らし、クエリの速度を上げます。

サブスクリプションを使用したログデータのリアルタイム処理

サブスクリプションを使用して CloudWatch Logs からのログイベントのリアルタイムフィードにアクセスし、カスタム処理、分析、他のシステムへのロードを行うために、Amazon Kinesis ストリーム、Amazon Kinesis Data Firehose ストリーム、AWS Lambda などの他のサービスに配信することができます。ログイベントのサブスクリプションを開始するには、Kinesis ストリームなど、イベントを配信する宛先ソースを作成します。サブスクリプションフィルタは、AWS リソースに配信されるログイベントのフィルタリングに使用するフィルタパターンと、一致するログイベントの送信先に関する情報を定義します。

また、CloudWatch Logs は、サブスクリプションへのログイベントの転送に関する CloudWatch メトリクスを作成します。詳細については、「Amazon CloudWatch Logs のメトリクスとディメンション」を参照してください。

コンテンツ

概念

各サブスクリプションフィルタは以下のキー要素で構成されています。

log group name

サブスクリプションフィルタを関連付けるロググループ。このロググループにアップロードされたすべてログイベントは、サブスクリプションフィルタの対象となり、フィルタパターンがログイベントに一致する場合は選択された Kinesis ストリームに配信されます。

フィルタパターン

CloudWatch Logs が各ログイベントのデータをどのように解釈するかの記号による説明と、送信先 AWS リソースに配信される内容を制限するフィルタリング表現。フィルタパターンの構文の詳細については、「フィルターとパターンの構文 (p. 57)」を参照してください。

destination arn

サブスクリプションフィードの送信先として使用する Kinesis ストリーム、Kinesis Data Firehose ストリーム、または Lambda 関数の Amazon リソースネーム (ARN)。

role arn

選択された Kinesis ストリームにデータを置くのに必要な権限を CloudWatch Logs に付与する IAM ロール。CloudWatch Logs は Lambda 関数自体のアクセスコントロール設定から必要なアクセス権限を取得できるため、Lambda の送信先にはこのロールは必要ありません。

ディストリビューション

送信先にログデータを配信するのに使用するメソッド。この場合、宛先は Amazon Kinesis のストリーム です。デフォルトでは、ログデータは、ログストリームによってグループ化されています。さらに詳細に分散する場合でも、ログデータはランダムにグループ化することができます。

CloudWatch Logs サブスクリプションフィルタの使用

Kinesis、Lambda、または Kinesis Data Firehose では、サブスクリプションフィルタを使用できます。

例

- 例 1: Kinesis のサブスクリプションフィルタ (p. 75)
- 例 2: AWS Lambda のサブスクリプションフィルタ (p. 78)
- 例 3: Amazon Kinesis Data Firehose のサブスクリプションフィルタ (p. 81)

例 1: Kinesis のサブスクリプションフィルタ

次の例では、サブスクリプションフィルタが、AWS CloudTrail イベントを含むロググループと関連付けられ、"ルート" AWS 認証情報により実行されたすべてのログアクティビティが "RootAccess" と呼ばれる Kinesis ストリームに配信されます。 AWS CloudTrail イベントを CloudWatch Logs に送信する方法の詳細については、『AWS CloudTrail User Guide』の「CloudTrail イベントの CloudWatch Logs への送信」を参照してください。

> **Note**
>
> Kinesis ストリームを作成する前に、生成するログデータのボリュームを計算します。このボリュームを処理するために十分なシャードで Kinesis ストリームを作成するように注意してください。ストリームに十分なシャードがないと、ログストリームはスロットリングされます。Kinesis のストリームボリューム制限に関する詳細は、「Amazon Kinesis データストリーム制限」を参照してください。

Kinesis のサブスクリプションフィルタを作成するには

1. 次のコマンドを使用して送信先 Kinesis ストリームを作成します。

```
$ C:\>  aws kinesis create-stream --stream-name "RootAccess" --shard-count 1
```

2. Kinesis ストリームが [アクティブ] になるまで待ちます (これには 1 ~ 2 分かかる可能性があります)。次の Kinesis describe-stream コマンドを使用して、StreamDescription.StreamStatus プロパティをチェックできます。さらに、後のステップで必要になるため StreamDescription.StreamARN 値を書き留めます。

```
aws kinesis describe-stream --stream-name "RootAccess"
```

出力例を次に示します。

```
{
    "StreamDescription": {
        "StreamStatus": "ACTIVE",
        "StreamName": "RootAccess",
        "StreamARN": "arn:aws:kinesis:us-east-1:123456789012:stream/RootAccess",
        "Shards": [
            {
                "ShardId": "shardId-000000000000",
                "HashKeyRange": {
                    "EndingHashKey": "340282366920938463463374607431768211455",
                    "StartingHashKey": "0"
                },
                "SequenceNumberRange": {
```

```
                    "StartingSequenceNumber":
                    "49551135218688818456679503831981458784591352702181572610"
                }
            }
        ]
    }
}
```

3. Kinesis ストリームにデータを置く権限を CloudWatch Logs に付与する IAM ロールを作成します。まず、ファイル (~/TrustPolicyForCWL.json など) で信頼ポリシーを作成する必要があります。テキストエディタを使用してこのポリシーを作成します。IAM コンソールを使用してポリシーを作成しないでください。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.region.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

4. create-role コマンドを使用し、信頼ポリシーファイルを指定して IAM ロールを作成します。後のステップで必要になるため、返された Role.Arn 値も書き留めます。

```
aws iam create-role --role-name CWLtoKinesisRole --assume-role-policy-document file://
~/TrustPolicyForCWL.json
```

```
{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.region.amazonaws.com"
                }
            }
        },
        "RoleId": "AAOIIAH450GAB4HC5F431",
        "CreateDate": "2015-05-29T13:46:29.431Z",
        "RoleName": "CWLtoKinesisRole",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisRole"
    }
}
```

5. 権限ポリシーを作成し、CloudWatch Logs がアカウントで実行できるアクションを定義します。まず、ファイル (~/PermissionsForCWL.json など) で権限ポリシーを作成します。テキストエディタを使用してこのポリシーを作成します。IAM コンソールを使用してポリシーを作成しないでください。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:region:123456789012:stream/RootAccess"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
```

```
      "Resource": "arn:aws:iam::123456789012:role/CWLtoKinesisRole"
    }
  ]
}
```

6. 次の put-role-policy コマンドを使用して、アクセス許可ポリシーをロールに関連付けます。

```
aws iam put-role-policy  --role-name CWLtoKinesisRole  --policy-name Permissions-
Policy-For-CWL  --policy-document file://~/PermissionsForCWL.json
```

7. Kinesis ストリームが [Active] 状態になり、IAM ロールを作成したら、CloudWatch Logs サブスクリプションフィルタを作成できます。サブスクリプションフィルタにより、選択されたロググループから Kinesis ストリームへのリアルタイムログデータの流れがすぐに開始されます。

```
aws logs put-subscription-filter \
    --log-group-name "CloudTrail" \
    --filter-name "RootAccess" \
    --filter-pattern "{$.userIdentity.type = Root}" \
    --destination-arn "arn:aws:kinesis:region:123456789012:stream/RootAccess" \
    --role-arn "arn:aws:iam::123456789012:role/CWLtoKinesisRole"
```

8. サブスクリプションフィルタを設定したら、CloudWatch Logs によりフィルタパターンに一致するすべての受信ログイベントが Kinesis ストリームに転送されます。これが起きていることは、Kinesis シャードイテレータを取得し、Kinesis get-records コマンドを使用していくつかの Kinesis レコードを取得することで確認できます。

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

```
{
    "ShardIterator":
    "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+OIK8zM5My8ID+g6rMo7UKWeI4+IWiK2OSh0uP"
}
```

```
aws kinesis get-records --limit 10 --shard-iterator "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+OIK8zM5My8ID+g6rMo7UKWeI4+IWiK2OSh0uP"
```

Kinesis がデータを返し始めるまで、この呼び出しを数回行う必要があるかもしれない点に注意してください。

レコードの配列を含むレスポンスが表示されます。Kinesis レコードの Data 属性は、Base64 でエンコードされており、gzip 形式で圧縮されています。raw データは、コマンドラインから次の UNIX コマンドを使用して調べることができます。

```
echo -n "<Content of Data>" | base64 -d | zcat
```

Base64 でデコードおよび圧縮されたデータは、次の構造を使用して JSON としてフォーマットされます。

```
{
    "owner": "111111111111",
    "logGroup": "CloudTrail",
```

```
    "logStream": "111111111111_CloudTrail_us-east-1",
    "subscriptionFilters": [
        "Destination"
    ],
    "messageType": "DATA_MESSAGE",
    "logEvents": [
        {
            "id": "31953106606966983378809025079804211143289615424298221568",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221569",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221570",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        }
    ]
}
```

上のデータ構造の主な要素は次のとおりです。

owner

発行元ログデータの AWS アカウント ID。

logGroup

発行元ログデータのロググループ名。

logStream

発行元ログデータのログストリーム名。

subscriptionFilters

発行元ログデータと一致したサブスクリプションフィルタ名のリスト。

messageType

データメッセージは、"DATA_MESSAGE" 型を使用します。CloudWatch Logs は、主に送信先が到達可能であるかどうかをチェックするために、"CONTROL_MESSAGE" 型の Kinesis レコードを発行することがあります。

logEvents

ログイベントレコードの配列として表される実際のログデータ。"id" プロパティは、各ログイベントの一意識別子です。

例 2: AWS Lambda のサブスクリプションフィルタ

この例では、CloudWatch Logs サブスクリプションのフィルタを作成して、ログデータを AWS Lambda 関数に送信します。

Note

Lambda 関数を作成する前に、生成するログデータのボリュームを計算します。このボリュームを処理できる関数を作成するように注意してください。関数に十分なボリュームがないと、ログストリームはスロットリングされます。Lambda 制限の詳細については、「AWS Lambda の制限」を参照してください。

Lambda のサブスクリプションフィルタを作成するには

1. AWS Lambda 関数を作成します。

   Lambda 実行ロールをセットアップ済みであることを確認します。詳細については、『AWS Lambda Developer Guide』の「2.2: IAM ロールの作成 (実行ロール)」を参照してください。

2. テキストエディターを開き、以下の内容で `helloWorld.js` という名前のファイルを作成します。

```
var zlib = require('zlib');
exports.handler = function(input, context) {
    var payload = new Buffer(input.awslogs.data, 'base64');
    zlib.gunzip(payload, function(e, result) {
        if (e) {
            context.fail(e);
        } else {
            result = JSON.parse(result.toString('ascii'));
            console.log("Event Data:", JSON.stringify(result, null, 2));
            context.succeed();
        }
    });
};
```

3. helloWorld.js ファイルを圧縮して `helloWorld.zip` という名前で保存します。
4. 次のコマンドを使用します。ロールは、最初のステップで設定した Lambda 実行ロールです。

```
aws lambda create-function \
    --function-name helloworld \
    --zip-file file://file-path/helloWorld.zip \
    --role lambda-execution-role-arn \
    --handler helloWorld.handler \
    --runtime nodejs4.3
```

5. CloudWatch Logs に、関数を実行するためのアクセス権限を付与します。次のコマンドを使用します。プレースホルダーは自身のアカウントに置き換え、プレースホルダーロググループは処理するロググループに置き換えます。

```
aws lambda add-permission \
    --function-name "helloworld" \
    --statement-id "helloworld" \
    --principal "logs.region.amazonaws.com" \
    --action "lambda:InvokeFunction" \
    --source-arn "arn:aws:logs:region:123456789123:log-group:TestLambda:*" \
    --source-account "123456789012"
```

6. 次のコマンドを使用してサブスクリプションフィルタを作成します。プレースホルダーアカウントは自身のアカウントに置き換え、プレースホルダーロググループは処理するロググループに置き換えます。

```
aws logs put-subscription-filter \
    --log-group-name myLogGroup \
    --filter-name demo \
    --filter-pattern "" \
```

```
    --destination-arn arn:aws:lambda:region:123456789123:function:helloworld
```

7. (オプション) サンプルのログイベントを使用してテストします。コマンドプロンプトで、次のコマンドを実行します。これにより、サブスクライブしたストリームに単純なログメッセージを送信されます。

   Lambda 関数の出力を確認するには、Lambda 関数に移動して、/aws/lambda/helloworld にある出力を参照します。

```
aws logs put-log-events --log-group-name myLogGroup --log-stream-name stream1 --log-
events "[{\"timestamp\":<CURRENT TIMESTAMP MILLIS> , \"message\": \"Simple Lambda
 Test\"}]"
```

   Lambda の配列を含むレスポンスが表示されます。Lambda レコードの Data 属性は、Base64 でエンコードされており、gzip 形式で圧縮されています。Lambda が受け取る実際のペイロードは、{ "awslogs": {"data": "BASE64ENCODED_GZIP_COMPRESSED_DATA"} } の形式になります。raw データは、コマンドラインから次の UNIX コマンドを使用して調べることができます。

```
echo -n "<BASE64ENCODED_GZIP_COMPRESSED_DATA>" | base64 -d | zcat
```

   Base64 でデコードおよび圧縮されたデータは、次の構造を使用して JSON としてフォーマットされます。

```
{
    "owner": "123456789012",
    "logGroup": "CloudTrail",
    "logStream": "123456789012_CloudTrail_us-east-1",
    "subscriptionFilters": [
        "Destination"
    ],
    "messageType": "DATA_MESSAGE",
    "logEvents": [
        {
            "id": "31953106606966983378809025079804211143289615424298221568",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221569",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424298221570",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root
\"}"
        }
    ]
}
```

   上のデータ構造の主な要素は次のとおりです。

   owner

   発行元ログデータの AWS アカウント ID。

logGroup

発行元ログデータのロググループ名。

logStream

発行元ログデータのログストリーム名。

subscriptionFilters

発行元ログデータと一致したサブスクリプションフィルタ名のリスト。

messageType

データメッセージは、"DATA_MESSAGE" 型を使用します。CloudWatch Logs は、主に送信先が到達可能であるかどうかをチェックするために、"CONTROL_MESSAGE" 型の Lambda レコードを発行することがあります。

logEvents

ログイベントレコードの配列として表される実際のログデータ。"id" プロパティは、各ログイベントの一意識別子です。

例 3: Amazon Kinesis Data Firehose のサブスクリプションフィルタ

この例では、CloudWatch Logs サブスクリプションを作成して、Amazon Kinesis Data Firehose 配信ストリームに定義されたフィルタに一致する受信ログイベントを送信します。CloudWatch Logs から Amazon Kinesis Data Firehose に送信されたデータは、すでに gzip レベル 6 圧縮で圧縮されているため、Kinesis Data Firehose 配信ストリーム内で圧縮を使用する必要はありません。

Note

Kinesis Data Firehose ストリームを作成する前に、生成するログデータのボリュームを計算します。このボリュームを処理できる Kinesis Data Firehose ストリームを作成するように注意してください。ストリームがこのボリュームを処理できない場合、ログストリームはスロットリングされます。Kinesis Data Firehose のストリームボリューム制限に関する詳細は、「Amazon Kinesis Data Firehose データ制限」を参照してください。

Kinesis Data Firehose のサブスクリプションフィルタを作成するには

1. Amazon Simple Storage Service (Amazon S3) バケットの作成。CloudWatch Logs 専用に作成したバケットを使用することをお勧めします。ただし、既存のバケットを使用する場合は、ステップ 2 に進みます。

   次のコマンドを実行します。プレースホルダーリージョンは、使用するリージョンに置き換えます。

   ```
   aws s3api create-bucket --bucket my-bucket --create-bucket-configuration
    LocationConstraint=region
   ```

   出力例を次に示します。

   ```
   {
       "Location": "/my-bucket"
   }
   ```

2. IAM ロールを作成して、Amazon Kinesis Data Firehose に Amazon S3 バケットにデータを置く権限を付与します。

詳細については、『Amazon Kinesis Data Firehose 開発者ガイド』の「Amazon Kinesis Data Firehose を使用したユーザーアクセスの制御」を参照してください。

まず、テキストエディタを使用して次のようにファイル ~/TrustPolicyForFirehose.json で信頼ポリシーを作成します。account-id は、AWS アカウント ID で置き換えます。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "firehose.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": { "StringEquals": { "sts:ExternalId":"account-id" } }
  }
}
```

3. create-role コマンドを使用し、信頼ポリシーファイルを指定して IAM ロールを作成します。後のステップで必要になるため、返された Role.Arn 値を書き留めます。

```
aws iam create-role \
      --role-name FirehosetoS3Role \
      --assume-role-policy-document file://~/TrustPolicyForFirehose.json

{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "firehose.amazonaws.com"
                }
            }
        },
        "RoleId": "AAOIIAH450GAB4HC5F431",
        "CreateDate": "2015-05-29T13:46:29.431Z",
        "RoleName": "FirehosetoS3Role",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/FirehosetoS3Role"
    }
}
```

4. 権限ポリシーを作成し、Kinesis Data Firehose がアカウントで実行できるアクションを定義します。まず、テキストエディタを使用してファイル ~/PermissionsForFirehose.json で権限ポリシーを作成します。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "s3:AbortMultipartUpload",
          "s3:GetBucketLocation",
          "s3:GetObject",
          "s3:ListBucket",
          "s3:ListBucketMultipartUploads",
          "s3:PutObject" ],
      "Resource": [
          "arn:aws:s3:::my-bucket",
          "arn:aws:s3:::my-bucket/*" ]
    }
  ]
```

```
}
```

5. 次の put-role-policy コマンドを使用して、アクセス許可ポリシーをロールに関連付けます。

```
aws iam put-role-policy --role-name FirehosetoS3Role --policy-name Permissions-Policy-
For-Firehose --policy-document file://~/PermissionsForFirehose.json
```

6. 次のように、送信先 Kinesis Data Firehose 配信ストリームを作成します。RoleARN と BucketARN のプレースホルダー値を、作成したロールおよびバケット ARN に置き換えます。

```
aws firehose create-delivery-stream \
   --delivery-stream-name 'my-delivery-stream' \
   --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::123456789012:role/FirehosetoS3Role", "BucketARN":
 "arn:aws:s3:::my-bucket"}'
```

Kinesis Data Firehose は、Amazon S3 オブジェクトに提供された YYYY/MM/DD/HH UTC 時間形式をプレフィックスで自動的に使用することに注意してください。時間形式プレフィックスの前に、追加のプレフィックスを指定できます。プレフィックスの最後がフォワードスラッシュ (/) の場合は、Amazon S3 バケット内のフォルダとして表示されます。

7. ストリームがアクティブになるまで待ちます (これには数分かかる可能性があります)。Kinesis Data Firehose describe-delivery-stream コマンドを使用して、DeliveryStreamDescription.DeliveryStreamStatus プロパティをチェックできます。さらに、後のステップで必要になるため、DeliveryStreamDescription.DeliveryStreamARN 値を書き留めます。

```
aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
{
    "DeliveryStreamDescription": {
        "HasMoreDestinations": false,
        "VersionId": "1",
        "CreateTimestamp": 1446075815.822,
        "DeliveryStreamARN": "arn:aws:firehose:us-east-1:123456789012:deliverystream/
my-delivery-stream",
        "DeliveryStreamStatus": "ACTIVE",
        "DeliveryStreamName": "my-delivery-stream",
        "Destinations": [
            {
                "DestinationId": "destinationId-000000000001",
                "S3DestinationDescription": {
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "RoleARN": "delivery-stream-role",
                    "BucketARN": "arn:aws:s3:::my-bucket",
                    "BufferingHints": {
                        "IntervalInSeconds": 300,
                        "SizeInMBs": 5
                    }
                }
            }
        ]
    }
}
```

8. IAM ロールを作成して、CloudWatch Logs に Kinesis Data Firehose 送信ストリームにデータを置く権限を付与します。まず、テキストエディタを使用してファイル `~/TrustPolicyForCWL.json` で信頼ポリシーを作成します。

```
{
```

```
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.region.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

9. create-role コマンドを使用し、信頼ポリシーファイルを指定して IAM ロールを作成します。後のステップで必要になるため、返された Role.Arn 値を書き留めます。

```
aws iam create-role \
      --role-name CWLtoKinesisFirehoseRole \
      --assume-role-policy-document file://~/TrustPolicyForCWL.json

{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.region.amazonaws.com"
                }
            }
        },
        "RoleId": "AAOIIAH450GAB4HC5F431",
        "CreateDate": "2015-05-29T13:46:29.431Z",
        "RoleName": "CWLtoKinesisFirehoseRole",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"
    }
}
```

10. 権限ポリシーを作成し、CloudWatch Logs がアカウントで実行できるアクションを定義します。まず、テキストエディタを使用して権限ポリシーファイル (例: `~/PermissionsForCWL.json`) を作成します。

```
{
    "Statement":[
      {
        "Effect":"Allow",
        "Action":["firehose:*"],
        "Resource":["arn:aws:firehose:region:123456789012:*"]
      },
      {
        "Effect":"Allow",
        "Action":["iam:PassRole"],
        "Resource":["arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"]
      }
    ]
}
```

11. put-role-policy コマンドを使用して、権限ポリシーをロールに関連付けます。

```
aws iam put-role-policy --role-name CWLtoKinesisFirehoseRole --policy-name Permissions-Policy-For-CWL --policy-document file://~/PermissionsForCWL.json
```

12. Amazon Kinesis Data Firehose ストリームが Active 状態になり、IAM ロールを作成したら、CloudWatch Logs サブスクリプションフィルタを作成できます。サブスクリプションフィルタにより、選択されたロググループから Amazon Kinesis Data Firehose 送信ストリームへのリアルタイムログデータの流れがすぐに開始されます。

```
aws logs put-subscription-filter \
    --log-group-name "CloudTrail" \
    --filter-name "Destination" \
    --filter-pattern "{$.userIdentity.type = Root}" \
    --destination-arn "arn:aws:firehose:region:123456789012:deliverystream/my-delivery-
stream" \
    --role-arn "arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"
```

13. サブスクリプションフィルタを設定したら、CloudWatch Logs によりフィルタパターンに一致するすべての受信ログイベントが Amazon Kinesis Data Firehose 送信ストリームに転送されます。データは、Amazon Kinesis Data Firehose 配信ストリームに設定された時間の間隔に基づいて、Amazon S3 に表示されます。十分な時間が経過すると、Amazon S3 バケットをチェックしてデータを確認できます。

```
aws s3api list-objects --bucket 'my-bucket' --prefix 'firehose/'
{
    "Contents": [
        {
            "LastModified": "2015-10-29T00:01:25.000Z",
            "ETag": "\"a14589f8897f4089d3264d9e2d1f1610\"",
            "StorageClass": "STANDARD",
            "Key": "firehose/2015/10/29/00/my-delivery-stream-2015-10-29-00-01-21-
a188030a-62d2-49e6-b7c2-b11f1a7ba250",
            "Owner": {
                "DisplayName": "cloudwatch-logs",
                "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b5"
            },
            "Size": 593
        },
        {
            "LastModified": "2015-10-29T00:35:41.000Z",
            "ETag": "\"a7035b65872bb2161388ffb63dd1aec5\"",
            "StorageClass": "STANDARD",
            "Key": "firehose/2015/10/29/00/my-delivery-
stream-2015-10-29-00-35-40-7cc92023-7e66-49bc-9fd4-fc9819cc8ed3",
            "Owner": {
                "DisplayName": "cloudwatch-logs",
                "ID": "1ec9cf700ef6be062b19584e0b7d84ecc19237f87b6"
            },
            "Size": 5752
        }
    ]
}
```

```
aws s3api get-object --bucket 'my-bucket' --key 'firehose/2015/10/29/00/my-delivery-
stream-2015-10-29-00-01-21-a188030a-62d2-49e6-b7c2-b11f1a7ba250' testfile.gz

{
    "AcceptRanges": "bytes",
    "ContentType": "application/octet-stream",
    "LastModified": "Thu, 29 Oct 2015 00:07:06 GMT",
    "ContentLength": 593,
    "Metadata": {}
}
```

Amazon S3 オブジェクトのデータは、gzip 形式で圧縮されます。raw データは、コマンドラインから次の UNIX コマンドを使用して調べることができます。

```
zcat testfile.gz
```

クロスアカウントのログデータをサブスクリプションと共有する

別の AWS アカウントの所有者と協力して、Amazon Kinesis ストリームなどのご自身の AWS リソースで相手のログイベントを受信することができます (これは、クロスアカウントデータ共有と呼ばれます)。たとえば、このログイベントデータを集中管理型の Amazon Kinesis ストリームから読み取り、カスタム処理や分析を実行することができます。カスタム処理は、多数のアカウントが協力しデータを分析する場合に特に便利です。たとえば、ある会社の情報セキュリティグループがリアルタイムで侵入または異常な挙動を検出するためにデータを分析するとします。この場合、会社の全部署のアカウントのフェデレーションされた本稼働ログを中央処理のために収集することによって、これらのアカウントの監査を行うことができます。これらすべてのアカウントのイベントデータのリアルタイムストリームは、アセンブルした後に、Kinesis を使用してデータを既存のセキュリティ分析システムにアタッチできる情報セキュリティグループに配信できます。

Kinesis ストリームは、クロスアカウントサブスクリプションの送信先として現在サポートされている唯一のリソースです。

複数のアカウントでログデータを共有するには、ログデータの送信者と受信者を確立する必要があります。

- [Log data sender]—受取人から送信先情報を取得し、そのログイベントを特定の送信先に送信する準備が完了していることを CloudWatch Logs に通知します。このセクションの残りの手順では、ログデータの送信者に、架空の AWS アカウント番号 111111111111 が表示されます。
- [Log data recipient]—Kinesis ストリームをカプセル化する送信先を設定し、受取人がログデータの受け取りを希望していることを CloudWatch Logs に通知します。この後、受取人は自分の送信先に関する情報を送信者と共有します。このセクションの残りの手順では、ログデータの受信者に、架空の AWS アカウント番号 999999999999 が表示されます。

クロスアカウントのユーザーからのログイベントの受け取りを開始するには、ログデータの受取人がまず CloudWatch Logs 送信先を作成する必要があります。各送信先は以下のキー要素で構成されています。

送信先名

作成する送信先の名前。

ターゲット ARN

サブスクリプションフィードの送信先として使用する AWS リソースの Amazon リソースネーム (ARN)。

ロールの ARN

特定の Kinesis ストリームにデータを入力するために必要なアクセス許可を CloudWatch Logs に付与する AWS Identity and Access Management (IAM) ロール。

アクセスポリシー

送信先に書き込むことが許可されている一連のユーザーを管理する IAM ポリシードキュメント (IAM ポリシー構文を使用して記述された JSON 形式のドキュメント) 。

ロググループと送信先は同じ AWS リージョンに存在している必要があります。ただし、送信先が指す AWS リソースは、別のリージョンに配置することができます。

トピック

送信先を作成する

この手順のステップは、ログデータの受取人アカウントで行われます。この例では、ログデータの受取人アカウントの AWS アカウント ID は 999999999999 で、ログデータの送信者の AWS アカウント ID は 111111111111 です。

この例では、RecipientStream という名前の Kinesis ストリームを使用して送信先を作成し、次に CloudWatch Logs がそれにデータを書き込むことを許可するロールを作成します。

送信先を作成するには

1. Kinesis で送信先ストリームを作成します。コマンドプロンプトで、次のように入力します。

```
aws kinesis create-stream --stream-name "RecipientStream" --shard-count 1
```

2. Kinesis ストリームがアクティブになるまで待ちます。aws kinesis describe-stream コマンドを使用して、StreamDescription.StreamStatus プロパティをチェックできます。さらに、StreamDescription.StreamARN 値を書き留めておきます。これは後で CloudWatch Logs に渡されるからです。

```
aws kinesis describe-stream --stream-name "RecipientStream"
{
  "StreamDescription": {
    "StreamStatus": "ACTIVE",
    "StreamName": "RecipientStream",
    "StreamARN": "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream",
    "Shards": [
      {
        "ShardId": "shardId-000000000000",
        "HashKeyRange": {
          "EndingHashKey": "340282366920938463463374607743176EXAMPLE",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
 "49551135218688818456679503831981458784591352702l8EXAMPLE"
        }
      }
    ]
  }
}
```

ストリームがアクティブ状態で表示されるまでに 1 ~ 2 分かかる場合があります。

3. IAM が CloudWatch Logs ストリームにデータを入力することを許可する Kinesis ロールを作成します。まず、ファイル ~/TrustPolicyForCWL.json で信頼ポリシーを作成する必要があります。このポリシーの作成にはテキストエディタを使用します。IAM コンソールは使用しないでください。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.region.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

4. aws iam create-role コマンドを使用して、信頼ポリシーファイルを指定する IAM ロールを作成します。返された Role.Arn 値を書き留めておきます。これも後で CloudWatch Logs に渡されるからです。

```
aws iam create-role \
    --role-name CWLtoKinesisRole \
    --assume-role-policy-document file://~/TrustPolicyForCWL.json

{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.region.amazonaws.com"
                }
            }
        },
        "RoleId": "AAOIIAH450GAB4HC5F431",
        "CreateDate": "2015-05-29T13:46:29.431Z",
        "RoleName": "CWLtoKinesisRole",
        "Path": "/",
        "Arn": "arn:aws:iam::999999999999:role/CWLtoKinesisRole"
    }
}
```

5. アクセス許可ポリシーを作成して、CloudWatch Logs がお客様のアカウントで実行できるアクションを定義します。まず、テキストエディタを使用してファイル ~/PermissionsForCWL.json で権限ポリシーを作成します。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:region:999999999999:stream/RecipientStream"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::999999999999:role/CWLtoKinesisRole"
    }
  ]
}
```

6. aws iam put-role-policy コマンドを使用して、アクセス許可ポリシーをロールに関連付けます。

```
aws iam put-role-policy \
    --role-name CWLtoKinesisRole \
    --policy-name Permissions-Policy-For-CWL \
    --policy-document file://~/PermissionsForCWL.json
```

7. Kinesis ストリームがアクティブ状態になり、IAM ロールが作成されたら、CloudWatch Logs の送信先を作成できます。

   a. このステップでは、アクセスポリシーと送信先は関連付けられません。送信先の作成を完了するには 2 つのステップを行う必要がありますが、このステップはその最初のステップです。ペイロードで返された DestinationArn を書き留めておいてください。

```
aws logs put-destination \
    --destination-name "testDestination" \
```

```
    --target-arn "arn:aws:kinesis:region:999999999999:stream/RecipientStream" \
    --role-arn "arn:aws:iam::999999999999:role/CWLtoKinesisRole"

{
  "DestinationName" : "testDestination",
  "RoleArn" : "arn:aws:iam::999999999999:role/CWLtoKinesisRole",
  "DestinationArn" : "arn:aws:logs:us-
east-1:999999999999:destination:testDestination",
  "TargetArn" : "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream"
}
```

b. ステップ 7a が完了したら、ログデータの受取人アカウントで、アクセスポリシーを送信先に関連付けます。このポリシーを使用すると、ログデータの送信者アカウント (111111111111) はログデータの受取人アカウント (999999999999) の送信先にアクセスできます。テキストエディタを使用して、このポリシーを ~/AccessPolicy.json ファイルに配置できます。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "111111111111"
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:region:999999999999:destination:testDestination"
    }
  ]
}
```

c. これにより、誰が送信先に書き込むことができるかを定義するポリシーが作成されます。このポリシーでは、送信先にアクセスするための logs:PutSubscriptionFilter アクションが指定されている必要があります。クロスアカウントのユーザーは、PutSubscriptionFilter アクションを使用して送信先にログイベントを送信します。

```
aws logs put-destination-policy \
    --destination-name "testDestination" \
    --access-policy file://~/AccessPolicy.json
```

このアクセスポリシーでは、ID 111111111111 の AWS アカウントのルートユーザーは、arn:aws:logs:region:999999999999:destination:testDestination という ARN を持つ送信先に対して、PutSubscriptionFilter を呼び出すことを許可します。他のユーザーがこの送信先に対して PutSubscriptionFilter を呼び出そうとしても、それは却下されます。

アクセスポリシーに照らし合わせてユーザーの特権を検証するには、『IAM ユーザーガイド』の「Policy Validator の使用」を参照してください。

サブスクリプションフィルタを作成する

送信先を作成したら、ログデータの受信者アカウントは、送信先の ARN (arn:aws:logs:us-east-1:999999999999:destination:testDestination) を他の AWS アカウントと共有できるようになります。これにより、これらのアカウントは同じ送信先にログイベントを送信できます。この後、これらの他の送信アカウントのユーザーは、この送信先に対するサブスクリプションフィルタをそれぞれのロググループに作成します。サブスクリプションフィルタは、特定のロググループから特定の送信先へのリアルタイムログデータの送信をすぐに開始します。

次の例は、AWS CloudTrail イベントを含んでいるロググループにサブスクリプションフィルタを関連付けます。これにより、ログに記録されている、"Root" AWS 認証情報によって行われたすべてのアクティビ

ティが、先ほど作成した "RecipientStream" という名前の Kinesis ストリームをカプセル化する送信先に配信されます。AWS CloudTrail イベントを CloudWatch Logs に送信する方法の詳細については、『AWS CloudTrail User Guide』の「CloudTrail イベントの CloudWatch Logs への送信」を参照してください。

```
aws logs put-subscription-filter \
    --log-group-name "CloudTrail" \
    --filter-name "RecipientStream" \
    --filter-pattern "{$.userIdentity.type = Root}" \
    --destination-arn "arn:aws:logs:region:999999999999:destination:testDestination"
```

ロググループと送信先は同じ AWS リージョンに存在している必要があります。ただし、送信先は、別のリージョンに配置された Kinesis ストリームなどの AWS リソースを指すことができます。

Note

サブスクリプションを使用したログデータのリアルタイム処理 (p. 74) のサブスクリプションの例とは異なり、この例では role-arn を指定する必要がありませんでした。role-arn は Kinesis ストリームへの書き込み中に "なりすます" ために必要とされるからであり、これは既に送信先の作成中に送信先の所有者によって指定されています。

ログイベントの送信を検証する

サブスクリプションフィルタを作成したら、CloudWatch Logs が、フィルタパターンと一致するすべての受信ログイベントを、送信先ストリーム内でカプセル化されている「RecipientStream」という名前の Kinesis ストリームに転送します。送信先の所有者は、aws kinesis get-shard-iterator コマンドを使用して Kinesis シャードを取得し、aws kinesis get-records コマンドを使用していくつかの Kinesis レコードをフェッチすることにより、これが実際に行われていることを確認できます。

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

{
    "ShardIterator":
    "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+OIK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
}

aws kinesis get-records \
      --limit 10 \
      --shard-iterator
      "AAAAAAAAAAFGU/
kLvNggvndHq2UIFOw5PZc6F01s3e3afsSscRM70JSbjIefg2ub07nk1y6CDxYR1UoGHJNP4m4NFUetzfL+wev
+e2P4djJg4L9wmXKvQYoE+rMUiFq+p4Cn3IgvqOb5dRA0yybNdRcdzvnC35KQANoHzzahKdRGb9v4scv+3vaq+f
+OIK8zM5My8ID+g6rMo7UKWeI4+IWiKEXAMPLE"
```

Note

場合によっては、Kinesis がデータを返し始めるまで、get-records コマンドを数回再実行する必要があります。

一連の Kinesis レコードを含んでいるレスポンスが表示されます。Kinesis レコードのデータ属性は Base64 でエンコードされており、gzip 形式で圧縮されています。raw データは、コマンドラインから次の UNIX コマンドを使用して調べることができます。

```
echo -n "<Content of Data>" | base64 -d | zcat
```

Base64 でデコードおよび圧縮されたデータは、次の構造を使用して JSON としてフォーマットされます。

```
{
    "owner": "111111111111",
    "logGroup": "CloudTrail",
    "logStream": "111111111111_CloudTrail_us-east-1",
    "subscriptionFilters": [
        "RecipientStream"
    ],
    "messageType": "DATA_MESSAGE",
    "logEvents": [
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root\"}"
        },
        {
            "id": "31953106606966983378809025079804211143289615424 29EXAMPLE",
            "timestamp": 1432826855000,
            "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root\"}"
        }
    ]
}
```

このデータ構造のキー要素は以下のとおりです。

owner

発行元ログデータの AWS アカウント ID。

logGroup

発行元ログデータのロググループ名。

logStream

発行元ログデータのログストリーム名。

subscriptionFilters

発行元ログデータと一致したサブスクリプションフィルタ名のリスト。

messageType

データメッセージは、"DATA_MESSAGE" 型を使用します。CloudWatch Logs は、主に送信先が到達可能であるかどうかをチェックするために、"CONTROL_MESSAGE" 型の Kinesis レコードを発行することがあります。

logEvents

ログイベントレコードの配列として表される実際のログデータ。ID プロパティは、各ログイベントの一意の識別子です。

実行時に送信先のメンバーシップを変更する

所有する送信先のユーザーのメンバーシップを追加または削除する必要がある場合があります。新しいアクセスポリシーが関連付けられている送信先に対して PutDestinationPolicy アクションを使用できま

す。次の例では、先ほど追加したアカウント 111111111111 がログデータの送信を停止し、アカウント 222222222222 が有効になります。

1. 現在 testDestination という送信先に関連付けられているポリシーをフェッチし、AccessPolicy を書き留めておきます。

```
aws logs describe-destinations \
    --destination-name-prefix "testDestination"

{
 "Destinations": [
   {
     "DestinationName": "testDestination",
     "RoleArn": "arn:aws:iam::123456789012:role/CWLtoKinesisRole",
     "DestinationArn": "arn:aws:logs:region:123456789012:destination:testDestination",
     "TargetArn": "arn:aws:kinesis:region:123456789012:stream/RecipientStream",
     "AccessPolicy": "{\"Version\": \"2012-10-17\", \"Statement\":
 [{\"Sid\": \"\", \"Effect\": \"Allow\", \"Principal\": {\"AWS\":
 \"111111111111\"}, \"Action\": \"logs:PutSubscriptionFilter\", \"Resource\":
 \"arn:aws:logs:region:123456789012:destination:testDestination\"}] }"
   }
 ]
}
```

2. アカウント 111111111111 が停止したこととアカウント 222222222222 が有効になったことを反映させるためにポリシーを更新します。このポリシーを ~/AccessPolicy.json ファイルに配置します。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "222222222222"
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:region:123456789012:destination:testDestination"
    }
  ]
}
```

3. PutDestinationPolicy を呼び出して、NewAccessPolicy.json ファイルで定義されているポリシーを送信先に関連付けます。

```
aws logs put-destination-policy \
--destination-name "testDestination" \
--access-policy file://~/NewAccessPolicy.json
```

これにより、最終的には、アカウント ID 111111111111 からのログイベントが無効になります。アカウント ID 222222222222 の所有者が PutSubscriptionFilter を使用してサブスクリプションフィルタを作成すると、すぐに 222222222222 からのログイベントが送信先に送信されるようになります。

Amazon S3 へのログの送信

一部の AWS サービスは、ログを直接 Amazon S3 に発行できます。これにより、ログの主な要件が Amazon S3 のストレージである場合、追加のインフラストラクチャを設定しなくても、サービスで簡単にログを作成し、直接 Amazon S3 に配信することができます。

Amazon S3 に発行されたログは、指定する既存のバケットに発行されます。指定したバケットで、5 分おきに 1 つ以上のログが作成されます。

ログが S3 バケットに直接発行された場合でも、CloudWatch Logs の変更は適用されます。詳細については、Amazon CloudWatch 料金表の「ログを S3 に配信する」を参照してください。

次のログは、Amazon S3 に直接発行できます。

- VPC フローログ。詳細については、Amazon VPC ユーザーガイド の「Amazon S3 へのフローログの発行」を参照してください。
- AWS Global Accelerator のフローログ。詳細については、AWS Global Accelerator 開発者ガイドの「Amazon S3 へのフローログの発行」を参照してください。

Amazon S3 へのログデータのエクスポート

ロググループのログデータを Amazon S3 バケットにエクスポートし、このデータをカスタム処理や分析で使用したり、別のシステムの読み込んだりすることができます。

エクスポート処理を開始するには、エクスポートされたログデータを保存する S3 バケットを作成する必要があります。エクスポートしたファイルは Amazon S3 バケットに保存し、エクスポートしたファイルを自動的にアーカイブまたは削除するための Amazon S3 ライフサイクルルールを定義することができます。

> Note
>
> 2019 年 2 月 15 日以降、Amazon S3 にエクスポートする機能を使用するには、送信先のバケットへの `s3:PutObject` アクセス権が発信者に必要です。

複数のロググループからのログや、複数の時間範囲のログを同じ S3 バケットにエクスポートすることができます。エクスポートタスクごとにログデータを分割するためには、エクスポートされたすべてのオブジェクトに対する Amazon S3 キープレフィックスとして使用するプレフィックスを指定する必要があります。

ログデータは、エクスポートできるようになるまで最大 12時間かかる場合があります。ほぼリアルタイムのログデータ分析については、「CloudWatch Logs Insights で ログデータを分析する (p. 34)」または「サブスクリプションを使用したログデータのリアルタイム処理 (p. 74)」を参照してください。

コンテンツ

概念

作業を開始する前に、エクスポートに関する以下の概念を理解してください。

log group name

エクスポートタスクに関連付けられるロググループの名前。このロググループのログデータが、指定された Amazon S3 バケットにエクスポートされます。

from (timestamp)

1970 年 1 月 1 日 00:00:00 UTC からの経過ミリ秒数で表されるタイムスタンプであり、指定は必須です。この時より後に取り込まれたロググループのすべてのログイベントがエクスポートされます。

to (timestamp)

1970 年 1 月 1 日 00:00:00 UTC からの経過ミリ秒数で表されるタイムスタンプであり、指定は必須です。この時刻より前に取り込まれたロググループのすべてのログイベントがエクスポートされます。

送信先バケット

エクスポートタスクに関連付けられる Amazon S3 バケットの名前。このバケットは、指定されたロググループのログデータをエクスポートするために使用されます。

送信先プレフィックス

エクスポートされたすべてのオブジェクトの S3 キープレフィックスとして使用される、オプションの属性。バケット内でフォルダのような構成を作成するのに役立ちます。

コンソールを使用したログデータの Amazon S3 へのエクスポート

次の例では、Amazon CloudWatch コンソールを使用して、すべてのデータを `my-log-group` という名前の Amazon CloudWatch Logs ロググループから `my-exported-logs` という名前の Amazon S3 バケットにエクスポートします。

ステップ 1: Amazon S3 バケットを作成する

CloudWatch Logs 専用に作成したバケットを使用することをお勧めします。ただし、既存のバケットを使用する場合は、ステップ 2 に進むことができます。

Note

Amazon S3 バケットは、エクスポートするログデータと同じリージョンに存在している必要があります。CloudWatch Logs では、別のリージョン内の Amazon S3 バケットへのデータのエクスポートをサポートしていません。

Amazon S3 バケットを作成するには

1. https://console.aws.amazon.com/s3/ にある Amazon S3 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、CloudWatch Logs があるリージョンを選択します。
3. [Create Bucket] を選択します。
4. [バケット名] にバケットの名前を入力します。
5. [リージョン] で、CloudWatch Logs データが存在するリージョンを選択します。
6. [Create] を選択します。

ステップ 2: Amazon S3 および CloudWatch Logs へのフルアクセスを持つ IAM ユーザーを作成する

以下のステップで、必要なアクセス許可を持つ IAM ユーザーを作成します。

必要な IAM ユーザーを作成するには

1. https://console.aws.amazon.com/iam/ にある IAM コンソールを開きます。
2. [ユーザー]、[ユーザーの追加] を選択します。
3. ユーザー名 (例: *`CWLExportUser`*) を入力します。
4. [Programmatic access] と [AWS マネジメントコンソールへのアクセス] の両方を選択します。
5. [Autogenerated password] または [Custom password] を選択します。
6. [Next: Permissions] を選択します。

7. [Attach existing policies directly (既存のポリシーを直接アタッチ)] を選択して、[AmazonS3FullAccess] および [CloudWatchLogsFullAccess] ポリシーをユーザーにアタッチします。ポリシーを検索するには、検索ボックスを使用します。
8. [Next: Tags]、[Next: Review]、[Create user] の順に選択します。

ステップ 3: Amazon S3 バケットにアクセス許可を設定する

デフォルトでは、すべての Amazon S3 バケットとオブジェクトはプライベートです。バケットを作成した AWS アカウント (リソース所有者) のみが、バケットとそれに含まれるオブジェクトにアクセスできます。ただし、リソース所有者は、アクセスポリシーを記述することで他のリソースおよびユーザーにアクセス権限を付与することができます。

ポリシーを設定する場合は、ランダムに生成された文字列をバケットのプレフィックスとして含めることをお勧めします。これにより、意図したログストリームのみがバケットにエクスポートされます。

Amazon S3 バケットに対する権限を設定するには

1. Amazon S3 コンソールで、ステップ 1 で作成したバケットを選択します。
2. [Permissions (アクセス許可)]、[Add bucket policy (バケットポリシーの追加)] の順に選択します。
3. [バケットポリシーエディター] で、以下のいずれかのポリシーを追加します。my-exported-logs を S3 バケットの名前、random-string をランダムに生成された文字列に変更します。[プリンシパル] に正しいリージョンエンドポイントを指定してください。

   - バケットが自分のアカウントにある場合は、次のポリシーを追加します。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs",
          "Principal": { "Service": "logs.us-west-2.amazonaws.com" }
      },
      {
          "Action": "s3:PutObject" ,
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs/random-string/*",
          "Condition": { "StringEquals": { "s3:x-amz-acl": "bucket-owner-full-
control" } },
          "Principal": { "Service": "logs.us-west-2.amazonaws.com" }
      }
    ]
}
```

   - バケットが別のアカウントにある場合は、次のポリシーを使用します。その際、前のステップで作成した IAM ユーザーを使用してステートメントを追加します。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs",
          "Principal": { "Service": "logs.us-west-2.amazonaws.com" }
      },
```

```
      {
          "Action": "s3:PutObject" ,
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs/random-string/*",
          "Condition": { "StringEquals": { "s3:x-amz-acl": "bucket-owner-full-
control" } },
          "Principal": { "Service": "logs.us-west-2.amazonaws.com" }
      },
      {
          "Action": "s3:PutObject" ,
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs/random-string/*",
          "Condition": { "StringEquals": { "s3:x-amz-acl": "bucket-owner-full-
control" } },
          "Principal": { "AWS": "arn:aws:iam::SendingAccountID:user/CWLExportUser" }
      }
    ]
}
```

4. [Save] を選択して、バケットに対するアクセスポリシーとして追加したポリシーを設定します。このポリシーにより、CloudWatch Logs がログデータを Amazon S3 バケットにエクスポートできるようになります。バケット所有者には、エクスポートされたすべてのオブジェクトに対する完全なアクセス権限があります。

   Warning

   バケットにアタッチされているポリシーがすでに 1 つ以上ある場合は、そのポリシーに CloudWatch Logs アクセスのステートメントを追加します。バケットにアクセスするユーザーに適したアクセス許可であることを確認するために、アクセス許可の結果セットを評価することをお勧めします。

ステップ 4: エクスポートタスクを作成する

このステップでは、ロググループからログをエクスポートするためのエクスポートタスクを作成します。

CloudWatch コンソールを使用して Amazon S3 にデータをエクスポートするには

1. [ステップ 2: Amazon S3 および CloudWatch Logs へのフルアクセスを持つ IAM ユーザーを作成する] で作成した IAM ユーザーとしてサインインします。
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 [Logs] を選択します。
4. [ロググループ] 画面でロググループの横のボタンを選択し、[アクション]、[データを Amazon S3 にエクスポートする] の順に選択します。
5. [データを Amazon S3 にエクスポートする] 画面の [エクスポートするデータを定義する] で、[開始日] と [終了日] を使用してデータをエクスポートする時間の範囲を設定します。
6. ロググループに複数のログストリームがある場合は、特定のストリームのロググループデータを制限するログストリームプレフィックスを指定できます。[Advanced (詳細設定)] を選択して、[ストリームプレフィックス] にログストリームプレフィックスを入力します。
7. [Choose S3 bucket (S3 バケットの選択)] で、Amazon S3 バケットに関連付けるアカウントを選択します。
8. [S3 バケット名] で、Amazon S3 バケットを選択します。
9. [Advanced (詳細設定)] を選択し、[S3 バケットプレフィックス] にバケットポリシーで指定した、ランダムに生成された文字列を入力します。
10. [データのエクスポート] を選択して、ログデータを Amazon S3 にエクスポートします。
11. Amazon S3 にエクスポートしたログデータのステータスを表示するには、[アクション]、[Amazon S3 へのすべてのエクスポートを表示] を選択します。

AWS CLI を使用した Amazon S3 へのログデータのエクスポート

次の例では、エクスポートタスクを使用して、すべてのデータを my-log-group という名前の CloudWatch Logs ロググループから my-exported-logs という名前の Amazon S3 バケットにエクスポートします。この例では、「my-log-group」というロググループを作成済みであることを前提としています。

ステップ 1: Amazon S3 バケットを作成する

CloudWatch Logs 専用に作成したバケットを使用することをお勧めします。ただし、既存のバケットを使用する場合は、ステップ 2 に進むことができます。

> Note
>
> Amazon S3 バケットは、エクスポートするログデータと同じリージョンに存在している必要があります。CloudWatch Logs では、別のリージョン内の Amazon S3 バケットへのデータのエクスポートをサポートしていません。

AWS CLI を使用して Amazon S3 バケットを作成するには

コマンドプロンプトで、次の create-bucket コマンドを実行します。ここで、LocationConstraint はログデータをエクスポートするリージョンです。

```
aws s3api create-bucket --bucket my-exported-logs --create-bucket-configuration
 LocationConstraint=us-east-2
```

出力例を次に示します。

```
{
    "Location": "/my-exported-logs"
}
```

ステップ 2: Amazon S3 および CloudWatch Logs へのフルアクセスを持つ IAM ユーザーを作成する

以下のステップで、必要なアクセス許可を持つ IAM ユーザーを作成します。

ユーザーを作成し、アクセス許可を割り当てるには

1. IAM ユーザーを作成するには、次のコマンドを入力します。

   ```
   aws iam create-user --user-name CWLExportUser
   ```

2. IAM 管理ポリシーを、先ほど作成した IAM ユーザーにアタッチします。

   ```
   export S3POLICYARN=$(aws iam list-policies --query 'Policies[?
   PolicyName==`AmazonS3FullAccess`].{ARN:Arn}' --output text)
   ```

   ```
   export CWLPOLICYARN=$( aws iam list-policies --query 'Policies[?
   PolicyName==`CloudWatchLogsFullAccess`].{ARN:Arn}' --output text)
   ```

   ```
   aws iam attach-user-policy --user-name CWLExportUser --policy-arn $S3POLICYARN
   ```

```
aws iam attach-user-policy --user-name CWLExportUser --policy-arn $CWLPOLICYARN
```

3. 2 つの管理ポリシーがアタッチされていることを確認します。

```
aws iam list-attached-user-policies --user-name CWLExportUser
```

4. IAM ユーザー *CWLExportUser* の IAM 認証情報を含むように AWS CLI を設定します。詳細については、「AWS CLI の設定」を参照してください。

ステップ 3: Amazon S3 バケットにアクセス許可を設定する

デフォルトでは、すべての Amazon S3 バケットとオブジェクトはプライベートです。バケットを作成したアカウント (リソース所有者) のみが、バケットとそれに含まれるオブジェクトにアクセスできます。ただし、リソース所有者は、アクセスポリシーを記述することで他のリソースおよびユーザーにアクセス権限を付与することができます。

Amazon S3 バケットに対する権限を設定するには

1. `policy.json` という名前のファイルを作成し、次のアクセスポリシーを追加します。このとき、`Resource` を S3 バケットの名前に変更し、`Principal` をログデータのエクスポート先のリージョンのエンドポイントに変更します。テキストエディタを使用してこのポリシーファイルを作成します。IAM コンソールは使用しないでください。

   - バケットが自分のアカウントにある場合は、次のポリシーを使用します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "s3:GetBucketAcl",
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::my-exported-logs",
            "Principal": { "Service": "logs.us-west-2.amazonaws.com" }
        },
        {
            "Action": "s3:PutObject" ,
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::my-exported-logs/*",
            "Condition": { "StringEquals": { "s3:x-amz-acl": "bucket-owner-full-
control" } },
            "Principal": { "Service": "logs.us-west-2.amazonaws.com" }
        }
    ]
}
```

   - バケットが別のアカウントにある場合は、次のポリシーを使用します。その際、前のステップで作成した IAM ユーザーを使用してステートメントを追加します。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs",
          "Principal": { "Service": "logs.us-west-2.amazonaws.com" }
```

```
      },
      {
          "Action": "s3:PutObject" ,
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs/random-string/*",
          "Condition": { "StringEquals": { "s3:x-amz-acl": "bucket-owner-full-
control" } },
          "Principal": { "Service": "logs.us-west-2.amazonaws.com" }
      },
      {
          "Action": "s3:PutObject" ,
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs/random-string/*",
          "Condition": { "StringEquals": { "s3:x-amz-acl": "bucket-owner-full-
control" } },
          "Principal": { "AWS": "arn:aws:iam::SendingAccountID:user/CWLExportUser" }
      }
    ]
}
```

- バケットが別のアカウントにあり、IAM ユーザーではなく IAM ロールを使用している場合は、次のポリシーを使用します。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs",
          "Principal": { "Service": "logs.us-west-2.amazonaws.com" }
      },
      {
          "Action": "s3:PutObject" ,
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs/random-string/*",
          "Condition": { "StringEquals": { "s3:x-amz-acl": "bucket-owner-full-
control" } },
          "Principal": { "Service": "logs.us-west-2.amazonaws.com" }
      },
      {
          "Action": "s3:PutObject" ,
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs/random-string/*",
          "Condition": { "StringEquals": { "s3:x-amz-acl": "bucket-owner-full-
control" } },
          "Principal": { "AWS": "arn:aws:iam::SendingAccountID:role/CWLExportUser" }
      }
    ]
}
```

2. put-bucket-policy コマンドを使用して、バケットでアクセスポリシーとして先ほど追加したポリシーを設定します。このポリシーにより、CloudWatch Logs がログデータを Amazon S3 バケットにエクスポートできるようになります。バケット所有者には、エクスポートされたすべてのオブジェクトに対する完全なアクセス権限があります。

```
aws s3api put-bucket-policy --bucket my-exported-logs --policy file://policy.json
```

Warning

バケットにアタッチされているポリシーがすでに 1 つ以上ある場合は、そのポリシーに CloudWatch Logs アクセスのステートメントを追加します。バケットにアクセスするユー

ザーに適したアクセス許可であることを確認するために、アクセス許可の結果セットを評価することをお勧めします。

ステップ 4: エクスポートタスクを作成する

ロググループからログをエクスポートするためのエクスポートタスクを作成すると、エクスポートするデータのサイズに応じて、エクスポートタスクに数秒から数時間かかる可能性があります。

AWS CLI を使用してエクスポートタスクを作成するには

コマンドプロンプトで、次の create-export-task コマンドを使用してエクスポートタスクを作成します。

```
aws logs create-export-task --profile CWLExportUser --task-name "my-log-group-09-10-2015"
 --log-group-name "my-log-group" --from 1441490400000 --to 1441494000000 --destination "my-exported-logs" --destination-prefix "export-task-output"
```

出力例を次に示します。

```
{
    "taskId": "cda45419-90ea-4db5-9833-aade86253e66"
}
```

ステップ 5: エクスポートタスクを記述する

エクスポートタスクを作成すると、タスクの現在のステータスを取得できます。

AWS CLI を使用してエクスポートタスクを記述するには

コマンドプロンプトで、次の describe-export-tasks コマンドを使用します。

```
aws logs --profile CWLExportUser describe-export-tasks --task-id "cda45419-90ea-4db5-9833-aade86253e66"
```

出力例を次に示します。

```
{
   "exportTasks": [
   {
      "destination": "my-exported-logs",
      "destinationPrefix": "export-task-output",
      "executionInfo": {
         "creationTime": 1441495400000
      },
      "from": 1441490400000,
      "logGroupName": "my-log-group",
      "status": {
         "code": "RUNNING",
         "message": "Started Successfully"
      },
      "taskId": "cda45419-90ea-4db5-9833-aade86253e66",
      "taskName": "my-log-group-09-10-2015",
      "tTo": 1441494000000
   }]
}
```

describe-export-tasks コマンドを使用する方法は 3 通りあります。

- フィルタなし: すべてのエクスポートタスクが、作成順とは逆の順序でリストされます。
- タスク ID によるフィルタ: 指定された ID のエクスポートタスクが存在する場合に、そのエクスポートタスクのみがリストされます。
- タスクステータスによるフィルタ: 指定されたステータスのエクスポートタスクがリストされます。

たとえば、次のコマンドを使用して FAILED ステータスによってフィルタリングします。

```
aws logs --profile CWLExportUser describe-export-tasks --status-code "FAILED"
```

出力例を次に示します。

```
{
   "exportTasks": [
   {
      "destination": "my-exported-logs",
      "destinationPrefix": "export-task-output",
      "executionInfo": {
         "completionTime": 1441498600000
         "creationTime": 1441495400000
      },
      "from": 1441490400000,
      "logGroupName": "my-log-group",
      "status": {
         "code": "FAILED",
         "message": "FAILED"
      },
      "taskId": "cda45419-90ea-4db5-9833-aade86253e66",
      "taskName": "my-log-group-09-10-2015",
      "to": 1441494000000
   }]
}
```

ステップ 6: エクスポートタスクをキャンセルする

エクスポートタスクが PENDING または RUNNING の状態の場合、そのタスクをキャンセルできます。

AWS CLI を使用してエクスポートタスクをキャンセルするには

コマンドプロンプトで、次の cancel-export-task コマンドを使用します。

```
aws logs --profile CWLExportUser cancel-export-task --task-id "cda45419-90ea-4db5-9833-
aade86253e66"
```

describe-export-tasks コマンドを使用して、タスクが正常にキャンセルされたことを確認できる点に注意してください。

CloudWatch Logs データの Amazon Elasticsearch Service へのストリーミング

CloudWatch Logs サブスクリプションを通して、ほぼリアルタイムでAmazon Elasticsearch Service (Amazon ES) クラスターで受け取るCloudWatch Logs ロググループをストリームデータに設定することができます。詳細については、「サブスクリプションを使用したログデータのリアルタイム処理 (p. 74)」を参照してください。

Note

大量の CloudWatch Logs データを Amazon ES にストリーミングすると、利用料金が高額になる可能性があります。請求情報とコスト管理コンソールで予算を作成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。

前提条件

開始する前に、Amazon ES ドメインを作成します。Amazon ES ドメインではパブリックアクセスまたは VPC アクセスが可能ですが、その場合はドメインの作成後にアクセスのタイプを変更することはできません。後で Amazon ES ドメイン設定を確認し、クラスターが処理するデータの量に基づいてクラスター設定を変更することができます。

Amazon ES の詳細については、「Amazon Elasticsearch Service 開発者ガイド」を参照してください。

Amazon ESドメインの作成方法

コマンドプロンプトで、次の create-elasticsearch-domain コマンドを使用します。

```
aws es create-elasticsearch-domain --domain-name my-domain
```

ログ グループを Amazon ES にサブスクライブする

CloudWatch コンソールを使用すると、ロググループを Amazon ES にサブスクライブできます。

ログ グループを Amazon ES にサブスクライブする方法

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. ロググループを選択してサブスクライブします。
4. [アクション]、[Amazon Elasticsearch Service にストリーミング] を選択します。
5. [Amazon Elasticsearch Service へのストリーミングを開始する] 画面の [Amazon ES クラスター] ドロップダウンリストで、前のステップで作成したクラスターを選択してから、[次へ] を選択します。
6. [Lambda 関数] の [Lambda IAM 実行ロール] で、Amazon ES の呼び出し時に Lambda が使用する IAM ロールを選択してから、[次へ] を選択します。

選択した IAM ロールは、これらの要件を満たす必要があります。

- 信頼関係に `lambda.amazonaws.com` が含まれている必要があります。
- 以下のポリシーが含まれている必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "es:*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:es:region:account-id:domain/target-domain-name/*"
        }
    ]
}
```

- ターゲットの Amazon ES ドメインで VPC アクセスを使用する場合、ロールには AWSLambdaVPCAccessExecutionRole ポリシーがアタッチされている必要があります。Amazon が管理するこのポリシーにより、お客様の VPC へのアクセスが Lambda に許可され、Lambda は VPC の Amazon ES エンドポイントに書き込むことができます。

7. [Configure Log Format and Filters] 画面の [Log Format] で、ログ形式を選択します。
8. [Subscription Filter Pattern] に、ログイベントで検索する用語やパターンを入力します。これにより、関心のあるデータだけが Amazon ES クラスターに送信されるようになります。詳細については、「ログデータの検索およびフィルタリング (p. 56)」を参照してください。
9. (オプション) [Select Log Data to Test] を開き、ログストリームを選択し、[Test Pattern] をクリックして、期待通りの結果が出ることを確認します。
10. [次へ] を選択し、[Amazon Elasticsearch Service へのストリーミングを確認し開始する] 画面で [ストリーミングの開始] を選択します。

Amazon CloudWatch Logs に対する認証とアクセスコントロール

Amazon CloudWatch Logs へのアクセスには、AWS によってリクエストの認証に使用される認証情報が必要です。これらの認証情報には、クラウドリソースに関する CloudWatch Logs データの取得などの AWS リソースへのアクセス権限が必要です。次のセクションでは、AWS Identity and Access Management (IAM) と CloudWatch Logs を使用して、リソースにアクセスできるユーザーを制御することで、リソースをセキュリティで保護する方法について詳しく説明します。

認証

AWS には、次のタイプのアイデンティティでアクセスできます。

- AWS アカウントのルートユーザー – AWS にサインアップするときは、AWS アカウントに関連付けられた E メールアドレスとパスワードを指定します。これらは ルート認証情報であり、これらの情報を使用すると、すべての AWS リソースへの完全なアクセスが可能になります。

  Important

  セキュリティ上の理由から、AWS アカウントへの完全なアクセス権限を持つ管理者ユーザー (IAM ユーザー) を作成するためにのみ、ルート認証情報を使用することをお勧めします。その後、この管理者ユーザーを使用して、制限されたアクセス権限を持つ他の IAM ユーザーとロールを作成できます。詳細については、『IAM ユーザーガイド』の「IAM のベストプラクティス」および「管理者のユーザーおよびグループの作成」を参照してください。
- IAM ユーザー – IAM ユーザーは、特定のカスタム権限 (たとえば、CloudWatch Logs でメトリクスを表示するアクセス権限) を持つ AWS アカウント内のアイデンティティです。IAM のユーザー名とパスワードを使用して、AWS マネジメントコンソール、AWS ディスカッションフォーラム、AWS Support Center などのセキュリティ保護された AWS ウェブページにサインインできます。

  ユーザー名とパスワードに加えて、各ユーザーのアクセスキーを生成することもできます。いくつかの SDK の 1 つ、または AWS Command Line Interface (AWS CLI) を使ってプログラムで AWS サービスにアクセスするときに、これらのキーを使用します。SDK と CLI ツールでは、アクセスキーを使用してリクエストが暗号で署名されます。AWS ツールを使用しない場合は、リクエストを自分で署名する必要があります。CloudWatch Logs supports では、署名バージョン 4 がサポートされています。これは、インバウンド API リクエストを認証するためのプロトコルです。リクエストの認証の詳細については、『AWS General Reference』の「署名バージョン 4 の署名プロセス」を参照してください。

- IAMロール – IAM ロールは、特定のアクセス権限を持ち、アカウントで作成できるもう 1 つの IAM ID です。これは IAM ユーザーに似ていますが、特定のユーザーに関連付けられていません。IAM ロールでは、AWS のサービスおよびリソースにアクセスできる一時的なアクセスキーを取得することができます。IAM ロールと一時的な認証情報は、次の状況で役立ちます。

  - フェデレーティッドユーザーアクセス – IAM ユーザーを作成するのではなく、AWS Directory Service、エンタープライズユーザーディレクトリ、またはウェブアイデンティティプロバイダーの既

存のユーザーアイデンティティを使用することもできます。このようなユーザーはフェデレーティッドユーザーと呼ばれます。AWS では、アイデンティティプロバイダーを通じてアクセスがリクエストされたときに、フェデレーティッドユーザーにロールを割り当てます。フェデレーティッドユーザーの詳細については、『IAM ユーザーガイド』の「フェデレーティッドユーザーとロール」を参照してください。

- クロスアカウントアクセス – アカウントで IAM ロールを使って、お客様のアカウントのリソースへのアクセス権を別の AWS アカウントに付与できます。例については、IAM ユーザーガイド の「チュートリアル: AWS アカウント間の IAM ロールを使用したアクセスの委任」を参照してください。

- AWS サービスアカウント – アカウントで IAM ロールを使って、お客様のアカウントのリソースにアクセスする AWS サービスのアクセス権限を付与できます。たとえば、Amazon Redshift がお客様に代わって Amazon S3 バケットにアクセスし、バケットに保存されたデータを Amazon Redshift クラスターにロードすることを許可するロールを作成できます。詳細については、IAM ユーザーガイド の「AWS のサービスにアクセス権限を委任するロールの作成」を参照してください。

- Amazon EC2 で実行されるアプリケーション – インスタンスで実行し、AWS API リクエストを作成するアプリケーションで使用されるアクセスキーを EC2 インスタンス内に保存する代わりに、IAM ロールを使用して、これらのアプリケーション用の一時認証情報を管理できます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成できます。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時認証情報を取得することができます。詳細については、『IAM ユーザーガイド』の「Amazon EC2 上のアプリケーションに対するロールの使用」を参照してください。

アクセスコントロール

有効な認証情報があればリクエストを認証できますが、アクセス許可が付与されている場合を除き、CloudWatch Logs リソースの作成やアクセスはできません。たとえば、ログストリーム、ロググループなどを作成する許可が必要となります。

以下のセクションでは、CloudWatch Logs のアクセス許可を管理する方法について説明します。最初に概要のセクションを読むことをお勧めします。

- CloudWatch Logs リソースへのアクセス許可の管理の概要 (p. 106)
- CloudWatch Logs でアイデンティティベースのポリシー (IAM ポリシー) を使用する (p. 110)
- CloudWatch Logs の権限リファレンス (p. 114)

CloudWatch Logs リソースへのアクセス許可の管理の概要

すべての AWS リソースは AWS アカウントによって所有され、リソースを作成またはアクセスするためのアクセス権限は、アクセス権限ポリシーによって管理されます。アカウント管理者は、アクセス権限ポリシーを IAM ID (ユーザー、グループ、ロール) にアタッチでき、一部のサービス (AWS Lambda など) では、アクセス権限ポリシーをリソースにアタッチすることをサポートしています。

Note

アカウント管理者 (または管理者 IAM ユーザー) は、管理者権限を持つユーザーです。詳細については、「IAM のベストプラクティス」 (IAM ユーザーガイド) を参照してください。

アクセス権限を付与する場合、アクセス権限を取得するユーザー、取得するアクセス権限の対象となるリソース、およびそれらのリソースに対して許可される特定のアクションを決定します。

トピック

CloudWatch Logs リソースおよびオペレーション

CloudWatch Logs では、プライマリリソースはロググループ、ログストリーム、送信先です。CloudWatch Logs はサブリソース (プライマリリソースと使用する他のリソース) をサポートしていません。

これらのリソースとサブリソースには、次の表に示すとおり、一意の Amazon リソースネーム (ARN) が関連付けられています。

リソースタイプ	ARN 形式
ロググループ	arn:aws:logs:*region*:*account-id*:log-group:*log_group_name*
ログストリーム	arn:aws:logs:*region*:*account-id*:log-group:*log_group_name*:log-stream:*log-stream-name*
送信先	arn:aws:logs:*region*:*account-id*:destination:*destination_name*

ARN の詳細については、IAM ユーザーガイド の「ARN」を参照してください。CloudWatch Logs ARN 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「Amazon リソースネーム (ARN) と AWS サービスの名前空間」を参照してくださいCloudWatch Logs を対象とするポリシーの例については、「CloudWatch Logs でアイデンティティベースのポリシー (IAM ポリシー) を使用する (p. 110)」を参照してください。

CloudWatch Logs には、CloudWatch Logs リソースを操作するための一連のオペレーションが用意されています。使用可能なオペレーションのリストについては、「CloudWatch Logs の権限リファレンス (p. 114)」を参照してください。

リソース所有権について

AWS アカウントは、誰がリソースを作成したかにかかわらず、アカウントで作成されたリソースを所有します。具体的には、リソース所有者は、リソースの作成リクエストを認証するプリンシパルエンティティ (ルートアカウント、IAM ユーザー、または IAM ロール) の AWS アカウントです。以下の例では、このしくみを示しています。

- AWS アカウントの root アカウントの認証情報を使用してロググループを作成する場合、AWS アカウントは CloudWatch Logs リソースの所有者です。
- AWS アカウントに IAM ユーザーを作成し、そのユーザーに CloudWatch Logs リソースを作成するアクセス権限を付与する場合、そのユーザーは CloudWatch Logs リソースを作成できます。ただし、ユーザーが属する AWS アカウントは CloudWatch Logs リソースを所有しているとします。

- CloudWatch Logs リソースを作成するためのアクセス許可を持つ AWS アカウントに IAM ロールを作成する場合は、ロールを引き受けることのできるいずれのユーザーも CloudWatch Logs リソースを作成できます。ロールが属する AWS アカウントは CloudWatch Logs リソースを所有しているとします。

リソースへのアクセスの管理

アクセスポリシーでは、誰が何にアクセスできるかを記述します。以下のセクションで、アクセス権限のポリシーを作成するために使用可能なオプションについて説明します。

> Note
>
> このセクションでは、CloudWatch Logs のコンテキストでの IAM の使用について説明します。これは、IAM サービスに関する詳細情報を取得できません。完全な IAM ドキュメントについては、「IAM とは?」 (IAM ユーザーガイド) を参照してください。IAM ポリシー構文の詳細および説明については、IAM ユーザーガイド の「AWS IAM ポリシーリファレンス」を参照してください。

IAM アイデンティティにアタッチされたポリシーはアイデンティティベースのポリシー (IAM ポリシー) と呼ばれ、リソースにアタッチされたポリシーはリソースベースのポリシーと呼ばれます。CloudWatch Logs では、アイデンティティベースのポリシーと、送信先のリソースベースのポリシー (クロスアカウントサブスクリプションを有効にするために使用されます) がサポートされています。詳細については、「クロスアカウントのログデータをサブスクリプションと共有する (p. 86)」を参照してください。

トピック

アイデンティティベースのポリシー (IAM ポリシー)

ポリシーを IAM アイデンティティにアタッチできます。たとえば、次の操作を実行できます。

- アカウントのユーザーまたはグループにアクセス権限ポリシーをアタッチする – CloudWatch Logs コンソールのログを表示するアクセス権限を付与するために、ユーザーが所属するユーザーまたはグループにアクセス許可のポリシーをアタッチできます。
- アクセス権限ポリシーをロールにアタッチする (クロスアカウントのアクセス権限を付与する) – アイデンティティベースのアクセス権限ポリシーを IAM ロールにアタッチして、クロスアカウントのアクセス権限を付与することができます。たとえば、アカウント A の管理者は、次のように他のまたは AWS にクロスアカウントのアクセス権限を別の AWS アカウント (アカウント B) または AWS サービスに付与するロールを作成することができます。
  1. アカウント A の管理者は、IAM ロールを作成して、アカウント A のリソースに権限を付与するロールに権限ポリシーをアタッチします。
  2. アカウント A の管理者は、アカウント B をそのロールを引き受けるプリンシパルとして識別するロールに、信頼ポリシーをアタッチします。
  3. アカウント B の管理者は、アカウント B のユーザーにロールを引き受ける権限を委任できるようになります。これにより、アカウント B のユーザーにアカウント A のリソースの作成とアクセスが許可されます。AWS サービスのアクセス権限を付与してロールを引き受けさせたい場合は、信頼ポリシー内のプリンシパルも、AWS サービスのプリンシパルとなることができます。

  IAM を使用したアクセス許可の委任の詳細については、『IAM ユーザーガイド』の「アクセス管理」を参照してください。

us-east-1 のすべてのリソースの`logs:PutLogEvents`、`logs:CreateLogGroup`、`logs:CreateLogStream` アクションのアクセス権限を付与するポリシーの例を次に示します。ロググループの場合、一部の API アクションについて、CloudWatch Logs はリソー

ス ARN (リソースレベルのアクセス許可とも呼ばれる) を使用した特定のリソースの識別をサポートします。すべてのロググループを含める場合、ワイルドカード文字 (*) を指定する必要があります。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"",
         "Effect":"Allow",
         "Principal":{
            "AWS":"234567890123"
         },
         "Action":[
            "logs:PutLogEvents",
            "logs:CreateLogGroup",
            "logs:CreateLogStream"
         ],
         "Resource":"arn:aws:logs:us-east-1:*:*"
      }
   ]
}
```

CloudWatch Logs でアイデンティティベースのポリシーを使用する場合の詳細については、「CloudWatch Logs でアイデンティティベースのポリシー (IAM ポリシー) を使用する (p. 110)」を参照してください。ユーザー、グループ、ロール、アクセス権限の詳細については、IAM ユーザーガイド の「アイデンティティ (ユーザー、グループ、ロール)」を参照してください。

リソースベースのポリシー

CloudWatch Logs は、クロスアカウントのサブスクリプションを有効にするために使用する、送信先のリソースベースのポリシーをサポートします。詳細については、「送信先を作成する (p. 87)」を参照してください。PutDestination API を使用して送信先を作成でき、PutDestination API を使用して送信先にリソースポリシーを追加できます。次の例では、アカウント ID 111122223333 の他の AWS アカウントが、送信先 `arn:aws:logs:us-east-1:123456789012:destination:testDestination` にロググループをサブスクライブできるようにします。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "111122223333"
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:us-east-1:123456789012:destination:testDestination"
    }
  ]
}
```

ポリシー要素の指定 : アクション、効果、プリンシパル

サービスは、CloudWatch Logs リソースごとに一連の API オペレーションを定義します。こうした API オペレーションへのアクセス権限を付与するために、CloudWatch Logs は一連のアクションをポリシーに定義します。一部の API オペレーションは、API オペレーションを実行するために複数のアクションに対するアクセス許可を要求できます。リソースおよび API オペレーションに関する詳細については、

「CloudWatch Logs リソースおよびオペレーション (p. 107)」および「CloudWatch Logs の権限リファレンス (p. 114)」を参照してください。

以下は、基本的なポリシーの要素です。

- リソース – Amazon リソースネーム (ARN) を使用して、ポリシーを適用するリソースを識別します。詳細については、「CloudWatch Logs リソースおよびオペレーション (p. 107)」を参照してください。
- アクション – アクションのキーワードを使用して、許可または拒否するリソースオペレーションを識別します。たとえば、`logs.DescribeLogGroups` 権限は、`DescribeLogGroups` オペレーションの実行をユーザーに許可します。
- 効果 – ユーザーが特定のアクションをリクエストする際の効果 (許可または拒否) を指定します。リソースへのアクセスを明示的に許可していない場合、アクセスは暗黙的に拒否されます。また、明示的にリソースへのアクセスを拒否すると、別のポリシーによってアクセスが許可されている場合でも、ユーザーはそのリソースにアクセスできなくなります。
- プリンシパル – アイデンティティベースのポリシー (IAM ポリシー) で、ポリシーがアタッチされているユーザーが黙示的なプリンシパルとなります。リソースベースのポリシーでは、権限 (リソースベースのポリシーにのみ適用) を受け取りたいユーザー、アカウント、サービス、またはその他のエンティティを指定します。CloudWatch Logs では、送信先のリソースベースのポリシーはサポートしています。

IAM ポリシーの構文と説明についての詳細については、『IAM ユーザーガイド』の「AWS IAM ポリシーのリファレンス」を参照してください。

すべての CloudWatch Logs API アクションとそれらが適用されるリソースの表については、「CloudWatch Logs の権限リファレンス (p. 114)」を参照してください。

ポリシーでの条件の指定

アクセス権限を付与するとき、アクセスポリシー言語を使用して、ポリシーが有効になる必要がある条件を指定できます。たとえば、特定の日付の後にのみ適用されるポリシーが必要になる場合があります。ポリシー言語での条件の指定の詳細については、IAM ユーザーガイドの「条件」を参照してください。

条件を表すには、あらかじめ定義された条件キーを使用します。各 AWS サービスでサポートされるコンテキストキーと AWS 全体のポリシーキーのリストについては、IAM ユーザーガイド の「AWS のサービスアクションと条件コンテキストキー」および「グローバルキーと IAM 条件コンテキストキー」を参照してください。

CloudWatch Logs でアイデンティティベースのポリシー (IAM ポリシー) を使用する

このトピックでは、アカウント管理者が識別 IAM アイデンティティ (ユーザー、グループ、ロール) へのアクセス権限ポリシーをアタッチする、アイデンティティベースのポリシーの例を示します。

> Important
>
> 初めに、CloudWatch Logs リソースへのアクセスを管理するための基本概念と使用可能なオプションについて説明する概要トピックを読むことをお勧めします。詳細については、「CloudWatch Logs リソースへのアクセス許可の管理の概要 (p. 106)」を参照してください。

このトピックでは次の内容について説明します。

- CloudWatch コンソールを使用するために必要なアクセス権限 (p. 111)
- CloudWatch Logs での AWS 管理 (事前定義) ポリシー (p. 113)
- お客様が管理するポリシーの例 (p. 113)

以下は、アクセス権限ポリシーの例です。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
    ],
      "Resource": [
        "arn:aws:logs:*:*:*"
    ]
  }
 ]
}
```

このポリシーには、ロググループとログストリームを作成して、ログストリームにログイベントをアップロードし、ログストリームの詳細を一覧表示する権限を付与する 1 つのステートメントがあります。

このステートメントで `Resource` 値の末尾のワイルドカード文字 (*) は、任意のロググループに対して `logs:CreateLogGroup`、`logs:CreateLogStream`、`logs:PutLogEvents`、および `logs:DescribeLogStreams` アクションを実行するためのアクセス権限を付与することを意味します。このアクセス権限を特定のロググループに制限するには、リソース ARN 内のワイルドカード文字 (*) を特定のロググループ ARN に置き換えます。IAM ポリシーステートメント内のセクションに関する詳細については、『IAM ユーザーガイド』の「IAM ポリシーエレメントリファレンス」を参照してください。すべての CloudWatch Logs アクションを示すリストについては、「CloudWatch Logs の権限リファレンス (p. 114)」を参照してください。

CloudWatch コンソールを使用するために必要なアクセス権限

CloudWatch コンソールを使用して CloudWatch Logs の作業を行うユーザーの場合、そのユーザーは、他の AWS リソースを AWS アカウントで記述できる、最小限のアクセス権限を持っている必要があります。CloudWatch コンソールで CloudWatch Logs を使用するには、次のサービスからのアクセス許可が必要になります。

- CloudWatch
- CloudWatch Logs
- Amazon ES
- IAM
- Kinesis
- Lambda
- Amazon S3

これらの最小限必要なアクセス権限よりも制限された IAMIAM ポリシーを作成している場合、その IAM ポリシーを使用するユーザーに対してコンソールは意図したとおりには機能しません。「CloudWatch Logs での AWS 管理 (事前定義) ポリシー (p. 113)」で説明されているとおり、ユーザーが CloudWatch コンソールを使用できること、および、`CloudWatchReadOnlyAccess` 管理対象ポリシーがユーザーにアタッチされていることを確認してください。

AWS CLI または CloudWatch Logs API のみを呼び出すユーザーには、最小限のコンソールアクセス権限を付与する必要はありません。

CloudWatch コンソールを使用してログのサブスクリプションを管理していないユーザーが、コンソールを使用するために必要なフルセットのアクセス許可は、次のとおりです。

- cloudwatch:getMetricData
- cloudwatch:listMetrics
- logs:cancelExportTask
- logs:createExportTask
- logs:createLogGroup
- logs:createLogStream
- logs:deleteLogGroup
- logs:deleteLogStream
- logs:deleteMetricFilter
- logs:deleteRetentionPolicy
- logs:deleteSubscriptionFilter
- logs:describeExportTasks
- logs:describeLogGroups
- logs:describeLogStreams
- logs:describeMetricFilters
- logs:describeSubscriptionFilters
- logs:filterLogEvents
- logs:getLogEvents
- logs:putMetricFilter
- logs:putRetentionPolicy
- logs:putSubscriptionFilter
- logs:testMetricFilter

コンソールを使用してログのサブスクリプションを管理するユーザーには、以下のアクセス許可も必要です。

- es:describeElasticsearchDomain
- es:listDomainNames
- iam:attachRolePolicy
- iam:createRole
- iam:getPolicy
- iam:getPolicyVersion
- iam:getRole
- iam:listAttachedRolePolicies
- iam:listRoles
- kinesis:describeStreams
- kinesis:listStreams
- lambda:addPermission
- lambda:createFunction
- lambda:getFunctionConfiguration
- lambda:listAliases
- lambda:listFunctions
- lambda:listVersionsByFunction
- lambda:removePermission

- s3:listBuckets

CloudWatch Logs での AWS 管理 (事前定義) ポリシー

AWS は、AWS によって作成され管理されるスタンドアロンの IAM ポリシーが提供する多くの一般的ユースケースに対応します。管理ポリシーは、一般的ユースケースに必要なアクセス権限を付与することで、どの権限が必要なのかをユーザーが調査する必要をなくすることができます。詳細については、『IAM ユーザーガイド』の「AWS 管理ポリシー」を参照してください。

アカウントのユーザーにアタッチできる以下の AWS 管理ポリシーは、CloudWatch Logs に固有のものです。

- CloudWatchLogsFullAccess – CloudWatch Logs へのフルアクセスを付与します。
- CloudWatchLogsReadOnlyAccess – CloudWatch Logs への読み取り専用アクセスを付与します。

> Note
>
> IAM コンソールにサインインし、特定のポリシーを検索することで、これらのアクセス権限ポリシーを確認することができます。

独自のカスタム IAM ポリシーを作成して、CloudWatch Logs のアクションとリソースのための権限を許可することもできます。こうしたカスタムポリシーは、該当するアクセス権限が必要な IAM ユーザーまたはグループにアタッチできます。

お客様が管理するポリシーの例

このセクションでは、さまざまな CloudWatch Logs アクションのアクセス権限を付与するユーザーポリシー例を示しています。これらのポリシーは、CloudWatch Logs API、AWS SDK、または AWS CLI を使用しているときに機能します。

例

- 例 1: CloudWatch Logs へのフルアクセスを許可する (p. 113)
- 例 2: CloudWatch Logs への読み取り専用アクセスを許可する (p. 113)

例 1: CloudWatch Logs へのフルアクセスを許可する

以下のポリシーでは、ユーザーにすべての CloudWatch Logs アクションへのアクセスを許可します。

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

例 2: CloudWatch Logs への読み取り専用アクセスを許可する

以下のポリシーでは、ユーザーに CloudWatch Logs データへの読み取り専用アクセスを許可します。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "logs:Describe*",
            "logs:Get*",
            "logs:TestMetricFilter",
            "logs:FilterLogEvents"
         ],
         "Effect":"Allow",
         "Resource":"*"
      }
   ]
}
```

CloudWatch Logs の権限リファレンス

アクセスコントロール (p. 106) をセットアップし、IAM アイデンティティにアタッチできる書き込みのアクセス許可ポリシー (アイデンティティベースのポリシー) を作成するときは、以下の表をリファレンスとして使用できます。この表には、各 CloudWatch Logs API オペレーション、およびその実行のためのアクセス権限を付与できる対応するアクションを示しています。ポリシーの `Action` フィールドでアクションを指定し、ポリシーの `Resource` フィールドでリソース値としてワイルドカード文字 (*) を指定します。

CloudWatch Logs ポリシーで AWS 全体の条件キーを使用して、条件を表現することができます。AWS 全体を対象とするすべてのキーのリストについては、IAM ユーザーガイド の「AWS グローバルキーと IAM 条件コンテキストキー」を参照してください。

> **Note**
>
> アクションを指定するには、API オペレーション名の前に `logs:` プレフィックスを使用します。例: `logs:CreateLogGroup`、`logs:CreateLogStream`、`logs:*` (すべての CloudWatch Logs アクションの場合)。

CloudWatch Logs API オペレーションおよびアクションで必要なアクセス許可

CloudWatch Logs API オペレーション	必要なアクセス権限 (API アクション)
CancelExportTask	`logs:CancelExportTask` 保留中または実行中のエクスポートタスクをキャンセルするのに必要です。
CreateExportTask	`logs:CreateExportTask` ログ グループからAmazon S3バケットにデータをエクスポートするのに必要です。
CreateLogGroup	`logs:CreateLogGroup` 新しいロググループを作成するのに必要です。
CreateLogStream	`logs:CreateLogStream` ロググループに新しいログストリームを作成するのに必要です。
DeleteDestination	`logs:DeleteDestination`

CloudWatch Logs API オペレーション	必要なアクセス権限 (API アクション)
	ログ宛先を削除したり、サブスクリプションのフィルタを無効化するのに必要です。
DeleteLogGroup	`logs:DeleteLogGroup` ロググループや関連したアーカイブログイベントを削除するのに必要です。
DeleteLogStream	`logs:DeleteLogStream` ログストリームや関連したアーカイブログイベントを削除するのに必要です。
DeleteMetricFilter	`logs:DeleteMetricFilter` ロググループに関連したメトリクスフィルタを削除するのに必要です。
DeleteRetentionPolicy	`logs:DeleteRetentionPolicy` ロググループの保持ポリシーを削除するのに必要です。
DeleteSubscriptionFilter	`logs:DeleteSubscriptionFilter` ロググループに関連したサブスクリプションのフィルタを削除するのに必要です。
DescribeDestinations	`logs:DescribeDestinations` アカウントに関連したすべての送信先を表示するのに必要です。
DescribeExportTasks	`logs:DescribeExportTasks` アカウントに関連したすべてのエクスポートタスクを表示するのに必要です。
DescribeLogGroups	`logs:DescribeLogGroups` アカウントに関連したすべてのロググループを表示するのに必要です。
DescribeLogStreams	`logs:DescribeLogStreams` ロググループに関連したすべてのログストリームを表示するのに必要です。
DescribeMetricFilters	`logs:DescribeMetricFilters` ロググループに関連したすべてのメトリクスを表示するのに必要です。
DescribeSubscriptionFilters	`logs:DescribeSubscriptionFilters` ロググループに関連したすべてのサブスクリプションフィルタを表示するのに必要です。

CloudWatch Logs API オペレーション	必要なアクセス権限 (API アクション)
FilterLogEvents	`logs:FilterLogEvents` ロググループのフィルタパターンでログイベントをソートするのに必要です。
GetLogEvents	`logs:GetLogEvents` ログストリームからログイベントを取得するのに必要です。
ListTagsLogGroup	`logs:ListTagsLogGroup` ロググループに関連したタグをリストするのに必要です。
PutDestination	`logs:PutDestination` 宛先ログストリーム (Kinesis ストリームなど) の作成や更新に必要です。
PutDestinationPolicy	`logs:PutDestinationPolicy` 既存のログ宛先に関連するアクセスポリシーの作成や更新に必要です。
PutLogEvents	`logs:PutLogEvents` 一連のログイベントをログストリームに更新するのに必要です。
PutMetricFilter	`logs:PutMetricFilter` メトリクスフィルタの作成や更新、またはそれをロググループに関連付けるのに必要です。
PutRetentionPolicy	`logs:PutRetentionPolicy` ロググループでログイベント (保持) を維持する日数を設定するのに必要です。
PutSubscriptionFilter	`logs:PutSubscriptionFilter` サブスクリプションフィルタの作成や更新、またはそれをロググループに関連付けるのに必要です。
TagLogGroup	`logs:TagLogGroup` ロググループのタグを追加または更新するのに必要です。
TestMetricFilter	`logs:TestMetricFilter` ログイベントメッセージのサンプリングに対してフィルタパターンをテストするのに必要です。

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

Amazon Virtual Private Cloud (Amazon VPC) を使用して AWS リソースをホストする場合は、VPC と CloudWatch Logs の間にプライベート接続を確立できます。この接続を使用して、インターネットを経由して送信せずに CloudWatch Logs にログを送信できます。

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 Logs に接続するには、CloudWatch Logs 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 Logs 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 Logs 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)
- EU (パリ)
- 南米 (サンパウロ)

CloudWatch Logs の VPC エンドポイントの作成

VPC で CloudWatch Logs の使用を開始するには、CloudWatch Logs のインターフェイス VPC エンドポイントを作成します。エンドポイント名は `com.amazonaws.Region.logs` になります。詳細については、Amazon VPC ユーザーガイド の「インターフェイスエンドポイントの作成」を参照してください。

CloudWatch Logs の設定を変更する必要はありません。CloudWatch Logs 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 Logs のインターフェイス VPC エンドポイントを作成する際、Kinesis Data Streams の CloudWatch Logs サブスクリプションフィルタおよび Kinesis Data Streams のインターフェイス VPC エンドポイントがすでにある場合は、CloudWatch Logs と Kinesis Data Streams の間の呼び出しは、インターフェイス VPC エンドポイントを介して流れ始めます。

VPC と CloudWatch Logs との間の接続のテスト

エンドポイントの作成が完了したら、接続をテストできます。

VPC と CloudWatch Logs のエンドポイント間の接続をテストするには

1. VPC にある Amazon EC2 インスタンスに接続します。接続の詳細については、Amazon EC2 のドキュメントの「Linux インスタンスへの接続」または「Windows インスタンスへの接続」を参照してください。
2. インスタンスから、AWS CLI を使用して既存のロググループのいずれかにログエントリを作成します。

   まず、ログイベントを持つ JSON ファイルを作成します。タイムスタンプは、1970 年 1 月 1 日 00:00:00 UTC からの経過ミリ秒数で指定する必要があります。

   ```
   [
     {
       "timestamp": 1533854071310,
       "message": "VPC Connection Test"
     }
   ]
   ```

   次に、`put-log-events` コマンドを使用してログエントリを作成します。

   ```
   aws logs put-log-events --log-group-name LogGroupName --log-stream-name LogStreamName
    --log-events file://JSONFileName
   ```

   コマンドに対する応答に `nextSequenceToken` が含まれていた場合、コマンドは成功しており VPC エンドポイントが機能しています。

VPC コンテキストキーのサポート

CloudWatch Logs では、特定の VPC または特定の VPC エンドポイントへのアクセスの制限に使用できる `aws:SourceVpc` および `aws:SourceVpce` コンテキストキーをサポートしています。これらのキーは、ユーザーが VPC エンドポイントを使用している場合にのみ使用できます。詳細については、「https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_condition-keys.html#condition-keys-service-available.html」を参照してください。

Amazon CloudWatch Logs での AWS CloudTrail API コールのログ記録

Amazon CloudWatch Logs は、CloudWatch Logs 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 Logs 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 Logs 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 Logs 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 Logs 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 Logs 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 Logs 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 Logs 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。

- CancelExportTask
- CreateExportTask
- CreateLogGroup

- CreateLogStream
- DeleteDestination
- DeleteLogGroup
- DeleteLogStream
- DeleteMetricFilter
- DeleteRetentionPolicy
- DeleteSubscriptionFilter
- PutDestination
- PutDestinationPolicy
- PutMetricFilter
- PutRetentionPolicy
- PutSubscriptionFilter
- TestMetricFilter

次の CloudWatch Logs API アクションの場合、リクエスト要素のみが CloudTrail に記録されます。

- DescribeDestinations
- DescribeExportTasks
- DescribeLogGroups
- DescribeLogStreams
- DescribeMetricFilters
- DescribeSubscriptionFilters

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。

ログファイルエントリの概要

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。

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

```
{
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "IAMUser",
```

```
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/someuser",
            "accountId": "123456789012",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "someuser"
        },
        "eventTime": "2016-02-08T06:35:14Z",
        "eventSource": "logs.amazonaws.com",
        "eventName": "CreateExportTask",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
        "requestParameters": {
            "destination": "yourdestination",
            "logGroupName": "yourloggroup",
            "to": 123456789012,
            "from": 0,
            "taskName": "yourtask"
        },
        "responseElements": {
            "taskId": "15e5e534-9548-44ab-a221-64d9d2b27b9b"
        },
        "requestID": "1cd74c1c-ce2e-12e6-99a9-8dbb26bd06c9",
        "eventID": "fd072859-bd7c-4865-9e76-8e364e89307c",
        "eventType": "AwsApiCall",
        "apiVersion": "20140328",
        "recipientAccountId": "123456789012"
}
```

CloudWatch Logs エージェントのリファレンス

CloudWatch Logs エージェントには、ログデータを Amazon EC2 インスタンスから CloudWatch Logs に送信する自動化された方法が用意されています。エージェントは次のコンポーネントで構成されています。

- ログデータを CloudWatch Logs にプッシュする AWS CLI プラグイン。
- データを CloudWatch Logs にプッシュするプロセスを開始するスクリプト (デーモン)。
- デーモンが常に実行中であることを確認する cron ジョブ。

エージェント設定ファイル

CloudWatch Logs エージェント設定ファイルは、CloudWatch Logs エージェントが必要とする情報を記述します。エージェント設定ファイルの [general] セクションは、すべてログストリームに適用する一般的な設定を定義します。[logstream] セクションは、リモートなログストリームにローカルファイルを送信するために必要な情報を定義します。複数の [logstream] セクションを持つことができますが、設定ファイル内にそれぞれ [logstream1]、[logstream2] などの一意の名前を持つ必要があります。ログファイルのデータの最初の行とともにある [logstream] 値は、ログファイルの ID を定義します。

```
[general]
state_file = value
logging_config_file = value
use_gzip_http_content_encoding = [true | false]

[logstream1]
log_group_name = value
log_stream_name = value
datetime_format = value
time_zone = [LOCAL|UTC]
file = value
file_fingerprint_lines = integer | integer-integer
multi_line_start_pattern = regex | {datetime_format}
initial_position = [start_of_file | end_of_file]
encoding = [ascii|utf_8|..]
buffer_duration = integer
batch_count = integer
batch_size = integer

[logstream2]
...
```

state_file

状態ファイルをどこに保存するかを指定します。

logging_config_file

(オプション) エージェントのログ config ファイルの場所を指定します。ここでエージェントのログ config ファイルを指定しない場合は、デフォルトファイル awslogs.conf が使用されます。スクリプトでエージェントをインストールした場合、デフォルトのファイルの場所は `/var/awslogs/`

`etc/awslogs.conf` です。rpm でエージェントをインストールした場合は、`/etc/awslogs/awslogs.conf` です。このファイルは、Python の設定ファイル形式 (https://docs.python.org/2/library/logging.config.html#logging-config-fileformat) です。以下の名前のロガーはカスタマイズできます。

```
cwlogs.push
cwlogs.push.reader
cwlogs.push.publisher
cwlogs.push.event
cwlogs.push.batch
cwlogs.push.stream
cwlogs.push.watcher
```

以下のサンプルは、デフォルトの値が INFO であるリーダーとパブリッシャーのレベルを WARNING に変更します。

```
[loggers]
keys=root,cwlogs,reader,publisher

[handlers]
keys=consoleHandler

[formatters]
keys=simpleFormatter

[logger_root]
level=INFO
handlers=consoleHandler

[logger_cwlogs]
level=INFO
handlers=consoleHandler
qualname=cwlogs.push
propagate=0

[logger_reader]
level=WARNING
handlers=consoleHandler
qualname=cwlogs.push.reader
propagate=0

[logger_publisher]
level=WARNING
handlers=consoleHandler
qualname=cwlogs.push.publisher
propagate=0

[handler_consoleHandler]
class=logging.StreamHandler
level=INFO
formatter=simpleFormatter
args=(sys.stderr,)

[formatter_simpleFormatter]
format=%(asctime)s - %(name)s - %(levelname)s - %(process)d - %(threadName)s -
 %(message)s
```

use_gzip_http_content_encoding

true (デフォルト) に設定すると、CloudWatch Logs への圧縮されたペイロードの送信に対して、gzip による HTTP コンテンツのエンコードが有効になります。これにより、CPU の使用率が減り、NetworkOut が少なくなり、Put のレイテンシーが短くなります。この機能

を無効にするには、CloudWatch Logs エージェント設定ファイルの [general] セクションに use_gzip_http_content_encoding = false を追加してから、エージェントを再起動します。

Note

この設定は awscli-cwlogs バージョン 1.3.3 以降でのみ使用できます。

log_group_name

送信先ロググループを指定します。ロググループが存在しない場合には、自動的に作成されます。ロググループの名前は 1～512 文字で指定します。ここで使えるのは、a～z、A～Z、0～9、"_" (アンダーバー)、"-" (ハイフン)、"/" (スラッシュ) および "." (ピリオド) です。

log_stream_name

送信先ログストリームを指定します。リテラル文字列、定義済み変数 ({instance_id}、{hostname}、{ip_address})、またはこれらの組み合わせを使用して、ログストリーム名を定義できます。ログストリームが存在しない場合には、自動的に作成されます。

datetime_format

ログからタイムスタンプを入手する方法を指定します。タイムスタンプはログイベントを取得し、メトリクスを生成するために使用されます。現在の時刻は、[datetime_format] が提供されていない場合に各ログイベントで使用されます。提供された [datetime_format] の値がそのログメッセージに対して無効の場合は、適切に解析されたタイムスタンプを持つ最後のログイベントのタイムスタンプが使用されます。以前のログイベントが存在しない場合は、現在の時刻が使用されます。

一般的な datetime_format コードは次のとおりです。Python がサポートする datetime_format コード (datetime.strptime()) も使用できます。タイムゾーンオフセット (%z) もサポートされています。ただし、Python 3.2 までは、コロン (:) のない [+-] HHMM はサポートされていません。詳細については、「strftime() および strptime() Behavior ()」を参照してください。

[%y]: ゼロ詰め 10 進数での年 (世紀なし) です。00, 01, ..., 99

%Y: 10 進数での年 (世紀あり) です。1970、1988、2001、2013

%b: ロケールの省略名称での月です。Jan、Feb ... Dec (en_US) ;

%B: ロケールの正式名称での月です。January、February...December (en_US) ;

%m: ゼロ詰め 10 進数での月です。01, 02, ..., 12

%d: ゼロ詰め 10 進数での日です。01, 02, ..., 31

%H: ゼロ詰め 10 進数での時 (24 時間形式の時計) です。00, 01, ..., 23

%I: ゼロ詰め 10 進数での時 (12 時間形式の時計) です。01, 02, ..., 12

%p: ロケールで AM または PM に相当するものです。

%M: ゼロ詰め 10 進数での分です。00, 01, ..., 59

%S: ゼロ詰め 10 進数での秒です。00, 01, ..., 59

%f: 左ゼロ詰め 10 進数でのマイクロ秒です。000000, ..., 999999

%z: +HHMM または -HHMM 形式の UTC オフセットです。+0000, -0400, +1030

形式の例:

```
Syslog: '%b %d %H:%M:%S', e.g. Jan 23 20:59:29

Log4j: '%d %b %Y %H:%M:%S', e.g. 24 Jan 2014 05:00:00
```

```
ISO8601: '%Y-%m-%dT%H:%M:%S%z', e.g. 2014-02-20T05:20:20+0000
```

time_zone

ログイベントのタイムスタンプのタイムゾーンを指定します。サポートされる 2 つの値は UTC および LOCAL です。デフォルトは LOCAL です。タイムゾーンが [datetime_format] に基づいて推定できない場合に使用されます。

file

CloudWatch Logs にプッシュするログファイルを指定します。ファイルは、特定のファイルまたは複数のファイルを指すことができます (/var/log/system.log* のようにワイルドカードを使用) 。ファイルの変更時間に基づいて、最新のファイルのみが CloudWatch Logs にプッシュされます。access_log.2014-06-01-01 と access_log.2014-06-01-02 など同じ形式の一連のファイルを指定するにはワイルドカードの使用をお勧めします。ただし、access_log_80 と access_log_443 のように複数の種類のファイルには使用しないでください。複数の種類のファイルを指定するには、設定ファイルに別のストリームログのエントリを追加して、各種類のログファイルが異なるログストリームに行くようにします。圧縮ファイルはサポートされていません。

file_fingerprint_lines

ファイルを識別するための行範囲を指定します。有効な値は「1」「2-5」のように単一の数字またはハイフンで区切られた 2 つの数字です。デフォルト値は「1」です。最初の行を使用してフィンガープリントを計算します。指定された行がすべて存在しない限り、フィンガープリント行は CloudWatch Logs に送信されません。

multi_line_start_pattern

ログメッセージの開始を識別するパターンを指定します。ログメッセージは、パターンに一致する 1 行と、それに続くパターンに一致しない行で構成されます。有効な値は正規表現または {datetime_format} です。{datetime_format} を使用する場合は、datetime_format オプションを指定する必要があります。デフォルト値は「^[^\s]」です。よって、空白文字以外の文字で始まる行で前のログメッセージを終了し、新しいログメッセージを開始します。

initial_position

データの読み出しをどこから開始するかを指定します (start_of_file または end_of_file) 。デフォルトは start_of_file です。そのログストリームに保持されている状態がない場合にのみ使用されます。

encoding

ファイルを正しく読み込むことができるように、ログファイルのエンコードを指定します。デフォルトは utf_8 です。Python の codecs.decode() がサポートするエンコードを使用できます。

> Warning
>
> 正しくないエンコードを指定すると、デコードできない文字がそのほかの文字に置き換えられるため、データ損失が生じる可能性があります。

一般的なエンコードを次に示します。

```
ascii, big5, big5hkscs, cp037, cp424, cp437, cp500, cp720, cp737, cp775,
cp850, cp852, cp855, cp856, cp857, cp858, cp860, cp861, cp862, cp863, cp864,
cp865, cp866, cp869, cp874, cp875, cp932, cp949, cp950, cp1006, cp1026,
cp1140, cp1250, cp1251, cp1252, cp1253, cp1254, cp1255, cp1256, cp1257,
cp1258, euc_jp, euc_jis_2004, euc_jisx0213, euc_kr, gb2312, gbk, gb18030,
hz, iso2022_jp, iso2022_jp_1, iso2022_jp_2, iso2022_jp_2004, iso2022_jp_3,
iso2022_jp_ext, iso2022_kr, latin_1, iso8859_2, iso8859_3, iso8859_4,
iso8859_5, iso8859_6, iso8859_7, iso8859_8, iso8859_9, iso8859_10,
iso8859_13, iso8859_14, iso8859_15, iso8859_16, johab, koi8_r, koi8_u,
mac_cyrillic, mac_greek, mac_iceland, mac_latin2, mac_roman, mac_turkish,
ptcp154, shift_jis, shift_jis_2004, shift_jisx0213, utf_32, utf_32_be,
utf_32_le, utf_16, utf_16_be, utf_16_le, utf_7, utf_8, utf_8_sig
```

buffer_duration

ログイベントのバッチ期間を指定します。最小値は 5000ms で、デフォルト値は 5000ms です。

batch_count

バッチのログイベントの最大値を 10000 までの値で指定します。デフォルト値は 10000 です。

batch_size

バッチのログイベントの最大値を 1048576 バイトまでのバイト値で指定します。デフォルト値は 1048576 バイトです。このサイズは、UTF-8 のすべてのイベントメッセージの合計に各ログイベントにつき 26 バイトを加算して計算されます。

HTTP プロキシでの CloudWatch Logs エージェントの使用

HTTP プロキシで CloudWatch Logs エージェントを使用できます。

Note

HTTP プロキシは awslogs-agent-setup.py バージョン 1.3.8 以降でサポートされています。

HTTP プロキシで CloudWatch Logs エージェントを使用するには

1. 以下のいずれかを行います。

   a. CloudWatch Logs エージェントを新たにインストールする場合は、以下のコマンドを実行します。

   ```
   curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-setup.py -O
   ```

   ```
   sudo python awslogs-agent-setup.py --region us-east-1 --http-proxy http://your/proxy --https-proxy http://your/proxy --no-proxy 169.254.169.254
   ```

   EC2 インスタンスで Amazon EC2 メタデータサービスへのアクセスを維持するには、[--no-proxy 169.254.169.254] (推奨) を使用します。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「インスタンスメタデータとユーザーデータ」を参照してください。

   `http-proxy` および `https-proxy` の値で、URL 全体を指定します。

   b. CloudWatch Logs エージェントが既にインストールされている場合は、/var/awslogs/etc/proxy.conf 編集し、プロキシを追加します。

   ```
   HTTP_PROXY=
   HTTPS_PROXY=
   NO_PROXY=
   ```

2. エージェントを再起動して、変更を有効にします。

   ```
   sudo service awslogs restart
   ```

   Amazon Linux 2 を使用している場合は、次のコマンドを使用してエージェントを再起動します。

   ```
   sudo service awslogsd restart
   ```

CloudWatch Logs エージェント設定ファイルのコンパートメント化

awslogs-agent-setup.py バージョン 1.3.8 以降と awscli-cwlogs 1.3.3 以降を使用している場合は、/var/awslogs/etc/config/ に追加の設定ファイルを作成することで、さまざまなコンポーネントのストリーム設定をそれぞれ個別にインポートできます。CloudWatch Logs エージェントが起動すると、これらの追加の設定ファイルにストリーム設定が追加されます。[general] セクションの設定プロパティはメインの設定ファイル (/var/awslogs/etc/awslogs.conf) で定義する必要があり、/var/awslogs/etc/config/ にある追加の設定ファイルで定義しても無視されます。

rpm でエージェントをインストールしたため /var/awslogs/etc/config/ ディレクトリがない場合は、代わりに /etc/awslogs/config/ ディレクトリを使用できます。

エージェントを再起動して、変更を有効にします。

```
sudo service awslogs restart
```

Amazon Linux 2 を使用している場合は、次のコマンドを使用してエージェントを再起動します。

```
sudo service awslogsd restart
```

CloudWatch Logs エージェントのよくある質問

どのようなファイルローテーションがサポートされていますか。

次のファイルローテーション機能がサポートされています。

- 既存のファイルを数字サフィックスをつけた名前に変更し、その後、元の名前の空のログファイルを作成し直します。たとえば、/var/log/syslog.log が /var/log/syslog.log.1 という名前に変更されます。/var/log/syslog.log.1 が前回のローテーションにより既に存在する場合は、/var/log/syslog.log.2 という名前に変更されます。
- 元のログファイルを、コピーを作成した後切り捨てます。たとえば、/var/log/syslog.log を /var/log/syslog.log.1 にコピーし、/var/log/syslog.log を切り捨てます。この場合、データを損失する恐れがあるため、このファイルローテーション機能の使用にはご注意ください。
- 古いファイルと共通のパターンを持つ新しいファイルを作成します。たとえば /var/log/syslog.log.2014-01-01 をそのまま残し、/var/log/syslog.log.2014-01-02 を作成します。

ファイルのフィンガープリント (ソース ID) は、ログストリームキーとファイルのコンテンツの 1 行目をハッシュして計算されます。この動作をオーバーライドするには、[file_fingerprint_lines] オプションを使用できます。ファイルのローテーションが発生した場合、新しいファイルには新しいコンテンツがあり古いファイルにはコンテンツの追加がないと思われるため、エージェントは古いファイルの読み込みが完了した後は、新しいファイルをプッシュします。

使用しているエージェントのバージョンを確認する方法

セットアップスクリプトから CloudWatch Logs エージェントをインストールした場合、[/var/awslogs/bin/awslogs-version.sh] で使用しているエージェントのバージョンを確認することができます。エージェントのバージョンと主要な依存関係がプリントアウトされます。yum から CloudWatch Logs エージェントをインストールした場合には、["yum info awslogs"] と ["yum info aws-cli-plugin-cloudwatch-logs"] で CloudWatch Logs エージェントとプラグインのバージョンを確認することができます。

ログのエントリは、どのようにログイベントに変換されるのですか。

ログイベントには 2 つのプロパティが含まれます。イベント発生時のタイムスタンプおよび生のログメッセージです。デフォルトでは、空白文字以外の文字で始まる行は、前のログメッセージが

ある場合はこれを終了して新しいログメッセージを開始します。この動作をオーバーライドするには、[multi_line_start_pattern] を使用します。パターンに一致する行が新しいログメッセージを開始します。パターンには正規表現または「{datetime_format}」を使用できます。たとえば、それぞれのログメッセージの 1 行目が「2014-01-02T13:13:01Z」のようなタイムスタンプを持っている場合、multi_line_start_pattern は「\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}Z'」と設定できます。設定を簡略化するために、datetime_format オプションが指定されている場合は「{datetime_format}」変数を使用できます。同じ例で、datetime_format が「%Y-%m-%dT%H:%M:%S%z」に設定されている場合、multi_line_start_pattern は「{datetime_format}」だけにできます。

現在の時刻は、[datetime_format] が提供されていない場合に各ログイベントで使用されます。提供された [datetime_format] がそのログメッセージに対して無効の場合は、適切に解析されたタイムスタンプを持つ最後のログイベントのタイムスタンプが使用されます。前のログイベントがない場合は、現在の時刻が使用されます。ログイベントが現在の時刻または前のログイベントの時刻にフォールバックした場合は、警告メッセージが記録されます。

タイムスタンプはログイベントを取得し、メトリクスを生成するために使用されます。誤った形式を指定した場合、ログイベントが取得できなくなり誤ったメトリクスが生成されます。

ログイベントはどのようにバッチされていますか。

次の条件のいずれかが満たされる場合、バッチがフルになり発行されます。

1. 最初のログイベントが追加されてから、[buffer_duration] の時間が経過した。
2. 累積されたログイベントの [batch_size] 未満ですが、新しいログイベントを追加すると [batch_size] を超過します。
3. ログイベント数は [batch_count] に達しました。
4. バッチのログイベントは 24 時間以上になりませんが、新しいログイベントを追加すると 24 時間の制約を超過します。

ログエントリ、ログイベント、またはバッチがスキップまたは切り捨てられるのはどのような原因がありますか。

`PutLogEvents` オペレーションの制約に従って、次の問題によりログイベントまたはバッチがスキップされる場合があります。

> Note
>
> データがスキップされた場合、CloudWatch Logs エージェントはログに警告を書き込みます。

1. ログイベントのサイズが 256 KB を超過した場合、ログイベントは完全にスキップされます。
2. ログイベントのタイムスタンプが 2 時間以上未来の場合、ログイベントはスキップされます。
3. ログイベントのタイムスタンプが 14 日以上過去の場合、ログイベントはスキップされます。
4. ログイベントがロググループの保持期間よりも古い場合、バッチはすべてスキップされます。
5. 単一の `PutLogEvents` リクエストでログイベントのバッチが 24 時間実行されている場合、`PutLogEvents` オペレーションは失敗します。

エージェントを停止させた場合、データ損失や重複が発生しますか。

状態ファイルが使用可能であり、最後に実行されたときからファイルのローテーションが発生していなければ、発生しません。CloudWatch Logs エージェントは停止した場所から再開してログデータのプッシュを続行できます。

同一または異なるホストの異なるログデータを同じログストリームに指定できますか。

複数のログソースから単一のログストリームにデータを送信する設定はサポートされていません。

エージェントはどの API に呼び出しを作成しますか (またはどのアクションを IAM ポリシーに含める必要がありますか)。

CloudWatch Logs エージェントには、`CreateLogGroup`、`CreateLogStream`、`DescribeLogStreams`、および `PutLogEvents` オ

ペレーションが必要です。最新のエージェントを使用している場合には、DescribeLogStreams は必要ありません。以下の IAM ポリシーの例を参照してください。

```
{
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "logs:CreateLogGroup",
      "logs:CreateLogStream",
      "logs:PutLogEvents",
      "logs:DescribeLogStreams"
    ],
    "Resource": [
      "arn:aws:logs:*:*:*"
    ]
  }
 ]
}
```

CloudWatch Logs エージェントに自動的にロググループまたはログストリームを作成させたくありません。エージェントによるロググループとログストリームの再作成を禁止する方法を教えてください。

IAM ポリシーで、エージェントを次のオペレーションのみに制限できます。DescribeLogStreams、PutLogEvents。

トラブルシューティング時にはどのログを調べますか。

エージェントのインストールログは /var/log/awslogs-agent-setup.log に、エージェントログは /var/log/awslogs.log にあります。

CloudWatch メトリクスの使用状況のモニタリング

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

CloudWatch Logs のメトリクス

AWS/Logs 名前空間には、次のメトリクスが含まれます。

メトリクス	説明
IncomingBytes	CloudWatch Logs にアップロードされたログイベントのボリューム (非圧縮バイト数)。LogGroupName ディメンションと同時に使用すると、ロググループにアップロードされたログイベントのボリューム (非圧縮バイト数) になります。 有効なディメンション: LogGroupName 有効な統計: Sum 単位: バイト
IncomingLogEvents	CloudWatch Logs にアップロードされたログイベントの数。LogGroupName ディメンションと同時に使用すると、ロググループにアップロードされたログイベントの数になります。 有効なディメンション: LogGroupName 有効な統計: Sum 単位: なし
ForwardedBytes	サブスクリプション送信先に転送されたログイベントのボリューム (圧縮済みバイト数)。 有効なディメンション: LogGroupName、DestinationType、FilterName 有効な統計: Sum 単位: バイト
ForwardedLogEvents	サブスクリプション送信先に転送されたログイベントの数。 有効なディメンション: LogGroupName、DestinationType、FilterName 有効な統計: Sum 単位: なし

メトリクス	説明
`DeliveryErrors`	データをサブスクリプション送信先に転送するときに CloudWatch Logs がエラーを受け取ったログイベントの数。 有効なディメンション: LogGroupName、DestinationType、FilterName 有効な統計: Sum 単位: なし
`DeliveryThrottling`	データをサブスクリプション送信先に転送するときに CloudWatch Logs がスロットルされたログイベントの数。 有効なディメンション: LogGroupName、DestinationType、FilterName 有効な統計: Sum 単位: なし

CloudWatch Logs メトリクスのディメンション

CloudWatch Logs メトリクスで使用できるディメンションを以下に示します。

ディメンション	説明
`LogGroupName`	メトリクスを表示する CloudWatch Logs ロググループの名前。
`DestinationType`	CloudWatch Logs データのサブスクリプション送信先 (AWS Lambda、Amazon Kinesis Data Streams、または Amazon Kinesis Data Firehose)。
`FilterName`	ロググループから送信先にデータを転送するサブスクリプションフィルタの名前。サブスクリプションフィルタ名は CloudWatch で自動的に ASCII に変換され、サポートされていない文字は疑問符 (?) に置き換えられます。

CloudWatch Logs の制限

CloudWatch Logs には以下の制限があります。

リソース	デフォルトの制限
バッチサイズ	1 MB (最大)。この制限は変更できません。
データアーカイブ	データアーカイブは 5GB まで無料です。この制限は変更できません。
DescribeLogStreams	1 リージョン、1 アカウントあたり 5 件のトランザクション/秒 (TPS)。
検出されるログフィールド	CloudWatch Logs Insights は、ロググループ内の最大 1000 個のログイベントフィールドを検出できます。この制限は変更できません。
イベントサイズ	256 KB (最大)。この制限は変更できません。
エクスポートタスク	アカウントごとに、一度に 1 つのアクティブ (実行中または保留中) のエクスポートタスクがあります。この制限は変更できません。
FilterLogEvents	1 リージョン、1 アカウントあたり 5 件のトランザクション/秒 (TPS)。この制限は変更できません。
GetLogEvents	1 秒、1 アカウント、1 リージョンあたり 10 リクエスト。この制限は変更できません。 継続的に新しいデータを処理している場合は、サブスクリプションをお勧めします。履歴データが必要な場合は、データを Amazon S3 にエクスポートすることをお勧めします。
受信データ	受信データは 5 GB まで無料です。この制限は変更できません。
ロググループ	1 アカウント、1 リージョンあたり 5000 ロググループ。制限の引き上げをリクエストできます。 1 つのロググループに属することができるログストリームの数に制限はありません。
メトリクスフィルタ	1 ロググループあたり 100。この制限は変更できません。
PutLogEvents	1 秒、1 ログストリームあたり 5 リクエスト。追加のリクエストは調整されます。この制限は変更できません。 PutLogEvents の最大バッチサイズは 1 MB です。 1 秒、1 アカウント、1 リージョンあたり 1500 件のトランザクション。ただし、以下のリージョンでは 1 秒、1 アカウント、1 リージョンあたり 800 件のトランザクションに制限されます。ap-south-1、ap-northeast-1、ap-northeast-2、ap-southeast-1、ap-southeast-2、eu-central-1、eu-west-2、sa-east-1、us-east-2、および us-west-1。制限の引き上げをリクエストできます。

リソース	デフォルトの制限
クエリの同時実行数	最大 4 個の同時実行 CloudWatch Logs Insights クエリ (ダッシュボードに追加したクエリも含まれます)。制限の引き上げをリクエストできます。
コンソールに表示されるクエリ結果	CloudWatch Logs Insights クエリの結果として、最大 10,000 個のログイベントがコンソールに表示されます。この制限は変更できません。
サブスクリプションフィルタ	1 ロググループあたり 1。この制限は変更できません。

ドキュメント履歴

次の表に、2018 年 6 月以降の CloudWatch Logs 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
Amazon VPC エンドポイントのサポート (p. 134)	これで、VPC と CloudWatch Logs 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 Logs User Guide』の「VPC エンドポイントでの CloudWatch Logs の使用」を参照してください。	June 28, 2018

以下の表は、Amazon CloudWatch Logs 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
CloudWatch Logs Insights	CloudWatch Logs Insights では、ログデータをインタラクティブに検索および分析できます。詳細については、「CloudWatch Logs Insights でログデータを分析する (p. 34)」を参照してください。	2018 年 11 月 27 日
インターフェイス VPC エンドポイント	一部のリージョンでは、インターフェイス VPC エンドポイントを使用して、Amazon VPC と CloudWatch Logs との間のトラフィックが Amazon ネットワークから離れないように維持できます。詳細については、「CloudWatch Logs とインターフェイス VPC エンドポイントの使用 (p. 117)」を参照してください。	2018 年 3 月 7 日
Route 53 DNS クエリログ	CloudWatch Logs を使用して Route 53 が受け取った DNS クエリについてのログを保存できます。詳細については、『Amazon Route 53 開発者ガイド』の「Amazon CloudWatch Logs とは (p. 1)」または「DNS クエリの口グ記録」を参照してください。	2017 年 9 月 7 日
ロググループのタグ付け	タグを使用すると、ロググループを分類できます。詳細については、「Amazon CloudWatch Logs ロググループのタグ付け (p. 51)」を参照してください。	2016 年 12 月 13 日
コンソールの改善	メトリクスグラフから関連するロググループに移動できます。詳細については、「メトリクスから口グへのピボット (p. 72)」を参照してください。	2016 年 11 月 7 日
コンソールの再利用可能性の向上	検索、フィルタ、トラブルシューティングの作業が容易になりました。たとえば、日時の範囲でログデータをフィルタすることができます。詳細については、「CloudWatch Logs に送信されたログデータの表示 (p. 50)」を参照してください。	2016 年 8 月 29 日

変更	説明	リリース日
Amazon CloudWatch Logs および新しい CloudWatch Logs メトリクス向けに追加された AWS CloudTrail サポート	CloudWatch Logs に対する AWS CloudTrail のサポートが追加されました。詳細については、「Amazon CloudWatch Logs での AWS CloudTrail API コールのログ記録 (p. 119)」を参照してください。	2016 年 3 月 10 日
CloudWatch Logs の Amazon S3 へのエクスポートのサポートを追加しました。	CloudWatch Logs データの Amazon S3 へのエクスポートのサポートを追加しました。詳細については、「Amazon S3 へのログデータのエクスポート (p. 94)」を参照してください。	2015 年 12 月 7 日
Amazon CloudWatch Logs で AWS CloudTrail が記録したイベントのサポートを追加	CloudWatch にアラームを作成して、CloudTrail がキャプチャした特定 API アクティビティの通知を受け取り、通知をトラブルシューティングの実行に使用できます。	2014 年 11 月 10 日
Amazon CloudWatch Logs のサポートの追加	Amazon CloudWatch Logs を使用して、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスまたはその他のソースのシステム、アプリケーション、およびカスタムログファイルの監視、保存、アクセスができます。その後、Amazon CloudWatch コンソール、AWS CLI の CloudWatch Logs コマンド、または CloudWatch Logs SDK を使用して、CloudWatch Logs から関連ログデータを取得できます。詳細については、「Amazon CloudWatch Logs とは (p. 1)」を参照してください。	2014 年 7 月 10 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。