aws

ユーザーガイド

Amazon CloudWatch Logs

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、Route 53、およびその他のソースからログファイルをモニタリング AWS CloudTrail、保存、およびアクセスできます。

CloudWatch ログを使用すると、使用するすべてのシステム、アプリケーション、 AWS サービスのログを、スケーラブルな単一のサービスに一元化できます。その後、それらを簡単に表示したり、特定のエラーコードやパターンを検索したり、特定のフィールドに基づいてフィルタリングしたり、将来の分析のために安全にアーカイブしたりできます。 CloudWatch Logs を使用すると、ソースに関係なく、すべてのログをイベントの単一の一貫したフローとして時間順に表示できます。

CloudWatch ログは、強力なクエリ言語によるログのクエリ、ログ内の機密データの監査とマスキング、フィルターまたは埋め込みログ形式を使用したログからのメトリクスの生成もサポートします。

CloudWatch ログは 2 つのログクラス をサポートします。CloudWatch Logs Standard ログクラスのロググループは、すべての CloudWatch ログ機能をサポートします。CloudWatch Logs 低頻度アクセスログクラスのロググループでは、取り込み料金が低くなり、標準クラスの機能のサブセットがサポートされます。詳細については、「ログクラス」を参照してください。

機能

- 柔軟性のための 2 つのログクラス — CloudWatch Logs には 2 つのログクラスが用意されているため、アクセス頻度の低いログに対して費用対効果の高いオプションを使用できます。リアルタイムモニタリングやその他の機能を必要とするログには、フル機能オプションもあります。詳細については、「ログクラス」を参照してください。
- ログデータのクエリ — CloudWatch Logs Insights を使用して、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。クエリを実行すると、運用上の問題により効率的かつ効果的に対応できます。 CloudWatch Logs Insights には、シンプルで強力なコマンドがいくつか用意された専用のクエリ言語が含まれています。提供されているサンプルのクエリ、コマンドの説明、クエリの自動補完、およびログフィールドの検出を利用して簡単に使用を開始できます。サンプルクエリは、いくつかのタイプの AWS サービスログに含まれています。開始するには、 CloudWatch Logs Insights を使用したログデータの分析 を参照してください。
- Live Tail を使用した検出とデバッグ — Live Tail を使用すれば、新しいログイベントのストリーミングリストを取り込みに表示して、インシデントのトラブルシューティングをすばやく行うことができます。取り込まれたログをほぼリアルタイムで表示、フィルタリング、強調表示できるため、

問題をすばやく検出して解決することができます。指定した用語に基づいてログをフィルタリングしたり、特定の用語を含むログを強調表示したりすることで、探しているものをすぐに見つけることができます。詳細については、「Live Tail を使用すると、ログをほぼリアルタイムで表示できます。」を参照してください。

- Amazon EC2 インスタンスからのログのモニタリング – CloudWatch ログを使用して、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。例えば、 CloudWatch Logs はアプリケーションログで発生したエラーの数を追跡し、エラー率が指定したしきい値を超えるたびに通知を送信できます。 CloudWatch Logs はログデータをモニタリングに使用するため、コードの変更は必要ありません。例えば、特定のリテラル語 (「」などNullReferenceException) のアプリケーションログをモニタリングしたり、ログデータ内の特定の位置 (Apache アクセスログの「404」ステータスコードなど) でのリテラル語の出現回数をカウントしたりできます。検索する用語が見つかると、 CloudWatch Logs は指定した 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、「Logs CloudWatch の開始方法」を参照してください。
- AWS CloudTrail ログに記録されたイベントのモニタリング – でアラームを作成し CloudWatch 、によってキャプチャされた特定の API アクティビティの通知を受信 CloudTrail し、通知を使用してトラブルシューティングを実行できます。開始するには、「 AWS CloudTrail ユーザーガイド」のCloudWatch 「ログへの CloudTrail イベントの送信」を参照してください。
- 機密データの監査とマスキング — ログに機密データがある場合は、データ保護ポリシーを使用して保護できます。これらのポリシーにより、機密性の高いデータを監査してマスクできます。データ保護を有効にすると、デフォルトでは、選択したデータ識別子と一致する機密データがマスクされます。詳細については、「機密性の高いログデータをマスキングで保護する」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。
- ログデータのアーカイブ – CloudWatch ログを使用して、ログデータを耐久性の高いストレージに保存できます。 CloudWatch Logs エージェントを使用すると、ローテーションされたログデータとローテーションされていないログデータの両方をホストからログサービス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。
- ログ Route 53 DNS クエリ – CloudWatch ログを使用して、Route 53 が受信した DNS クエリに関する情報をログに記録できます。詳細については、Amazon Route 53 デベロッパーガイドの「DNS クエリのログ」を参照してください。

関連 AWS サービス

次の サービスは、 CloudWatch ログと組み合わせて使用されます。

- AWS CloudTrail は、、 AWS Management Console AWS Command Line Interface （ AWS CLI ）、およびその他の のサービスによる呼び出しなど、アカウントの CloudWatch Logs API に対する呼び出しをモニタリングできるウェブサービスです。 CloudTrail ログ記録が有効になっている場合、 CloudTrail はアカウント内の API コールをキャプチャし、指定した Amazon S3 バケットにログファイルを配信します。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。の詳細については AWS CloudTrail、「 ユーザーガイド」の「 とは AWS CloudTrailAWS CloudTrail 」を参照してください。が CloudTrail ログファイルに CloudWatch 書き込むデータのタイプの例については、「」を参照してくださいでの CloudWatch Logs API およびコンソールオペレーションの ログ記録 AWS CloudTrail。
- AWS Identity and Access Management (IAM) は、ユーザーの AWS リソースへのアクセスを安全に制御するのに役立つ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 デベロッパーガイドの「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ギングなどが含まれます。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 デベロッパーガイド」の「 とは AWS Lambda」を参照してください。

料金

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

標準料金は、 ログを使用して他の のサービスによって保存される CloudWatch ログ (Amazon VPC フローログや Lambda ログなど) に適用されます。

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

CloudWatch ログと のコストと使用状況を分析する方法、およびコストを削減する方法のベストプラクティスの詳細については CloudWatch、CloudWatch 「請求とコスト」を参照してください。

Amazon CloudWatch Logs の概念

CloudWatch ログの理解と使用の中心となる用語と概念を以下に示します。

ログクラス

CloudWatch ログには 2 つのクラスのロググループがあります。標準ログクラスは、リアルタイムモニタリングが必要なログまたは頻繁にアクセスするログ用のフル機能オプションです。低頻度アクセスログクラスは、アクセス頻度の低いログの低コストオプションです。標準ログクラスの機能のサブセットをサポートします。

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、イベントが発生したときのタイムスタンプと raw イベントメッセージの 2 つのプロパティが含まれています。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。ログストリームが不要になった場合は、aws logs delete-log-stream コマンドを使用して削除できます。

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ー

メトリクスフィルターを使用して、取り込まれたイベントからメトリクスの観測値を抽出し、メトリクス内のデータポイントに変換できますCloudWatch 。メトリクスフィルター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ーはそのログストリームに適用されます。

保持設定

保持設定を使用して、ログイベントをCloudWatch ログに保持する期間を指定できます。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ーと同様に、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。

Amazon CloudWatch Logs の請求とコスト

CloudWatch Logs および CloudWatch のコストと使用状況を分析する方法、およびコストを節約するためのベストプラクティスについては、「CloudWatch の請求とコスト」を参照してください。

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

AWS にサインアップすると、AWS 無料利用枠を利用して、CloudWatch Logs 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格納されたログ (Amazon VPC フローログおよび Lambda ログなど) に適用されます。

ログクラス

CloudWatch ログには、次の 2 つのクラスのロググループがあります。

- CloudWatch Logs Standard ログクラスは、リアルタイムモニタリングまたは頻繁にアクセスするログを必要とするログ用のフル機能オプションです。
- CloudWatch Logs 低頻度アクセスログクラスは、コスト効率よくログを統合するために使用できる新しいログクラスです。このログクラスは、マネージドインジェスト、ストレージ、クロスアカウント CloudWatch ログ分析、GB あたりの取り込み料金の引き下げによる暗号化など、ログ機能のサブセットを提供します。低頻度アクセスログクラスは、アクセス頻度の低いログに対するアドホッククエリと after-the-fact フォレンジック分析に最適です。

> **Note**
>
> 料金については、標準アクセスログクラスと低頻度アクセスログクラスは、取り込みコストのみに異なります。ストレージ料金と CloudWatch Logs Insights 料金は、各ログクラスで同じです。

CloudWatch Logs の料金の詳細については、「Amazon の CloudWatch 料金」を参照してください。

> **Important**
>
> ロググループが作成されると、そのログクラスを変更することはできません。

サポートされている機能

次の表に、各ログクラスの機能を示します。

	規格	低頻度アクセス	
フルマネージドログの取り込みとストレージ	✓	✓	

	規格	低頻度アクセス	
クロスアカウント機能	✓	✓	
による暗号化 AWS KMS	✓	✓	
CloudWatch Logs Insights クエリコマンド	✓	✓ (ほとんどのコマンド - 「」を参照ログクラスでサポートされるコマンド)	
CloudWatch Logs Insights の検出フィールド	✓		
自然言語クエリアシスト	✓		
CloudWatch ログ異常検出	✓		
以前の時間範囲と比較する	✓		
サブスクリプションフィルター	✓		
Amazon S3 へのエクスポート	✓		
GetLogEvents および FilterLogEvents API オペレーション	✓	サポート外。CloudWatch Logs Insights を使用して、低頻度アクセスログクラスのロググループに保存されているログイベントを表示します。	

	規格	低頻度アクセス	
メトリクスフィルター	✓		
Container Insights ログの取り込み	✓		
Lambda Insights ログの取り込み	✓		
マスキングによる機密データの保護	✓		
埋め込みメトリクス形式	✓		

Logs CloudWatch の開始方法

Amazon EC2 インスタンスとオンプレミスサーバーからログを CloudWatch Logs に収集するには、統合 CloudWatch エージェントを使用します。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 CloudWatch エージェントを使用して CloudWatch メトリクスを収集している場合、追加のシステムメトリクスを収集して、ゲストを可視化できます。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 「 エージェントのインストール」を参照してください。

Linux を実行しているサーバーからのログのコレクションのみをサポートする古い CloudWatch Logs エージェントは廃止され、サポートされなくなりました。古い CloudWatch Logs エージェントから統合エージェントへの移行については、「ウィザードを使用して 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。

内容

前提条件

Amazon CloudWatch Logs を使用するには、 AWS アカウントが必要です。 AWS アカウントでは、サービス (Amazon EC2 など) を使用して、ウェブベースのインターフェイスである CloudWatch コンソールで表示できるログを生成できます。さらに、 AWS Command Line Interface () をインストールして設定できますAWS CLI。

にサインアップする AWS アカウント

がない場合は AWS アカウント、次のステップを実行して作成します。

にサインアップするには AWS アカウント

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

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

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

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

管理アクセスを持つユーザーを作成する

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

のセキュリティ保護 AWS アカウントのルートユーザー

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

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

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

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

手順については、「AWS IAM Identity Center ユーザーガイド」の「AWS IAM Identity Centerの有効化」を参照してください。

2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

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

管理アクセス権を持つユーザーとしてサインインする

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

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

追加のユーザーにアクセス権を割り当てる

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

   手順については、「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」を参照してください。

2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

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

Command Line Interface をセットアップする

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

をインストールして設定する方法については AWS CLI、「 ユーザーガイド」の AWS 「 コマンドラインインターフェイスのセットアップAWS Command Line Interface 」を参照してください。

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

統合 CloudWatch エージェントを使用して CloudWatch ログ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は、メトリクスを参照するセクションを無視できます。

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取って、同じログを CloudWatch収集するようにエージェントを設定できます。ウィザードの詳細については、「Amazon ユーザーガイド」の「ウィザードを使用して 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。 CloudWatch

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

> **Important**
>
> 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統合 CloudWatch エージェントが含まれています。ログのみの古いエージェントは非推奨となり、サポートされなくなりました。
> 古いログ専用エージェントから統合エージェントへの移行については、「ウィザードを使用して 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。
> このセクションの残りの部分では、古い CloudWatch Logs エージェントをまだ使用しているお客様向けの使用について説明します。

CloudWatch Logs エージェントを使用すると、Linux または Windows Server を実行している Amazon EC2 インスタンスのログデータと、 からログに記録されたイベントを発行できます AWS CloudTrail。代わりに、 CloudWatch 統合エージェントを使用してログデータを公開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 ユーザーガイド」の「 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 CloudWatch 」を参照してください。 CloudWatch

内容

CloudWatch ログエージェントの前提条件

CloudWatch Logs エージェントには、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。Amazon Linux 2 はサポートされていません
- 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

Important

古いログエージェントは非推奨です。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統合エージェント CloudWatch が含まれています。詳細については、「Logs CloudWatch の開始方法」を参照してください。
古い CloudWatch Logs エージェントから統合エージェントへの移行については、「ウィザードを使用して 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。
古いログエージェントは、Python の 2.6 から 3.5 までのバージョンしかサポートしていません。さらに、古い CloudWatch Logs エージェントはインスタンスメタデータサービスバージョン 2 (IMDSv2) をサポートしていません。サーバーが IMDSv2 を使用している場合は、古い CloudWatch Logs エージェントではなく、新しい統合エージェントを使用する必要があります。
このセクションの残りの部分では、古い CloudWatch Logs エージェントをまだ使用しているお客様向けの使用について説明します。

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 エージェントをまだ使用していない場合は、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、「Logs CloudWatch の開始方法」を参照してください。
さらに、古いエージェントでは、Instance Metadata Service Version 2 (IMDSv2) がサポートされていません。サーバーが IMDSv2 を使用している場合は、古い CloudWatch Logs エージェントではなく、新しい統合エージェントを使用する必要があります。
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ーを使用して、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。

エージェントの使用に加えて、、 CloudWatch Logs SDK AWS CLI、または Logs API を使用して CloudWatch ログデータを発行することもできます。 AWS CLI は、コマンドラインまたはスクリプトでデータを発行するのに最適です。 CloudWatch Logs SDK は、アプリケーションから直接ログデータを発行したり、独自のログ発行アプリケーションを構築したりする場合に最適です。

ステップ 1: Logs の IAM CloudWatch ロールまたは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、「インスタンスへの IAM ロールのアタッチ」を参照してください。

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

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで [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": [
        "*"
```

```
    ]
  }
 ]
}
```

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は、Policy Validator (ポリシー検証) によってレポートされます。
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 [Summary] (概要) を参照して、ポリシーによって付与された許可を確認します。次に、[Create policy] (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 ログをインストールして設定する

CloudWatch Logs エージェントをインストールするプロセスは、Amazon EC2 インスタンスが Amazon Linux、Ubuntu、CentOS、Red Hat のいず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 Amazon Linux インスタンスに CloudWatch Logs をインストールして設定するには

Amazon Linux AMI 2014.09 以降、 CloudWatch Logs エージェントは awslogs パッケージを使用した RPM インストールとして使用できます。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。Logs インストーラを使用する代わりに awslogs パッケージを RPM CloudWatch としてインストールすることで、インスタンスは から定期的なパッケージの更新とパッチを受け取り、CloudWatch Logs エージェントを手動で再インストール AWS 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。そうすると、Logs エージェントが CloudWatch ログを に送信できない設定の問題が発生する可能性があります CloudWatch。

1. Amazon Linux インスタンスに接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。 Amazon EC2

   接続の問題の詳細については、Amazon EC2 ユーザーガイド」の「インスタンスへの接続の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 エージェントリファレンス」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. エージェントがしばらく実行された後、 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が表示されます。

   詳細については、「Logs に送信された CloudWatch ログデータを表示する」を参照してください。

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

Ubuntu Server、CentOS、または Red Hat を実行している AMI を使用している場合は、次の手順を使用してインスタンスに CloudWatch Logs エージェントを手動で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、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。 Amazon EC2

   接続の問題の詳細については、Amazon EC2 ユーザーガイド」の「インスタンスへの接続のトラブルシューティング」を参照してください。

2. 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できません。

   > **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 エージェントをインストールするには、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、または sa-east-1 リージョンを指定します。

> **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 エージェントインストーラは、セットアップ中に特定の情報を必要とします。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。デフォルトは us-east-2 です。これは、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、または 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 エージェントリファレンス」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. エージェントがしばらく実行された後、 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が表示されます。

   詳細については、「Logs に送信された CloudWatch ログデータを表示する」を参照してください。

クイックスタート: 起動時に 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**
>
> このセクションで説明した古い CloudWatch Logs エージェントは、非推奨になる予定です。代わりに、ログとメトリクスの両方を収集できる新しい統合 CloudWatch エージェントを使用することを強くお勧めします。さらに、古い CloudWatch Logs エージェントには Python 3.3 以前が必要であり、これらのバージョンはデフォルトでは新しい EC2 インスタンスにインストールされません。統合 CloudWatch エージェントの詳細については、「 エージェントのインストール」を参照してください CloudWatch 。
> 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

起動時にインスタンスにパラメータ情報を渡すことができる Amazon EC2 の機能である Amazon EC2 ユーザーデータを使用して、そのインスタンスに CloudWatch Logs エージェントをインストールして設定できます。 CloudWatch Logs エージェントのインストールおよび設定情報を Amazon EC2 に渡すには、Amazon S3 バケットなどのネットワークロケーションに設定ファイルを指定します。

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 エージェントリファレンス」を参照してください。

以下は、Amazon Linux 2 のサンプルエージェント設定ファイルです。

```
[general]
state_file = /var/lib/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. IAM コンソール (https://console.aws.amazon.com/iam/) 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、Amazon EC2 ユーザーガイド」の「Amazon EC2 の IAM ポリシーAmazon EC2」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. [ポリシーの作成] 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、[Next Step (次のステップ)] を選択します。
12. [ロールの作成] を選択します。

    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」および「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 ログを有効にするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. [Launch Instance] (インスタンスの起動) を選択します。

    詳細については、Amazon EC2 ユーザーガイド」の「インスタンスの起動」を参照してください。
3. [ステップ 1: Amazon Machine Image (AMI) を選択する] ページで、起動する Linux インスタンスタイプを選択し、[ステップ 2: インスタンスタイプを選択する] 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 AMIs、および Ubuntu と RHEL の AMIs にはすでに cloud-init が含まれていますが、 の CentOS やその他の AMIs には含まれていない AWS Marketplace 場合があります。
4. [ステップ 3: インスタンスの詳細を設定する] ページの [IAM role (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://DOC-EXAMPLE-BUCKET1/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. エージェントがしばらく実行された後、 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が表示されます。

   詳細については、「Logs に送信された CloudWatch ログデータを表示する」を参照してください。

クイックスタート: Windows Server 2016 を実行している Amazon EC2 インスタンスが CloudWatch Logs エージェントを使用してログを Logs CloudWatch に送信できるようにします。

> **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、「Logs CloudWatch の開始方法」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 インスタンスが、古い CloudWatch Logs エージェントを使用してログを Logs CloudWatch に送信できるようにします。

Windows Server 2016 を実行しているインスタンスが CloudWatch ログにログを送信できるようにするには、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。その他の可能な方法の詳細については、「Amazon へのログ、イベント、パフォーマンスカウンターの送信 CloudWatch」を参照してください。

ステップ

- サンプル設定ファイルをダウンロードする
- の JSON ファイルを設定する CloudWatch
- Systems Manager 用 IAM ロールを作成する
- Systems Manager の前提条件を確認する
- インターネットアクセスを確認する

- Systems Manager Run Command を使用して CloudWatch ログを有効にする

サンプル設定ファイルをダウンロードする

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

の JSON ファイルを設定する CloudWatch

送信するログは、設定ファイルで選択を指定 CloudWatch することで決定します。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 ログを有効にする
- ステップ 2: の設定を構成する CloudWatch
- ステップ 3: 送信するデータを設定する
- ステップ 4: フロー制御を設定する
- ステップ 5: JSON コンテンツを保存する

ステップ 1: CloudWatch ログを有効にする

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: の設定を構成する CloudWatch

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、インスタンスはログデータを Logs CloudWatch に送信できます。同じログデータを異なる場所に送信するには、一意の IDs CloudWatchLogs「3」など) と、ID ごとに異なるリージョンを持つセクションを追加します。CloudWatchLogs

ログデータを Logs CloudWatch 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、コンソールのロググループ画面 CloudWatch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指定すると、 CloudWatch Logs によって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、およびその他のログデータを CloudWatch ログに送信できます。

Windows アプリケーションイベントログデータを Logs CloudWatch 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 Logs CloudWatch 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 Logs CloudWatch 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。

他のタイプのイベントログデータを Logs CloudWatch 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。

Windows データのイベントトレースを CloudWatch ログに送信するには

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 (UTF-8 など) 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。サポートされている値の詳細については、MSDN のFileSystemWatcherFilter 「プロパティ」トピック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。このパラメータを空白のままにして、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 ログはデフォルトでローカルタイムゾーンになります。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 Logs CloudWatch 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 (C:\inetpub\logs\LogFiles\W3SVC*n* など) 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 (UTF-8 など) 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。サポートされている値の詳細については、MSDN のFileSystemWatcherFilter 「プロパティ」トピック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。このパラメータを空白のままにして、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 ログはデフォルトでローカルタイムゾーンになります。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。例えば、カスタムログ、ETW ログ、システムログを CloudWatch Logs に送信するには、 Flowsセクション(CustomLogs,ETW,SystemEventLog),CloudWatchLogsに を追加します。

> **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 用 IAM 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。詳細については、AWS Systems Manager ユーザーガイドの「Systems Manager 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、Amazon EC2 ユーザーガイド」の「インスタンスに IAM ロールをアタッチする」を参照してください。

Systems Manager の前提条件を確認する

Systems Manager Run Command を使用して CloudWatch ログとの統合を設定する前に、インスタンスが最小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

ログとイベントデータを に送信するには、Amazon EC2 Windows Server 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インターネットアクセスが必要です CloudWatch。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 を使用して CloudWatch ログを有効にする

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 ログとの統合を設定するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. SSM コンソール (https://console.aws.amazon.com/systems-manager/) を開きます。
3. ナビゲーションペインで、[Run Command] を選択します。
4. [Run a command] を選択します。
5. コマンドドキュメント で、AWS-ConfigureCloudWatch を選択します。
6. ターゲットインスタンス で、 CloudWatch ログ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、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。
7. [Status] で、[Enabled] を選択します。
8. [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。
9. 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 インスタンスがログを CloudWatch Logs に送信できるようにします。

> **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、「Logs CloudWatch の開始方法」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 インスタンスでログを CloudWatch Logs に送信できるようにする

Windows Server 2012 および Windows Server 2008 を実行しているインスタンスが CloudWatch ログにログを送信できるようにするには、次のステップを使用します。

サンプル設定ファイルをダウンロードする

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

の JSON ファイルを設定する CloudWatch

JSON 設定ファイルで選択を指定 CloudWatch して、送信先のログを決定します。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 ログを有効にする
- ステップ 2: の設定を構成する CloudWatch
- ステップ 3: 送信するデータを設定する
- ステップ 4: フロー制御を設定する

ステップ 1: CloudWatch ログを有効にする

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: の設定を構成する CloudWatch

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、インスタンスはログデータを Logs CloudWatch に送信できます。同じログデータを異なる場所

に送信するには、一意の IDs CloudWatchLogs「3」など) と、ID ごとに異なるリージョンを持つセクションを追加します。CloudWatchLogs

ログデータを Logs CloudWatch 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、コンソールの CloudWatch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指定すると、 CloudWatch Logs によって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、およびその他のログデータを CloudWatch ログに送信できます。

Windows アプリケーションイベントログデータを Logs CloudWatch 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 Logs CloudWatch 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 Logs CloudWatch 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。

他のタイプのイベントログデータを Logs CloudWatch 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。

Windows データのイベントトレースを CloudWatch ログに送信するには

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 (UTF-8 など) 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。サポートされている値の詳細については、MSDN のFileSystemWatcherFilter 「プロパティ」トピック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。このパラメータを空白のままにして、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 ログはデフォルトでローカルタイムゾーンになります。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 Logs CloudWatch 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 (C:\inetpub\logs\LogFiles\W3SVC*n* など) 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 (UTF-8 など) 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。サポートされている値の詳細については、MSDN のFileSystemWatcherFilter 「プロパティ」トピック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。このパラメータを空白のままにして、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 ログはデフォルトでローカルタイムゾーンになります。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。例えば、カスタムログ、ETW ログ、システムログを CloudWatch Logs に送信するには、 Flowsセクション(CustomLogs,ETW,SystemEventLog),CloudWatchLogsに を追加します。

> **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。

エージェントを起動する

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 ログにログを送信できるようにするには、EC2Config サービス () を使用しますEC2Config.exe)。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 の使用の詳細については、「Amazon EC2 EC2Config 3.x 以前を使用して を設定する CloudWatch」を参照してください。 Amazon EC2

EC2Config 4.x CloudWatch を使用して 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 サービスコントロールパネルまたは次の PowerShell コマンドを使用して、SSM エージェント (AmazonSSMAgent.exe) を起動または再起動します。

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

SSM エージェントが再起動すると、設定ファイルが検出され、 CloudWatch 統合のためにインスタンスが設定されます。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 の統合を無効にするには、 IsEnabledを に変更falseし、変更を設定ファイルに保存します。

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

CloudWatch Logs エージェントをインストールし、サードパーティーのシステム AWS OpsWorks およびクラウドインフラストラクチャ自動化ツールである Chef を使用してログストリームを作成できます。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

recipes を保存するリポジトリを作成します。 は Git と Subversion AWS OpsWorks をサポートします。または、アーカイブを Amazon S3 に保存できます。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 (CloudWatchLogs-Cookbooks.zip) をダウンロードすることもできます。

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 ログ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て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**
>
> 上記の例では、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、または sa-east-1 *#####*のいずれかに置き換えます。

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 エージェントリファレンス」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/ で AWS OpsWorks コンソールを開きます。

2. OpsWorks ダッシュボード で、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

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

1. IAM コンソール (https://console.aws.amazon.com/iam/) 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、Amazon EC2 ユーザーガイド」の「Amazon EC2 の IAM ポリシーAmazon EC2」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. [ポリシーの作成] 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、コンテンツペインでロール名 で、 AWS OpsWorks スタックで使用され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. [Customer Managed Policies (カスタマーマネージドポリシー)] で、上で作成した IAM ポリシーを選択し、[Attach Policy (ポリシーを添付する)] を選択します。

    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」および「IAM ポリシーを管理する」を参照してください。

ステップ 4: レイヤーを追加する

1. https://console.aws.amazon.com/opsworks/ で 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 タブのレイヤータイプ で、カスタム 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. レシピタブのカスタム Chef レシピ には、セットアップ、設定、デプロイ、デプロイ解除、シャットダウンの複数の見出しがあります。これらは AWS OpsWorks ライフサイクルイベントに対応します。 は、関連するレシピを実行するインスタンスのライフサイクル内のこれらのキーポイントでこれらのイベントを AWS OpsWorks トリガーします。

> **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、インスタンスの起動直後に、このレイヤー内の新しいインスタンスごと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/ で 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 は新しい EC2 インスタンスを起動し、 CloudWatch ログを設定します。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

エージェントがしばらく実行された後、 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が表示されます。

詳細については、「Logs に送信された CloudWatch ログデータを表示する」を参照してください。

CloudWatch Logs エージェントのステータスを報告する

EC2 インスタンスの CloudWatch Logs エージェントのステータスを報告するには、次の手順に従います。

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、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。 Amazon EC2

   接続の問題の詳細については、Amazon EC2 ユーザーガイド」の「インスタンスへの接続の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 エージェントにエラー、警告、または問題がないか確認します。

CloudWatch Logs エージェントを起動する

EC2 インスタンスの CloudWatch Logs エージェントをインストール後に自動的に起動しなかった場合、またはエージェントを停止した場合は、次の手順を使用してエージェントを起動できます。

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、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。 Amazon EC2

   接続の問題の詳細については、Amazon EC2 ユーザーガイド」の「インスタンスへの接続の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 エージェントを停止する

EC2 インスタンスで CloudWatch Logs エージェントを停止するには、次の手順に従います。

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、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。 Amazon EC2

   接続の問題の詳細については、Amazon EC2 ユーザーガイド」の「インスタンスへの接続の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: AWS CloudFormation を使用して CloudWatch ログの使用を開始する

AWS CloudFormation では、JSON 形式で AWS リソースを記述および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 1 つのユニットとして管理できることや、リージョン間で AWS リソースを簡単にレプリケートできることなどがあります。

AWS を使用してプロビジョニングするときは AWS CloudFormation、使用するリソースを AWS 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。を使用して、より豊富な CloudWatch Logs デプロイを設定できます AWS CloudFormation。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。開始方法の詳細については、AWS CloudFormation ユーザーガイドの「AWS CloudFormationの開始方法」を参照してください。

AWS SDK での CloudWatch ログの使用

AWS Software Development Kit (SDKsは、多くの一般的なプログラミング言語で使用できます。各 SDK には、デベロッパーが好みの言語でアプリケーションを簡単に構築できるようにする API、コード例、およびドキュメントが提供されています。

SDK ドキュメント	コード例
AWS SDK for C++	AWS SDK for C++ コード例
AWS CLI	AWS CLI コード例
AWS SDK for Go	AWS SDK for Go コード例
AWS SDK for Java	AWS SDK for Java コード例
AWS SDK for JavaScript	AWS SDK for JavaScript コード例
AWS SDK for Kotlin	AWS SDK for Kotlin コード例
AWS SDK for .NET	AWS SDK for .NET コード例
AWS SDK for PHP	AWS SDK for PHP コード例
AWS Tools for PowerShell	PowerShell コード例のツール
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) コード例
AWS SDK for Ruby	AWS SDK for Ruby コード例
AWS SDK for Rust	AWS SDK for Rust コード例
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コード例
AWS SDK for Swift	AWS SDK for Swift コード例

CloudWatch ログ固有の例については、「」を参照してくださいAWS SDKs を使用した CloudWatch ログのコード例。

可用性の例

必要なものが見つからなかった場合。このページの下側にある [Provide feedback (フィードバックを送信)] リンクから、コードの例をリクエストしてください。

CloudWatch Logs Insights を使用したログデータの分析

CloudWatch Logs Insights を使用すると、Amazon CloudWatch Logs で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。クエリを実行することで、運用上の問題に効率的かつ効果的に対応できます。問題が発生した場合は、Logs Insights CloudWatch を使用して潜在的な原因を特定し、デプロイされた修正を検証できます。

CloudWatch Logs Insights には、シンプルで強力なコマンドがいくつか用意された専用のクエリ言語が含まれています。 CloudWatch Logs Insights には、サンプルクエリ、コマンドの説明、クエリの自動補完、ログフィールド検出が用意されており、使用開始に役立ちます。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、Amazon Route 53、Amazon VPC AWS Lambda AWS CloudTrail、および JSON としてログイベントを発行するアプリケーションまたはカスタムログなどの AWS サービスから、ログ内のフィールドを自動的に検出します。

CloudWatch Logs Insights を使用して、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**
>
> CloudWatch Logs Insights は、ロググループの作成時刻より前のタイムスタンプを持つログイベントにアクセスできません。

自然言語を使用して Logs Insights CloudWatch クエリを作成することもできます。そのためには、どのようなデータを探しているのかを質問したり、具体的に説明したりしてみてください。この AI 支援機能は、プロンプトに基づいてクエリを生成し、クエリの仕組み line-by-line を説明します。詳細については、「自然言語を使用して CloudWatch Logs Insights クエリを生成および更新する」を参照してください。

CloudWatch クロスアカウントオブザーバビリティでモニタリングアカウントとして設定されたアカウントにサインインしている場合は、このモニタリングアカウントにリンクされたソースアカウントのロググループに対して CloudWatch Logs Insights クエリを実行できます。異なるアカウントにある複数のロググループをクエリするクエリを実行できます。詳細については、CloudWatch 「クロスアカウントオブザーバビリティ」を参照してください。

1 つのリクエストで最大 50 個のロググループをクエリできます。クエリが完了していない場合、60 分後にタイムアウトします。クエリ結果は 7 日間利用できます。

作成したクエリは保存できます。そのため、必要なときに複雑なクエリを実行でき、実行するたびにクエリを再作成する必要はありません。

CloudWatch Logs Insights クエリには、クエリされるデータの量に基づいて料金が発生します。詳細については、「Amazon CloudWatch の料金」を参照してください。

> **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ていない場合、現在 CloudWatch コンソールの CloudWatch Logs Insights 部分にアクセスすることはできません。APIs を使用して CloudWatch Logs Insights クエリ機能を使用できます。詳細については、「Amazon Logs API リファレンスStartQuery」の「」を参照してください。CloudWatch

内容

ログクラスでサポートされるコマンド

すべての CloudWatch Logs Insights クエリコマンドは、標準ログクラスのロググループでサポートされています。低頻度アクセスログクラスのロググループは、`pattern`、`diff`、および を除くすべてのクエリコマンドをサポートしますunmask。

開始方法: クエリのチュートリアル

以下のセクションには、 CloudWatch Logs Insights の使用を開始するのに役立つサンプルクエリチュートリアルが含まれています。

トピック

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は、Logs Insights CloudWatch の使用を開始するのに役立ちます。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、ログがすでに Logs CloudWatch に保存されている必要があります。既に CloudWatch ログを使用していて、ロググループとログストリームを設定している場合は、開始する準備が整います。、Amazon Route 53 AWS CloudTrail、Amazon VPC などのサービスを使用していて、それらのサービスのログが Logs に記録されるように設定されている場合も、 CloudWatch 既にログがある可能性があります。ログへの CloudWatch ログの送信の詳細については、「」を参照してくださいLogs CloudWatch の開始方法。

CloudWatch Logs Insights のクエリは、ログイベントから一連のフィールド、またはログイベントに対して実行された数学的集計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

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

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Logs] (ログ)、[Logs Insights] (ログのインサイト) の順に選択します。

   [Logs Insights] (ログのインサイト) ページでは、クエリエディタ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. [Select log group] (ロググループの選択) ドロップダウンから、クエリを実行するロググループを 1 つ以上選択します。

   クロスアカウントオブザーバー CloudWatch バビリティのモニタリングアカウントの場合は、ソースアカウントとモニタリングアカウントのロググループを選択できます。1 つのクエリで複数のアカウントのログを一度にクエリできます。

   ロググループは、ロググループ名、アカウント ID、またはアカウントラベルでフィルタリングできます。

   標準ログクラスでロググループを選択すると、Logs Insights CloudWatch はグループ内のデータフィールドを自動的に検出します。検出されたフィールドを表示するには、ページの右上あたりにある [Fields] (フィールド) メニューを選択します。

   > **Note**
   >
   > 検出されたフィールドは、標準ログクラスのロググループでのみサポートされます。ログクラスの詳細については、「」を参照してくださいログクラス。

4. (オプション) 時間間隔セレクタを使用して、クエリを実行する期間を選択します。

   5～30 分間隔、1 時間、3 時間、12 時間間隔、またはカスタム時間枠を選択できます。

5. [Run] (実行) を選択して結果を表示します。

   このチュートリアルでは、最近追加されたログイベントが 20 件表示されます。

   CloudWatch ログには、ロググループのログイベントの棒グラフが時間の経過とともに表示されます。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も示します。

6. 返されたログイベントのすべてのフィールドを表示するには、番号付きイベントの左にある三角形のドロップダウンアイコンを選択します。

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 で提供されるサンプルクエリでは、 の代わりに `head`または `tail` コマンドを使用します`limit`。これらのコマンドは非推奨であり、`limit` に置き換えられています。ユーザーが記述するすべてのクエリで、`limit` または `head` の代わりに `tail` を使用します。

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

1. クエリエディタで、20 を 50 に変更し、[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。
2. (オプション) 作成したクエリは保存できます。このクエリを保存するには、[保存] を選択します。詳細については、「CloudWatch Logs Insights クエリを保存して再実行する」を参照してください。

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。過去 15 分間に選択したロググループに含まれるログイベントで 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るログイベントの割合を確認するには、ページの右側にあるフィールドを選択します。

   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、その行の左にあるアイコンを選択します。

   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. [Run (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 (例: **`` `error-code`="102" ``**)。

   英数字以外の文字を含むフィールド名にはバックティック文字を使用する必要がありますが、値には必要ありません。値は常に引用符 (") で囲まれます。

CloudWatch Logs Insights には、いくつかのコマンドや正規表現、数学、統計オペレーションのサポートなど、強力なクエリ機能が含まれています。詳細については、「CloudWatch Logs Insights クエリ構文」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

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。このチュートリアルでは、指定したフィールドを含むログイベントの数をカウントするクエリコマンドを実行します。このクエリコマンドは、指定したフィールドの値でグループ化された合計数を返します。集計関数の詳細については、「Amazon Logs ユーザーガイド」の「サポートされているオペレーションと関数」を参照してください。 CloudWatch

集計関数を使用したクエリの実行方法

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Logs] (ログ)、[Logs Insights] (ログのインサイト) の順に選択します。
3. [Select log group] (ロググループの選択) ドロップダウンから、クエリを実行するロググループを 1 つ以上選択します。

クロスアカウントオブザーバビリティのモニタリングアカウントの場合は、ソースアカウントとモニタリングアカウントのロググループを選択できます。1 つのクエリで複数のアカウントのログを一度にクエリできます。

ロググループは、ロググループ名、アカウント ID、またはアカウントラベルでフィルタリングできます。

ロググループを選択すると、Logs Insights CloudWatch は標準クラスのロググループである場合、ロググループのデータフィールドを自動的に検出します。検出されたフィールドを表示するには、ページの右上あたりにある [Fields] (フィールド) メニューを選択します。

4. クエリエディタでデフォルトのクエリを削除し、次のコマンドを入力します。

```
stats count(*) by fieldName
```

5. *fieldName* を [Fields] (フィールド) メニューから検出されたフィールドに置換します。

フィールドメニューはページの右上にあり、 CloudWatch Logs Insights がロググループ内で検出したすべての検出フィールドが表示されます。

6. [Run] (実行) を選択してクエリの結果を表示します。

クエリの結果には、クエリコマンドに一致するロググループ内のレコード数と、指定したフィールドの値でグループ化された合計数が表示されます。

チュートリアル: ログフィールド別にグループ化された視覚化を生成するクエリを実行する

`stats` 関数を使用するクエリを実行して、返された値をログエントリ内の 1 つ以上のフィールドの値別にグループ化すると、結果を棒グラフ、円グラフ、折れ線グラフ、積み上げ面グラフとして表示できます。これにより、ログの傾向をより効率的に視覚化できます。

視覚化用のクエリを実行するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Logs] (ログ)、[Logs Insights] (ログのインサイト) の順に選択します。
3. [Select log group] (ロググループの選択) ドロップダウンから、クエリを実行するロググループを 1 つ以上選択します。

クロスアカウントオブザーバビリティの CloudWatch モニタリングアカウントの場合は、ソースアカウントとモニタリングアカウントのロググループを選択できます。1 つのクエリで複数のアカウントのログを一度にクエリできます。

ロググループは、ロググループ名、アカウント ID、またはアカウントラベルでフィルタリングできます。

4. クエリエディタで、現在の表示内容を削除し、以下の `stats` 関数を入力して、[クエリの実行] を選択します。

```
stats count(*) by @logStream
    | limit 100
```

結果には、各ログストリームのロググループ内のログイベント数が表示されます。結果は 100 行に制限されます。

5. [Visualization (視覚化)] タブを選択します。
6. [線] の横にある矢印を選択し、[バー] を選択します。

棒グラフが表示され、ロググループ内のログストリームごとに棒が表示されます。

チュートリアル: 時系列の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、積み上げ面グラフ、円グラフ、棒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より効率的に視覚化できます。

視覚化用のクエリを実行するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Logs] (ログ)、[Logs Insights] (ログのインサイト) の順に選択します。
3. [Select log group] (ロググループの選択) ドロップダウンから、クエリを実行するロググループを 1 つ以上選択します。

クロスアカウントオブザーバビリティの CloudWatch モニタリングアカウントの場合は、ソースアカウントとモニタリングアカウントのロググループを選択できます。1 つのクエリで複数のアカウントのログを一度にクエリできます。

ロググループは、ロググループ名、アカウント ID、またはアカウントラベルでフィルタリングできます。

4. クエリエディタで、現在の表示内容を削除し、以下の `stats` 関数を入力して、[クエリの実行] を選択します。

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

結果は、ロググループ内のログイベントのうち、30 秒ごとに CloudWatch ログによって受信されたログイベントの数を示します。

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。棒グラフ、円グラフ、積み上げ面グラフに切り替えるには、グラフの右上で [Line (線)] を選択します。

サポートされるログと検出されるフィールド

CloudWatch Logs Insights は、さまざまなログタイプをサポートしています。標準クラスのロググループ Amazon CloudWatch Logs に送信されるログごとに、Logs Insights は 5 CloudWatch つのシステムフィールドを自動的に生成します。

- `@message` は、生の未解析のログイベントを示します。これは、 の `message`フィールドと同等ですInputLogevent。
- `@timestamp` には、ログイベントの `timestamp` フィールドに含まれるイベントタイムスタンプが含まれます。これは、 の `timestamp`フィールドと同等ですInputLogevent。
- `@ingestionTime` には、 CloudWatch ログがログイベントを受信した時刻が含まれます。
- `@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。ログストリームは、生成時と同じプロセスでログをグループ化します。
- `@log` は、 の形式のロググループ識別子です。*`account-id`*`:`*`log-group-name`*これは、複数のロググループにクエリを実行する場合に、特定のイベントが属しているロググループを識別するのに役立ちます。

> **Note**
>
> フィールド検出は、標準ログクラスのロググループでのみサポートされます。ログクラスの詳細については、「」を参照してくださいログクラス。

CloudWatch Logs Insights は、生成するフィールドの先頭に @ 記号を挿入します。

多くのログタイプでは、 CloudWatch Logs はログに含まれる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タイプのログについては、`parse` コマンドを使用して、そのクエリで使用する抽出フィールドを抽出および作成できます。詳細については、「CloudWatch Logs Insights クエリ構文」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、Logs Insights CloudWatch はそのフィールドを先頭に追加@して表示します。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` , `logStatus` , `startTime` , `version`, `action`, `bytes`, `dstAddr`, `dstPort`, `packets`, `protocol`, `srcAddr`, `srcPort`
Route 53 ログ	`@timestamp` , `@logStream` , `@message`, `edgeLocation` , `ednsClientSubnet` , `hostZoneId` , `protocol`, `queryName` , `queryTimestamp` , `queryType` , `resolverIp` , `responseCode` , `version`
Lambda ログ	`@timestamp` , `@logStream` , `@message`, `@requestId` , `@duration`, `@billedDuration` , `@type`, `@maxMemoryUsed` , `@memorySize` Lambda ログ行に X-Ray トレース ID が含まれている場合は、`@xrayTraceId` および `@xraySegmentId` フィールドも含まれます。 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が、各ログイベントに最初に埋め込まれた JSON フラグメントに対してのみ検出します。Lambda ログイベントに複数の JSON フラグメントが含

ログタイプ	検出されるログフィールド
	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」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」を参照してください。
その他のログタイプ	`@timestamp` , `@ingestionTime` , `@logStream` , `@message`, `@log`.

JSON ログのフィールド

CloudWatch Logs Insights では、ドット表記を使用して JSON フィールドを表します。このセクションでは、ドット表記を使用して JSON フィールドにアクセスする方法を、JSON イベントとコードスニペットによる例で説明します。

例: JSON イベント

```
{
    "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": "192.0.2.255",
    "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"
                    }
                }
            ]
        }
    }
}
```

サンプルの JSON イベントには、userIdentity という名前のオブジェクトが含まれています。userIdentity には type という名前のフィールドが含まれます。ドット表記を使用して type の値を表すには、userIdentity.type を使用します。

サンプル JSON イベントには、ネストされたフィールド名と値のリストにフラット化された配列が含まれています。requestParameters.instancesSet の最初の項目である instanceId の値を表すには、requestParameters.instancesSet.items.0.instanceId を使用します。instanceID フィールドの前にある番号 0 は、items フィールドの値の場所を指します。次の例には、JSON ログイベントでネストされた JSON フィールドにアクセスする方法を示すコードスニペットが含まれています。

例: クエリ

```
fields @timestamp, @message
| filter requestParameters.instancesSet.items.0.instanceId="i-abcde123"
| sort @timestamp desc
```

このコードスニペットは、ネストされた JSON フィールド `instanceId` の値にアクセスする `filter` コマンドと共にドット表記を使用するクエリを示します。このクエリは、`instanceId` の値が `"i-abcde123"` に等しいメッセージをフィルタリングし、指定した値を含むログイベントをすべて返します。

> **Note**
>
> CloudWatch Logs Insights は、JSON ログから最大 200 個のログイベントフィールドを抽出できます。抽出されない追加のフィールドについては、`parse` コマンドを使用して、メッセージフィールドの未処理の未解析ログイベントからこれらのフィールドを抽出できます。`parse` コマンドの詳細については、「Amazon ユーザーガイド」の「クエリ構文 CloudWatch 」を参照してください。

CloudWatch Logs Insights クエリ構文

CloudWatch Logs Insights では、クエリ言語を使用してロググループをクエリします。クエリ構文は、一般的な関数、算術演算と比較演算、正規表現など、さまざまな関数とオペレーションをサポートしています。

複数のコマンドを含むクエリを作成するときは、コマンドをパイプ文字 (|) で区切ります。

コメントを含むクエリを作成するときは、コメントをハッシュ文字 (#) で区切ります。

> **Note**
>
> CloudWatch Logs Insights は、さまざまなログタイプのフィールドを自動的に検出し、@ 文字で始まるフィールドを生成します。これらのフィールドの詳細については、「Amazon CloudWatch ユーザーガイド」の「サポートされているログと検出されたフィールド」を参照してください。

次の表で、各コマンドについて簡単に説明します。この表の後に、各コマンドついての詳細な説明と例とを示します。

> **Note**
>
> すべての CloudWatch Logs Insights クエリコマンドは、標準ログクラスのロググループでサポートされています。低頻度アクセスログクラスのロググループは、`pattern`、`diff`、および `unmask` を除くすべてのクエリコマンドをサポートします。

display	クエリ結果に特定のフィールドを表示します。
fields	クエリ結果に特定のフィールドを表示し、クエリで使用するフィールド値を変更したり新しいフィールドを作成したりするときに使用できる関数と演算をサポートします。
filter	クエリをフィルタリングし、1 つ以上の条件に一致するログイベントのみを返します。
pattern	自動的にログデータをパターンにクラスター化します。パターンは、ログフィールド間で繰り返される共有テキスト構造です。 CloudWatch Logs Insights には、ログイベントで見つかったパターンを分析する方法が用意されています。詳細については、「パターン分析」を参照してください。
diff	リクエストされた期間に見つかったログイベントと以前の期間と同じ長さのログイベントを比較し、傾向を調べて特定のログイベントが新しいかどうかを調べることができます。
parse	ログフィールドからデータを抽出し、クエリで処理できる抽出フィールドを作成します。**parse** は、ワイルドカードを使用する glob モードと正規表現の両方をサポートします。
sort	返されたログイベントを昇順 (asc) または降順 (desc) で表示します。
stats	ログフィールドの値を使って集計した統計を算出します。
limit	クエリで返すログイベントの最大数を指定します。**sort** で「上位 20 件」または「最新の 20 件」の結果を返すソートと一緒に使用すると便利です。

dedup	指定したフィールドの特定の値に基づいて、重複した結果を削除します。
unmask	データ保護ポリシーにより一部のコンテンツがマスクされているログイベントの、すべてのコンテンツを表示します。ロググループのデータ保護の詳細については、「機密性の高いログデータをマスキングで保護する」を参照してください。
その他の演算と関数	CloudWatch Logs Insights は、多くの比較、算術、日時、数値、文字列、IP アドレス、および一般的な関数とオペレーションもサポートしています。

以下のセクションでは、 CloudWatch Logs Insights クエリコマンドの詳細について説明します。

トピック

- display
- fields
- フィルター
- pattern
- 差分
- parse
- sort
- stats
- limit
- 重複排除
- マスクを外す
- ブール、比較、数値、日時、その他の関数
- 特殊文字を含むフィールド
- クエリでのエイリアスとコメントの使用

display

`display` を使用して、クエリ結果の特定のフィールドを表示します。

display コマンドは、指定したフィールドのみを表示します。クエリに複数の display コマンドが含まれている場合、クエリ結果には、最後の display コマンドで指定したフィールドのみが表示されます。

例: 1 つのフィールドを表示する

コードスニペットは、解析コマンドを使用して @message からデータを抽出し、抽出フィールド loggingType および loggingMessage を作成するクエリの例を示します。クエリは、loggingType の値が ERROR であるすべてのログイベントを返します。display は、クエリ結果に loggingMessage の値のみを表示します。

```
fields @message
| parse @message "[*] *" as loggingType, loggingMessage
| filter loggingType = "ERROR"
| display loggingMessage
```

> **Tip**
>
> クエリで 1 回だけ display を使用します。クエリで display を 2 回以上使用すると、クエリの結果には、使用されている display コマンドの直近の実行で指定されたフィールドが表示されます。

fields

fields を使用して、クエリ結果の特定のフィールドを表示します。

クエリに複数の fields コマンドが含まれ、display コマンドが含まれていない場合は、結果に、fields コマンドで指定されたすべてのフィールドが表示されます。

例: 特定のフィールドを表示する

以下は、20 個のログイベントを返し、それらを降順で表示するクエリの例です。@timestamp と @message の値がクエリ結果に表示されます。

```
fields @timestamp, @message
| sort @timestamp desc
| limit 20
```

フィールド値を変更したり、クエリで使用できる新しいフィールドを作成したりするため、fields がサポートしている異なる関数や演算を使用するときは、display ではなく fields を使用します。

fields コマンドを as キーワードと共に使用すると、ログイベント内の関数とフィールドを使用して抽出フィールドを作成できます。例えば、fields ispresent as isRes はクエリの残りの部分で使用できる isRes という名前の抽出フィールドを作成します。

フィルター

filter を使用して、1 つ以上の条件に一致するログイベントを取得します。

例: 1 つの条件を使用してログイベントをフィルタリングする

コードスニペットは、range の値が 3000 より大きいすべてのログイベントを返すクエリの例を示します。このクエリは、結果を 20 個のログイベントに制限し、ログイベントを @timestamp 別に降順で並べ替えます。

```
fields @timestamp, @message
| filter (range>3000)
| sort @timestamp desc
| limit 20
```

例: 複数の条件を使用してログイベントをフィルタリングする

キーワード and および or を使用して、複数の条件を組み合わせることができます。

コードスニペットは、range の値が 3000 より大きく、accountId の値が 123456789012 に等しいログイベントを返すクエリの例を示します。このクエリは、結果を 20 個のログイベントに制限し、ログイベントを @timestamp 別に降順で並べ替えます。

```
fields @timestamp, @message
| filter (range>3000 and accountId=123456789012)
| sort @timestamp desc
| limit 20
```

フィルターコマンドの一致と正規表現

フィルターコマンドは、正規表現の使用をサポートします。以下の比較演算子 (=、!=、<、<=、>、>=) とブール演算子 (and、or、および not) を使用できます。

キーワード in を使用して集合要素関係をテストし、配列内の要素をチェックできます。配列の要素をチェックするには、in の後に対象の配列を配置します。ブール演算子 not および in を使用できます。in を使用するクエリを作成して、フィールドに文字列の一致がある口グイベントを返すことができます。フィールドは完全な文字列でなければなりません。例えば、次のコードス二ペットは、フィールド logGroup が完全な文字列 example_group である口グイベントを返すために in を使用するクエリを示しています。

```
fields @timestamp, @message
| filter logGroup in ["example_group"]
```

キーワードフレーズ like および not like を使用して、部分文字列を一致させることができます。正規表現の演算子 =~ を使用して部分文字列を一致させることができます。like および not like で部分文字列を一致させるには、単一引用符または二重引用符で一致させたい部分文字列を囲みます。正規表現パターンは、like および not like と共に使用できます。部分文字列を正規表現の演算子と一致させるには、一致させたい部分文字列をスラッシュで囲みます。次の例には、filter コマンドを使用して部分文字列を照合する方法を示すコードス二ペットが含まれます。

例: 部分文字列の一致

以下の例では、f1 に単語 Exception が含まれている口グイベントを返します。これら 3 つの例すべてで、大文字と小文字が区別されます。

最初の例では、部分文字列を like と一致させます。

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

2 番目の例では、部分文字列を like および正規表現パターンと一致させます。

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

3 番目の例では、部分文字列を正規表現と一致させます。

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

例: 部分文字列をワイルドカードと一致させる

ピリオド記号 (.) を正規表現のワイルドカードとして使用して、部分文字列に一致させることができます。次の例では、クエリは f1 の値が文字列 ServiceLog で始まる一致を返します。

```
fields f1, f2, f3
| filter f1 like /ServiceLog./
```

ピリオド記号 (.*) の後にアスタリスク記号を置いて、できるだけ多くの一致を返す貪欲な量指定子を作成することができます。例えば、次のクエリは f1 の値が文字列 ServiceLog で始まるだけでなく、文字列 ServiceLog も含む一致を返します。

```
fields f1, f2, f3
| filter f1 like /ServiceLog.*/
```

考えられる一致は、次のようにフォーマットされている可能性があります:

- ServiceLogSampleApiLogGroup
- SampleApiLogGroupServiceLog

例: 一致から部分文字列を除外する

次の例は、f1 に Exception という単語が含まれないログイベントを返すクエリを示しています。この例では大文字と小文字が区別されます。

```
fields f1, f2, f3
| filter f1 not like "Exception"
```

例: 大文字と小文字を区別しないパターンで部分文字列を一致させる

大文字と小文字を区別しない部分文字列を、like および正規表現と一致させることができます。次のパラメータ (?i) を、一致させる部分文字列の前に配置します。次の例は、f1 に単語 Exception または exception が含まれるログベントを返すクエリを示しています。

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

pattern

pattern を使用してログデータを自動的にパターンにクラスター化します。

パターンは、ログフィールド間で繰り返される共有テキスト構造です。pattern を使用して、新たな傾向を明らかにし、既知のエラーをモニタリングし、頻繁に発生するログ行やコストの高いログ行を特定できます。 CloudWatch Logs Insights は、ログイベントのパターンを検索してさらに分析するために使用できるコンソールエクスペリエンスも提供します。詳細については、「パターン分析」を参照してください。

pattern コマンドは一般的なパターンを自動的に識別するため、それを開始点として使用してログを検索および分析できます。また、pattern を filter、 parse、または sort コマンドと組み合わせて、より微調整されたクエリでパターンを識別することもできます。

パターンコマンド入力

pattern コマンドでは、@message フィールド、 parse コマンドを使用して作成された抽出フィールド、または 1 つ以上の String 関数を使用して操作された文字列のいずれかの入力が予期されます。

パターンコマンド出力

pattern コマンドは以下の出力を生成します。

- @pattern: ログイベントフィールド間で繰り返される共有テキスト構造。リクエスト ID やタイムスタンプなど、パターン内で異なるフィールドは <*> によって表現されます。例えば、[INFO] Request time: <*> ms はログメッセージ [INFO] Request time: 327 ms の出力候補です。
- @ratio: 選択した期間のログイベントと、識別されたパターンに一致する特定のロググループのログイベントの割合。例えば、選択したロググループと期間のログイベントの半分がパターンと一致する場合、@ratio は 0.50 を返します。
- @sampleCount: 選択した期間のログイベントと、識別されたパターンに一致する特定のロググループのログイベントの数。
- @severityLabel: ログの重要度またはレベル。ログに含まれる情報の種類を示します。Error、Warning、Info、Debug などが該当します。

例

次のコマンドは、選択した時間範囲内の指定されたロググループ内の構造が似ているログを識別し、パターンと数でグループ化します。

```
pattern @message
```

pattern コマンドは filter コマンドと組み合わせて使用できます

```
filter @message like /ERROR/
| pattern @message
```

pattern コマンドは、 parse および sort コマンドと共に使用できます。

```
filter @message like /ERROR/
| parse @message 'Failed to do: *' as cause
| pattern cause
| sort @sampleCount asc
```

差分

リクエストされた期間に見つかったログイベントと、以前の期間と同じ長さのログイベントを比較します。これにより、傾向を探し、特定のログイベントが新しいかどうかを確認できます。

diff コマンドに修飾子を追加して、比較する期間を指定します。

- diff は、現在選択されている時間範囲のログイベントを、直前の時間範囲のログイベントと比較します。
- diff previousDay は、現在選択されている時間範囲のログイベントを、前日の同じ時刻のログイベントと比較します。
- diff previousWeek は、現在選択されている時間範囲のログイベントを、前週の同じ時刻のログイベントと比較します。
- diff previousMonth は、現在選択されている時間範囲のログイベントを、前月と同じ時刻のログイベントと比較します。

詳細については、「(差分) を以前の時間範囲と比較する」を参照してください。

parse

parse を使用して、ログフィールドからデータを抽出し、クエリで処理できる抽出フィールドを作成します。 **parse** は、ワイルドカードを使用する glob モードと正規表現の両方をサポートします。正規表現構文の詳細については、「」を参照してくださいサポートされている正規表現 (regex) 構文。

ネストされた JSON フィールドは正規表現で解析できます。

例: ネストされた JSON フィールドの解析

コードスニペットは、取り込み中にフラット化された JSON ログイベントを解析する方法を示します。

```
{'fieldsA': 'logs', 'fieldsB': [{'fA': 'a1'}, {'fA': 'a2'}]}
```

コードスニペットは、fieldsA および fieldsB の値を抽出し、抽出フィールド fld および array を作成する正規表現を含むクエリを示します。

```
parse @message "'fieldsA': '*', 'fieldsB': ['*']" as fld, array
```

名前付きキャプチャグループ

正規表現で **parse** を使用すると、名前付きキャプチャグループを使用してパターンをフィールドに取り込むことができます。構文は `parse @message (?<Name>pattern).` です。

次の例では、VPC フローログのキャプチャグループを使用して、ENI を NetworkInterface という名前のフィールドに抽出します。

```
parse @message /(?<NetworkInterface>eni-.*?) / display @timestamp, NetworkInterface
```

> **Note**
>
> JSON ログイベントは取り込み中にフラット化されます。現在、ネストされた JSON フィールドを glob 式で解析することはサポートされていません。解析できるのは、200 個以下のログイベントフィールドを含む JSON ログイベントのみです。ネストされた JSON フィールドを解析するときは、クエリ内の正規表現を JSON ログイベントの形式と一致するようにフォーマットする必要があります。

解析コマンドの例

glob 式を使用して、ログフィールド **@message** から、抽出フィールド **@user**、**@method**、**@latency** を抽出し、**@method** および **@user** との一意の組み合わせごとに平均レイテンシーを返します。

```
parse @message "user=*, method:*, latency := *" as @user,
    @method, @latency | stats avg(@latency) by @method,
```

```
    @user
```

正規表現を使用して、ログフィールド **@message** から、フィールド **@user2**、**@method2**、**@latency2** を抽出し、**@method2** および **@user2** との一意の組み合わせごとに平均レイテンシーを返します。

```
parse @message /user=(?<user2>.*?), method:(?<method2>.*?),
    latency := (?<latency2>.*?)/ | stats avg(latency2) by @method2,
    @user2
```

フィールド **loggingTime**、**loggingType**、**loggingMessage** を抽出し、**ERROR** または **INFO** 文字列を含むログイベントをフィルタリングし、**ERROR** 文字列を含むイベントの **loggingMessage** および **loggingType** フィールドのみを表示します。

```
FIELDS @message
    | PARSE @message "* [*] *" as loggingTime, loggingType, loggingMessage
    | FILTER loggingType IN ["ERROR", "INFO"]
    | DISPLAY loggingMessage, loggingType = "ERROR" as isError
```

sort

`sort` を使用して、ログイベントを指定したフィールドごとに昇順 (asc) または降順 (desc) で表示します。これを `limit` コマンドと一緒に使用すれば、「上位 N 件」または「下位 N 件」のクエリを作成できます。

ソートアルゴリズムは、自然ソートの更新バージョンです。昇順でソートする場合、次のロジックが使用されます。

- 数値以外の値はすべて、すべての数値より前になります。数値は数値のみを含む値であり、数値と他の文字を混在させるものではありません。
- 数値以外の値の場合、アルゴリズムは比較のために連続する数値と連続するアルファベット文字を別々のチャンクにグループ化します。数値以外の部分は Unicode 値で順序付けされ、数値部分は最初に長さで順序付けされ、次に数値で順序付けされます。

Unicode の順序の詳細については、「Unicode 文字のリスト」を参照してください。

例えば、昇順でソートした結果を次に示します。

```
!: >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> sorted by unicode order
```

```
#
*%04
0# >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Alphanumeric starting with numbers
5A
111A   >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  Starts with more digits than 5A, so it sorted
 to be later than 5A
2345_
@ >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> 2345 is compared with @ in the unicode order,
@_
A >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Values starting with letters
A9876fghj
a12345hfh
0 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Number values
01
1
2
3
```

降順でソートする場合、ソート結果は逆になります。

例えば、Amazon VPC フローログの次のクエリでは、ホスト間のパケット転送の上位 15 件を検索します。

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

stats

stats を使用して、ログデータを棒グラフ、折れ線グラフ、積み上げ面グラフなどで視覚化します。これにより、ログデータのパターンをより効率的に識別できます。 CloudWatch Logs Insights は、 stats関数と 1 つ以上の集計関数を使用するクエリの視覚化を生成します。

例えば、Route 53 ロググループの次のクエリは、Route 53 レコードの 1 時間あたりのディストリビューションをクエリタイプ別に視覚化して返します。

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

このようなクエリはすべて、棒グラフを生成できます。クエリで bin() 関数を使用して、時間の経過とともにデータを 1 つのフィールドでグループ化する場合、折れ線グラフや積み上げ面グラフも表示できます。

bin 関数では、次の時間単位と略語がサポートされています。複数の文字を含むすべての単位と略語では、s の複数形への追加がサポートされています。したがって、hr および hrs の両方とも時間を指定して機能します。

- millisecond ms msec
- second s sec
- minute m min
- hour h hr
- day d
- week w
- month mo mon
- quarter q qtr
- year y yr

トピック

- 時系列データを視覚化
- フィールド別にグループ化されたログデータを視覚化
- 1 つのクエリで複数の stats コマンドを使用する
- 統計と併用する関数

時系列データを視覚化

時系列の視覚化は、次の特性を持つクエリで機能します。

- 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、「Aggregation Functions in the Stats Command」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。

これらのクエリは、折れ線グラフ、積み上げ面グラフ、棒グラフ、円グラフを生成できます。

例

完全なチュートリアルについては、「the section called “チュートリアル: 時系列の視覚化を生成するクエリを実行する”」を参照してください。

時系列の視覚化で機能するクエリの他の例を以下に示します。

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)
```

折れ線グラフと積み上げ面グラフの制限

ログエントリ情報を集計するが、bin() 関数を使用しないクエリでは、棒グラフを生成できます。ただし、これらのクエリは折れ線グラフや積み上げ面グラフを生成できません。これらのタイプのクエリの詳細については、「the section called “フィールド別にグループ化されたログデータを視覚化”」を参照してください。

フィールド別にグループ化されたログデータを視覚化

stats 関数と 1 つ以上の集計関数を使用するクエリの棒グラフを作成できます。詳細については、「Aggregation Functions in the Stats Command」を参照してください。

視覚化を表示するには、クエリを実行します。次に、[Visualization (視覚化)] タブを選択し、[Line (線)] の横にある矢印を選択して、[Bar (棒)] を選択します。棒グラフでは、視覚化は最大 100 本の棒に制限されています。

例

完全なチュートリアルについては、「the section called “チュートリアル: ログフィールド別にグループ化された視覚化を生成するクエリを実行する”」を参照してください。次の段落では、フィールド別の視覚化のクエリに関する他の例を示します。

次の VPC フローログクエリは、各宛先アドレスについて、セッションごとに転送された平均バイト数を検出します。

```
stats avg(bytes) by dstAddr
```

また、結果の値ごとに複数の棒を含むグラフを生成することもできます。たとえば、次の VPC フローログクエリは、各宛先アドレスについて、セッションごとに転送された平均および最大バイト数を検出します。

```
stats avg(bytes), max(bytes) by dstAddr
```

次のクエリは、各クエリタイプの Amazon Route 53 クエリログの数を検出します。

```
stats count(*) by queryType
```

1 つのクエリで複数の stats コマンドを使用する

1 つのクエリで最大 2 つの `stats` コマンドを使用できます。これにより、最初の集計の出力に対して追加の集計を実行できます。

例: 2 つの **`stats`** コマンドによるクエリ

例えば、次のクエリは、最初に 5 分間のビンの合計トラフィック量を検索し、次に、その 5 分間のビンの中で最大、最低、および平均のトラフィック量を計算します。

```
FIELDS strlen(@message) AS message_length
| STATS sum(message_length)/1024/1024 as logs_mb BY bin(5m)
| STATS max(logs_mb) AS peak_ingest_mb,
        min(logs_mb) AS min_ingest_mb,
        avg(logs_mb) AS avg_ingest_mb
```

例: 複数の stats コマンドを **`filter`**、**`fields`**、**`bin`** などの他の関数と組み合わせます。

1 つのクエリで、2 つの `stats` コマンドを、`filter` や `fields` などの他のコマンドと組み合わせることができます。例えば、次のクエリは、セッション内の異なる IP アドレス数を調べ、クライアントプラットフォームごとにセッション数を調べて、それらの IP アドレスをフィルタリングして、最後にクライアントプラットフォームごとのセッションリクエストの平均を求めます。

```
STATS count_distinct(client_ip) AS session_ips,
      count(*) AS requests BY session_id, client_platform
| FILTER session_ips > 1
| STATS count(*) AS multiple_ip_sessions,
        sum(requests) / count(*) AS avg_session_requests BY client_platform
```

クエリでは、bin と dateceil の関数を複数の stats コマンドと共に使用できます。例えば、次のクエリは、最初にメッセージを 5 分のブロックに結合し、次に 5 分間のブロックを 10 分のブロックに集約して、各 10 分ブロック内の最大、最低、および平均のトラフィック量を計算します。

```
FIELDS strlen(@message) AS message_length
| STATS sum(message_length) / 1024 / 1024 AS logs_mb BY BIN(5m) as @t
| STATS max(logs_mb) AS peak_ingest_mb,
        min(logs_mb) AS min_ingest_mb,
        avg(logs_mb) AS avg_ingest_mb BY dateceil(@t, 10m)
```

注意事項と制限事項

1 つのクエリにつき、最大 2 つの stats コマンドを持つことができます。このクォータは変更できません。

sort または limit コマンドを使用する場合は、2 番目の stats コマンドの後に指定する必要があります。2 番目の stats コマンドより前に置くと、クエリは無効になります。

クエリに 2 つの stats コマンドがある場合、1 つ目の stats 集計が完了するまで、クエリの結果の一部は表示されなくなります。

1 つのクエリにある 2 番目の stats コマンドでは、1 番目の stats コマンドで定義されているフィールドのみを参照できます。例えば、最初の stats 集計以降 @message フィールドが使用できなくなるため、次のクエリは無効です。

```
FIELDS @message
| STATS SUM(Fault) by Operation
# You can only reference `SUM(Fault)` or Operation at this point
| STATS MAX(strlen(@message)) AS MaxMessageSize # Invalid reference to @message
```

最初の stats コマンドの後に参照するフィールドは、すべて最初の stats コマンドで定義する必要があります。

```
STATS sum(x) as sum_x by y, z
| STATS max(sum_x) as max_x by z
# You can only reference `max(sum_x)`, max_x or z at this point
```

Important

この bin 関数は常に @timestamp フィールドを暗黙的に使用します。つまり、2 番目の stats コマンドでは、1 番目の stats コマンドを使用して timestamp フィールドを伝達

しないと bin を使用できないということです。例えば、以下のクエリは有効ではありません。

```
FIELDS strlen(@message) AS message_length
 | STATS sum(message_length) AS ingested_bytes BY @logStream
 | STATS avg(ingested_bytes) BY bin(5m) # Invalid reference to @timestamp field
```

代わりに、最初の stats コマンドで @timestamp フィールドを定義し、次の例のように 2 番目の stats コマンドで dateceil と共にそれを使用できます。

```
FIELDS strlen(@message) AS message_length
 | STATS sum(message_length) AS ingested_bytes, max(@timestamp) as @t BY
 @logStream
 | STATS avg(ingested_bytes) BY dateceil(@t, 5m)
```

統計と併用する関数

CloudWatch Logs Insights は、統計集約関数と統計非集約関数の両方をサポートしています。

statsaggregation 関数は、stats コマンドで使用します。また、他の関数の引数としても使用します。

機能	結果タイプ	説明
avg(fieldName: NumericLogField)	数値	指定したフィールドの値の平均。
count() count(fieldName: LogField)	数値	ログイベントをカウントします。count() (または count(*)) は、クエリによって返されたすべてのイベントをカウントし、count(fieldName) は指定されたフィールド名を含むすべてのレコードをカウントします。
count_distinct(fieldName: LogField)	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。

機能	結果タイプ	説明
max(fieldName: LogField)	LogFieldValue	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: LogField)	LogFieldValue	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: LogFieldValue, percent: number)	LogFieldValue	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。
stddev(fieldName: NumericLogField)	数値	指定されたフィールドの値の標準偏差。
sum(fieldName: NumericLogField)	数値	指定したフィールドの値の合計。

統計非集計関数

非集約関数は、stats コマンドで使用します。また、他の関数の引数としても使用します。

機能	結果タイプ	説明
earliest(fieldName: LogField)	LogField	クエリを実行したうち最も早いタイムスタンプがあるログイベントから fieldName の値を返します。
latest(fieldName: LogField)	LogField	クエリを実行したうち最も遅いタイムスタンプがあるログイベントから fieldName の値を返します。
sortsFirst(fieldName: LogField)	LogField	クエリを実行したログをソートすると最初に来る fieldName の値を返します。

機能	結果タイプ	説明
sortsLast(fieldName: LogField)	LogField	クエリを実行したログをソートすると最後に来る fieldName の値を返します。

limit

limit を使用して、クエリで返すログイベントの数を指定します。

例えば、以下の例は、最新の 25 のログイベントのみを返しています。

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

重複排除

指定したフィールドの特定の値に基づいて、重複した結果を削除するときは、dedup を使用します。dedup は 1 つ以上のフィールドで使用できます。dedup を使ってフィールドを 1 つ指定すると、そのフィールドの一意の値ごとに 1 つのログイベントのみが返されます。複数のフィールドを指定すると、そのフィールドの一意の値の組み合わせごとに 1 つのログイベントが返されます。

重複はソート順に基づいて破棄され、ソート順の最初の結果だけが保持されます。dedup コマンドを実行する前に、結果をソートすることが推奨されます。dedup を実行する前に結果がソートされていない場合は、@timestamp を使用しているデフォルトの降順のソート順が使用されます。

NULL 値は、評価において重複とは見なされません。指定したフィールドのいずれかに NULL 値が含まれるログイベントは保持されます。NULL 値のフィールドを削除するには、isPresent(field) 関数を使用して **filter** を実行します。

dedup コマンドの後のクエリで使用できるクエリコマンドは、limit だけです。

例: **server** という名前のフィールドの、一意の値ごとに、最新のログイベントのみを表示 します。

次の例では、server の一意の値ごとに、最新のイベントの timestamp、server、severity、message フィールドのみを表示します。

```
fields @timestamp, server, severity, message
| sort @timestamp desc
| dedup server
```

CloudWatch Logs Insights クエリのその他のサンプルについては、「」を参照してください一般的なクエリ。

マスクを外す

データ保護ポリシーにより一部のコンテンツがマスクされているログイベントのすべてのコンテンツを表示するには unmask を使用します。このコマンドを使用するには、logs:Unmask アクセス許可が必要です。

ロググループのデータ保護の詳細については、「機密性の高いログデータをマスキングで保護する」を参照してください。

ブール、比較、数値、日時、その他の関数

CloudWatch Logs Insights は、次のセクションで説明するように、クエリで他の多くのオペレーションと関数をサポートしています。

トピック

- 算術演算子
- ブール演算子
- 比較演算子
- 数値演算子
- 日時関数
- 一般関数
- IP アドレス文字列関数
- 文字列関数

算術演算子

算術演算子は、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。算術演算子は、filter コマンドと fields コマンドで使用します。また、他の関数の引数としても使用します。

操作	説明
a + b	加算

操作	説明
`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)

ブール演算子

ブール演算子 **`and`**、**`or`**、および **`not`** を使用します。

> **Note**
> ブール演算子は、TRUE または FALSE の値を返す関数でのみ使用します。

比較演算子

比較演算子は、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。比較オペレーションは、`filter` コマンドで使用します。また、他の関数の引数としても使用します。

演算子	説明
=	Equal
!=	Not equal
<	Less than
>	Greater than
<=	以下
>=	以上

数値演算子

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。数値オペレーションは、filter コマンドと fields コマンドで使用します。また、他の関数の引数としても使用します。

操作	結果タイプ	説明
abs(a: number)	数値	絶対値
ceil(a: number)	数値	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: number)	数値	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: number, ...numbers: number[])	数値	最大値を返します
least(a: number, ...numbers: number[])	数値	最小値を返します
log(a: number)	数値	自然対数
sqrt(a: number)	数値	平方根

日時関数

日時関数

日時関数は、fields コマンドと filter コマンドで使用します。また、他の関数の引数としても使用します。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します。数値と次のいずれかで構成される期間を使用します。

- ms ミリ秒
- s 秒
- m 分間

- h 時間

たとえば、10m は 10 分、1h は 1 時間です。

> **Note**
>
> 日時関数に最も適した時間単位を使用します。 CloudWatch ログは、選択した時間単位に従ってリクエストを上限します。例えば、 を使用するすべてのリクエストの最大値として 60 を上限としますs。したがって、 を指定した場合bin(300s)、 CloudWatch Logs はこれを実際に 60 秒として実装します。これは、60 が 1 分間の秒数であるため、CloudWatch Logs は で 60 を超える数値を使用しませんs。5 分間のバケットを作成するには、bin(5m)代わりに を使用します。
> の上限msは 1000、 sと の上限mは 60、 の上限hは 24 です。

次の表は、クエリコマンドで使用できるさまざまな日付時刻関数のリストを示したものです。このリストには、各関数の結果タイプと説明が記載されています。

> **Tip**
>
> クエリコマンドを作成するときに、時間間隔セレクタを使用してクエリの対象とする期間を選択できます。例えば、5～30 分間隔、1 時間、3 時間、12 時間間隔、またはカスタム時間枠の期間を設定できます。また、特定の日付の間で期間を指定することもできます。

機能	結果タイプ	説明
bin(period: Period)	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。例えば、bin(5m) は @timestamp の値を最も近い 5 分に四捨五入します。 これを使用して、複数のログエントリをクエリにまとめることができます。次の例では、1 時間あたりの例外の数を返します。 `filter @message like /Exception/`

機能	結果タイプ	説明
		<pre> | stats count(*) as exceptionCount by bin(1h) | sort exceptionCount desc</pre> bin 関数では、次の時間単位と略語がサポートされています。複数の文字を含むすべての単位と略語では、s の複数形への追加がサポートされています。したがって、hr および hrs の両方とも時間を指定して機能します。 • millisecond ms msec • second s sec • minute m min • hour h hr • day d • week w • month mo mon • quarter q qtr • year y yr
datefloor(timestamp: Timestamp, period: Period)	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(timestamp : Timestamp, period: Period)	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。
fromMillis(fieldNa me: number)	タイムスタンプ	入力フィールドを Unix エポックからのミリ秒数として解釈し、タイムスタンプに変換します。

機能	結果タイプ	説明
`toMillis(fieldName: Timestamp)`	数値	指定されたフィールドで見つかったタイムスタンプを、Unix エポックからのミリ秒を表す数値に変換します。例えば、`toMillis(@timestamp)` はタイムスタンプを `2022-01-14T13:18:031.000-08:00` から `1642195111000` に変換します。

> **Note**
>
> 現在、 CloudWatch Logs Insights は、人間が読み取れるタイムスタンプを使用したログのフィルタリングをサポートしていません。

一般関数

一般関数

一般関数は、`fields` コマンドと `filter` コマンドで使用します。また、他の関数の引数としても使用します。

機能	結果タイプ	説明
`ispresent(fieldName: LogField)`	ブール値	フィールドが存在する場合は `true` を返します
`coalesce(fieldName: LogField, ...fieldNames: LogField[])`	LogField	リストから最初の null でない値を返します

IP アドレス文字列関数

IP アドレス文字列関数

IP アドレス文字列関数は、`filter` コマンドと `fields` コマンドで使用します。また、他の関数の引数としても使用します。

機能	結果タイプ	説明
`isValidIp(fieldName: string)`	ブール型	フィールドが有効な IPv4 または IPv6 アドレスである場合、`true` を返します。
`isValidIpV4(fieldName: string)`	ブール型	フィールドが有効な IPv4 アドレスである場合、`true` を返します。
`isValidIpV6(fieldName: string)`	ブール型	フィールドが有効な IPv6 アドレスである場合、`true` を返します。
`isIpInSubnet(fieldName: string, subnet: string)`	ブール型	指定された v4 または v6 サブネット内でフィールドが有効な IPv4 または IPv6 アドレスである場合、`true` を返します。サブネットを指定するときは、`192.0.2.0/24` または `2001:db8::/32` などの CIDR 表記を使用します。`192.0.2.0` または `2001:db8::` は CIDR ブロックの開始アドレスです。
`isIpv4InSubnet(fieldName: string, subnet: string)`	ブール値	指定された v4 サブネット内でフィールドが有効な IPv4 アドレスである場合、`true` を返します。サブネットを指定するときは、`192.0.2.0/24` などの CIDR 表記を使用します。`192.0.2.0` は CIDR ブロックの開始アドレスです。
`isIpv6InSubnet(fieldName: string, subnet: string)`	ブール値	指定された v6 サブネット内でフィールドが有効な IPv6 アドレスである場合、`true` を返します。サブネットを指定するときは、`2001:db8::/32` などの CIDR 表記を使用します。`2001:db8::` は CIDR ブロックの開始アドレスです。

文字列関数

文字列関数

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

機能	結果タイプ	説明
isempty(fieldName: string)	数	フィールドが欠落しているか、空の文字列である場合、1を返します。
isblank(fieldName: string)	数	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、1 を返します。
concat(str: string, ...strings: string[])	文字列	複数の文字列を連結します。
ltrim(str: string) ltrim(str: string, trimChars: string)	文字列	関数に 2 番目の文字列引数がない場合、文字列の左側からホワイトスペースを削除します。関数に 2 番目の文字列引数がある場合、ホワイトスペースは削除されません。その場合、strの左から trimChars 個の文字が削除されます。たとえば、ltrim("xyZxyfooxyZ","xyZ") は "fooxyZ" を返します。
rtrim(str: string) rtrim(str: string, trimChars: string)	文字列	関数に 2 番目の文字列引数がない場合、文字列の右側からホワイトスペースを削除します。関数に 2 番目の文字列引数がある場合、ホワイトスペースは削除されません。その場合、strの右から trimChars 個の文

機能	結果タイプ	説明
		字が削除されます。たとえば、rtrim("xyZfooxyxyZ","xyZ") は "xyZfoo" を返します。
trim(str: string) trim(str: string, trimChars: string)	文字列	関数に 2 番目の文字列引数がない場合、文字列の両方の端からホワイトスペースを削除します。関数に 2 番目の文字列引数がある場合、ホワイトスペースは削除されません。その場合、strの両方から trimChars 個の文字が削除されます。たとえば、trim("xyZxyfooxyxyZ","xyZ") は "foo" を返します。
strlen(str: string)	数値	文字列の長さを Unicode コードポイントで返します。
toupper(str: string)	文字列	文字列を大文字に変換します。
tolower(str: string)	文字列	文字列を小文字に変換します。

機能	結果タイプ	説明
substr(str: string, startIndex: number) substr(str: string, startIndex: number, length: number)	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(fieldName: string, searchValue: string, replaceValue: string)	文字列	searchValue の fieldName: string のすべてのインスタンスを replaceValue に置き換えます。 例えば、関数 replace(logGroup,"smoke_test","Smoke") はフィールド logGroup に文字列値 smoke_test を含む口グイベントを検索し、その値を文字列 Smoke に置き換えます。
strcontains(str: string, searchValue: string)	数値	str に searchValue 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。

特殊文字を含むフィールド

フィールドに@記号またはピリオド (.) 以外の英数字以外の文字が含まれている場合は、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`。例えば、ログフィールド foo-bar では英数字以

外の文字であるハイフン (`foo-bar`) が含まれているため、バッククォート (-) で囲む必要があります。

クエリでのエイリアスとコメントの使用

エイリアスを含むクエリを作成します。ログフィールドの名前を変更するために、またはフィールドに値を抽出する場合にエイリアスを使用します。キーワード as を使用して、ログフィールドまたは結果にエイリアスを指定します。クエリ内で複数のエイリアスを使用できます。次のコマンド内でエイリアスを使用できます。

- fields
- parse
- sort
- stats

次の例では、エイリアスを含むクエリを作成する方法を示します。

例

クエリの fields コマンドはエイリアスを含みます。

```
fields @timestamp, @message, accountId as ID
| sort @timestamp desc
| limit 20
```

クエリは、フィールド @timestamp、@message、および accountId の値を返します。結果は降順でソートされ、20 に制限されます。ID の値は、エイリアス accountId の下に一覧表示されます。

例

クエリの sort および stats コマンドはエイリアスを含みます。

```
stats count(*) by duration as time
| sort time desc
```

クエリは、ロググループでフィールド duration が発生した回数をカウントし、結果を降順で並べ替えます。duration の値は、エイリアス time の下に一覧表示されます。

コメントの使用

CloudWatch Logs Insights は、クエリでコメントをサポートします。ハッシュ文字 (#) を使用してコメントを開始します。コメントを使用して、クエリまたはドキュメントクエリの行を無視できます。

例: クエリ

次のクエリを実行すると、2 行目は無視されます。

```
fields @timestamp, @message, accountId
# | filter accountId not like "7983124201998"
| sort @timestamp desc
| limit 20
```

パターン分析

CloudWatch Logs Insights は、機械学習アルゴリズムを使用して、ログをクエリするときにパターンを検索します。パターンは、ログフィールド間で繰り返される共有テキスト構造です。クエリの結果を表示するときは、パターンタブを選択して、結果のサンプルに基づいて CloudWatch Logs が検出したパターンを表示できます。または、クエリに pattern コマンドを追加して、一致するログイベントのセット全体のパターンを分析することもできます。

多数のログイベントをいくつかのパターンに圧縮できるため、パターンは大きなログセットの分析に役立ちます。

次の 3 つのログイベントのサンプルを検討してください。

```
2023-01-01 19:00:01 [INFO] Calling DynamoDB to store for resource id 12342342k124-12345
2023-01-01 19:00:02 [INFO] Calling DynamoDB to store for resource id 324892398123-12345
2023-01-01 19:00:03 [INFO] Calling DynamoDB to store for resource id 3ff231242342-12345
```

前のサンプルでは、3 つのログイベントはすべて 1 つのパターンに従います。

```
<*> <*> [INFO] Calling DynamoDB to store for resource id <*>
```

パターン内のフィールドはトークン と呼ばれます。リクエスト ID やタイムスタンプなど、パターン内で異なるフィールドは動的トークン です。各動的トークンは、 CloudWatch ログに表示される<*>ときに によって表されます。

動的トークンの一般的な例には、エラーコード、タイムスタンプ、リクエスト IDs。トークン値は、動的トークンの特定の値を表します。例えば、動的トークンが HTTP エラーコードを表す場合、トークン値は になります`501`。

パターン検出は、 CloudWatch Logs 異常ディテクターで機能を比較する際にも使用されます。詳細については、「ログ異常検出」および「(差分) を以前の時間範囲と比較する」を参照してください。

パターン分析の開始方法

パターン検出は、Logs Insights CloudWatch クエリで自動的に実行されます。`pattern` コマンドを含まないクエリは、結果にログイベントとパターンの両方を取得します。

クエリに `pattern` コマンドを含めると、一致するログイベントのセット全体に対してパターン分析が実行されます。これにより、より正確なパターン結果が得られますが、`pattern`コマンドを使用すると raw ログイベントは返されません。クエリに が含まれていない場合`pattern`、パターン結果は、最初に返された 1000 個のログイベント、またはクエリで使用した制限値のいずれかに基づきます。クエリ`pattern`に を含めると、パターン タブに表示される結果は、クエリに一致するすべてのログイベントから取得されます。

CloudWatch Logs Insights でパターン分析を開始するには

1. https://console.aws.amazon.com/cloudwatch/ で 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 、ログインサイト を選択します。

   [Logs Insights] (ログのインサイト) ページでは、クエリエディタ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. クエリボックスの`| limit 20`行を削除して、クエリが次のようになります。

   ```
   fields @timestamp, @message, @logStream, @log
   | sort @timestamp desc
   ```
4. ロググループを選択 (複数可) ドロップダウンで、クエリするロググループを 1 つ以上選択します。
5. (オプション) 時間間隔セレクタを使用して、クエリを実行する期間を選択します。

   5 分間隔と 30 分間隔、1 時間間隔、3 時間間隔、12 時間間隔、またはカスタム時間枠から選択できます。
6. クエリの実行 を選択してクエリを開始します。

クエリの実行が完了すると、ログタブにクエリによって返されるログイベントのテーブルが表示されます。上の表は、クエリに一致したレコードの数に関するメッセージです。これは、一致した 71,101 件のレコードのうち 1,000 件を表示するのと同様です。

7. パターンタブを選択します。
8. テーブルにクエリで見つかったパターンが表示されるようになりました。クエリに pattern コマンドが含まれていなかったため、このタブには、ログタブの テーブルに表示された 1000 件のログイベントで検出されたパターンのみが表示されます。

   パターンごとに、次の情報が表示されます。

   - パターン 。各動的トークンは として表示されます<*>。
   - イベント数 は、クエリされたログイベントにパターンが表示された回数です。イベント数列の見出しを選択して、パターンを頻度でソートします。
   - イベント比率 は、このパターンを含むクエリされたログイベントの割合です。
   - 重要度タイプ 。次のいずれかになります。
     - パターンに「エラー」という単語が含まれている場合、エラー。
     - パターンに「Warn」という単語が含まれているが、「エラー」が含まれていない場合は WARN。
     - パターンに警告またはエラー が含まれていない場合の INFO。

   重要度情報列の見出しを選択して、パターンを重要度別にソートします。
9. 次に、クエリを変更します。クエリの| sort @timestamp desc行を に置き換えて| pattern @message、完全なクエリを次のようにします。

```
fields @timestamp, @message, @logStream, @log
| pattern @message
```

10. [Run query] (クエリの実行) を選択します。

    クエリが終了すると、ログタブに結果はありません。ただし、クエリされたログイベントの合計数によっては、パターンタブに表示されるパターンの数が多い可能性があります。
11. クエリpatternに を含めたかどうかにかかわらず、クエリが返すパターンをさらに検査できます。そのためには、いずれかのパターンについて Inspect 列のアイコンを選択します。

    パターン検査ペインが表示され、以下が表示されます。

- パターン 。パターン内のトークンを選択して、そのトークンの値を分析します。
- クエリされた時間範囲におけるパターンの出現回数を示すヒストグラム。これにより、パターンの突然の出現の増加など、興味深い傾向を特定できます。
- ログサンプルタブには、選択したパターンに一致するログイベントがいくつか表示されます。
- トークン値 タブには、選択した動的トークンの値が表示されます。

> **Note**
>
> トークンごとに最大 10 個のトークン値がキャプチャされます。トークン数は正確ではない可能性があります。 CloudWatch Logs は確率カウンターを使用して、絶対値ではなくトークン数を生成します。

- 関連パターンタブには、検査するパターンとほぼ同じ時間に頻繁に発生する他のパターンが表示されます。例えば、ERRORメッセージのパターンに通常、追加の詳細とともに INFOとマークされた別のログイベントが伴う場合、そのパターンがここに表示されます。

pattern コマンドの詳細

このセクションでは、patternコマンドとその使用について詳しく説明します。

- 前のチュートリアルでは、 コマンドのpattern後にsortコマンドが含まれている場合、クエリが有効ではないpatternため、 コマンドを追加したときに sort コマンドを削除しました。pattern の前に があることは有効ですsort。

  pattern 構文の詳細については、「」を参照してくださいpattern。
- クエリpatternで を使用する場合、 はpatternコマンドで選択されたフィールドのいずれか@messageである必要があります。
- filter コマンドの前に pattern コマンドを含めると、フィルタリングされたログイベントのセットのみがパターン分析の入力として使用されます。
- parse コマンドから派生したフィールドなど、特定のフィールドのパターン結果を表示するには、 を使用しますpattern @fieldname。
- stats コマンドを使用したクエリなど、ログ以外の出力を持つクエリは、パターン結果を返しません。

(差分) を以前の時間範囲と比較する

CloudWatch Logs Insights を使用して、ログイベントの変更を経時的に比較できます。最近の時間範囲で取り込まれたログイベントと、直前の期間のログを比較できます。または、同様の過去の期間と比較することもできます。これは、ログのエラーが最近発生したか、すでに発生しているかを確認するのに役立ち、他の傾向を見つけるのに役立ちます。

比較クエリは、未加工のログイベントではなく、結果のパターンのみを返します。返されるパターンは、ログイベントの傾向と変化を時間の経過とともにすばやく確認するのに役立ちます。比較クエリを実行してパターン結果を取得したら、関心のあるパターンのサンプル raw ログイベントを表示できます。ログパターンの詳細については、「」を参照してくださいパターン分析。

比較クエリを実行すると、選択した元のクエリ期間と比較期間の 2 つの異なる期間に対してクエリが分析されます。比較期間の長さは常に元のクエリ期間と同じです。比較のデフォルトの時間間隔は次のとおりです。

- 前の期間 — クエリ期間の直前の期間と比較します。
- 前日 — クエリ期間の 1 日前の期間と比較します。
- 前の週 — クエリ期間の 1 週間前の期間と比較します。
- 前月 — クエリ期間の 1 か月前の期間と比較します。

> **Note**
>
> 比較を使用するクエリには、結合された時間範囲で単一の CloudWatch Logs Insights クエリを実行する場合と同様の料金が発生します。詳細については、「Amazon CloudWatch の料金」を参照してください。

比較クエリを実行するには

1. https://console.aws.amazon.com/cloudwatch/ で 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 、ログインサイト を選択します。

   デフォルトのクエリがクエリボックスに表示されます。
3. デフォルトのクエリを保持するか、別のクエリを入力します。
4. ロググループを選択 (複数可) ドロップダウンで、クエリするロググループを 1 つ以上選択します。

5. (オプション) 時間間隔セレクタを使用して、クエリを実行する期間を選択します。デフォルトのクエリは、過去 1 時間のログデータ用です。
6. 時間範囲セレクタで、比較 を選択します。次に、元のログを比較する前の期間を選択し、適用を選択します。
7. [Run query] (クエリの実行) を選択します。

   クエリが比較期間からデータを取得するようにするには、 `diff` コマンドがクエリに追加されます。
8. パターンタブを選択すると、結果が表示されます。

   テーブルには、次の情報が表示されます。

   - 各パターン 。パターンの可変部分は動的トークン記号 に置き換えられます`<*>`。詳細については、「パターン分析」を参照してください。
   - イベント数は、元のより現在の期間の、そのパターンを持つログイベントの数です。
   - 差分イベント数は、現在の期間と比較期間の一致するログイベントの数の差です。正の異なるは、現在の期間にそのようなイベントが多いことを意味します。
   - 差分の説明は、現在の期間と比較期間の間のパターンの変化を簡潔にまとめたものです。
   - 重要度タイプは、、、 などのログイベントで見つかった単語に基づいて`FATAL`ERROR、このパターンでログイベントの考えられる重要度です`WARN`。
9. リスト内のパターンの 1 つをさらに検査するには、Inspect 列のアイコンを選択してパターンの 1 つを探します。

   パターン検査ペインが表示され、以下が表示されます。

   - パターン 。パターン内のトークンを選択して、そのトークンの値を分析します。
   - クエリされた時間範囲におけるパターンの出現回数を示すヒストグラム。これにより、パターンの突然の出現の増加など、興味深い傾向を特定できます。
   - ログサンプルタブには、選択したパターンに一致するログイベントがいくつか表示されます。
   - トークン値 タブには、選択した動的トークンの値が表示されます。

   > **Note**
   >
   > トークンごとに最大 10 個のトークン値がキャプチャされます。トークン数は正確ではない可能性があります。 CloudWatch Logs は確率カウンターを使用して、絶対値ではなくトークン数を生成します。

- 関連パターンタブには、検査するパターンとほぼ同じ時間に頻繁に発生する他のパターンが表示されます。例えば、ERRORメッセージのパターンに通常、追加の詳細が INFOとマークされた別のログイベントが伴う場合、そのパターンがここに表示されます。

サンプルクエリ

このセクションでは、CloudWatch コンソール で実行できる一般的なクエリコマンドと便利なクエリコマンドのリストを示します。クエリコマンドの実行方法については、「Amazon Logs ユーザーガイド」の「チュートリアル: サンプルクエリの実行と変更」を参照してください。 CloudWatch

クエリ構文の詳細については、「」を参照してくださいCloudWatch Logs Insights クエリ構文。

トピック

一般的なクエリ

最近追加された 25 件のログイベントを検索します。

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

1 時間あたりの例外数のリストを表示します。

```
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します。

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

server フィールドの一意の値ごとに最新のログイベントを表示します。

```
fields @timestamp, server, severity, message
| sort @timestamp asc
| dedup server
```

各 **severity** タイプの、**server** フィールドの一意の値ごとに最新のログイベントを表示します。

```
fields @timestamp, server, severity, message
| sort @timestamp desc
| dedup server, severity
```

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します。

```
filter @type = "REPORT"
    | stats max(@memorySize / 1000 / 1000) as provisonedMemoryMB,
        min(@maxMemoryUsed / 1000 / 1000) as smallestMemoryRequestMB,
        avg(@maxMemoryUsed / 1000 / 1000) as avgMemoryUsedMB,
        max(@maxMemoryUsed / 1000 / 1000) as maxMemoryUsedMB,
        provisonedMemoryMB - maxMemoryUsedMB as overProvisionedMB
```

レイテンシーレポートを作成します。

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

遅い関数呼び出しを検索し、再試行やクライアント側コードが原因で発生する可能性のある重複リクエストを削除します。このクエリでは、**@duration** はミリ秒単位です。

```
fields @timestamp, @requestId, @message, @logStream
| filter @type = "REPORT" and @duration > 1000
| sort @timestamp desc
| dedup @requestId
| limit 20
```

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検索します:

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

特定のサブネットにおけるホストの上位 15 バイトの転送を検索します。

```
filter isIpv4InSubnet(srcAddr, "192.0.2.0/24")
    | 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 を使用する IP アドレスを検索します。

```
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 アドレスを検索します。

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

接続のたびに 1 つのレコードを検索し、ネットワークの接続問題の解決を促します。

```
fields @timestamp, srcAddr, dstAddr, srcPort, dstPort, protocol, bytes
| filter logStream = 'vpc-flow-logs' and interfaceId = 'eni-0123456789abcdef0'
| sort @timestamp desc
```

```
| dedup srcAddr, dstAddr, srcPort, dstPort, protocol
| limit 20
```

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検索します。

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

リクエスト数が最大である 10 件の DNS リゾルバーを検索します。

```
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検索します。

```
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、 AWS リージョン別のログエントリ数を検索します。

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

特定の AWS リージョンで開始または停止された Amazon EC2 ホストを検索します。

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

新しく作成された IAM ユーザーの AWS リージョン、ユーザー名ARNs を検索します。

```
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検索します。

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

TLS 1.0 または 1.1 が使用されたログエントリを検索します。

```
filter tlsDetails.tlsVersion in [ "TLSv1", "TLSv1.1" ]
| stats count(*) as numOutdatedTlsCalls by userIdentity.accountId, recipientAccountId,
 eventSource, eventName, awsRegion, tlsDetails.tlsVersion, tlsDetails.cipherSuite,
 userAgent
| sort eventSource, eventName, awsRegion, tlsDetails.tlsVersion
```

TLS バージョン 1.0 または 1.1 を使用したサービスごとの呼び出し数を検索します。

```
filter tlsDetails.tlsVersion in [ "TLSv1", "TLSv1.1" ]
| stats count(*) as numOutdatedTlsCalls by eventSource
| sort numOutdatedTlsCalls desc
```

のクエリ Amazon API Gateway

最新の 4XX エラーを 10 件検索します。

```
fields @timestamp, status, ip, path, httpMethod
| filter status>=400 and status<=499
| sort @timestamp desc
| limit 10
```

Amazon API Gateway アクセスロググループで最も実行時間が長い Amazon API Gateway リクエストを 10 件特定する

```
fields @timestamp, status, ip, path, httpMethod, responseLatency
| sort responseLatency desc
| limit 10
```

Amazon API Gateway アクセスロググループで最も人気のある API パスのリストを返します。

```
stats count(*) as requestCount by path
| sort requestCount desc
| limit 10
```

Amazon API Gateway アクセスロググループの統合レイテンシーレポートを作成する

```
filter status=200
| stats avg(integrationLatency), max(integrationLatency),
min(integrationLatency) by bin(1m)
```

NAT ゲートウェイに対するクエリ

AWS 請求額が通常よりも高い場合は、Logs Insights CloudWatch を使用して上位の寄稿者を見つけることができます。次のクエリコマンドの詳細については、 AWS プレミアムサポートページの「VPC の NAT ゲートウェイを通過するトラフィックの上位の寄稿者を見つけるにはどうすればよいですか？」を参照してください。

> **Note**
>
> 次のクエリコマンドの「x.x.x.x」の部分をお使いの NAT ゲートウェイのプライベート IP に置き換え、「y.y」を VPC CIDR アドレス範囲の第 1 および 第 2 オクテットの値に置き換えます。

NAT ゲートウェイ経由で最も多くのトラフィックを送信しているインスタンスを検索します。

```
filter (dstAddr like 'x.x.x.x' and srcAddr like 'y.y.')
| stats sum(bytes) as bytesTransferred by srcAddr, dstAddr
| sort bytesTransferred desc
| limit 10
```

NAT ゲートウェイ内のインスタンスとの間で送受信されているトラフィックを特定します。

```
filter (dstAddr like 'x.x.x.x' and srcAddr like 'y.y.') or (srcAddr like 'xxx.xx.xx.xx'
 and dstAddr like 'y.y.')
| stats sum(bytes) as bytesTransferred by srcAddr, dstAddr
| sort bytesTransferred desc
| limit 10
```

VPC 内のインスタンスがアップロードとダウンロードの通信で最も頻繁に使用している、インターネット上の送信先を特定します。

アップロードの場合

```
filter (srcAddr like 'x.x.x.x' and dstAddr not like 'y.y.')
| stats sum(bytes) as bytesTransferred by srcAddr, dstAddr
| sort bytesTransferred desc
| limit 10
```

ダウンロードの場合

```
filter (dstAddr like 'x.x.x.x' and srcAddr not like 'y.y.')
| stats sum(bytes) as bytesTransferred by srcAddr, dstAddr
| sort bytesTransferred desc
| limit 10
```

Apache サーバーのログに対するクエリ

CloudWatch Logs Insights を使用して Apache サーバーログをクエリできます。以下のクエリの詳細については、 AWS クラウド運用と移行ブログの CloudWatch 「Logs Insights を使用した Apache サーバーログの簡素化」を参照してください。

アクセスログを確認してアプリケーションの /admin パスでトラフィックをチェックできるよう、最も関連性の高いフィールドを検索します。

```
fields @timestamp, remoteIP, request, status, filename| sort @timestamp desc
| filter filename="/var/www/html/admin"
| limit 20
```

メインページにアクセスした際のステータスコードが「200」(成功) になっている箇所を探し、一意の GET リクエストの数を見つけます。

```
fields @timestamp, remoteIP, method, status
| filter status="200" and referrer= http://34.250.27.141/ and method= "GET"
| stats count_distinct(remoteIP) as UniqueVisits
| limit 10
```

Apache サービスが再起動した回数を確認します。

```
fields @timestamp, function, process, message
| filter message like "resuming normal operations"
| sort @timestamp desc
| limit 20
```

Amazon のクエリ EventBridge

EventBridge イベント詳細タイプ別にグループ化されたイベントの数を取得する

```
fields @timestamp, @message
| stats count(*) as numberOfEvents by `detail-type`
| sort numberOfEvents desc
```

解析コマンドの例

glob 式を使用して、ログフィールド **@message** から、抽出フィールド **@user**、**@method**、**@latency** を抽出し、**@method** および **@user** との一意の組み合わせごとに平均レイテンシーを返します。

```
parse @message "user=*, method:*, latency := *" as @user,
    @method, @latency | stats avg(@latency) by @method,
    @user
```

正規表現を使用して、ログフィールド **@message** から、フィールド **@user2**、**@method2**、**@latency2** を抽出し、**@method2** および **@user2** との一意の組み合わせごとに平均レイテンシーを返します。

```
parse @message /user=(?<user2>.*?), method:(?<method2>.*?),
    latency := (?<latency2>.*?)/ | stats avg(latency2) by @method2,
    @user2
```

フィールド **loggingTime**、**loggingType**、**loggingMessage** を抽出し、**ERROR** または **INFO** 文字列を含むログイベントをフィルタリングし、**ERROR** 文字列を含むイベントの **loggingMessage** および **loggingType** フィールドのみを表示します。

```
FIELDS @message
    | PARSE @message "* [*] *" as loggingTime, loggingType, loggingMessage
    | FILTER loggingType IN ["ERROR", "INFO"]
```

```
    | DISPLAY loggingMessage, loggingType = "ERROR" as isError
```

グラフでログデータを視覚化する

棒グラフ、折れ線グラフ、積み上げ面グラフなどの視覚化を使用して、ログデータのパターンをより効率的に識別できます。 CloudWatch Logs Insights は、 `stats`関数と 1 つ以上の集計関数を使用するクエリの視覚化を生成します。詳細については、「stats」を参照してください。

CloudWatch Logs Insights クエリを保存して再実行する

作成したクエリは、後で再度実行できるように保存できます。クエリはフォルダ構造に保存されるため、整理できます。アカウントごとに、リージョンあたり最大 1000 件保存できます。

クエリを保存するには、アクセス許可 `logs:PutQueryDefinition` を持つロールにログインする必要があります。保存されたクエリのリストを表示するには、アクセス許可 `logs:DescribeQueryDefinitions` を持つロールにログインする必要があります。

クエリを保存するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Logs] (ログ)、[Logs Insights] (ログのインサイト) の順に選択します。
3. クエリエディタで、クエリを作成します。
4. [Save] を選択します。

   保存ボタンが表示されない場合は、 CloudWatch ログコンソールの新しい設計に変更する必要があります。そのためには、次の操作を行います。

   a. ナビゲーションペインで、[Log groups] (ロググループ) を選択します。
   b. [新しいデザインを試す] を選択します。
   c. ナビゲーションペインで [Insights] を選択し、この手順のステップ 3 に戻ります。
5. クエリの名前を入力します。
6. (オプション) クエリを保存するフォルダを選択します。[新規作成] を選択して、フォルダを作成します。新しいフォルダを作成した場合、フォルダ名にスラッシュ (/) 文字を使用してフォルダ構造を定義できます。たとえば、新しいフォルダに **`folder-level-1/folder-level-2`** という名前を付けると、**`folder-level-1`** という最上位フォルダが作成され、そのフォルダ内に

folder-level-2 という別のフォルダが作成されます。クエリは **folder-level-2** に保存されます。

7. (オプション) クエリのロググループまたはクエリテキストを変更します。
8. [Save] を選択します。

> **Tip**
>
> PutQueryDefinition で保存したクエリー用のフォルダを作成することができます。保存したクエリ用のフォルダを作成するには、スラッシュ (/) を使用して、目的のクエリ名の前に目的のフォルダ名を付加します: <*folder-name*>/<*query-name*>。このアクションの詳細については、「」を参照してくださいPutQueryDefinition。

保存されたクエリを実行するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Logs] (ログ)、[Logs Insights] (ログのインサイト) の順に選択します。
3. 右側の [クエリ] を選択します。
4. [保存されたクエリ] リストからクエリを選択します。クエリエディタに表示されます。
5. [Run (実行)] を選択します。

保存したクエリの新しいバージョンを保存するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Logs] (ログ)、[Logs Insights] (ログのインサイト) の順に選択します。
3. 右側の [クエリ] を選択します。
4. [保存されたクエリ] リストからクエリを選択します。クエリエディタに表示されます。
5. クエリを修正します。作業を確認するために実行する必要がある場合は、[クエリの実行] を選択します。
6. 新しいバージョンを保存する準備ができたら、[アクション]、[名前を付けて保存] の順に選択します。
7. クエリの名前を入力します。

8. (オプション) クエリを保存するフォルダを選択します。[新規作成] を選択して、フォルダを作成します。新しいフォルダを作成した場合、フォルダ名にスラッシュ (/) 文字を使用してフォルダ構造を定義できます。たとえば、新しいフォルダに **folder-level-1/folder-level-2** という名前を付けると、**folder-level-1** という最上位フォルダが作成され、そのフォルダ内に **folder-level-2** という別のフォルダが作成されます。クエリは **folder-level-2** に保存されます。
9. (オプション) クエリのロググループまたはクエリテキストを変更します。
10. [Save] を選択します。

クエリを削除するには、`logs:DeleteQueryDefinition` アクセス許可を持つロールにログインする必要があります。

保存したクエリを編集または削除するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Logs] (ログ)、[Logs Insights] (ログのインサイト) の順に選択します。
3. 右側の [クエリ] を選択します。
4. [保存されたクエリ] リストからクエリを選択します。クエリエディタに表示されます。
5. [アクション]、[編集]、または [アクション]、[削除] を選択します。

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

クエリを実行したら、クエリを CloudWatch ダッシュボードに追加するか、結果をクリップボードにコピーできます。

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再実行されます。これらのクエリは、30 件の同時 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/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Logs] (ログ)、[Logs Insights] (ログのインサイト) の順に選択します。

3. 1 つ以上のロググループを選択し、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、クエリ結果用のダッシュボードを作成します。
6. クエリ結果に使用するウィジェットの種類を選択します。
7. ウィジェットの名前を入力します。
8. [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か、クエリ結果をダウンロードするには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Logs] (ログ)、[Logs Insights] (ログのインサイト) の順に選択します。
3. 1 つ以上のロググループを選択し、クエリを実行します。
4. [結果のエクスポート] を選択し、必要なオプションを選択します。

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されたクエリを含め、1 アカウントあたり 30 件の CloudWatch Logs Insights クエリを同時に実行できます。

最近のクエリ履歴を表示するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Logs] (ログ)、[Logs Insights] (ログのインサイト) の順に選択します。
3. Logs コンソールの新しい設計を使用している場合は、履歴 CloudWatch を選択します。古いデザインを使用している場合は、[アクション]、[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。クエリを選択して [実行] を選択すると、それらのいずれかを再度実行できます。

ステータス で、現在実行中のクエリの進行中が CloudWatch ログに表示されます。

でクエリ結果を暗号化する AWS Key Management Service

デフォルトでは、 CloudWatch Logs は、デフォルトの CloudWatch Logs サーバー側の暗号化方法を使用して Logs Insights CloudWatch クエリの保存結果を暗号化します。代わりに、 AWS KMS キーを使用してこれらの結果を暗号化することもできます。 AWS KMS キーを暗号化結果に関連付けると、 CloudWatch Logs はそのキーを使用して、アカウント内のすべてのクエリの保存された結果を暗号化します。

後でクエリ結果からキーの関連付けを解除すると、 CloudWatch Logs は後のクエリのデフォルトの暗号化方法に戻ります。ただし、キーが関連付けられている間に実行されたクエリは、そのキーで暗号化されます。 CloudWatch ログは、引き続きキーを参照できるため、KMS キーの関連付けが解除された後も、 CloudWatch これらの結果を返すことができます。ただし、後でキーが無効になると、CloudWatch ログはそのキーで暗号化されたクエリ結果を読み取ることができません。

> **Important**
>
> CloudWatch ログは対称 KMS キーのみをサポートします。クエリ結果の暗号化に非対称キーを使用しないでください。詳細については、「対称キーと非対称キーの使用」を参照してください。

制限

- 以下の手順を実行するには、`kms:CreateKey`、`kms:GetKeyPolicy`、および `kms:PutKeyPolicy` アクセス許可が必要です。
- キーとクエリ結果を関連付けた後、または関連付けを解除した後、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。
- 関連付けられたキーへの CloudWatch Logs アクセスを取り消したり、関連付けられた KMS キーを削除したりすると、 CloudWatch Logs 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キーを関連付けることはできません。 AWS CLI または CloudWatch Logs API を使用する必要があります。

ステップ 1: を作成する AWS KMS key

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

```
aws kms create-key
```

出力には、キーのキー 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": {
        "Origin": "AWS_KMS",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Description": "",
        "KeyManager": "CUSTOMER",
        "Enabled": true,
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "CreationDate": 1478910250.94,
        "Arn": "arn:aws:kms:us-west-2:123456789012:key/6f815f63-e628-448c-8251-
e40cb0d29f59",
        "AWSAccountId": "123456789012",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

ステップ 2: KMS キーでアクセス許可を設定する

デフォルトでは、すべての KMS キーはプライベートです。リソースの所有者のみがその CMK を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キーへのアクセス許可を付与することができます。このステップでは、 キーを使用するアクセス許可を CloudWatch Logs サービスプリンシパルに付与します。このサービスプリンシパルは、キーが保存されているのと同じ AWS リージョンに存在する必要があります。

ベストプラクティスとして、キーの使用を指定した AWS アカウントのみに制限することをお勧めします。

まず、次のget-key-policyコマンドpolicy.jsonを使用して、KMS キーのデフォルトポリシーを 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、以下のいずれかのステートメントから太字のセクションを追加します。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。これらのステートメントでは、 Conditionセクションを使用して AWS KMS キーのセキュリティを強化します。詳細については、「AWS KMS キーと暗号化コンテキスト」を参照してください。

この例の Conditionセクションでは、 AWS KMS キーの使用を、指定されたアカウントの CloudWatch Logs Insights クエリ結果に制限します。

```
{
 "Version": "2012-10-17",
    "Id": "key-default-1",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::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": "*",
            "Condition": {
                "ArnEquals": {
```

```
                    "aws:SourceArn": "arn:aws:logs:region:account_ID:query-result:*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "Your_account_ID"
                }
            }
        }
    ]
}
```

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: KMS キーをクエリ結果に関連付ける

KMS キーをアカウントのクエリ結果に関連付けるには

次のように disassociate-kms-key コマンドを使用します。

```
aws logs associate-kms-key --resource-identifier "arn:aws:logs:region:account-id:query-
result:*" --kms-key-id "key-arn"
```

ステップ 4: アカウントのクエリ結果からキーの関連付けを解除する

クエリ結果に関連付けられた KMS キー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 --resource-identifier "arn:aws:logs:region:account-
id:query-result:*"
```

自然言語を使用して Logs Insights CloudWatch クエリを生成および更新する

> **Note**
>
> この機能は、 CloudWatch ログの米国東部 (バージニア北部) 、米国西部 (オレゴン) 、アジアパシフィック (東京) で一般利用可能です。

CloudWatch Logs は自然言語クエリ機能をサポートしており、CloudWatch Logs Insights および CloudWatch Metrics Insights のクエリの生成と更新に役立ちます。

この機能を使用すると、探している CloudWatch ログデータをプレーン英語で質問したり、説明したりできます。自然言語機能は、入力したプロンプトに基づいてクエリを生成し、クエリの仕組み line-by-line を説明します。また、さらにデータを詳しく調査できるように、生成されたクエリを更新することもできます。

環境に応じて、「転送されたバイト数で上位 100 のソース IP アドレスは何ですか？」などのプロンプトを入力できます。および 「Lambda 関数の最も遅いリクエストを 10 件見つけてください」

この機能を使用して CloudWatch Logs Insights クエリを生成するには、 CloudWatch Logs Insights クエリエディタを開き、クエリするロググループを選択し、クエリの生成 を選択します。

> **Important**
>
> 自然言語クエリ機能を使用するには、CloudWatchLogsFullAccess、、CloudWatchLogsReadOnlyAccess、AdministratorAccessまたは ReadOnlyAccessポリシーを使用する必要があります。
> また、新規または既存のカスタマー管理ポリシーまたはインラインポリシーに `cloudwatch:GenerateQuery` アクションを含めることもできます。

クエリの例

このセクションの例では、自然言語機能を使用して、クエリを生成し更新する方法について説明します。

> **Note**
>
> CloudWatch Logs Insights クエリエディタと構文の詳細については、「Logs CloudWatch Insights クエリ構文」を参照してください。

例: 自然言語クエリを生成する

自然言語を使用してクエリを生成するには、プロンプトを入力し、[新しいクエリを生成] を選択します。この例は、基本的な検索を実行するクエリを示しています。

プロンプト

以下は、Lambda 関数の呼び出しが最も遅い 10 件を検索する機能を指定するプロンプトの例です。

```
Find the 10 slowest requests
```

Query

次に、このプロンプトに基づいて自然言語機能が生成するクエリの例を示します。プロンプトがクエリの前にあるコメントにどのように表示されるかに注意してください。クエリの後に、クエリの仕組みを説明した文があります。

```
# Find the 10 slowest requests
fields @timestamp, @message, @duration
| sort @duration desc
| limit 10
# This query retrieves the timestamp, message and duration fields from the logs and
 sorts them in descending order by duration to find the 10 slowest requests.
```

> **Note**
>
> プロンプトとクエリの仕組みを説明した文が表示されないようにするには、エディタの歯車アイコンを使用します。

例: 自然言語クエリを更新する

生成済みのプロンプトを編集し、[クエリを更新] を選択することで、クエリを更新できます。

プロンプトの更新

次の例は、先ほどのプロンプトを更新したものを示しています。このプロンプトは、Lambda 関数の呼び出しが最も遅い 10 件を検索するプロンプトの代わりに、Lambda 関数の呼び出しが最も遅い 20 件を検索し、追加のログイベントに別の列を含めるように指示するようになりました。

```
Show top 20 slowest requests instead and display requestId as a column
```

クエリの更新

次に、更新したクエリの例を示します。更新したプロンプトが更新したクエリの前にあるコメントにどのように表示されるかに注意してください。クエリの後に、元のクエリがどのように更新されたかを説明した文があります。

```
# Show top 20 slowest requests instead and display requestId as a column
fields @timestamp, @message, @requestId, @duration
| sort @duration desc
| limit 20
# This query modifies the original query by replacing the @message field with the
 @requestId field and changing the limit from 10 to 20 to return the top 20 log events
 by duration instead of the top 10.
```

サービス改善のためのデータ使用をオプトアウトする

AI モデルをトレーニングし、該当するクエリを生成するために提供した自然言語プロンプト
データは、サービスを提供して維持するためにのみ使用されます。このデータは、Logs Insights
CloudWatch の品質を向上させるために使用される場合があります。お客様の信頼、プライバシー、
コンテンツのセキュリティが当社の最優先事項です。詳細については、「AWS サービス規約」と
「AWS responsible AI policy」を参照してください。

自然言語クエリの開発や品質改善に自分のコンテンツが使用されないようにオプトアウトするには、AI サービスオプトアウトポリシーを作成します。クエリ生成機能を含むすべての CloudWatch Logs AI 機能のデータ収集をオプトアウトするには、 CloudWatch Logs のオプトアウトポリシーを作成する必要があります。詳細については、「AWS Organizations ユーザーガイド」の「AI サービスオプトアウトポリシー」を参照してください。

ログ異常検出

ロググループごとにログ異常ディテクターを作成できます。異常ディテクターは、ロググループに取り込まれたログイベントをスキャンし、ログデータで異常を検出します。異常検出では、機械学習とパターン認識を使用して、一般的なログコンテンツのベースラインを確立します。

ロググループの異常ディテクターを作成すると、過去 2 週間のログイベントを使用してトレーニングされます。トレーニング期間は最大 15 分かかる場合があります。トレーニングが完了すると、受信ログの分析が開始されて異常が識別され、その異常が Logs CloudWatch コンソールに表示されて調べることができます。

CloudWatch ログパターン認識は、ログ内の静的コンテンツと動的コンテンツを識別することで、ログパターンを抽出します。多数のログイベントをいくつかのパターンに圧縮できるため、パターンは大きなログセットの分析に役立ちます。

例えば、次の 3 つのログイベントのサンプルを参照してください。

```
2023-01-01 19:00:01 [INFO] Calling DynamoDB to store for resource id 12342342k124-12345
2023-01-01 19:00:02 [INFO] Calling DynamoDB to store for resource id 324892398123-12345
2023-01-01 19:00:03 [INFO] Calling DynamoDB to store for resource id 3ff231242342-12345
```

前のサンプルでは、3 つのログイベントはすべて 1 つのパターンに従います。

```
<*> <*> [INFO] Calling DynamoDB to store for resource id <*>
```

パターン内のフィールドはトークン と呼ばれます。リクエスト ID やタイムスタンプなど、パターン内で異なるフィールドは、動的トークン と呼ばれます。動的トークンは、 CloudWatch ログにパターンが表示される<*>ときに によって表されます。動的トークンに対して見つかった各値は、トークン値 と呼ばれます。

動的トークンの一般的な例には、エラーコード、タイムスタンプ、リクエスト IDs。

ログの異常検出では、これらのパターンを使用して異常を検出します。異常ディテクターモデルのトレーニング期間後、ログは既知の傾向に対して評価されます。異常ディテクターは、異常として大きな変動にフラグを付けます。

ログ異常ディテクターを作成しても料金は発生しません。

異常とパターンの重要度と優先度

ログ異常ディテクターによって検出された各異常には、優先度 が割り当てられます。見つかった各パターンには重要度 が割り当てられます。

- Priority は自動的に計算され、パターンの重要度レベルと想定値からの偏差量の両方に基づきます。例えば、特定のトークン値が突然 500% 増加した場合、その重要度が であっても、その異常がHIGH優先度として指定されることがありますNONE。
- 重要度は、、FATAL、 などのパターンで見つかったキーワードのみに基づいていますERRORWARN。これらのキーワードが見つからない場合、パターンの重要度は とマークされますNONE。

異常可視性時間

異常ディテクターを作成するときは、そのディテクターの最大異常可視性期間を指定します。これは、異常がコンソールに表示され、ListAnomaliesAPI オペレーションによって返される日数です。この期間が経過した後、異常が発生し続けると、自動的に通常の動作として受け入れられ、異常ディテクターモデルは異常としてフラグ付けを停止します。

異常ディテクターの作成時に可視性時間を調整しない場合、デフォルトとして 21 日が使用されます。

異常の抑制

異常が見つかったら、それを一時的または永続的に抑制することを選択できます。異常を抑制すると、異常ディテクターは、指定した時間、この発生を異常としてフラグ付けしなくなります。異常を抑制する場合、その特定の異常のみを抑制するか、異常が見つかったパターンに関連するすべての異常を抑制するかを選択できます。

抑制された異常はコンソールで引き続き表示できます。抑制を停止することもできます。

よくある質問

データ AWS を使用して機械学習アルゴリズムをトレーニングしたり、他の顧客 AWS 向けにトレーニングしたりしていますか?

いいえ。トレーニングによって作成された異常検出モデルは、ロググループのログイベントに基づいており、そのロググループとその AWS アカウント内でのみ使用されます。

異常検出にはどのようなタイプのログイベントが適していますか？

ログ異常検出は、アプリケーションログやその他のタイプのログで、ほとんどのログエントリが一般的なパターンに適合する場合に適しています。INFO、ERROR、DEBUG などのログレベルまたは重要度キーワードを含むイベントを含むロググループは、異常検出のログ記録に特に適しています。

ログ異常検出は、Logs などの非常に長い JSON 構造を持つログイベントには適していません。CloudTrail パターン分析では、ログ行の最初の 1500 文字までしか分析されないため、その制限を超える文字はスキップされます。

VPC フローログなどの監査ログやアクセスログも、異常検出で成功する回数が少なくなります。異常検出はアプリケーションの問題を検出することを目的としているため、ネットワークやアクセスの異常には適していない可能性があります。

異常ディテクターが特定のロググループに適しているかどうかを判断するには、 CloudWatch ログパターン分析を使用してグループ内のログイベントのパターン数を見つけます。パターンの数が約 300 以下の場合、異常検出はうまく機能する可能性があります。パターン分析の詳細については、「」を参照してくださいパターン分析。

何が異常としてフラグ付けされますか？

次の出現により、ログイベントが異常としてフラグ付けされる可能性があります。

- ロググループで以前には見られないパターンのログイベント。
- 既知のパターンに対する大きなバリエーション。
- 通常の値の個別のセットを持つ動的トークンの新しい値。
- 動的トークンの値の出現回数の大きな変更。

上記の項目はすべて異常としてフラグ付けされている可能性がありますが、すべてがアプリケーションのパフォーマンスが低いことを意味するわけではありません。例えば、いくつかの higher-than-usual200成功値が異常としてフラグ付けされる場合があります。このような場合は、問題を示していないこれらの異常を抑制することを検討してください。

マスキングされている機密データはどうなりますか？

機密データとしてマスクされたログイベントの部分は、異常についてスキャンされません。機密データのマスキングの詳細については、「 マスキングによる機密データの保護に役立つ」を参照してください。

ロググループで異常検出を有効にする

次の手順を使用して、 CloudWatch コンソールを使用して、ロググループをスキャンして異常を検出するログ異常ディテクターを作成します。

異常ディテクターはプログラムで作成することもできます。詳細については、「」を参照してくださいCreateLogAnomalyDetector。

ログ異常ディテクターを作成するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ログ 、ログ異常 を選択します。
3. 異常ディテクターの作成 を選択します。
4. この異常ディテクターを作成するロググループを選択します。
5. 異常ディテクター名 にディテクターの名前を入力します。
6. (オプション) 評価の頻度をデフォルトの 5 分から変更します。ロググループが新しいログを受信する頻度に従って、この値を設定します。例えば、ロググループが 10 分ごとに新しいログイベントをバッチで受信する場合、評価頻度を 15 分に設定するのが適切な場合があります。
7. (オプション) 特定の単語または文字列を含むログイベントでのみ異常を検索するように異常ディテクターを設定するには、フィルターパターン を選択します。

   次に、異常検出フィルターパターン にパターンを入力します。パターン構文の詳細については、「」を参照してくださいメトリクスフィルター、サブスクリプションフィルター、フィルターログイベント、およびライブテールのフィルターパターン構文。

   (オプション) フィルターパターンをテストするには、ログイベントメッセージにいくつかのログメッセージを入力し、テストパターン を選択します。
8. (オプション) 異常可視性期間をデフォルトから変更したり、 キーを AWS KMS この異常ディテクターに関連付けるには、詳細設定 を選択します。

   a. 異常可視性期間をデフォルトから変更するには、「異常可視性期間の最大 (日数) 」に新しい値を入力します。

   b. AWS KMS キーをこの異常ディテクターに関連付けるには、KMS キー ARN に ARN を入力します。キーを割り当てると、このディテクターで見つかった異常情報は、保管時にキーで暗号化されます。ユーザーは、このキーと異常ディテクターが検出した異常に関する情報を取得するためのアクセス許可を持っている必要があります。

また、 CloudWatch Logs サービスプリンシパルにキーを使用するアクセス許可があることを確認する必要があります。詳細については、「異常ディテクターとその結果を で暗号化する AWS KMS」を参照してください。

9. 異常検出を有効にする を選択します。

   異常ディテクターが作成され、ロググループが取り込んでいるログイベントに基づいてモデルのトレーニングが開始されます。約 15 分後、異常検出がアクティブになり、異常の検出と表面化が開始されます。

見つかった異常を表示する

1 つ以上のログ異常ディテクターを作成したら、 CloudWatch コンソールを使用して、検出した異常を表示できます。

異常はプログラムで表示できます。詳細については、「」を参照してくださいListAnomalies。

すべてのログ異常ディテクターで見つかった異常を表示するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ログ 、ログ異常 を選択します。

   ログ異常テーブルが表示されます。ログ異常の横にある上部の数値には、テーブルにリストされているログ異常の数が表示されます。テーブルの各行には、次の情報が表示されます。

   - 異常 列には、異常の簡単な概要が表示されます。これらの概要は Logs CloudWatch によって生成されます。
   - 異常の優先度。Priority は、ログイベントの変化量、ログイベントでException発生する などのキーワードに基づいて自動的に計算されます。
   - 異常が基づいているログパターン。パターンの詳細については、「」を参照してくださいログ異常検出。
   - 異常ログの傾向には、パターンに一致するログの量を示すヒストグラムが表示されます。
   - 最終検出時刻には、この異常が最後に検出された時刻が表示されます。
   - 最初の検出時刻には、この異常が最初に検出された時刻が表示されます。
   - 異常ディテクターは、この異常に関連するログイベントを含むロググループの名前を表示します。この名前を選択すると、ロググループの詳細ページを表示できます。
3. 1 つの異常をさらに検査するには、その行のラジオボタンを選択します。

パターン検査ペインが表示され、以下が表示されます。

- この異常が基づいているパターン。パターン内のトークンを選択して、そのトークンの値を分析します。
- クエリされた時間範囲における異常の出現回数を示すヒストグラム。
- ログサンプルタブには、異常の一部であるログイベントがいくつか表示されます。
- トークン値 タブには、選択した動的トークンの値が表示されます。

> **Note**
>
> トークンごとに最大 10 個のトークン値がキャプチャされます。トークン数は正確ではない可能性があります。 CloudWatch Logs は確率カウンターを使用して、絶対値ではなくトークン数を生成します。

4. 異常を抑制するには、その行のラジオボタンを選択し、次の操作を行います。

   a. アクション を選択し、異常 を抑制します。

   b. 次に、異常を抑制する期間を指定します。

   c. このパターンに関連するすべての異常を抑制するには、パターンの抑制 を選択します。

   d. Suppress anomaly を選択します。

1 つのロググループで見つかった異常を表示するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. [Logs] (ログ)、[Log groups] (ロググループ) を選択します。
3. ロググループの名前を選択し、異常検出タブを選択します。

   異常検出テーブルが表示されます。ログ異常の横にある上部の数値には、テーブルにリストされているログ異常の数が表示されます。テーブルの各行には、次の情報が表示されます。

   - 異常 列には、異常の簡単な概要が表示されます。これらの概要は Logs CloudWatch によって生成されます。
   - 異常の優先度。Priority は、ログイベントの変化量、ログイベントで`Exception`発生する などのキーワードに基づいて自動的に計算されます。
   - 異常が基づいているログパターン。パターンの詳細については、「」を参照してください[ログ異常検出]。

- 異常ログの傾向には、パターンに一致するログの量を示すヒストグラムが表示されます。
- 最終検出時刻には、この異常が最後に検出された時刻が表示されます。
- 最初の検出時刻には、この異常が最初に検出された時刻が表示されます。

4. 1 つの異常をさらに検査するには、その行のラジオボタンを選択します。

   パターン検査ペインが表示され、以下が表示されます。

   - この異常が基づいているパターン。パターン内のトークンを選択して、そのトークンの値を分析します。
   - クエリされた時間範囲における異常の出現回数を示すヒストグラム。
   - ログサンプルタブには、異常の一部であるログイベントがいくつか表示されます。
   - トークン値 タブには、選択した動的トークンの値が表示されます。

   > **Note**
   >
   > トークンごとに最大 10 個のトークン値がキャプチャされます。トークン数は正確ではない可能性があります。 CloudWatch Logs は確率カウンターを使用して、絶対値ではなくトークン数を生成します。

5. 異常を抑制するには、その行のラジオボタンを選択し、次の操作を行います。

   a. アクション を選択し、異常 を抑制します。

   b. 次に、異常を抑制する期間を指定します。

   c. このパターンに関連するすべての異常を抑制するには、パターンの抑制 を選択します。

   d. 異常を抑制 を選択します。

ログ異常ディテクターにアラームを作成する

ロググループのログ異常ディテクターのアラームを作成できます。指定した期間中にロググループで指定した数の異常が見つかったときにアラームが ALARM 状態になるように を指定できます。フィルターを使用して、指定した優先順位の異常のみがアラームによってカウントされるようにすることもできます。

ログ異常ディテクターのアラームを作成するには

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

2. ナビゲーションペインで、ログ 、ログ異常 を選択します。

   ログ異常ディテクターの表が表示されます。

3. アラームを設定する異常ディテクターのラジオボタンを選択し、アラームの作成 を選択します。

   CloudWatch アラーム作成ウィザードが表示されます。LogAnomalyDetector フィールドには、選択した異常ディテクターの名前が表示されます。メトリクス名フィールドには が表示されますAnomalyCount。

4. (オプション) このアラームを異常優先度でフィルタリングするには、次のいずれかを実行します。

   - アラームカウントで優先度の高い異常のみを指定するには、 **HIGH**に を入力しますLogAnomalyPriority。
   - アラーム数を高優先度と中優先度の異常のみにするには、 **MEDIUM**に と入力しますLogAnomalyPriority。

   優先度レベルの詳細については、「」を参照してください異常とパターンの重要度と優先度。

5. アラームに静的またはメトリクスの異常検出しきい値を使用するかどうかを選択します。この選択により、アラームしきい値の設定方法が決まります。静的しきい値は、アラームのしきい値が、選択した静的な定数数であることを意味します。異常検出しきい値とは、 が通常の値の範囲 CloudWatch を決定し、実際の数がこの帯域のしきい値を超えた場合にアラームがトリガーされることを意味します。ログ異常検出アラームの異常検出を選択する必要はありません。メトリクス異常検出の詳細については、 CloudWatch 「異常検出の使用」を参照してください。

6. ***your-metric-name*** が であるときはいつでも、より大きい 、より大きい/等しい 、小さい/等しい 、または小さい を選択します。 次に、[... よりも] で、しきい値の数値を指定します。期間で指定された時間内に異常ディテクターがこの数を超えるアラームを検出した場合、アラームはALARM状態になります。

7. [Additional configuration (追加設定)] を選択します。[Datapoints to alarm (アラームを発生させるデータポイント数)] で、アラームをトリガーするために ALARM 状態を維持する必要がある評価期間 (データポイント) の数を指定します。2 つの値が一致する場合は、該当する数の連続した期間でしきい値を超過したときに ALARM 状態に移行するアラームを作成します。

   N 個中 M 個のアラームを作成するには、2 番目の値よりも小さい数字を最初の値に指定します。詳細については、「アラームの評価」を参照してください。

8. [Missing data treatment] (欠落データの処理) で、一部のデータポイントが欠落している際のアラームによる対処方法を選択します。詳細については、 CloudWatch 「アラームが欠落データを処理する方法の設定」を参照してください。
9. [次へ] をクリックします。
10. 通知 で、通知の追加 を選択し、アラームが 、、または 状態に移行したときに通知する Amazon SNS トピックを指定します。 `ALARM` `OK` `INSUFFICIENT_DATA`

    a. (オプション) 同じアラーム状態または異なるアラーム状態について複数の通知を送信するには、[Add notification] (通知の追加)を選択します。

    > **Note**
    >
    > [アラーム] 状態になったときだけでなく、[データ不足] 状態になったときにもアクションを実行するようにアラームを設定することをお勧めします。これは、データソースに接続する Lambda 関数に関する多くの問題により、アラームが [データ不足] に遷移する可能性があるためです。

    b. (オプション) Amazon SNS の通知を送信しない場合は、[削除] を選択します。
11. (オプション) アラームで Amazon EC2 Auto Scaling 、Amazon EC2、チケット、または のアクションを実行する場合は AWS Systems Manager、適切なボタンを選択し、アラームの状態とアクションを指定します。

    > **Note**
    >
    > アラームは、`ALARM` 状態になった時にのみ、Systems Manager のアクションを実行できます。Systems Manager アクションの詳細については、「作成 CloudWatch するための設定 OpsItems」および「インシデント作成」を参照してください。
12. [次へ] をクリックします。
13. [Name and description] (名前と説明) にアラームの名前と説明を入力し、[Next] (次へ) をクリックします。アラーム名には UTF-8 文字のみを使用する必要があり、ASCII 制御文字は使用できません。説明には、 CloudWatch コンソールのアラームの詳細タブにのみ表示されるマークダウン形式を含めることができます。マークダウンは、ランブックや他の内部リソースへのリンクを追加するのに役立ちます。

> **Tip**
>
> アラーム名には UTF-8 文字のみを使用する必要があります。ASCII 制御文字を含めることはできません。

14. [Preview and create] (プレビューと作成) で、アラームの情報と条件が正しいことを確認し、[Create alarm] (アラームの作成) をクリックします。

ログ異常ディテクターによって発行されるメトリクス

CloudWatch ログはメトリクスを CloudWatch メトリクスAnomalyCountに発行します。このメトリクスは `AWS/Logs`名前空間に発行されます。

AnomalyCount メトリクスは、次のディメンションで発行されます。

- LogAnomalyDetector– 異常ディテクターの名前
- LogAnomalyPriority- 異常の優先度レベル

異常ディテクターとその結果を で暗号化する AWS KMS

異常ディテクターデータは常に CloudWatch ログで暗号化されます。デフォルトでは、 CloudWatch Logs は保管中のデータに対してサーバー側の暗号化を使用します。別の方法として、この暗号化には AWS Key Management Service を使用できます。その場合、暗号化は AWS KMS キーを使用して行われます。を使用した暗号化 AWS KMS は、KMS キーを異常ディテクターに関連付けることで、異常ディテクターレベルで有効になります。

> **Important**
>
> CloudWatch ログは対称 KMS キーのみをサポートします。非対称キーを使用してロググループのデータを暗号化しないでください。詳細については、「対称キーと非対称キーの使用」を参照してください。

制限

- 以下の手順を実行するには、kms:CreateKey、kms:GetKeyPolicy、および kms:PutKeyPolicy アクセス許可が必要です。
- 異常ディテクターからのキーの関連付けまたは関連付け解除後、オペレーションが有効になるまでに最大 5 分かかることがあります。
- 関連付けられたキーへの CloudWatch Logs アクセスを取り消したり、関連付けられた KMS キーを削除したりすると、 CloudWatch Logs の暗号化されたデータを取得できなくなります。

ステップ 1: AWS KMS キーを作成する

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

```
aws kms create-key
```

出力には、キーのキー 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": {
        "Origin": "AWS_KMS",
        "KeyId": "key-default-1",
        "Description": "",
        "KeyManager": "CUSTOMER",
        "Enabled": true,
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "CreationDate": 1478910250.94,
        "Arn": "arn:aws:kms:us-west-2:123456789012:key/key-default-1",
        "AWSAccountId": "123456789012",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

ステップ 2: KMS キーでアクセス許可を設定する

デフォルトでは、すべての AWS KMS キーはプライベートです。リソースの所有者のみがその CMK を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 KMS キーへのアクセス許可を付与することができます。このステップでは、 キーを使用するアクセス許可を CloudWatch Logs サービスプリンシパルに付与します。このサービスプリンシパルは、KMS キーが保存されているのと同じ AWS リージョンに存在する必要があります。

ベストプラクティスとして、KMS キーの使用を、指定した AWS アカウントまたは異常ディテクターのみに制限することをお勧めします。

まず、次のget-key-policyコマンドpolicy.jsonを使用して、KMS キーのデフォルトポリシーを 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、以下のいずれかのステートメントから太字のセクションを追加します。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。これらのステートメントでは、 Conditionセクションを使用して AWS KMS キーのセキュリティを強化します。詳細については、「AWS KMS キーと暗号化コンテキスト」を参照してください。

この例の Conditionセクションでは、 AWS KMS キーの使用を指定されたアカウントに制限していますが、任意の異常ディテクターに使用できます。

```
{
  "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:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*",
      "Condition": {
        "ArnLike": {
          "kms:EncryptionContext:aws:logs:arn":
 "arn:aws:logs:REGION:Your_account_ID:anomaly-detector:*"
        }
      }
    },
    {
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.REGION.amazonaws.com"
        },
        "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
        ],
        "Resource": "*",
        "Condition": {
            "ArnLike": {
                "kms:EncryptionContext:aws-crypto-ec:aws:logs:arn":
 "arn:aws:logs:REGION:Your_account_ID:anomaly-detector:*"
            }
        }
    }
  ]
}
```

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: KMS キーを異常ディテクターに関連付ける

KMS キーは、コンソールで作成するとき、または AWS CLI または APIsを使用して作成するときに、異常ディテクターに関連付けることができます。

ステップ 4: 異常ディテクターからキーの関連付けを解除する

キーが異常ディテクターに関連付けられていると、キーを更新することはできません。キーを削除する唯一の方法は、異常ディテクターを削除してから再作成することです。

ロググループとログストリームの操作

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 内のログの個別のソースはそれぞれ、個別のログストリームを構成します。

ロググループは、保持、モニタリング、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ミング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。

Logs で CloudWatch ロググループを作成する

「Amazon CloudWatch Logs ユーザーガイド」の前のセクションのステップを使用して Amazon EC2 CloudWatch インスタンスに Logs エージェントをインストールすると、そのプロセスの一部としてロググループが作成されます。 Amazon EC2 CloudWatch コンソールでロググループを直接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/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[ロググループ] を選択します。
3. [Actions (アクション)] を選択し、[Create log group (ロググループの作成)] を選択します。
4. ロググループの名前を入力し、[Create log group (ロググループの作成)] を選択します。

> **Tip**
>
> ロググループ、ダッシュボード、アラームは、ナビゲーションペインの [お気に入りと最近使ったコンテンツ] メニューからお気に入りに登録できます。[最近アクセスしたサービス] 列で、お気に入りに登録するロググループにカーソルを合わせ、その横にある星の記号を選択します。

ロググループへのログの送信

CloudWatch ログは、複数の AWS サービスからログイベントを自動的に受信します。次のいずれかの方法を使用して、他のログイベントを CloudWatch Logs に送信することもできます。

- CloudWatch エージェント — 統合 CloudWatch エージェントはメトリクスとログの両方を CloudWatch ログに送信できます。 CloudWatch エージェントのインストールと使用の詳細については、「Amazon ユーザーガイド」の「 エージェントを使用した Amazon EC2 インスタンスとオンプレミスサーバーからのメトリクスとログの収集 CloudWatch 」を参照してください。 CloudWatch
- AWS CLI— はログイベントのバッチを CloudWatch ログput-log-eventsにアップロードします。
- プログラムによる - PutLogEvents API を使用すると、ログイベントのバッチをプログラムで CloudWatch Logs にアップロードできます。

Logs に送信された CloudWatch ログデータを表示する

CloudWatch Logs エージェントから CloudWatch Logs に stream-by-stream 送信されたログデータを表示および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/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[ロググループ] 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. ロググループのリストで、表示するロググループの名前を選択します。
5. ログストリームのリストで、表示するログストリームの名前を選択します。
6. 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。
   - 1 つのログイベントを展開するには、そのログイベントの横にある矢印を選択します。
   - 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ーするには、検索フィールドに目的の検索フィルターを入力します。詳細については、「フィルターを使用したログイベントからのメトリクスの作成」を参照してください。
   - 指定した日時範囲のログデータを表示するには、検索フィルターの隣の日付と時刻の横にある矢印を選択します。日付と時間の範囲を指定するには、[Absolute (絶対)] を選択します。事前定義された分、時間、日数、または週数を選択するには、[Relative (相対)] を選択します。UTC とローカルタイムゾーンを切り替えることもできます。

Live Tail を使用すると、ログをほぼリアルタイムで表示できます。

CloudWatch Logs Live Tail は、取り込まれた新しいログイベントのストリーミングリストを表示することで、インシデントを迅速にトラブルシューティングするのに役立ちます。取り込まれたログをほぼリアルタイムで表示、フィルタリング、強調表示できるため、問題をすばやく検出して解決することができます。指定した用語に基づいてログをフィルタリングしたり、特定の用語を含むログを強調表示したりすることで、探しているものをすぐに見つけることができます。

Live Tail セッションでは、セッションの使用時間ごとに 1 分間隔でコストが発生します。料金の詳細については、「Amazon CloudWatch 料金表」の「ログ」タブを参照してください。

> **Note**
>
> Live Tail は、標準ログクラスのロググループでのみサポートされます。ログクラスの詳細については、「」を参照してくださいログクラス。

以下のセクションでは、 コンソールで Live Tail を使用する方法について説明します。Live Tail セッションをプログラムで開始することもできます。詳細については、「」を参照してくださいStartLiveTail。SDK の例については、AWS 「 SDK を使用して Live Tail セッションを開始する」を参照してください。

Live Tail セッションを開始する

CloudWatch コンソールを使用して Live Tail セッションを開始します。以下の手順では、ナビゲーションペインの [Live tail] を選択して Live Tail セッションを開始する方法について説明します。ロググループページまたはログ CloudWatch インサイトページからライブテールセッションを開始することもできます。

> **Note**
>
> Live Tail で表示されるロググループの機密データを、データ保護ポリシーを使用してマスクしている場合、Live Tail セッションでは、機密データは常にマスクされて表示されます。ロググループの機密データのマスキングに関する詳細は、「機密性の高いログデータをマスキングで保護する」を参照してください。

Live Tail セッションを開始するには

1. https://console.aws.amazon.com/cloudwatch/ で 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]、[Live tail] の順に選択します。
3. [ロググループを選択] で、Live Tail セッションでイベントを表示するロググループを選択します。ロググループは 10 個まで選択できます。
4. (オプション) ロググループを 1 つのみ選択する場合は、ログイベントを表示するログストリームを 1 つ以上選択すれば、Live Tail セッションをさらに絞り込むことができます。それには、[ログストリームを選択] で、ドロップダウンリストからログストリームの名前を選択します。あるいは、[ログストリームを選択] の 2 番目のボックスにログストリーム名のプレフィックスを入力すれば、このプレフィックスに一致する名前を持つすべてのログストリームが選択されます。
5. (オプション) 特定の単語やその他の文字列を含むログイベントのみを表示するときは、その単語または文字列を `Add filter patterns` に入力します。

   例えば、`Warning` という語を含むログイベントのみを表示するときは、**`Warning`** と入力します。フィルターフィールドでは、大文字と小文字が区別されます。このフィールドには、次に示す複数の用語とパターン演算子を含めることができます。

   - **`error 404`** は、`error` と `404` の両方を含むログイベントのみを表示します。
   - **`?Error ?error`** は、`Error` または `error` を含むログイベントを表示します。
   - **`-INFO`** は、`INFO` を含まないログイベントをすべて表示します。
   - **`{ $.eventType = "UpdateTrail" }`** は、イベントタイプフィールドの値が `UpdateTrail` である JSON ログイベントをすべて表示します。

   正規表現を使用してフィルタリングすることもできます。

   - **`%ERROR%`** は regex を使用して、ERROR キーワードを含むすべてのログイベントを表示します。
   - **`{ $.names = %Steve% }`** は Steve が `"name"` プロパティ内にいる場合、regex を使用して JSON ログイベントを表示します。
   - **`[ w1 = %abc%, w2 ]`** は regex を使用して、最初の単語が abc の場合にスペースで区切られたログイベントを表示します。

   パターン構文の詳細については、「フィルターパターン構文」を参照してください。

6. (オプション) 表示されたログイベントの一部を強調表示するには、検索する用語を入力し、[Live Tail] で強調表示します。強調表示する用語は 1 度に 1 つずつ入力します。複数の用語を追加して強調表示すると、用語ごとに異なる色が割り当てられます。指定した用語を含むログイベントの左側に強調表示のインジケーターが表示されます。また、メインウィンドウでログイベントを展開してログイベント全体を表示すると、用語自体の下にも表示されます。

   フィルタリングと強調表示を併用することで、問題をすばやくトラブルシューティングできます。例えば、イベントをフィルタリングして、`Error` を含むイベントのみを表示し、さらに、`404` を含むイベントを強調表示することもできます。

7. セッションを開始するには、[フィルターを適用] を選択します

   一致するログイベントがウィンドウに表示されます。以下の情報も表示されます。

   - timer には、Live Tail セッションの実行時間が表示されます。
   - events/sec には、設定したフィルターに一致するログイベントが 1 秒間にいくつ取り込まれたかが表示されます。
   - 多くのイベントがフィルターと一致するため、セッションのスクロールが速すぎるように、CloudWatch Logs には一致するイベントの一部しか表示されない場合があります。その場合は、画面に表示されているイベントが一致するイベントの何割であるのかが % で表示されます。

8. イベントのフローを一時停止して、現在表示されている内容を調べるには、イベントウィンドウの任意の場所をクリックします。

9. セッション中は、以下を使って各ログイベントの詳細を確認できます。

   - メインウィンドウにログイベントのテキスト全体を表示するには、そのログイベントの横にある矢印をクリックします。
   - サイドウィンドウにログイベントのテキスト全体を表示するには、そのログイベントの横にある虫眼鏡の [+] をクリックします。イベントフローが一時停止し、サイドウィンドウが表示されます。

     サイドウィンドウにログイベントのテキストを表示すると、そのテキストをメインウィンドウの他のイベントと比較するのに便利です。

10. Live Tail セッションを停止するには、[停止] をクリックします。

11. セッションを再開するには、[フィルター] パネルを使用してフィルター条件を変更し、[フィルターを適用] をクリックします。次に、[Start (開始)] を選択します。

フィルターパターンを使用してログデータを検索する

ログデータは、メトリクスフィルター、サブスクリプションフィルター、フィルターログイベント、およびライブテールのフィルターパターン構文 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することも、 を使用して特定のログストリームを検索 AWS CLI 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。

CloudWatch クロスアカウントオブザーバビリティでモニタリングアカウントとして設定されたアカウントにサインインしている場合は、このモニタリングアカウントにリンクされたソースアカウントからログイベントを検索してフィルタリングできます。詳細については、CloudWatch 「クロスアカウントオブザーバビリティ」を参照してください。

コンソールを使用してログエントリを検索する

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/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[ロググループ] 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. [Log Events (ログイベント)] で、使用するフィルター構文を入力します。

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/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[ロググループ] 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. [ロググループの検索] を選択します。
5. [Log Events (ログイベント)] で、日付と時刻の範囲を選択し、フィルター構文を入力します。

を使用してログエントリを検索する AWS CLI

を使用して、指定された条件を満たすログエントリを検索できます AWS CLI。

を使用してログエントリを検索するには AWS CLI

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ストリームに限定するには `--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]
```

を使用して特定の時間範囲のログエントリを検索するには AWS CLI

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/ で 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/ で 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:

- を使用している場合は AWS CLI、検索対象を目的のログストリームのみに制限できます。例えば、ロググループに 1000 個のログストリームがあり、関連性があることがわかっているログストリームを 3 つだけ表示したい場合は、 を使用して、ロググループ内のこれら 3 つのログストリームのみに検索 AWS CLI を制限できます。
- 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。

CloudWatch Logs でログデータ保持を変更する

デフォルトでは、ログデータは無期限に CloudWatch Logs 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。

> **Note**
>
> CloudWatch Logs は、保持設定に達したログイベントをすぐに削除しません。通常、ログイベントが削除されるまでに最大 72 時間かかりますが、まれにそれ以上かかる場合もあります。
> つまり、有効期限を過ぎているが実際には削除されていないログイベントが含まれている場合に、ロググループを長い保持設定に変更すると、新しい保持期間に達してからこれらの

ログイベントが削除されるまでに最大 72 時間かかります。ログデータを完全に削除するには、前の保持期間が終了してから 72 時間が経過するか、古いログイベントが削除されることを確認するまで、ロググループを低い保持設定にしておきます。
ログイベントが保持設定に達すると、削除対象としてマークされます。削除対象としてマークされた後は、後で実際に削除されない場合でも、アーカイブストレージのコストが追加されることはありません。また、削除対象としてマークされたこれらのログイベントは、API を使用して `storedBytes` の値を取得し、ロググループが保存しているバイト数を確認する場合にも含まれません。

ログの保持設定を変更するには

1. https://console.aws.amazon.com/cloudwatch/ で 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]、[ロググループ] の順に選択します。
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保持列で、有効期限なし など、現在の保持設定を選択します。
5. 保持設定 で、 の後にイベントを期限切れにするには、ログ保持値を選択し、保存 を選択します。

Amazon CloudWatch Logs のロググループにタグを付ける

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 の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 リソースを管理し、請求データを含むデータを整理するためのシンプルで強力な方法です。

Note

タグを使用して、 CloudWatch ロググループや送信先などの Logs リソースへのアクセスを制御できます。ロググループとログストリームの間には階層的な関係があるため、ログストリームへのアクセスはロググループレベルで制御されます。リソースへのアクセスを制御するタグの使用の詳細については、タグを使用した Amazon Web Services のリソースへのアクセスの制御を参照してください。

内容

- タグ付けを使用したコストの追跡
- タグの制限
- を使用したロググループのタグ付け AWS CLI
- CloudWatch Logs API を使用したロググループのタグ付け

タグの基本

AWS CloudFormation AWS CLI、、または CloudWatch Logs API を使用して、次のタスクを完了します。

- 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一覧表示する
- 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 コストを分類および追跡できます。ロググループを含む AWS リソースにタグを適用すると、 AWS コスト配分レポートにはタグ別に集計された使用量とコストが含まれます。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 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。
- タグのキーと値では、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。
- このプレフィックスは が使用できるように予約aws:されているため、 でタグキーを開始することはできません AWS。 は、ユーザーに代わってこのプレフィックスで始まるタグ AWS を作成しますが、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 (_ . / = + - @)。

を使用したロググループのタグ付け AWS CLI

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。

タグリソース

指定された Logs リソースに 1 つ以上のタグ (キーと値のペア) CloudWatch を割り当てます。

list-tags-for-resource

が CloudWatch Logs リソースに関連付けられているタグを表示します。

タグなしリソース

指定された CloudWatch Logs リソースから 1 つ以上のタグを削除します。

CloudWatch Logs API を使用したロググループのタグ付け

CloudWatch Logs API を使用してタグを追加、一覧表示、削除できます。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。

TagResource

指定された Logs リソースに 1 つ以上のタグ (キーと値のペア) CloudWatch を割り当てます。

ListTagsForResource

が CloudWatch Logs リソースに関連付けられているタグを表示します。

UntagResource

指定された CloudWatch Logs リソースから 1 つ以上のタグを削除します。

を使用して Logs CloudWatch のログデータを暗号化する AWS Key Management Service

ロググループのデータは常に Logs CloudWatch で暗号化されます。デフォルトでは、 CloudWatch Logs は保管中のログデータにサーバー側の暗号化を使用します。別の方法として、この暗号化には AWS Key Management Service を使用できます。その場合、暗号化は AWS KMS キーを使用して行われます。を使用した暗号化 AWS KMS は、ロググループの作成時または作成後に、KMS キーをロググループに関連付けることで、ロググループレベルで有効になります。

> **Important**
>
> CloudWatch ログは暗号化コンテキストをサポートするようになりました。 をキー`kms:EncryptionContext:aws:logs:arn`として使用し、ロググループの ARN をそのキーの値として使用します。KMS キーで暗号化したロググループがあり、そのキーが 1 つのアカウントとロググループで使用されるように制限する場合は、新しい KMS キーを割り当て、そのための条件を IAM ポリシーに含める必要があります。詳細については、「AWS KMS キーと暗号化コンテキスト」を参照してください。

KMS キー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は、このキー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された形式で保存されます。 CloudWatch ログは、リクエストされるたびにこのデータを復号します。 CloudWatch ログは、暗号化されたデータがリクエストされるたびに KMS キーに対するアクセス許可を持っている必要があります。

後でロググループから KMS キーの関連付けを解除すると、 CloudWatch Logs は Logs のデフォルトの暗号化方法を使用して新しく取り込まれたデータを暗号化 CloudWatch します。KMS キーで暗号化された以前に取り込まれたデータはすべて KMS キーで暗号化されたままになります。CloudWatch ログは引き続きキーを参照できるため、KMS CloudWatch キーの関連付けが解除された後もそのデータを返すことができます。ただし、後でキーが無効になると、 CloudWatch ログはそのキーで暗号化されたログを読み取ることができません。

> **Important**
>
> CloudWatch ログは対称 KMS キーのみをサポートします。非対称キーを使用してロググループのデータを暗号化しないでください。詳細については、「対称キーと非対称キーの使用」を参照してください。

制限

- 以下の手順を実行するには、`kms:CreateKey`、`kms:GetKeyPolicy`、および `kms:PutKeyPolicy` アクセス許可が必要です。
- キー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。

- 関連付けられたキーへの CloudWatch Logs アクセスを取り消したり、関連付けられた KMS キーを削除したりすると、 CloudWatch Logs 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 KMS キーをロググループに関連付けることはできません。

ステップ 1: AWS KMS キーを作成する

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

```
aws kms create-key
```

出力には、キーのキー 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": {
        "Origin": "AWS_KMS",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Description": "",
        "KeyManager": "CUSTOMER",
        "Enabled": true,
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "CreationDate": 1478910250.94,
        "Arn": "arn:aws:kms:us-west-2:123456789012:key/6f815f63-e628-448c-8251-
e40cb0d29f59",
        "AWSAccountId": "123456789012",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

ステップ 2: KMS キーでアクセス許可を設定する

デフォルトでは、すべての AWS KMS キーはプライベートです。リソースの所有者のみがその CMK を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 KMS キーへのアクセス許可を付与することができます。このステップでは、 キーを使用するアクセス許可を CloudWatch Logs サービスプリンシパルに付与します。このサービスプリンシパルは、KMS キーが保存されているのと同じ AWS リージョンに存在する必要があります。

ベストプラクティスとして、KMS キーの使用は、指定した AWS アカウントまたはロググループのみに制限することをお勧めします。

まず、次のget-key-policyコマンドpolicy.jsonを使用して、KMS キーのデフォルトポリシーを 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、以下のいずれかのステートメントから太字のセクションを追加します。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。これらのステートメントでは、 Conditionセクションを使用して AWS KMS キーのセキュリティを強化します。詳細については、「AWS KMS キーと暗号化コンテキスト」を参照してください。

この例の Condition セクションでは、キーを 1 つのロググループ ARN に制限しています。

```
{
 "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": "*",
```

```
            "Condition": {
                "ArnEquals": {
                    "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:region:account-
id:log-group:log-group-name"
                }
            }
        }
    ]
}
```

この例の Condition セクションは AWS KMS キーの使用を指定したアカウントに制限しますが、これを使用できるロググループに制限はありません。

```
{
    "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": "*",
            "Condition": {
                "ArnLike": {
                    "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:region:account-
id:*"
```

```
                }
            }
        }
    ]
}
```

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: KMS キーをログ グループに関連付ける

KMS キーとロググループは、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。

ロググループに既に KMS キーが関連付けられているかどうかを確認するには、次のdescribe-log-groupsコマンドを使用します。

```
aws logs describe-log-groups --log-group-name-prefix "log-group-name-prefix"
```

出力に kmsKeyId フィールドが含まれている場合、ロググループはそのフィールドの値に対して表示されるキーに関連付けられます。

ロググループの作成時に KMS キー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"
```

KMS キー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: キーをロググループ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 KMS キー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
```

AWS KMS キーと暗号化コンテキスト

AWS Key Management Service キーと暗号化されたロググループのセキュリティを強化するために、 CloudWatch Logs はロググループの ARNs をログデータの暗号化に使用される暗号化コンテキストの一部として配置するようになりました。暗号化コンテキストは、追加の認証済みデータとして使用されるキーと値のペアのセットです。暗号化コンテキストを使用すると、IAM ポリシー条件を使用して、アカウントとロググループごとに AWS KMS AWS キーへのアクセスを制限できます。詳細については、暗号化コンテキストおよび IAM JSON ポリシー要素: 条件を参照してください。

暗号化されたロググループごとに異なる KMS キーを使用することをお勧めします。

前に暗号化したロググループがあり、そのロググループを変更して、そのグループでのみ機能する新しい KMS キーを使用する場合は、次の手順に従います。

暗号化されたロググループを変更して、KMS キーの使用をそのグループのみに制限するには

1. 次のコマンドを入力して、ロググループの現在のキーの ARN を見つけます。

   ```
   aws logs describe-log-groups
   ```

   出力には以下の行が含まれます。ARN を書きとめておきます。ステップ 7 で使用する必要があります。

   ```
   ...
   "kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/01234567-89ab-
   cdef-0123-456789abcdef"
   ...
   ```

2. 以下のコマンドを入力して、新しい KMS キーを作成します。

   ```
   aws kms create-key
   ```

3. 以下のコマンドを入力して、新しいキーのポリシーを policy.json ファイルに保存します。

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

4. テキストエディタを使用して policy.json を開き、Condition 式をポリシーに追加します。

```
{
    "Version": "2012-10-17",
    "Id": "key-default-1",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::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": "*",
            "Condition": {
                "ArnLike": {
                    "kms:EncryptionContext:aws:logs:arn":
 "arn:aws:logs:REGION:ACCOUNT-ID:log-
group:LOG-GROUP-NAME"
                }
            }
        }
    ]
}
```

5. 次のコマンドを入力して、更新されたポリシーを新しい KMS キーに追加します。

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

6. 以下のコマンドを入力して、そのポリシーをロググループに関連付けます。

```
aws logs associate-kms-key --log-group-name my-log-group --kms-key-id new-key-ARN
```

CloudWatch ログは、新しいキーを使用してすべての新しいデータを暗号化するようになりました。

7. 次に、Decrypt を除くすべてのアクセス許可を古いキーから取り消します。まず、以下のコマンドを入力して古いポリシーを取得します。

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

8. テキストエディタを使用して policy.json を開き、Action リストから kms:Decrypt* を除くすべての値を削除します。

```
{
    "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:Decrypt*"
            ],
            "Resource": "*"
        }
    ]
}
```

9. 次のコマンドを入力して、更新されたポリシーを古いキーに追加します。

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

機密性の高いログデータをマスキングで保護する

ロググループのデータ保護ポリシー を使用して、 CloudWatch ログによって取り込まれる機密データを保護するのに役立ちます。これらのポリシーを使うことで、アカウントのロググループが取り込んだログイベントに表示される機密データを、監査およびマスクできます。

データ保護ポリシーを作成すると、デフォルトで、選択したデータ識別子に一致する機密データは、CloudWatch Logs Insights、メトリクスフィルター、サブスクリプションフィルターなど、すべての出力ポイントでマスクされます。マスクされていないデータを閲覧できるのは、`logs:Unmask` IAM アクセス許可を持つユーザーのみです。

アカウントのすべてのロググループに対してデータ保護ポリシーを作成できます。また、個々のロググループのデータ保護ポリシーも作成できます。アカウント全体に対するポリシーを作成すると、既存のロググループと今後作成するロググループの両方に、ポリシーが適用されます。

アカウント全体に対するデータ保護ポリシーを作成し、1 つのロググループに対するポリシーも作成すると、そのロググループには両方のポリシーが適用されます。いずれかのポリシーで指定されたマネージドデータ識別子は、すべてそのロググループで監査およびマスクされます。

> **Note**
>
> 機密データのマスキングは、標準ログクラスのロググループでのみサポートされます。アカウント内のすべてのロググループにデータ保護ポリシーを作成すると、標準ログクラスのロググループにのみ適用されます。ログクラスの詳細については、「」を参照してください[ログクラス]。

各ロググループで設定できるロググループレベルのデータ保護ポリシーは 1 つのみです。ただしそのポリシーでは、監査およびマスキングの対象となるマネージドデータ識別子を複数指定できます。データ保護ポリシーの文字数の上限は、30,720 文字です。

> **Important**
>
> 機密データは、ロググループに取り込まれるときに検出され、マスクされます。データ保護ポリシーを設定しても、それ以前にロググループに取り込まれたログイベントはマスクされません。

CloudWatch Logs は、財務データ、個人健康情報 (PHI)、個人を特定できる情報 (PII) を保護するために選択できる事前設定されたデータ型を提供する、多くのマネージドデータ識別子 をサポートしています。 CloudWatch ログデータ保護を使用すると、パターンマッチングモデルと機械学習モデルを活用して機密データを検出できます。一部のタイプのマネージドデータ識別子では、検出は機密データの近くにある特定のキーワードを見つけることにも依存します。カスタムデータ識別子を使用して、特定のユースケースに合わせたデータ識別子を作成することもできます。

選択したデータ識別子に一致する機密データが検出され CloudWatch ると、メトリクスが に出力されます。これはLogEventsWithFindingsメトリクスであり、AWS/Logs 名前空間に出力されます。このメトリクスを使用して CloudWatch アラームを作成し、グラフやダッシュボードで視覚化できます。データ保護によって発行されたメトリクスは無料で提供されるメトリクスなので、料金はかかりません。 CloudWatch Logs が に送信するメトリクスの詳細については、 CloudWatch「」を参照してください CloudWatch メトリクスによるモニタリング。

各マネージドデータ識別子は、特定の国または地域のクレジットカード番号、 AWS シークレットアクセスキー、パスポート番号など、特定のタイプの機密データを検出するように設計されています。データ保護ポリシーを作成する際に、これらの識別子を使用してロググループが取り込んだログを分析し、検出された場合にアクションを実行するように設定できます。

CloudWatch ログデータ保護は、マネージドデータ識別子を使用して、次のカテゴリの機密データを検出できます。

- プライベートキーや AWS シークレットアクセスキーなどの認証情報
- クレジットカード番号などの財務情報
- 運転免許証や社会保障番号などの個人を特定できる情報 (PII)
- 健康保険または医療識別番号などの保護対象保健情報 (PHI)
- IP アドレスや MAC アドレスなどのデバイス識別子

保護できるデータの種類の詳細については、「保護できるデータの種類」を参照してください。

目次

        - 保護対象の医療情報 (PHI) データタイプのデータ識別子 ARN
    - 個人を特定できる情報 (PII)
        - 運転免許証識別番号のキーワード
        - 国民識別番号のキーワード
        - パスポート番号のキーワード
        - 納税者識別と参照番号のキーワード
        - 個人を特定できる情報 (PII) のデータ識別子 ARN
- カスタムデータ識別子
    - SNS カスタムデータ識別子とは
    - カスタムデータ識別子の制約
    - コンソールでのカスタムデータ識別子の使用
    - データ保護ポリシーでカスタムデータ識別子を使用する

データ保護ポリシーを理解する

トピック

- データ保護ポリシーとは
- データ保護ポリシーの構成の仕組み

データ保護ポリシーとは

CloudWatch ログは、データ保護ポリシーを使用して、スキャンする機密データと、そのデータを保護するために実行するアクションを選択します。対象の機密データを選択するには、データ識別子を使用します。 CloudWatch ログデータ保護は、機械学習とパターンマッチングを使用して機密データを検出します。検出されたデータ識別子に基づいてアクションを実行するには、Audit (監査) および De-identify (匿名化) 操作を定義できます。これらの操作は、検出された (または検出されなかった) 機密データをログに記録し、ログイベントが表示されるときに機密データをマスクすることを可能にします。

データ保護ポリシーの構成の仕組み

次の図に示すように、データ保護ポリシードキュメントには次の要素が含まれています。

- ドキュメントの最上部に記載されるポリシー全体の情報 (任意)

- 監査および匿名化アクションを定義する 1 つのステートメント

Logs ロググループごとに定義できるデータ保護ポリシーは 1 CloudWatch つだけです。データ保護ポリシーには、1 つまたは複数の拒否または識別解除ステートメントと 1 つの監査ステートメントのみを含めることができます。

データ保護ポリシーの JSON プロパティ

データ保護ポリシーでは、識別のために以下の基本ポリシー情報が必要です。

- Name – ポリシーの名前。
- Description (オプション) – ポリシーの説明。
- Version – ポリシー言語のバージョン。現在のバージョンは 2021-06-01. です。
- Statement – データ保護ポリシーアクションを指定するステートメントのリスト。

```
{
  "Name": "CloudWatchLogs-PersonalInformation-Protection",
  "Description": "Protect basic types of sensitive data",
  "Version": "2021-06-01",
  "Statement": [
        ...
  ]
}
```

ポリシーステートメントの JSON プロパティ

ポリシーステートメントは、データ保護オペレーションの検出コンテキストを設定します。

- Sid (オプション) – ステートメント識別子。
- DataIdentifier – CloudWatch Logs がスキャンする機密データ。名前、住所、電話番号などです。
- オペレーション – 監査または識別解除のいずれかのフォローアップアクション。 CloudWatch ログは、機密データが見つかったときにこれらのアクションを実行します。

```
{
  ...
  "Statement": [
    {
```

```
      "Sid": "audit-policy",
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/Address"
      ],
      "Operation": {
        "Audit": {
          "FindingsDestination": {}
        }
      }
    },
```

ポリシーステートメントオペレーションの JSON プロパティ

ポリシーステートメントは、次のデータ保護オペレーションのいずれかを設定します。

- Audit – ロギングを中断することなく、メトリクスと結果レポートを発行します。一致する文字列は、 CloudWatch Logs が の AWS/Logs 名前空間に発行するLogEventsWithFindingsメトリクスをインクリメントします CloudWatch。これらのメトリクスは、アラームを作成するために使用できます。

  結果レポートの例については、「監査結果レポート」を参照してください。

  CloudWatch Logs が に送信するメトリクスの詳細については CloudWatch、「」を参照してください CloudWatch メトリクスによるモニタリング。
- De-identify – ロギングを中断することなく機密データをマスクします。

データ保護ポリシーの作成または操作に必要な IAM 権限

ロググループのデータ保護ポリシーにアクセスできるようにするには、次の表で表示されている特定のアクセス権限を持っている必要があります。アクセス権限は、アカウント全体のデータ保護ポリシーと、単一のロググループに適用されるデータ保護ポリシーとで異なります。

アカウントレベルのデータ保護ポリシーに必要なアクセス権限

> **Note**
>
> Lambda 関数内でこれらの操作のいずれかを実行する場合、Lambda 実行ロールとアクセス許可の境界には次の権限も含める必要があります。

操作	IAM 権限が必要です	リソース
監査先を指定しないデータ保護ポリシーを作成する	logs:PutAccountPolicy	*
	logs:PutDataProtectionPolicy	*
CloudWatch ログを監査先とするデータ保護ポリシーを作成する	logs:PutAccountPolicy	*
	logs:PutDataProtectionPolicy	*
	logs:CreateLogDelivery	*
	logs:PutResourcePolicy	*
	logs:DescribeResourcePolicies	*
	logs:DescribeLogGroups	*
Firehose を監査先とするデータ保護ポリシーを作成する	logs:PutAccountPolicy	*
	logs:PutDataProtectionPolicy	*
	logs:CreateLogDelivery	*
	firehose:TagDeliveryStream	arn:aws:logs:::deliverystream/ *YOUR_DELIVERY_STREAM*

操作	IAM 権限が必要です	リソース
監査先として Amazon S3 を使用してデータ保護ポリシーを作成する	logs:PutAccountPolicy	*
	logs:PutDataProtectionPolicy	*
	logs:CreateLogDelivery	*
	s3:GetBucketPolicy	arn:aws:s3::: *YOUR_BUCKET*
	s3:PutBucketPolicy	arn:aws:s3::: *YOUR_BUCKET*
指定したロググループのマスクされたログイベントのマスクを外す	logs:Unmask	arn:aws:logs:::log-group:*
既存のデータ保護ポリシーを表示する	logs:GetDataProtectionPolicy	*
データ保護ポリシーを削除する	logs:DeleteAccountPolicy	*
	logs:DeleteDataProtectionPolicy	*

いずれかのデータ保護監査ログがすでに宛先に送信されている場合、同じ宛先にログを送信する他のポリシーに必要なのは logs:PutDataProtectionPolicy および logs:CreateLogDelivery 権限のみです。

1 つのロググループのデータ保護ポリシーに必要なアクセス権限

> **Note**
>
> Lambda 関数内でこれらの操作のいずれかを実行する場合、Lambda 実行ロールとアクセス許可の境界には次の権限も含める必要があります。

操作	IAM 権限が必要です	リソース
監査先を指定しないデータ保護ポリシーを作成する	`logs:PutDataProtectionPolicy`	`arn:aws:logs:::log-group:` *YOUR_LOG_GROUP* `:*`

CloudWatch ログを監査先とするデータ保護ポリシーを作成する	`logs:PutDataProtectionPolicy`
	`logs:CreateLogDelivery`
	`logs:PutResourcePolicy`
	`logs:DescribeResourcePolicies`
	`logs:DescribeLogGroups`
		`arn:aws:logs:::log-group:` *YOUR_LOG_GROUP* `:*`
		`*`
		`*`
		`*`
		`*`

Firehose を監査先とするデータ保護ポリシーを作成する	`logs:PutDataProtectionPolicy`
	`logs:CreateLogDelivery`
	`firehose:TagDeliveryStream`
		`arn:aws:logs:::log-group:` *YOUR_LOG_GROUP* `:*`
		`*`
		`arn:aws:logs:::deliverystream/` *YOUR_DELIVERY_STREAM*

操作	IAM 権限が必要です	リソース
監査先として Amazon S3 を使用してデータ保護ポリシーを作成する	logs:PutDataProtectionPolicy logs:CreateLogDelivery s3:GetBucketPolicy s3:PutBucketPolicy	arn:aws:logs:::log-group: *YOUR_LOG_GROUP* :* * arn:aws:s3::: *YOUR_BUCKET* arn:aws:s3::: *YOUR_BUCKET*
マスクされたログイベントをマスク解除する	logs:Unmask	arn:aws:logs:::log-group: *YOUR_LOG_GROUP* :*
既存のデータ保護ポリシーを表示する	logs:GetDataProtectionPolicy	arn:aws:logs:::log-group: *YOUR_LOG_GROUP* :*
データ保護ポリシーを削除する	logs:DeleteDataProtectionPolicy	arn:aws:logs:::log-group: *YOUR_LOG_GROUP* :*

いずれかのデータ保護監査ログがすでに宛先に送信されている場合、同じ宛先にログを送信する他のポリシーに必要なのは logs:PutDataProtectionPolicy および logs:CreateLogDelivery 権限のみです。

データ保護ポリシーのサンプル

次のサンプルポリシーにより、3 種類の監査先すべてに監査結果を送信できるデータ保護ポリシーを、ユーザーが作成、表示、削除できます。ユーザーはマスクされていないデータを確認することはできません。

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

```
        {
            "Sid": "YOUR_SID_1",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:PutResourcePolicy",
                "logs:DescribeLogGroups",
                "logs:DescribeResourcePolicies"
            ],
            "Resource": "*"
        },
        {
            "Sid": "YOUR_SID_2",
            "Effect": "Allow",
            "Action": [
                "logs:GetDataProtectionPolicy",
                "logs:DeleteDataProtectionPolicy",
                "logs:PutDataProtectionPolicy",
                "s3:PutBucketPolicy",
                "firehose:TagDeliveryStream",
                "s3:GetBucketPolicy"
            ],
            "Resource": [
                "arn:aws:firehose:::deliverystream/YOUR_DELIVERY_STREAM",
                "arn:aws:s3:::YOUR_BUCKET",
                "arn:aws:logs:::log-group:YOUR_LOG_GROUP:*"
            ]
        }
    ]
}
```

アカウント全体のデータ保護ポリシーを作成する

CloudWatch Logs コンソールまたは AWS CLI コマンドを使用して、アカウント内のすべてのロググループの機密データをマスクするデータ保護ポリシーを作成できます。作成すると、現在のロググループと今後作成するロググループの両方に影響します。

> **Important**
>
> 機密データは、ロググループに取り込まれるときに検出され、マスクされます。データ保護ポリシーを設定しても、それ以前にロググループに取り込まれたログイベントはマスクされません。

トピック

- コンソール
- AWS CLI

コンソール

コンソールを使用してアカウント全体のデータ保護ポリシーを作成するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで 設定 を選択します。リストの一番下付近にあります。
3. [ログ] タブを選択します。
4. [設定] を選択します。
5. マネージドデータ識別子 で、すべてのロググループに対して監査およびマスクするデータのタイプを選択します。選択ボックスに入力して、必要な識別子を見つけることができます。

   ログデータやビジネスに関連するデータ識別子のみを選択することをお勧めします。多くの種類のデータを選択すると、誤検出につながる可能性があります。

   保護できるデータの種類の詳細については、「保護できるデータの種類」を参照してください。
6. (オプション) カスタムデータ識別子を使用して他のタイプのデータを監査およびマスクする場合は、カスタムデータ識別子を追加 を選択します。次に、ログイベントでそのタイプのデータを検索するために使用するデータ型の名前と正規表現を入力します。詳細については、「カスタムデータ識別子」を参照してください。

   1 つのデータ保護ポリシーには、最大 10 個のカスタムデータ識別子を含めることができます。カスタムデータ識別子を定義する各正規表現は、200 文字以下である必要があります。
7. (オプション) 監査結果の送信先となるサービスを 1 つまたは複数選択します。監査結果をこれらのサービスのいずれにも送信しないことを選択した場合でも、選択した機密データタイプは引き続きマスクされます。

8. [Activate data protection] (データ保護をアクティブにする) を選択します。

AWS CLI

を使用してデータ保護ポリシー AWS CLI を作成するには

1. テキストエディタを使用して DataProtectionPolicy.json という名前のポリシーファイルを作成します。ポリシーの構文については、次のセクションを参照してください。
2. 次のコマンドを入力します。

```
aws logs put-account-policy \
--policy-name TEST_POLICY --policy-type "DATA_PROTECTION_POLICY" \
--policy-document file://policy.json \
--scope "ALL" \
--region us-west-2
```

AWS CLI または API オペレーションのデータ保護ポリシー構文

AWS CLI コマンドまたは API オペレーションで使用する JSON データ保護ポリシーを作成する場合、ポリシーには 2 つの JSON ブロックを含める必要があります。

- 最初のブロックには、DataIdentifer 配列と Audit アクションを含む Operation プロパティの両方が含まれている必要があります。DataIdentifer 配列には、マスクする機密データの種類が表示されます。利用できるすべてのオプションについての詳細は、「保護できるデータの種類」を参照してください。

  Audit アクションを含む Operation プロパティは、機密データ用語を検索するために必要です。この Audit アクションには FindingsDestination オブジェクトが含まれている必要があります。オプションで FindingsDestination オブジェクトを使用して、監査結果レポートの送信先を 1 つ、または複数リストできます。ロググループ、Amazon Data Firehose ストリーム、S3 バケットなどの送信先を指定する場合は、それらがすでに存在している必要があります。監査結果レポートの例については、「監査結果レポート」を参照してください。
- 2 番目のブロックには、DataIdentifer 配列と Deidentify アクションを含む Operation プロパティの両方が含まれている必要があります。DataIdentifer 配列は、ポリシーの最初のブロックにある DataIdentifer 配列と完全に一致する必要があります。

Deidentify アクションを含む Operation プロパティが実際にデータをマスクするものであり、そのアクションには "MaskConfig": {} オブジェクトが含まれている必要があります。"MaskConfig": {} オブジェクトは空である必要があります。

以下は、マネージドデータ識別子のみを使用するデータ保護ポリシーの例です。このポリシーは、Eメールアドレスと米国の運転免許証をマスクします。

カスタムデータ識別子を指定するポリシーの詳細については、「」を参照してください[データ保護ポリシーでカスタムデータ識別子を使用する](#)。

```
{
    "Name": "data-protection-policy",
    "Description": "test description",
    "Version": "2021-06-01",
    "Statement": [{
            "Sid": "audit-policy",
            "DataIdentifier": [
                "arn:aws:dataprotection::aws:data-identifier/EmailAddress",
                "arn:aws:dataprotection::aws:data-identifier/DriversLicense-US"
            ],
            "Operation": {
                "Audit": {
                    "FindingsDestination": {
                        "CloudWatchLogs": {
                            "LogGroup": "EXISTING_LOG_GROUP_IN_YOUR_ACCOUNT,"
                        },
                        "Firehose": {
                            "DeliveryStream": "EXISTING_STREAM_IN_YOUR_ACCOUNT"
                        },
                        "S3": {
                            "Bucket": "EXISTING_BUCKET"
                        }
                    }
                }
            }
        },
        {
            "Sid": "redact-policy",
            "DataIdentifier": [
                "arn:aws:dataprotection::aws:data-identifier/EmailAddress",
                "arn:aws:dataprotection::aws:data-identifier/DriversLicense-US"
```

```
            ],
            "Operation": {
                "Deidentify": {
                    "MaskConfig": {}
                }
            }
        }
    ]
}
```

1 つのロググループ用のデータ保護ポリシーを作成する

CloudWatch Logs コンソールまたは AWS CLI コマンドを使用して、機密データをマスクするデータ保護ポリシーを作成できます。

各ロググループに 1 つのデータ保護ポリシーを割り当てることができます。各データ保護ポリシーで、複数の種類の情報を監査できます。各データ保護ポリシーには、監査ステートメントを 1 つ含めることができます。

トピック

- コンソール
- AWS CLI

コンソール

コンソールを使用してデータ保護ポリシーを作成するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[ログ]、[ロググループ] の順に選択します。
3. ロググループの名前を選択します。
4. [Actions] (アクション)、[Create data protection policy] (データ保護ポリシーを作成) を選択します。
5. マネージドデータ識別子 で、このロググループで監査およびマスクするデータのタイプを選択します。選択ボックスに入力して、必要な識別子を見つけることができます。

   ログデータやビジネスに関連するデータ識別子のみを選択することをお勧めします。多くの種類のデータを選択すると、誤検出につながる可能性があります。

マネージドデータ識別子を使用して保護できるデータのタイプの詳細については、「」を参照してください保護できるデータの種類。

6. (オプション) カスタムデータ識別子を使用して他のタイプのデータを監査およびマスクする場合は、カスタムデータ識別子を追加 を選択します。次に、ログイベントでそのタイプのデータを検索するために使用するデータ型の名前と正規表現を入力します。詳細については、「カスタムデータ識別子」を参照してください。

   1 つのデータ保護ポリシーには、最大 10 個のカスタムデータ識別子を含めることができます。カスタムデータ識別子を定義する各正規表現は、200 文字以下である必要があります。

7. (オプション) 監査結果の送信先となるサービスを 1 つまたは複数選択します。監査結果をこれらのサービスのいずれにも送信しないことを選択した場合でも、選択した機密データタイプは引き続きマスクされます。

8. [Activate data protection] (データ保護をアクティブにする) を選択します。

AWS CLI

を使用してデータ保護ポリシー AWS CLI を作成するには

1. テキストエディタを使用して DataProtectionPolicy.json という名前のポリシーファイルを作成します。ポリシーの構文については、次のセクションを参照してください。

2. 次のコマンドを入力します。

```
aws logs put-data-protection-policy --log-group-identifier "my-log-group" --policy-
document file:///Path/DataProtectionPolicy.json --region us-west-2
```

AWS CLI または API オペレーションのデータ保護ポリシー構文

AWS CLI コマンドまたは API オペレーションで使用する JSON データ保護ポリシーを作成する場合、ポリシーには 2 つの JSON ブロックを含める必要があります。

- 最初のブロックには、DataIdentifer 配列と Audit アクションを含む Operation プロパティの両方が含まれている必要があります。DataIdentifer 配列には、マスクする機密データの種類が表示されます。利用できるすべてのオプションについての詳細は、「保護できるデータの種類」を参照してください。

Audit アクションを含む Operation プロパティは、機密データ用語を検索するために必要です。この Audit アクションには FindingsDestination オブジェクトが含まれている必要があります。オプションで FindingsDestination オブジェクトを使用して、監査結果レポートの送信先を 1 つ、または複数リストできます。ロググループ、Amazon Data Firehose ストリーム、S3 バケットなどの送信先を指定する場合は、それらがすでに存在している必要があります。監査結果レポートの例については、「監査結果レポート」を参照してください。

- 2 番目のブロックには、DataIdentifer 配列と Deidentify アクションを含む Operation プロパティの両方が含まれている必要があります。DataIdentifer 配列は、ポリシーの最初のブロックにある DataIdentifer 配列と完全に一致する必要があります。

  Deidentify アクションを含む Operation プロパティが実際にデータをマスクするものであり、そのアクションには "MaskConfig": {} オブジェクトが含まれている必要があります。"MaskConfig": {} オブジェクトは空である必要があります。

E メールアドレスと米国の運転免許証をマスクするデータ保護ポリシーの例を次に示します。

```
{
    "Name": "data-protection-policy",
    "Description": "test description",
    "Version": "2021-06-01",
    "Statement": [{
            "Sid": "audit-policy",
            "DataIdentifier": [
                "arn:aws:dataprotection::aws:data-identifier/EmailAddress",
                "arn:aws:dataprotection::aws:data-identifier/DriversLicense-US"
            ],
            "Operation": {
                "Audit": {
                    "FindingsDestination": {
                        "CloudWatchLogs": {
                            "LogGroup": "EXISTING_LOG_GROUP_IN_YOUR_ACCOUNT,"
                        },
                        "Firehose": {
                            "DeliveryStream": "EXISTING_STREAM_IN_YOUR_ACCOUNT"
                        },
                        "S3": {
                            "Bucket": "EXISTING_BUCKET"
                        }
                    }
                }
```

```
            }
        },
        {
            "Sid": "redact-policy",
            "DataIdentifier": [
                "arn:aws:dataprotection::aws:data-identifier/EmailAddress",
                "arn:aws:dataprotection::aws:data-identifier/DriversLicense-US"
            ],
            "Operation": {
                "Deidentify": {
                    "MaskConfig": {}
                }
            }
        }
    ]
}
```

データをマスクせずに表示する

データをマスクせずに表示するには、ユーザーに `logs:Unmask` アクセス許可が必要です。このアクセス許可を持つユーザーは、次の方法でデータをマスクせずに表示できます。

- ログストリームでイベントを表示するときは、[Display] (表示)、[Unmask] (マスク解除) を選択します。
- unmask(@message) コマンドを含む CloudWatch Logs Insights クエリを使用します。次のクエリ例では、ストリーム内の最新の 20 件のログイベントがマスクされずに表示されます。

  ```
  fields @timestamp, @message, unmask(@message)
  | sort @timestamp desc
  | limit 20
  ```

  CloudWatch Logs Insights コマンドの詳細については、「」を参照してくださいCloudWatch Logs Insights クエリ構文。
- unmask パラメータで GetLogEventsまたは FilterLogEventsオペレーションを使用します。

CloudWatchLogsFullAccess ポリシーには アクセス`logs:Unmask`許可が含まれます。を持たないユーザーに を付与`logs:Unmask`するにはCloudWatchLogsFullAccess、そのユーザーにカスタム IAM ポリシーをアタッチします。詳細については、「ユーザーへのアクセス許可の追加 (コンソール)」を参照してください。

監査結果レポート

CloudWatch Logs、Amazon S3、または Firehose に監査レポートを書き込むように CloudWatch Logs データ保護監査ポリシーを設定すると、これらの検出結果レポートは次の例のようになります。 CloudWatch Logs は、機密データを含む口グイベントごとに 1 つの検出結果レポートを書き込みます。

```
{
    "auditTimestamp": "2023-01-23T21:11:20Z",
    "resourceArn": "arn:aws:logs:us-west-2:111122223333:log-group:/aws/lambda/
MyLogGroup:*",
    "dataIdentifiers": [
        {
            "name": "EmailAddress",
            "count": 2,
            "detections": [
                {
                    "start": 13,
                    "end": 26
                },
{
                    "start": 30,
                    "end": 43
                }
            ]
        }
    ]
}
```

レポート内のフィールドは、以下のとおりです。

- resourceArn フィールドには、機密データが見つかった口グループが表示されます。
- dataIdentifiers オブジェクトには、監査している機密データのタイプの 1 つに関する結果が表示されます。
- name フィールドには、このセクションで報告されている機密データのタイプが特定されています。
- count フィールドには、口グイベントでこのタイプの機密データが出現する回数が表示されます。
- start および end フィールドには、機密データが口グイベントのどこで出現しているかが、出現ごとに文字数で表示されます。

上記の例は、1 つのログイベントで 2 件の E メールアドレスが見つかったレポートです。最初の E メールアドレスは、ログイベントの 13 文字目から始まり、26 文字目で終わります。2 番目のメールアドレスは 30 文字目から 43 文字目までとなっています。このログイベントには 2 件の E メールアドレスがありますが、`LogEventsWithFindings` メトリクスの値は 1 つしかインクリメントされていません。これは、メトリクスが数えているのが機密データの出現回数ではなく、機密データが含まれるログイベントの数だからです。

で保護された バケットに監査結果を送信するために必要なキーポリシー AWS KMS

Amazon S3 バケット内のデータを保護するには、Amazon S3 マネージドキーを使用したサーバー側の暗号化 (SSE-S3)、または KMS キーを使用したサーバー側の暗号化 (SSE-KMS) のいずれかを有効にします。詳細については、Amazon S3 ユーザーガイドの「サーバー側の暗号化を使用したデータの保護」を参照してください。

SSE-S3 で保護されているバケットに監査結果を送信した場合、追加の設定は必要ありません。Amazon S3 が暗号化キーを処理します。

SSE-KMS で保護されているバケットに監査結果を送信すると、ログ配信アカウントが S3 バケットに書き込めるように、KMS キーのキーポリシーを更新する必要があります。SSE-KMS で使用するために必要なキーポリシーの詳細については、「Amazon CloudWatch Logs ユーザーガイドAmazon S3」の「」を参照してください。

保護できるデータの種類

このセクションでは、 CloudWatch Logs データ保護ポリシーで保護できるデータの種類について説明します。 CloudWatch Logs マネージドデータ識別子は、財務データ、個人健康情報 (PHI)、および個人を特定できる情報 (PII) を保護するために事前設定されたデータ型を提供します。カスタムデータ識別子を使用して、特定のユースケースに合わせたデータ識別子を作成することもできます。

目次

- 保護対象保健情報 (PHI)
  - 保護対象の医療情報 (PHI) データタイプのデータ識別子 ARN
- 個人を特定できる情報 (PII)
  - 運転免許証識別番号のキーワード
  - 国民識別番号のキーワード
  - パスポート番号のキーワード
  - 納税者識別と参照番号のキーワード
  - 個人を特定できる情報 (PII) のデータ識別子 ARN
- カスタムデータ識別子
  - SNS カスタムデータ識別子とは
  - カスタムデータ識別子の制約
  - コンソールでのカスタムデータ識別子の使用
  - データ保護ポリシーでカスタムデータ識別子を使用する

CloudWatch 機密データタイプのマネージドデータ識別子を口グに記録する

このセクションでは、マネージドデータ識別子を使用して保護できるデータの種類と、それらの各種類のデータに関連する国と地域について説明します。

一部のタイプの機密データでは、 CloudWatch ログデータ保護はデータの近くにあるキーワードをスキャンし、そのキーワードが見つかった場合にのみ一致を検索します。キーワードが特定のタイプのデータの近くにある必要がある場合は、通常、キーワードはデータから 30 文字以内 (包括的) になければなりません。

キーワードにスペースが含まれている場合、 CloudWatch Logs データ保護は、スペースが欠落しているか、スペースの代わりにアンダースコア (_) またはハイフン (-) を含むキーワードバリエーションを自動的に一致させます。場合によっては、 CloudWatch Logs はキーワードの一般的なバリエーションに対応するためにキーワードを展開または省略します。

次の表に、Logs がマネージドデータ識別子を使用して検出できる認証情報、デバイス、財務、医療、保護医療情報 (PHI) CloudWatch のタイプを示します。これらは、個人を特定できる情報 (PII) としても認定される可能性のある特定のタイプのデータに加えたものです。

言語や地域に依存しないサポート対象の識別子

識別子	カテゴリ
Address	個人
AwsSecretKey	認証情報
CreditCardExpiration	金融
CreditCardNumber	金融
CreditCardSecurityCode	金融
EmailAddress	個人
IpAddress	個人
LatLong	個人
Name	個人
OpenSshPrivateKey	認証情報
PgpPrivateKey	認証情報
PkcsPrivateKey	認証情報
PuttyPrivateKey	認証情報
VehicleIdentificationNumber	個人

地域に依存するデータ識別子には、識別子名に続けてハイフンと 2 文字のコード (ISO 3166-1 alpha-2) が必要です。例えば DriversLicense-US です。

2 文字の国コードまたは地域コードを含む必要があるサポート対象の識別子

識別子	カテゴリ	国と言語
BankAccountNumber	金融	DE、ES、FR、GB、IT
CepCode	個人	BR

識別子	カテゴリ	国と言語
Cnpj	個人	BR
CpfCode	個人	BR
DriversLicense	個人	AT、AU、BE、 BG、CA、CY、 CZ、DE、DK、EE、ES、FI、 FR、GB、GR、 HR、HU、IE、IT、LT、LU、 LV、MT、NL、 PL、PT、RO、SE、SI、SK、 US
DrugEnforcementAgencyNumber	健康	米国
ElectoralRollNumber	個人	GB
HealthInsuranceCardNumber	健康	EU
HealthInsuranceClaimNumber	健康	米国
HealthInsuranceNumber	健康	FR
HealthcareProcedureCode	健康	米国
IndividualTaxIdentificationNumber	個人	米国
InseeCode	個人	FR
MedicareBeneficiaryNumber	健康	米国
NationalDrugCode	健康	米国
NationalIdentificationNumber	個人	DE、ES、IT
NationalInsuranceNumber	個人	GB

識別子	カテゴリ	国と言語
NationalProviderId	健康	米国
NhsNumber	健康	GB
NieNumber	個人	ES
NifNumber	個人	ES
PassportNumber	個人	CA、DE、ES、 FR、GB、IT、US
PermanentResidenceNumber	個人	CA
PersonalHealthNumber	健康	CA
PhoneNumber	個人	BR、DE、ES、 FR、GB、IT、US
PostalCode	個人	CA
RgNumber	個人	BR
SocialInsuranceNumber	個人	CA
SSN	個人	es-US
TaxId	個人	DE、ES、FR、GB
ZipCode	個人	米国

認証情報

CloudWatch ログデータ保護では、次のタイプの認証情報を見つけることができます。

データの種類	データ識別子 ID	キーワードが必須	国とリージョン
AWS シークレットアクセスキー	AwsSecretKey	aws_secret_access_key , credentials , secret access key, secret key, set-awscredential	すべて
OpenSSH プライベートキー	OpenSSHPrivateKey	なし	すべて
PGP プライベートキー	PgpPrivateKey	なし	すべて
Pkcs プライベートキー	PkcsPrivateKey	なし	すべて
PuTTY プライベートキー	PuttyPrivateKey	なし	すべて

認証情報データタイプのデータ識別子 ARN

以下は、データ保護ポリシーに追加できるデータ識別子の Amazon リソースネーム (ARN) のリストを示しています。

認証情報データ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/AwsSecretKey
arn:aws:dataprotection::aws:data-identifier/OpenSshPrivateKey
arn:aws:dataprotection::aws:data-identifier/PgpPrivateKey
arn:aws:dataprotection::aws:data-identifier/PkcsPrivateKey
arn:aws:dataprotection::aws:data-identifier/PuttyPrivateKey

デバイス識別子

CloudWatch ログデータ保護では、次のタイプのデバイス識別子を見つけることができます。

データの種類	データ識別子 ID	キーワードが必須	国とリージョン
IP アドレス	`IpAddress`	なし	すべて

デバイスデータタイプのデータ識別子 ARN

以下は、データ保護ポリシーに追加できるデータ識別子の Amazon リソースネーム (ARN) のリストを示しています。

デバイスデータ識別子 ARN
`arn:aws:dataprotection::aws:data-identifier/IpAddress`

財務情報

CloudWatch ログデータ保護では、次の種類の財務情報を見つけることができます。

データ保護ポリシーを設定すると、 CloudWatch ロググループはどのジオロケーションにあるかに関係なく、指定したデータ識別子をスキャンします。この表の国と地域列の情報は、データ識別子に 2 文字の国コードを追加して、それらの国や地域に適したキーワードを検出する必要があるかどうかを示しています。

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
銀行口座番号	`BankAccountNumber`	はい。国によって適用されるキーワードは異なります。詳細については、このセクションの後半にある銀	フランス、ドイツ、イタリア、ス	国コードなどの要素を含む、最

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
		行口座番号のキーワードの表を参照してください。	ペイン、英国	大 34 文字の英数字で構成される国際銀行口座番号 (IBAN) が含まれます。
クレジットカードの有効期限	`CreditCardExpiration`	`exp d`, `exp m`, `exp y`, `expiration` , `expiry`	すべて	

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
クレジットカード番号	CreditCardNumber	account number, american express, amex, bank card, card, card number, card num, cc #, ccn, check card, credit, credit card#, dankort, debit, debit card, diners club, discover, electron, japanese card bureau, jcb, mastercard , mc, pan, payment account number, payment card number, pcn, union pay, visa	すべて	検出では、データは Luhn チェック式に準拠する 13 ~ 19 桁のシーケンスである必要があります。また、American Express、Dankort、Diner's Club、Discover、Electron、日本語カード局 (JCB)、Mastercard UnionPay、Visa

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
				のいずれかのタイプのクレジットカードに標準のカード番号プレフィックスを使用します。
クレジットカード認証コード	CreditCardSecurityCode	card id, card identification code, card identification number , card security code, card validation code , card validation number , card verification data , card verification value, cvc, cvc2, cvv, cvv2, elo verification code	すべて	

銀行口座番号のキーワード

次のキーワードを使用して、国コードなどの要素を含む、最大 34 文字の英数字で構成される国際銀行口座番号 (IBAN) を検出します。

国	キーワード
フランス	account code, account number, accountno# , accountnumber# , bban, code bancaire, compte bancaire, customer account id, customer account number, customer bank account id, iban, numéro de compte
ドイツ	account code, account number, accountno# , accountnumber# , bankleitzahl , bban, customer account id, customer account number, customer bank account id, geheimzahl , iban, kartennummer , kontonummer , kreditkartennummer , sepa
イタリア	account code, account number, accountno# , accountnumber# , bban, codice bancario, conto bancario, customer account id, customer account number, customer bank account id, iban, numero di conto
スペイン	account code, account number, accountno# , accountnumber# , bban, código cuenta, código cuenta bancaria, cuenta cliente id, customer account ID, customer account number, customer bank account id, iban, número cuenta bancaria cliente, número cuenta cliente
英国	account code, account number, accountno# , accountnumber# , bban, customer account ID, customer account number, customer bank account id, iban, sepa
アメリカ	bank account, bank acct, checking account, checking acct, deposit account, deposit acct, savings account, savings acct, chequing account, chequing acct

CloudWatch ログは、クレジットカード発行者がパブリックテスト用に予約している次のシーケンスの出現を報告しません。

```
122000000000003, 2222405343248877, 2222990905257051, 2223007648726984,
 2223577120017656,
30569309025904, 34343434343434, 3528000700000000, 3530111333300000, 3566002020360505,
 36148900647913,
```

```
36700102000000, 371449635398431, 378282246310005, 378734493671000, 38520000023237,
 4012888888881881,
4111111111111111, 4222222222222, 4444333322221111, 4462030000000000, 4484070000000000,
 4911830000000,
4917300800000000, 4917610000000000, 4917610000000000003, 5019717010103742,
 5105105105105100,
5111010030175156, 5185540810000019, 5200828282828210, 5204230080000017,
 5204740009900014, 5420923878724339,
5454545454545454, 5455330760000018, 5506900490000436, 5506900490000444,
 5506900510000234, 5506920809243667,
5506922400634930, 5506927427317625, 5553042241984105, 5555553753048194,
 5555555555554444, 5610591081018250,
6011000990139424, 6011000400000000, 6011111111111117, 630490017740292441,
 630495060000000000,
6331101999990016, 6759649826438453, 6799990100000000019, and 76009244561.
```

財務データタイプのデータ識別子 ARN

以下は、データ保護ポリシーに追加できるデータ識別子の Amazon リソースネーム (ARN) のリストを示しています。

財務データ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-DE
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-ES
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-FR
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-GB
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-IT
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-US
arn:aws:dataprotection::aws:data-identifier/CreditCardExpiration
arn:aws:dataprotection::aws:data-identifier/CreditCardNumber
arn:aws:dataprotection::aws:data-identifier/CreditCardSecurityC ode

保護対象保健情報 (PHI)

CloudWatch ログデータ保護では、次のタイプの保護対象医療情報 (PHI) を検索できます。

データ保護ポリシーを設定すると、 CloudWatch ロググループはどのジオロケーションにあるかに関係なく、指定したデータ識別子をスキャンします。この表の国と地域列の情報は、データ識別子に 2 文字の国コードを追加して、それらの国や地域に適したキーワードを検出する必要があるかどうかを示しています。

データの種類	データ識別子 ID	キーワードが必須	国とリージョン
麻薬取締局 (DEA) 登録番号	DrugEnforcementAgencyNumber	dea number, dea registration	アメリカ
健康保険証番号 (EHIC)	HealthInsuranceCardNumber	assicurazione sanitaria numero, carta assicurazione numero, carte d’assurance maladie , carte européenne d'assurance maladie , ceam, ehic, ehic#, finlandehicnumber# , gesundheitskarte , hälsokort , health card, health card number, health insurance card, health insurance number, insurance card number, krankenversicherungskarte , krankenversicherungsnummer , medical account number, numero conto medico, numéro d’assuran	欧州連合

データの種類	データ識別子 ID	キーワードが必須	国とリージョン
		ce maladie , numéro de carte d’assurance , numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanhoitokortin , sairausvakuutuskortti , sairausvakuutusnumero , sjukförsäkringnummer, sjukförsäkringskort , suomi ehic-numero , tarjeta de salud, terveyskortti , tessera sanitaria assicurazione numero , versicherungsnummer	
健康保険請求番号 (HICN)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#, hicno#	アメリカ
健康保険または医療識別番号	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	フランス

データの種類	データ識別子 ID	キーワードが必須	国とリージョン
ヘルスケア共通手順コーディングシステム (HCPCS) コード	HealthcareProcedureCode	current procedural terminology , hcpcs, healthcare common procedure coding system	アメリカ
メディケア受給者番号 (MBN)	MedicareBeneficiaryNumber	mbi, medicare beneficiary	アメリカ
全米医薬品コード (NDC)	NationalDrugCode	national drug code, ndc	アメリカ
国家プロバイダー識別子 (NPI)	NationalProviderId	hipaa, n.p.i., national provider, npi	アメリカ
国民保健サービス (NHS) 番号	NhsNumber	national health service, NHS	グレートブリテン
個人健康管理番号	PersonalHealthNumber	canada healthcare number, msp number, care number, phn, soins de santé	カナダ

保護対象の医療情報 (PHI) データタイプのデータ識別子 ARN

保護対象保健情報 (PHI) データ保護ポリシーで使用できるデータ識別子 Amazon リソースネーム (ARN) を次に示します。

PHI データ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/DrugEnforcementAgencyNumber-US
arn:aws:dataprotection::aws:data-identifier/HealthcareProcedureCode-US
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceCardNumber-EU
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceClaimNumber-US
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceNumber-FR
arn:aws:dataprotection::aws:data-identifier/MedicareBeneficiaryNumber-US
arn:aws:dataprotection::aws:data-identifier/NationalDrugCode-US
arn:aws:dataprotection::aws:data-identifier/NationalInsuranceNumber-GB
arn:aws:dataprotection::aws:data-identifier/NationalProviderId-US
arn:aws:dataprotection::aws:data-identifier/NhsNumber-GB
arn:aws:dataprotection::aws:data-identifier/PersonalHealthNumber-CA

個人を特定できる情報 (PII)

CloudWatch ログデータ保護では、以下の種類の個人を特定できる情報 (PII) を見つけることができます。

データ保護ポリシーを設定すると、 CloudWatch ロググループはどのジオロケーションにあるかに関係なく、指定したデータ識別子をスキャンします。この表の国と地域列の情報は、データ識別子に 2

文字の国コードを追加して、それらの国や地域に適したキーワードを検出する必要があるかどうかを示しています。

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
生年月日	DateOfBirth	dob, date of birth, birthdate , birth date, birthday, b-day, bday	すべて	Supportには、すべての数字や数字と月の名前の組み合わせなど、ほとんどの日付形式が含まれます。日付コンポーネントは、スペース、スラッシュ (/)、またはハイフン (-) で区切ることが

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
				できます。
Código de Endereçamento Postal (CEP)	CepCode	cep, código de endereçamento postal, codigo de endereçamento postal	ブラジル	
Cadastro Nacional da Pessoa Jurídica (CNPJ)	Cnpj	cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj	ブラジル	
Cadastro de Pessoas Físicas (CPF)	CpfCode	Cadastro de pessoas fisicas, cadastro de pessoas físicas, cadastro de pessoa física, cadastro de pessoa fisica, cpf	ブラジル	
運転免許証識別番号	DriversLicense	はい。国によって適用されるキーワードは異なります。詳細については、このセクションの後半にある運転免許証識別番号の表を参照してください。	多くの国。詳細については、運転免許証識別番号の表を参照してください。	

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
選挙人名簿番号	ElectoralRollNumber	electoral #, electoral number, electoral roll #, electoral roll no., electoral roll number, electoral rollno	英国	
個人納税者識別	IndividualTaxIdentificationNumber	はい。国によって適用されるキーワードは異なります。詳細については、このセクションの後半にある個人納税者識別番号の表を参照してください。	ブラジル、フランス、ドイツ、スペイン、英国	
国立統計経済研究所 (INSEE)	InseeCode	はい。国によって適用されるキーワードは異なります。詳細については、このセクションの後半にある国民識別番号のキーワードの表を参照してください。	フランス	

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
国民識別番号	`NationalIdentificationNumber`	はい。詳細については、このセクションの後半にある国民識別番号のキーワードの表を参照してください。	ドイツ、イタリア、スペイン	これには、Documento Nacional de Identidad (DNI) 識別子 (スペイン)、Codice fiscale codes (イタリア)、国民 ID カード番号 (ドイツ語) が含まれます。

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
国民保険番号 (NINO)	NationalInsuranceNumber	insurance no., insurance number, insurance# , national insurance number, nationalinsurance# , nationalinsurancenumber , nin, nino	英国	–
Número de identidad de extranjero (NIE)	NieNumber	はい。国によって適用されるキーワードは異なります。詳細については、このセクションの後半にある個人納税者識別番号の表を参照してください。	スペイン	
Número de Identificación Fiscal (NIF)	NifNumber	はい。国によって適用されるキーワードは異なります。詳細については、このセクションの後半にある個人納税者識別番号の表を参照してください。	スペイン	

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
パスポート番号	PassportNumber	はい。国によって適用されるキーワードは異なります。詳細については、このセクションの後半にあるパスポート番号のキーワードの表を参照してください。	カナダ、フランス、ドイツ、イタリア、スペイン、英国、米国	
本籍地	PermanentResidenceNumber	carte résident permanent , numéro carte résident permanent , numéro résident permanent , permanent resident card, permanent resident card number, permanent resident no, permanent resident no., permanent resident number, pr no, pr no., pr non, pr number, résident permanent no., résident permanent non	カナダ	

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
電話番号	PhoneNumber	ブラジル: キーワードには、cel、celular、fone、residencial 、numero residencial 、telefone も含まれます。 その他: cell、contact、fax、fax number、mobile、phone、number、tel、telephone 、telephone number	ブラジル、カナダ、フランス、ドイツ、イタリア、スペイン、英国、米国	これには、米国の通話料無料の番号とファックス番号が含まれます。キーワードがデータの近くにある場合、番号に国コードを含める必要はありません。キーワードがデータの近くにない場合は、番

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
				号に国コードを含める必要があります。
郵便番号	PostalCode	なし	カナダ	
Registro Geral (RG)	RgNumber	はい。国によって適用されるキーワードは異なります。詳細については、このセクションの後半にある個人納税者識別番号の表を参照してください。	ブラジル	
社会保険番号 (SIN)	SocialInsuranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	カナダ	
社会保障番号 (SSN)	Ssn	スペイン – número de la seguridad social、social security no.、social security no、número de la seguridad social、social security number、socialsecurityno# 、ssn、ssn# 米国 – social security、ss#、ssn	スペイン、米国	

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
納税者識別番号または参照番号	`TaxId`	はい。国によって適用されるキーワードは異なります。詳細については、このセクションの後半にある個人納税者識別番号の表を参照してください。 .	フランス、ドイツ、スペイン、英国	これには、TIN (フランス)、Steueridentifikationsnummer (ドイツ)、CIF (スペイン)、TRN と UTR (英国) が含まれます。
ZIP コード	`ZipCode`	`zip code`, `zip+4`	アメリカ	米国の郵便番号。

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
郵送先住所	`Address`	なし	オーストラリア、カナダ、フランス、ドイツ、イタリア、スペイン、英国、米国	キーワードは必要ありませんが、検出では、住所には都市または場所の名前および ZIP コードまたは郵便番号を含める必要があります。
電子メールアドレス	`EmailAddress`	なし	すべて	

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
全地球測位システム (GPS) 座標	LatLong	coordinate , coordinates , lat long, latitude longitude , location, position	すべて	CloudWatch ログは、緯度と経度の座標がペアとして保存され、41.948614,-87.655311 などの 10 進数 (DD) 形式である場合に GPS 座標を検出できます。サポートには、度 10 進分 (DDM) 形式 (例: 41°56.916

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
				8'N 87°39.3187'W)、または、度、分、秒 (DMS) 形式 (例: 41°56'55.0104"N 87°39'19.1196"W) の座標は含まれません。

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
フルネーム	Name	なし	すべて	CloudWatch ログはフルネームのみを検出できます。Support はラテン文字セットに限定されます。

データの種類	データ識別子 ID	キーワードが必須	国とリージョン	メモ
車両識別番号 (VIN)	VehicleIdentificationNumber	Fahrgestellnummer , niv, numarul de identificare , numarul seriei de sasiu, serie sasiu, numer VIN, Número de Identificação do Veículo, Número de Identificación de Automóviles , numéro d'identification du véhicule, vehicle identification number, vin, VIN numeris	すべて	CloudWatch ログは、17 文字のシーケンスで構成され、ISO 3779 および 3780 標準に準拠する VINs を検出できます。これらの規格は、世界中で使用するために設計されています。

運転免許証識別番号のキーワード

さまざまなタイプの運転免許証識別番号を検出するために、 CloudWatch Logs では番号の近くにあるキーワードが必要です。次の表に、 CloudWatch Logs が特定の国とリージョンについて認識するキーワードを示します。

国またはリージョン	キーワード
オーストラリア	dl# dl:, dl :, dlno# driver licence, driver license, driver permit, drivers lic., drivers licence, driver's licence, drivers license, driver's license, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit
オーストリア	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik osterreich
ベルギー	fuehrerschein, fuehrerschein- nr, fuehrerscheinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrerscheinnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
ブルガリア	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
カナダ	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit,

国またはリージョン	キーワード
	drivers permit number, driving licence, driving license, driving permit, permis de conduire
クロアチア	vozačka dozvola
キプロス	άδεια οδήγησης
チェコ共和国	číslo licence, císlo licence řidiče, číslo řidičského průkazu, ovladače lic., povolení k jízdě, povolení řidiče, řidiči povolení, řidičský prúkaz, řidičský průkaz
デンマーク	kørekort, kørekortnummer
エストニア	juhi litsentsi number, juhiloa number, juhiluba, juhiluba number
フィンランド	ajokortin numero, ajokortti, förare lic., körkort, körkort nummer, kuljettaja lic., permis de conduire
フランス	permis de conduire
ドイツ	fuehrerschein, fuehrerschein- nr, fuehrerschein, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrerschein, führerscheinnummer
ギリシャ	δεια οδήγησης, adeia odigisis
ハンガリー	illesztőprogramok lic, jogosítvány, jogsi, licencszám, vezető engedély, vezetői engedély
アイルランド	ceadúnas tiomána
イタリア	patente di guida, patente di guida numero, patente guida, patente guida numero

国またはリージョン	キーワード
ラトビア	autovadītāja apliecība, licences numurs, vadītāja apliecība, vadītāja apliecības numurs, vadītāja atļauja, vadītāja licences numurs, vadītāji lic.
リトアニア	vairuotojo pažymėjimas
ルクセンブルグ	fahrerlaubnis, führerschäin
マルタ	liċenzja tas-sewqan
オランダ	permis de conduire, rijbewijs, rijbewijsnummer
ポーランド	numer licencyjny, prawo jazdy, zezwolenie na prowadzenie
ポルトガル	carta de condução, carteira de habilitação, carteira de motorist, carteira habilitação, carteira motorist, licença condução, licença de condução, número de licença, número licença, permissão condução, permissão de condução
ルーマニア	numărul permisului de conducere, permis de conducere
スロバキア	číslo licencie, číslo vodičského preukazu, ovládače lic., povolenia vodičov, povolenie jazdu, povolenie na jazdu, povolenie vodiča, vodičský preukaz
スロベニア	vozniško dovoljenje

国またはリージョン	キーワード
スペイン	carnet conducer, el carnet de conducer, licencia conducer, licencia de manejo, número carnet conducer, número de carnet de conducer, número de permiso conducer, número de permiso de conducer, número licencia conducer, número permiso conducer, permiso conducción, permiso conducer, permiso de conducción
スウェーデン	ajokortin numero, dlno# ajokortti, drivere lic., förare lic., körkort, körkort nummer, körkortsn ummer, kuljettajat lic.
英国	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit
アメリカ	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit

国民識別番号のキーワード

さまざまなタイプの国民識別番号を検出するために、 CloudWatch Logs では番号に近接したキーワードが必要です。これには、Documento Nacional de Identidad (DNI) 識別子 (スペイン)、フランス

国立統計経済研究所 (INSEE) コード、ドイツの国民 ID カード番号、Registro Geral (RG) 番号 (ブラジル) が含まれます。

次の表に、 CloudWatch Logs が特定の国とリージョンについて認識するキーワードを示します。

国またはリージョン	キーワード
ブラジル	registro geral, rg
フランス	assurance sociale, carte nationale d’identité, cni, code sécurité sociale, French social security number, fssn#, insee, insurance number, national id number, nationalid#, numéro d'assurance, sécurité sociale, sécurité sociale non., sécurité sociale numéro, social, social security, social security number, socialsecuritynumber, ss#, ssn, ssn#
ドイツ	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
イタリア	codice fiscal, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
スペイン	dni, dni#, dninúmero#, documento nacional de identidad, identidad único, identidadúnico#, insurance number, national identification number, national identity, nationalid#, nationaldno#, número nacional identidad, personal identification number, personal identity no, unique identity number, uniqueid#

パスポート番号のキーワード

さまざまなタイプのパスポート番号を検出するには、 CloudWatch Logs では番号の近くにあるキーワードが必要です。次の表に、 CloudWatch Logs が特定の国とリージョンについて認識するキーワードを示します。

国またはリージョン	キーワード
カナダ	passeport, passeport#, passport, passport#, passportno, passportno#
フランス	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport non
ドイツ	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reisepass, reisepass–nr, reisepassnummer
イタリア	italian passport number, numéro passeport, numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaporto
スペイン	españa pasaporte, libreta pasaporte, número pasaporte, pasaporte, passport, passport book, passport no, passport number, spain passport
英国	passeport #, passeport n °, passeportNon, passeport non, passeportn °, passport #, passport no, passport number, passport#, passportid
アメリカ	passport, travel document

納税者識別と参照番号のキーワード

さまざまなタイプの納税者識別番号と参照番号を検出するには、 CloudWatch Logs では番号の近くにあるキーワードが必要です。次の表に、 CloudWatch Logs が特定の国とリージョンについて認識するキーワードを示します。

国またはリージョン	キーワード
ブラジル	cadastro de pessoa física, cadastro de pessoa fisica, cadastro de pessoas físicas, cadastro de pessoas fisicas, cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj, cpf
フランス	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#
ドイツ	identifikationsnummer, steuer id, steueride ntifikationsnummer, steuernummer, tax id, tax identification number, tax number
スペイン	cif, cif número, cifnúmero#, nie, nif, número de contribuyente, número de identidad de extranjero, número de identificación fiscal, número de impuesto corporativo, personal tax number, tax id, tax identification number, tax number, tin, tin#
英国	paye, tax id, tax id no., tax id number, tax identification, tax identification#, tax no., tax number, tax reference, tax#, taxid#, temporary reference number, tin, trn, unique tax reference, unique taxpayer reference, utr
アメリカ	個別の納税者識別番号、itin、i.t.i.n。

個人を特定できる情報 (PII) のデータ識別子 ARN

次の表は、データ保護ポリシーに追加できる個人を特定できる情報 (PII) データ識別子の Amazon リソースネーム (ARN) のリストを示しています。

PII データ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/Address
arn:aws:dataprotection::aws:data-identifier/CepCode-BR
arn:aws:dataprotection::aws:data-identifier/Cnpj-BR
arn:aws:dataprotection::aws:data-identifier/CpfCode-BR
arn:aws:dataprotection::aws:data-identifier/DriversLicense-AT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-AU
arn:aws:dataprotection::aws:data-identifier/DriversLicense-BE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-BG
arn:aws:dataprotection::aws:data-identifier/DriversLicense-CA
arn:aws:dataprotection::aws:data-identifier/DriversLicense-CY
arn:aws:dataprotection::aws:data-identifier/DriversLicense-CZ
arn:aws:dataprotection::aws:data-identifier/DriversLicense-DE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-DK
arn:aws:dataprotection::aws:data-identifier/DriversLicense-EE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-ES
arn:aws:dataprotection::aws:data-identifier/DriversLicense-FI
arn:aws:dataprotection::aws:data-identifier/DriversLicense-FR
arn:aws:dataprotection::aws:data-identifier/DriversLicense-GB

PII データ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/DriversLicense-GR
arn:aws:dataprotection::aws:data-identifier/DriversLicense-HR
arn:aws:dataprotection::aws:data-identifier/DriversLicense-HU
arn:aws:dataprotection::aws:data-identifier/DriversLicense-IE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-IT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-LT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-LU
arn:aws:dataprotection::aws:data-identifier/DriversLicense-LV
arn:aws:dataprotection::aws:data-identifier/DriversLicense-MT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-NL
arn:aws:dataprotection::aws:data-identifier/DriversLicense-PL
arn:aws:dataprotection::aws:data-identifier/DriversLicense-PT
arn:aws:dataprotection::aws:data-identifier/DriversLicense-RO
arn:aws:dataprotection::aws:data-identifier/DriversLicense-SE
arn:aws:dataprotection::aws:data-identifier/DriversLicense-SI
arn:aws:dataprotection::aws:data-identifier/DriversLicense-SK
arn:aws:dataprotection::aws:data-identifier/DriversLicense-US
arn:aws:dataprotection::aws:data-identifier/ElectoralRollNumber-GB
arn:aws:dataprotection::aws:data-identifier/EmailAddress

PII データ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/IndividualTaxIdentificationNumber-US
arn:aws:dataprotection::aws:data-identifier/InseeCode-FR
arn:aws:dataprotection::aws:data-identifier/LatLong
arn:aws:dataprotection::aws:data-identifier/Name
arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-DE
arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-ES
arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-IT
arn:aws:dataprotection::aws:data-identifier/NieNumber-ES
arn:aws:dataprotection::aws:data-identifier/NifNumber-ES
arn:aws:dataprotection::aws:data-identifier/PassportNumber-CA
arn:aws:dataprotection::aws:data-identifier/PassportNumber-DE
arn:aws:dataprotection::aws:data-identifier/PassportNumber-ES
arn:aws:dataprotection::aws:data-identifier/PassportNumber-FR
arn:aws:dataprotection::aws:data-identifier/PassportNumber-GB
arn:aws:dataprotection::aws:data-identifier/PassportNumber-IT
arn:aws:dataprotection::aws:data-identifier/PassportNumber-US
arn:aws:dataprotection::aws:data-identifier/PermanentResidenceNumber-CA

PII データ識別子 ARN
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-BR
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-DE
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-ES
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-FR
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-GB
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-IT
arn:aws:dataprotection::aws:data-identifier/PhoneNumber-US
arn:aws:dataprotection::aws:data-identifier/PostalCode-CA
arn:aws:dataprotection::aws:data-identifier/RgNumber-BR
arn:aws:dataprotection::aws:data-identifier/SocialInsuranceNumber-CA
arn:aws:dataprotection::aws:data-identifier/Ssn-ES
arn:aws:dataprotection::aws:data-identifier/Ssn-US
arn:aws:dataprotection::aws:data-identifier/TaxId-DE
arn:aws:dataprotection::aws:data-identifier/TaxId-ES
arn:aws:dataprotection::aws:data-identifier/TaxId-FR
arn:aws:dataprotection::aws:data-identifier/TaxId-GB
arn:aws:dataprotection::aws:data-identifier/VehicleIdentificationNumber
arn:aws:dataprotection::aws:data-identifier/ZipCode-US

カスタムデータ識別子

トピック

SNS カスタムデータ識別子とは

カスタムデータ識別子 (CDI) を使用すると、データ保護ポリシーで使用できる独自のカスタム正規表現を定義できます。カスタムデータ識別子を使用すると、マネージドデータ識別子では提供できないビジネス固有の個人を特定できる情報 (PII) のユースケースをターゲットにすることができます。たとえば、カスタムデータ識別子を使用すると、会社固有の従業員 ID を検索できます。カスタムデータ識別子は、マネージドデータ ID と組み合わせて使用できます。

カスタムデータ識別子の制約

CloudWatch ログのカスタムデータ識別子には、次の制限があります。

- 各データ保護ポリシーに使用できるカスタムデータ識別子は最大 10個です。
- カスタムデータ識別子名に使用できるのは 128 文字までです。以下の文字がサポートされています。
  - 英数字: (a-zA-Z0-9)
  - 記号: ('_' | '-')
- RegEx の最大長は 200 文字です。以下の文字がサポートされています。
  - 英数字: (a-zA-Z0-9)
  - 記号: ('_' | '#' | '=' | '@' |'/' | ';' | ',' | '-' | ' ')
  - RegEx 予約文字: ('^' | '$' | '?' | '[' | ']' | '{' | '}' | '|' | '\\' | '*' | '+' | '.')
- カスタムデータ識別子は、マネージドデータ識別子と同じ名前を共有することはできません。
- カスタムデータ識別子は、アカウントレベルのデータ保護ポリシー内またはロググループレベルのデータ保護ポリシーで指定できます。マネージドデータ識別子と同様に、アカウントレベルのポリシー内で定義されたカスタムデータ識別子は、ロググループレベルのポリシーで定義されたカスタムデータ識別子と組み合わせて機能します。

コンソールでのカスタムデータ識別子の使用

CloudWatch コンソールを使用してデータ保護ポリシーを作成または編集する場合、カスタムデータ識別子を指定するには、データ識別子の名前と正規表現を入力するだけです。例えば、名前**Employee_ID**に と を正規表現**EmployeeID-\d{9}**として入力できます。この正規表現は、 の後に 9 つの数値を持つログイベントを検出してマスクしますEmployeeID-。例えば、次のようになります: EmployeeID-123456789

データ保護ポリシーでカスタムデータ識別子を使用する

AWS CLI または AWS API を使用してカスタムデータ識別子を指定する場合は、データ保護ポリシーの定義に使用される JSON ポリシーにデータ識別子名と正規表現を含める必要があります。次のデータ保護ポリシーは、会社固有の従業員 IDs。

1. データ保護ポリシー内で Configuration ブロックを作成します。
2. カスタムデータ識別子の Name を入力します。例えば **EmployeeId** です。
3. カスタムデータ識別子の Regex を入力します。例えば **EmployeeID-\d{9}** です。この正規表現は、 の後に 9 桁の EmployeeID-を含むログイベントと一致しますEmployeeID-。例えば、次のようになります: EmployeeID-123456789
4. ポリシーステートメントで、以下のカスタムデータ識別子を参照します。

```
{
    "Name": "example_data_protection_policy",
    "Description": "Example data protection policy with custom data identifiers",
    "Version": "2021-06-01",
    "Configuration": {
      "CustomDataIdentifier": [
        {"Name": "EmployeeId", "Regex": "EmployeeId-\\d{9}"}
      ]
    },
    "Statement": [
        {
            "Sid": "audit-policy",
            "DataIdentifier": [
                "EmployeeId"
            ],
            "Operation": {
                "Audit": {
                    "FindingsDestination": {
                        "S3": {
                            "Bucket": "EXISTING_BUCKET"
```

```
                        }
                    }
                }
            }
        },
        {
            "Sid": "redact-policy",
            "DataIdentifier": [
            "EmployeeId"
            ],
            "Operation": {
                "Deidentify": {
                    "MaskConfig": {
                    }
                }
            }
        }
    ]
}
```

5. (オプション) 必要に応じて、Configuration ブロックにさらに カスタムデータ識別子を追加します。現在、データ保護ポリシーでは最大 10 個のカスタムデータ識別子を使用できます。

フィルターを使用したログイベントからのメトリクスの作成

1 つ以上のメトリクスフィルター を作成することで、 CloudWatch ログに入るログデータを検索およびフィルタリングできます。 メトリクスフィルターは、ログデータの送信時にログデータで探す用語とパターンを定義します。 CloudWatch Logs CloudWatch 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化またはアラームを設定できる数値 CloudWatch メトリクスに変換します。

ログフィルターからメトリクスを作成するときに、ディメンションと単位をメトリクスに割り当てることもできます。単位を指定する場合は、フィルターの作成時に必ず正しい単位を指定してください。フィルターの単位を後で変更しても何も起こりません。

> **Note**
>
> メトリクスフィルターは、標準ログクラスのロググループでのみサポートされます。ログクラスの詳細については、「」を参照してくださいログクラス。

これらのメトリクスを表示したり CloudWatch 、アラームを設定したりするときに、パーセンタイル統計を含む任意のタイプの統計を使用できます。

> **Note**
>
> パーセンタイル統計は、メトリクスの値がいずれも負でない場合にのみメトリクスでサポートされます。負の数を報告できるようにメトリクスフィルタを設定した場合、値に負の数があると、パーセンタイル統計はそのメトリクスで使用できません。詳細については、パーセンタイルを参照してください。

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。

内容

- メトリクスフィルターの作成
- メトリクスフィルターの一覧表示
- メトリクスフィルターの削除

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。

デフォルト値

ログが取り込まれたものの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、データはこのような各期間の間にも報告されるため、一致する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。1 分間の期間内に取り込まれたログがない場合は、値は報告されません。

メトリクスフィルターによって作成されたメトリクスにディメンションを割り当てると、そのメトリクスにデフォルト値を割り当てることはできません。

ディメンション

ディメンションは、メトリクスをさらに定義するキーと値のペアです。メトリクスフィルターから作成されたメトリクスにディメンションを割り当てることができます。ディメンションはメトリクスの一意の識別子の一部であるため、ログから一意の名前/値のペアが抽出されるたびに、そのメトリクスの新しいバリエーションが作成されます。

フィルタパターン

Logs が各 CloudWatch ログイベントのデータを解釈する方法のシンボリックな説明。例えば、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

CloudWatch モニタリング対象のログ情報を公開するメトリクスの名前。例えば、 というメトリクスに発行できます ErrorCount。

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。

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。例えば、「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。

メトリックスフィルターのフィルターパターン構文

> **Note**
>
> メトリクスフィルターと CloudWatch Logs Insights クエリの違い
> メトリクスフィルターは、一致する CloudWatch ログが見つかるたびに指定された数値がメトリクスフィルターに追加されるという点で、Logs Insights クエリとは異なります。詳細については、「メトリクスフィルターのメトリクス値を設定する」を参照してください。
> Amazon CloudWatch Logs Insights クエリ言語でロググループをクエリする方法については、「」を参照してくださいCloudWatch Logs Insights クエリ構文。
> [一般的なフィルターパターンの例]
> メトリックスフィルターの他に、サブスクリプションフィルターとフィルターログイベントに適用される汎用フィルターパターン構文の詳細については、次の例を含むメトリックスフィルター、サブスクリプションフィルター、フィルターログイベントのフィルターパターン構文を参照してください。
>
> - サポートされている正規表現 (regex) 構文
> - 非構造化ログイベントでの語句の一致
> - JSON ログイベントの語句の一致
> - スペース区切りのログイベントの語句一致

メトリクスフィルターを使用すると、 CloudWatch ログに入力されたログデータを検索およびフィルタリングし、フィルタリングされたログデータからメトリクスの観測値を抽出し、データポイントを CloudWatch ログメトリクスに変換できます。 CloudWatch Logs に送信されるログデータを検索する用語とパターンを定義します。メトリクスフィルター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ーはそのログストリームに適用されます。

メトリックスフィルターが語句と一致するとき、メトリックスの数を特定の数値で増えます。例えば、ログイベントの中で ERROR という単語の出現回数を数えるメトリクスフィルターを作成します。

メトリックスに測定の単位と寸法を割り当てることができます。例えば、ログイベント内で [ERROR] という単語の出現回数を数えるメトリクスフィルターを作成する場合、[ERROR] という単語が含まれるログイベントの合計数を示す `ErrorCode` というディメンションを指定し、レポートされたエラーコードでデータをフィルタリングすることができます。

> **Tip**
>
> 測定の単位をメトリックスに割り当てるとき、必ず正しい単位を指定してください。後で単位を変更すると、変更が有効にならない場合があります。が CloudWatch サポートするユニットの完全なリストについては、Amazon CloudWatch API リファレンスMetricDatumの「」を参照してください。

トピック

- メトリクスフィルターのメトリクス値を設定する
- JSON の値またはスペース区切りログイベントからメトリクスとともにディメンションを発行する
- ログイベントの値を使用してメトリクスの値を増分する

メトリクスフィルターのメトリクス値を設定する

メトリクスフィルターを作成する際は、フィルターパターンを定義し、メトリクス値とデフォルト値を指定します。メトリクス値は、数値、名前付き識別子、または数値識別子に設定できます。デフォルト値を指定しない場合、メトリクスフィルター CloudWatch で一致が見つからなかった場合、 はデータをレポートしません。値が 0 であっても、デフォルト値を指定することをお勧めします。デフォルト値を設定すると、より正確にデータを CloudWatch レポートでき、むらのあるメトリクスが集約されなくなります CloudWatch 。 CloudWatch は、メトリクス値を 1 分ごとに集計してレポートします。

メトリクスフィルターがログイベント内で一致するものを見つけたら、メトリクスの数がメトリクスの値だけ増加します。メトリクスフィルターで一致が見つからない場合、 はメトリクスのデフォルト値 CloudWatch を報告します。例えば、ロググループが毎分 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 であるとします。最初の 1 分で両方のログレコードに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 分間にどちらのレコードでも一致が見つからなかった場合、その分のデフォルト値は 0 となります。メトリクスフィルターが生成するメトリクスにディメンションを割り当てると、それらのメトリクスのデフォルト値を指定することはできません。

また、静的値ではなく、ログイベントから抽出された値でメトリクスを増分するようにメトリクスフィルターを設定することもできます。詳細については、「ログイベントの値を使用してメトリクスの値を増分する」を参照してください。

JSON の値またはスペース区切りログイベントからメトリクスとともにディメンションを発行する

CloudWatch コンソールまたは AWS CLI を使用して、JSON およびスペース区切りのログイベントが生成するメトリクスを使用してディメンションを発行するメトリクスフィルターを作成できます。ディメンションとは名前と値のペアであり、JSON およびスペース区切りフィルターパターンでのみ使用できます。最大 3 つのディメンションを持つ JSON およびスペース区切りメトリクスフィルターを作成できます。ディメンションの詳細とディメンションをメトリクスに割り当てる方法の詳細については、以下のセクションを参照してください。

- Amazon CloudWatch ユーザーガイドのディメンション
- 例: Amazon Logs ユーザーガイドの「Apache ログからフィールドを抽出し、ディメンションを割り当てる CloudWatch 」

> **Important**
>
> ディメンションには、カスタムメトリクスと同じ請求を収集する値が含まれています。予期せぬ請求を防ぐため、`IPAddress` または `requestID` などをディメンションとするなど、高カーディナリティフィールドを指定しないでください。
> メトリクスをログイベントから抽出すると、カスタムメトリクスとして料金が発生します。意図しない高額請求の徴収を防ぐため、Amazon は、特定の時間内に指定したディメンションのために 1000 の異なる名前と値のペアを生成した場合、メトリクスフィルターを無効化することがあります。
> 見積費用を通知する請求アラームを作成できます。詳細については、「推定請求額をモニタリングするための請求アラームの作成 AWS」を参照してください。

JSON ログイベントからメトリクスとともにディメンションを発行する

次の例には、JSON メトリクスフィルターでディメンションを指定する方法を説明するコードスニペットが含まれています。

Example: JSON log event

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

}
```

> **Note**
>
> サンプルの JSON ログイベントを使用してサンプルメトリクスフィルターをテストする場合は、サンプル JSON ログを 1 行で入力する必要があります。

Example: Metric filter

JSON ログイベントにプロパティ `eventType` および `"sourceIPAddress"` が含まれるたびに、メトリクスフィルターによってメトリクスが増分されます。

```
{ $.eventType = "*" && $.sourceIPAddress != 123.123.* }
```

JSON メトリクスフィルターを作成するときに、メトリクスフィルター内の任意のプロパティをディメンションとして指定できます。例えば、`eventType` をディメンションとして設定するには、以下を使用します。

```
"eventType" : $.eventType
```

サンプルメトリクスには、"eventType" という名前のディメンションが含まれており、サンプルログイベント内のディメンションの値は "UpdateTrail" です。

スペース区切りのログイベントからメトリクスとともにディメンションを発行する

次の例には、スペース区切りのメトリクスフィルターでディメンションを指定する方法を説明するコードスニペットが含まれています。

Example: Space-delimited log event

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

Example: Metric filter

```
[ip, server, username, timestamp, request, status_code, bytes > 1000]
```

メトリクスフィルターは、スペース区切りのログイベントにフィルターで指定されているいずれかのフィールドが含まれる場合に、メトリクスを増分します。例えば、メトリクスフィルターは、サンプルのスペース区切りのログイベントで次のフィールドと値を検索します。

```
{
   "$bytes": "1534",
   "$status_code": "404",

   "$request": "GET /index.html HTTP/1.0",
   "$timestamp": "10/Oct/2000:13:25:15 -0700",
   "$username": "frank",
   "$server": "Prod",
   "$ip": "127.0.0.1"
}
```

スペース区切りのメトリクスフィルターを作成するときに、メトリクスフィルター内の任意のフィールドをディメンションとして指定できます。例えば、server をディメンションとして設定するには、以下を使用します。

```
"server" : $server
```

サンプルメトリクスフィルターには、server という名前のディメンションがあり、サンプルログイベント内のディメンションの値は "Prod" です。

Example: Match terms with AND (&&) and OR (||)

論理演算子 AND (「&&」) および OR (「||」) を使用して、条件を含むスペース区切りメトリクスフィルターを作成できます。次のメトリックスフィルターでは、イベントの最初の単語が ERROR (エラー)または WARN (警告)の超文字列としてログイベントが返されます。

```
[w1=ERROR || w1=%WARN%, w2]
```

ログイベントの値を使用してメトリクスの値を増分する

ログイベントで見つかった数値を公開するメトリクスフィルターを作成できます。このセクションの手順では、次のサンプルメトリクスフィルターを使用して、JSON ログイベントの数値をメトリクスに公開する方法を示します。

```
{ $.latency = * } metricValue: $.latency
```

ログイベントの値を発行するメトリクスフィルターを作成するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、ログ、ロググループ の順に選択します。
3. ロググループを選択または作成します。

   ロググループの作成方法については、「Amazon Logs ユーザーガイド」の CloudWatch 「ログでロググループを作成する」を参照してください。 CloudWatch
4. [アクション]、[メトリクスフィルターの作成] の順に選択します。

5. [Filter Pattern] (フィルターパターン) に **`{ $.latency = * }`** と入力し、[Next] (次へ) を選択します。
6. [Metric Name] (メトリクス名) に、「myMetric」と入力します。
7. [メトリクス値] に「**`$.latency`**」と入力します。
8. [Default Value] (デフォルト値) に 0 と入力し、[Next] (次へ) を選択します。

   値が 0 であっても、デフォルト値を指定することをお勧めします。デフォルト値を設定すると、より正確にデータを CloudWatch レポートでき、むらのあるメトリクスが集約されなくなります CloudWatch 。 CloudWatch は、メトリクス値を 1 分ごとに集計してレポートします。
9. [Create metric filter] (メトリクスフィルターの作成) を選択します。

サンプルメトリクスフィルターは、語句 "`latency`" をサンプル JSON ログイベントで照合し、数値 50 をメトリクスの [myMetric] に発行します。

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

メトリクスフィルターの作成

以下の手順は、メトリクスフィルターの作成方法を示しています。

例

- ロググループのメトリクスフィルターの作成
- 例: ログイベントのカウント
- 例: 語句の出現回数をカウントする
- 例: HTTP 404 コードをカウントする
- 例: HTTP 4xx コードをカウントする
- 例: Apache ログからフィールドを抽出してディメンションを割り当てる

ロググループのメトリクスフィルターの作成

ロググループのメトリクスフィルターを作成するには、次の手順に従います。メトリクスは、データポイントがいくつか見つかるまで表示されません。

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/ で 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、ロググループ の順に選択します。
3. ロググループの名前を選択します。
4. [Actions]、[メトリクスフィルターの作成] の順に選択します。
5. [フィルターパターン] に、フィルターパターンを入力します。詳細については、「メトリクスフィルター、サブスクリプションフィルター、フィルターログイベント、およびライブテールのフィルターパターン構文」を参照してください。
6. (オプション) フィルターパターンをテストするには、[テストパターン] に、パターンをテストする 1 つまたは複数のログイベントを入力します。各ログイベントは 1 行にフォーマットする必要があります。改行は、[ログイベントメッセージ] ボックスのログイベントを区切るために使用されます。
7. [次へ] を選択し、メトリクスフィルターの名前を入力します。
8. メトリクスの詳細 のメトリクス名前空間 に、メトリクスが公開される CloudWatch 名前空間の名前を入力します。名前空間がまだ存在しない場合は、[新規作成] が選択されていることを確認します。
9. [メトリクス名] に、新しいメトリクスの名前を入力します。
10. メトリクスフィルターでフィルター内のキーワードの出現回数をカウントする場合は、[メトリクス値] に「1」と入力します。これにより、キーワードの 1 つを含むログイベントごとに、メトリクスが 1 ずつ増加します。

    または、**$size** などのトークンを入力します。これにより、size フィールドを含むすべてのログイベントについて、size フィールド内の数値だけメトリクスが増加します。
11. (オプション) [Unit] (単位) で、メトリクスに割り当てる単位を選択します。単位を指定しない場合、単位は None に設定されます。
12. (オプション) メトリクスの 3 つのディメンションの名前とトークンを入力します。メトリクスフィルターが生成するメトリクスにディメンションを割り当てると、それらのメトリクスのデフォルト値を指定することはできません。

> **Note**
>
> ディメンションは、JSON またはスペース区切りメトリクスフィルターでのみサポートされます。

13. [Create metric filter] (メトリクスフィルターの作成) を選択します。ナビゲーションペインから作成したメトリクスフィルターを見つけることができます。[Logs] を選択し、ロググループを選択します。メトリクスフィルターを作成したロググループの名前を選択し、[メトリクスフィルター] タブを選択します。

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、 というメトリクスフィルター MyAppAccessCount がロググループ MyApp/access.log に適用され、 という名前空間 EventCount に CloudWatchメトリクスが作成されます MyNamespace。フィルタは、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/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Log groups] (ロググループ) を選択します。
3. ロググループの名前を選択します。
4. `Actions`、[メトリクスフィルターの作成] を選択します。
5. [フィルターパターン] および [テストするログデータの選択] は空白のままにします。
6. [次へ] を選択し、[フィルター名] に **`EventCount`** と入力します。
7. [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。
8. [メトリクス名] に「**`MyAppEventCount`**」と入力します。
9. [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。
10. [デフォルト値] に 0 と入力し、[次へ] 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。
11. [メトリクスフィルターの作成] を選択します。

を使用してメトリクスフィルターを作成するには AWS CLI

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 に発行されたデータポイントが表示されます MyAppAccessEventCount。

を使用してイベントデータを投稿するには AWS CLI

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。例えば、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。ポリシーが作成され、ロググループ MyApp/message.log に追加されました。 CloudWatch Logs は、Error を含むすべてのイベント ErrorCount に対して、値「1」の MyApp/message.log 名前空間の CloudWatch カスタムメトリクスに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。コンソールで CloudWatchこのデータをグラフ化する場合は、必ず合計統計を使用してください。

メトリクスフィルターを作成したら、 CloudWatch コンソールでメトリクスを表示できます。表示するメトリクスを選択するときに、ロググループ名と一致するメトリクス名前空間を選択します。詳細については、使用可能なメトリクスの表示を参照してください。

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/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Log groups] (ロググループ) を選択します。
3. ロググループの名前を選択します。
4. [アクション]、[メトリクスフィルターの作成] の順に選択します。
5. [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。

6. (オプション) フィルターパターンをテストするには、[テストパターン] に、パターンのテストに使用する 1 つまたは複数のログイベントを入力します。[ログイベントメッセージ] ボックスのログイベントを区切るために改行が使用されるため、各ログイベントは 1 行以内である必要があります。
7. [次へ] を選択し、[メトリクスの割り当て] ページの [フィルター名] 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 と入力しますMyNameSpace。
9. [メトリクス名] に「ErrorCount」と入力します。
10. [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、[次へ] を選択します。
12. [メトリクスフィルターの作成] を選択します。

を使用してメトリクスフィルターを作成するには AWS CLI

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=ErrorCount,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。

を使用してイベントを投稿するには AWS CLI

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 を使用すると、Apache サーバーが HTTP 404 レスポンスを返す回数をモニタリングできます。これは、見つからないページのレスポンスコードです。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/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Log groups] (ロググループ) を選択します。
3. Actions、[メトリクスフィルターの作成] を選択します。
4. [フィルターパターン] には **[IP, UserInfo, User, Timestamp, RequestInfo, StatusCode=404, Bytes]** と入力します。
5. (オプション) フィルターパターンをテストするには、[テストパターン] に、パターンのテストに使用する 1 つまたは複数のログイベントを入力します。[ログイベントメッセージ] ボックスのログイベントを区切るために改行が使用されるため、各ログイベントは 1 行以内である必要があります。
6. [次へ] を選択し、[フィルター名] に HTTP404Errors と入力します。
7. [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。
8. [メトリクス名] に、**ApacheNotFoundErrorCount** を入力します。
9. [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。
10. [デフォルト値] に 0 と入力し、[次へ] を選択します。
11. [メトリクスフィルターの作成] を選択します。

を使用してメトリクスフィルターを作成するには AWS CLI

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"
```

これらのサンプルログイベントを配置するとすぐに、 CloudWatch コンソールで という名前のメトリクスを として取得できます ApacheNotFoundErrorCount。

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。例えば、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」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/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Log groups] (ロググループ) を選択します。
3. Apache サーバーのロググループの名前を選択します。
4. Actions、[メトリクスフィルターの作成] を選択します。
5. [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。
6. (オプション) フィルターパターンをテストするには、[テストパターン] に、パターンのテストに使用する 1 つまたは複数のログイベントを入力します。[ログイベントメッセージ] ボックスのログイベントを区切るために改行が使用されるため、各ログイベントは 1 行以内である必要があります。
7. [次へ] を選択し、[フィルター名] 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. [メトリクス値] に「1」と入力します。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、[次へ] を選択します。
12. [メトリクスフィルターの作成] を選択します。

を使用してメトリクスフィルターを作成するには AWS CLI

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 ログからフィールドを抽出してディメンションを割り当てる

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。

この例では、作成するメトリクスにディメンションを割り当てる方法も示します。

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/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Log groups] (ロググループ) を選択します。
3. Apache サーバーのロググループの名前を選択します。
4. Actions、[メトリクスフィルターの作成] を選択します。
5. [フィルターパターン] に「**[ip, id, user, timestamp, request, status_code, size]**」と入力します。
6. (オプション) フィルターパターンをテストするには、[テストパターン] に、パターンのテストに使用する 1 つまたは複数のログイベントを入力します。[ログイベントメッセージ] ボックスのログイベントを区切るために改行が使用されるため、各ログイベントは 1 行以内である必要があります。
7. [次へ] を選択し、[フィルター名] 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。これは新しい名前空間であるため、[新規作成] が選択されていることを確認してください。
9. [メトリクス名] に、「**BytesTransferred**」と入力します。
10. [メトリクス値] に「**$size**」と入力します。
11. [Unit] (単位) で、[Bytes] (バイト) を選択します。
12. [Dimension Name] (ディメンション名) で、**IP** と入力します。

13. [Dimension Value] (ディメンションの値) に、**$ip** と入力し、[Next] (次へ) を選択します。
14. [メトリクスフィルターの作成] を選択します。

を使用してこのメトリクスフィルターを作成するには AWS CLI

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, size]' \
 --metric-transformations  \
 metricName=BytesTransferred,metricNamespace=MyNamespace,metricValue='$size'
```

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

> **Note**
>
> このコマンドでは、この形式を使用して複数のディメンションを指定します。
>
> ```
> aws logs put-metric-filter \
> --log-group-name my-log-group-name \
> --filter-name my-filter-name \
> --filter-pattern 'my-filter-pattern' \
> --metric-transformations  \
> metricName=my-metric-name,metricNamespace=my-metric-namespace,metricValue=my-
> token,unit=unit,dimensions='{dimension1=$dim,dimension2=$dim2,dim3=$dim3}'
> ```

put-log-event 呼び出しで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/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Log groups] (ロググループ) 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。

を使用してメトリクスフィルターを一覧表示するには AWS CLI

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/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[Log groups] (ロググループ) 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、ロググループのメトリクスフィルター数を選択します。
4. [メトリクスフィルター] 画面で、削除するフィルター名の右側にあるチェックボックスをオンにします。その後、[Delete] 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、[Delete] を選択します。

を使用してメトリクスフィルターを削除するには AWS CLI

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
```

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 ログ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、Amazon Kinesis ストリーム、Amazon Data Firehose ストリーム、または AWS Lambda カスタム処理、分析、他のシステムへのロードなどの他の サービスに配信できます。ログイベントが宛先サービスに送信されると、base64 でエンコードされ、gzip 形式で圧縮されます。

ログイベントのサブスクリプションを開始するには、Kinesis データストリームなど、イベントを配信する受信リソースを作成します。サブスクリプションフィルターは、 AWS リソースに配信されるログイベントをフィルタリングするために使用するフィルターパターンと、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。

サブスクリプションは、アカウントレベルとロググループレベルで作成できます。各アカウントには、1 つのアカウントレベルのサブスクリプションフィルターを設定できます。ロググループごとに、最大 2 つのサブスクリプションフィルターを関連付けることができます。

> **Note**
>
> 送信先サービスがスロットリング例外や再試行可能なサービス例外 (HTTP 5xx など) などの再試行可能なエラーを返した場合、 CloudWatch ログは最大 24 時間配信を再試行し続けます。エラーが AccessDeniedException や などの再試行不可能なエラーである場合、CloudWatch ログは再配信を試みません ResourceNotFoundException。このような場合、サブスクリプションフィルターは最大 10 分間無効になり、 CloudWatch ログは送信先へのログの送信を再試行します。この無効化期間中、ログはスキップされます。

CloudWatch ログは、ログイベントのサブスクリプションへの転送に関する CloudWatch メトリクスも生成します。詳細については、「 CloudWatch メトリクスによるモニタリング」を参照してください。

CloudWatch Logs サブスクリプションを使用して、ログデータをほぼリアルタイムで Amazon OpenSearch Service クラスターにストリーミングすることもできます。詳細については、「Amazon OpenSearch Service への CloudWatch ログデータのストリーミング」を参照してください。

サブスクリプションは、標準ログクラスのロググループでのみサポートされます。ログクラスの詳細については、「」を参照してくださいログクラス。

> **Note**
>
> サブスクリプションフィルターは、ログイベントをバッチ処理して送信を最適化し、送信先に対して行われる呼び出しの量を減らす場合があります。バッチ処理は保証されませんが、可能な場合は使用されます。

内容

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。

フィルタパターン

Logs が各 CloudWatch ログイベントのデータを解釈する方法のシンボリックな説明と、宛先 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式。フィルタパターンの構文の詳細については、「メトリクスフィルター、サブスクリプションフィルター、フィルターログイベント、およびライブテールのフィルターパターン構文」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 Data Streams ストリーム、Firehose ストリーム、または Lambda 関数の Amazon リソースネーム (ARN)。

role arn

選択した送信先にデータを配置するために必要なアクセス許可を CloudWatch ログに付与する IAM ロール。Logs は Lambda CloudWatch 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許可を取得できるため、このロールは Lambda 送信先では必要ありません。

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方法。この場合、宛先は Amazon Kinesis Data Streams 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。

ロググループレベルのサブスクリプションには、次のキー要素も含まれます。

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のログイベントにはサブスクリプションフィルターが適用され、フィルターに一致するログイベントは、一致するログイベントを受信する宛先サービスに配信されます。

アカウントレベルのサブスクリプションには、次のキー要素も含まれます。

選択基準

アカウントレベルのサブスクリプションフィルターが適用されているロググループを選択するために使用される条件。これを指定しない場合、アカウントレベルのサブスクリプションフィルターはアカウント内のすべてのロググループに適用されます。このフィールドは、無制限のロググループを防ぐために使用されます。無限ロググループの問題の詳細については、「」を参照してくださいログの再帰防止。

選択基準のサイズ制限は 25 KB です。

ロググループレベルのサブスクリプションフィルター

サブスクリプションフィルターは、Kinesis Data Streams、Lambda、または Firehose で使用できます。サブスクリプションフィルターを介して宛先サービスに送信されるログは、base64 でエンコードされ、gzip 形式で圧縮されます。

フィルターとパターン構文を使用してログデータを検索できます。

例

- 例 1: Kinesis データストリームのサブスクリプションフィルター
- 例 2: を使用したサブスクリプションフィルター AWS Lambda
- 例 3: Amazon Data Firehose を使用したサブスクリプションフィルター

例 1: Kinesis データストリームのサブスクリプションフィルター

次の例では、サブスクリプションフィルターをイベントを含むロググループに関連付け AWS CloudTrail ます。サブスクリプションフィルターは、「ルート AWS 」認証情報によって行われたすべてのログに記録されたアクティビティを、「」と呼ばれる Kinesis Data Streams のストリームに配信しますRootAccess。 CloudWatch ログに AWS CloudTrail イベントを送信する方法の詳細については、「 AWS CloudTrail ユーザーガイド」の CloudWatch 「ログへの CloudTrail イベントの送信」を参照してください。

> **Note**
>
> 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 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。ストリームボリューム制限に関する詳細は、「クォータと制限」を参照してください。
> スロットリングされた成果物は、最大 24 時間再試行されます。24 時間が経過すると、失敗した成果物は破棄されます。
> スロットリングのリスクを軽減するには、次のステップに従います。
>
> - PutSubscriptionFilter または を使用してサブスクリプションフィルターを作成するdistributionときに、 randomに put-subscription-filterを指定します。デフォルトでは、ストリームフィルターディストリビューションはログストリームによって行われ、スロットリングが発生する可能性があります。
> - CloudWatch メトリクスを使用してストリームをモニタリングします。これにより、スロットリングを特定し、必要に応じて構成を調整できます。例えば、DeliveryThrottlingメトリクスを使用して、サブスクリプションの送信先にデータを転送するときに CloudWatch Logs がスロットリングされたログイベントの数を追跡できます。モニタリングの詳細については、「 CloudWatch メトリクスによるモニタリング」をご参照ください。
> - Kinesis Data Streams のストリームにはオンデマンドキャパシティモードを使用します。オンデマンドモードは、ワークロードが増加または縮小すると、即座にワークロードに対応します。オンデマンドキャパシティモードの詳細については、「オンデマンドモード」を参照してください。
> - Kinesis Data Streams のストリームの容量に合わせて CloudWatch サブスクリプションフィルターパターンを制限します。ストリームに送信するデータが多すぎる場合、フィルターサイズを小さくするか、フィルター条件を調整する必要があります。

Kinesis データストリーム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 ストリームを作成します。

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

2. 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 Data Streams describe-stream コマンドを使用して、. プロパティを確認できますStreamDescription。StreamStatus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. ストリームにデータを配置するアクセス許可を CloudWatch Logs に付与する IAM ロールを作成します。まず、ファイル (~/TrustPolicyForCWL-Kinesis.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。

このポリシーには、「混乱した代理」のセキュリティ上の問題を防止するための aws:SourceArn グローバル条件コンテキストキーが含まれています。詳細については、「混乱した代理の防止」を参照してください。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
        "StringLike": { "aws:SourceArn": "arn:aws:logs:region:123456789012:*" }
     }
   }
}
```

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-Kinesis.json
```

次は出力の例です。

```
{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.amazonaws.com"
                },
                "Condition": {
                    "StringLike": {
                        "aws:SourceArn": { "arn:aws:logs:region:123456789012:*" }
                    }
                }
            }
        },
        "RoleId": "AAOIIAH450GAB4HC5F431",
        "CreateDate": "2015-05-29T13:46:29.431Z",
```

```
        "RoleName": "CWLtoKinesisRole",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisRole"
    }
}
```

5. アクセス許可ポリシーを作成して、Logs CloudWatch 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-Kinesis.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"
    }
  ]
}
```

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-Kinesis.json
```

7. ストリームがアクティブ状態になり、IAM ロールを作成したら、 CloudWatch ログサブスクリプションフィルターを作成できます。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 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/logs" \
    --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ーを設定すると、 CloudWatch Logs はフィルターパターンに一致するすべての受信ログイベントをストリームに転送します。これが起きていることは、Kinesis データストリームシャードイテレータを取得し、Kinesis データストリーム get-records コマンドを使用していくつかの Kinesis データストリーム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 データストリーム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 データストリームレコードの 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 でデコードおよび解凍されたデータは、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/logs",
    "logStream": "111111111111_CloudTrail/logs_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 (オーナー)

発信元ログデータの 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、主に送信先に到達可能かどうかを確認するために、「CONTROL_MESSAGE」タイプの Kinesis Data Streams レコードを出力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: を使用したサブスクリプションフィルター AWS Lambda

この例では、 CloudWatch ログデータを AWS Lambda 関数に送信する Logs サブスクリプションフィルターを作成します。

> **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 の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 デベロッパーガイドの「ステップ 2.2: IAM ロール (実行ロール) を作成する」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 = Buffer.from(input.awslogs.data, 'base64');
    zlib.gunzip(payload, function(e, result) {
        if (e) {
            context.fail(e);
        } else {
            result = JSON.parse(result.toString());
            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 fileb://file-path/helloWorld.zip \
    --role lambda-execution-role-arn \
    --handler helloWorld.handler \
    --runtime nodejs12.x
```

5. 関数を実行するアクセス許可を CloudWatch ログに付与します。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.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] (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。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 でデコードおよび解凍されたデータは、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 (オーナー)

発信元ログデータの 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、主に送信先に到達可能かどうかを確認するために、「CONTROL_MESSAGE」タイプの Lambda レコードを出力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 Data Firehose を使用したサブスクリプションフィルター

この例では、定義したフィルターに一致する受信 CloudWatch ログイベントを Amazon Data Firehose 配信ストリームに送信する Logs サブスクリプションを作成します。 CloudWatch Logs から Amazon Data Firehose に送信されるデータは、すでに gzip レベル 6 圧縮で圧縮されているため、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。その後、Firehose の解凍機能を使用して、ログを自動的に解凍できます。詳細については、CloudWatch 「ログを使用した Kinesis Data Firehose への書き込み」を参照してください。

> **Note**
>
> Firehose ストリームを作成する前に、生成され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 Firehose ストリームを必ず作成してください。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。Firehose ストリームのボリューム制限の詳細については、「Amazon Data Firehose データ制限」を参照してください。

Firehose のサブスクリプションフィルターを作成するには

1. Amazon Simple Storage Service (Amazon S3) バケットを作成します。 CloudWatch Logs 専用に作成され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. Amazon S3 バケットにデータを配置するアクセス許可を Amazon Data Firehose に付与する IAM ロールを作成します。

   詳細については、「Amazon Data Firehose デベロッパーガイド」の「Amazon Data Firehose によるアクセスの制御」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。

   ```
   {
     "Statement": {
       "Effect": "Allow",
       "Principal": { "Service": "firehose.amazonaws.com" },
   ```

```
    "Action": "sts:AssumeRole"
    }
}
```

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. アクセス許可ポリシーを作成して、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. 次のように送信先の Firehose 配信ストリームを作成し、RoleARN と BucketARN のプレースホルダー値を、作成したロールとバケット ARNs。

```
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"}'
```

Firehose は、配信された Amazon S3 オブジェクトに YYYY/MM/DD/HH UTC 時間形式のプレフィックスを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)。Firehose describe-delivery-stream コマンドを使用して、DeliveryStreamDescription.DeliveryStreamStatus プロパティを確認できます。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. Firehose 配信ストリームにデータを配置するアクセス許可を CloudWatch Logs に付与する IAM ロールを作成します。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。

   このポリシーには、「混乱した代理」のセキュリティ上の問題を防止するための aws:SourceArn グローバル条件コンテキストキーが含まれています。詳細については、「混乱した代理の防止」を参照してください。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
         "StringLike": {
             "aws:SourceArn": "arn:aws:logs:region:123456789012:*"
         }
     }
  }
}
```

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.amazonaws.com"
                },
                "Condition": {
                     "StringLike": {
                         "aws:SourceArn": "arn:aws:logs:region:123456789012:*"
                     }
                 }
            }
        },
        "RoleId": "AAOIIAH450GAB4HC5F431",
        "CreateDate": "2015-05-29T13:46:29.431Z",
        "RoleName": "CWLtoKinesisFirehoseRole",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"
    }
}
```

10. アクセス許可ポリシーを作成して、Logs CloudWatch 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:PutRecord"],
        "Resource":[
            "arn:aws:firehose:region:account-id:deliverystream/delivery-stream-
name"]
```

```
      }
    ]
}
```

11. 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 put-role-policy。

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

12. Amazon Data Firehose 配信ストリームがアクティブ状態になり、IAM ロールを作成したら、CloudWatch ログサブスクリプションフィルターを作成できます。サブスクリプションフィルターは、選択したロググループから Amazon Data Firehose 配信ストリームへのリアルタイムログデータのフローをすぐに開始します。

```
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ーを設定すると、 CloudWatch Logs はフィルターパターンに一致するすべての受信ログイベントを Amazon Data Firehose 配信ストリームに転送します。Amazon Data Firehose 配信ストリームに設定されている時間バッファ間隔に基づいて、データが Amazon S3 に表示され始めます。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": {}
}
```

Simple Storage Service (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
```

アカウントレベルのサブスクリプションフィルター

> **Important**
>
> サブスクリプションフィルターで無限の再帰ループが発生するリスクがあり、対処しないと取り込み料金が大幅に増加する可能性があります。このリスクを軽減するには、アカウントレベルのサブスクリプションフィルターで選択基準を使用して、サブスクリプション配信

ワークフローの一部であるリソースからログデータを取り込むロググループを除外することをお勧めします。この問題と除外するロググループの決定の詳細については、「」を参照してくださいログの再帰防止。

アカウントのロググループのサブセットを含むアカウントレベルのサブスクリプションポリシーを設定できます。アカウントサブスクリプションポリシーは、Kinesis Data Streams、Lambda、または Firehose で使用できます。アカウントレベルのサブスクリプションポリシーを介して受信サービスに送信されるログは、base64 でエンコードされ、gzip 形式で圧縮されます。

> **Note**
>
> アカウント内のすべてのサブスクリプションフィルターポリシーのリストを表示するには、 `SUBSCRIPTION_FILTER_POLICY--policy-type`パラメータに の値を指定して `describe-account-policies` コマンドを使用します。詳細については、 describe-account-policies「¶」を参照してください。

例

- 例 1: Kinesis データストリームのサブスクリプションフィルター
- 例 2: を使用したサブスクリプションフィルター AWS Lambda
- 例 3: Amazon Data Firehose を使用したサブスクリプションフィルター

例 1: Kinesis データストリームのサブスクリプションフィルター

アカウントレベルのサブスクリプションポリシーで使用する Kinesis Data Streams データストリームを作成する前に、生成され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 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場合は、スロットリングされます。ストリームボリュームの制限の詳細については、Kinesis Data Streams ドキュメントの「クォータと制限」を参照してください。

> **Warning**
>
> 複数のロググループのログイベントは送信先に転送されるため、スロットリングのリスクがあります。スロットリングされた成果物は、最大 24 時間再試行されます。24 時間が経過すると、失敗した成果物は破棄されます。
> スロットリングのリスクを軽減するには、次のステップに従います。

- CloudWatch メトリクスを使用して Kinesis Data Streams ストリームをモニタリングします。これにより、スロットリングを特定し、それに応じて設定を調整することができます。例えば、 `DeliveryThrottling`メトリクスは、サブスクリプションの送信先にデータを転送するときに CloudWatch ログがスロットリングされたログイベントの数を追跡します。詳細については、「CloudWatch メトリクスによるモニタリング」を参照してください。
- Kinesis Data Streams のストリームにはオンデマンドキャパシティモードを使用します。オンデマンドモードは、ワークロードが増加または縮小すると、即座にワークロードに対応します。詳細については、「オンデマンドモード」を参照してください。
- Kinesis Data Streams のストリームの容量に合わせて CloudWatch Logs サブスクリプションフィルターパターンを制限します。ストリームに送信するデータが多すぎる場合、フィルターサイズを小さくするか、フィルター条件を調整する必要があります。

次の例では、アカウントレベルのサブスクリプションポリシーを使用して、すべてのログイベントを Kinesis Data Streams のストリームに転送します。フィルターパターンは、すべてのログイベントをテキストと照合`Test`し、Kinesis Data Streams のストリームに転送します。

Kinesis Data Streams のアカウントレベルのサブスクリプションポリシーを作成するには

1. 次のコマンドを使用して送信先 ストリームを作成します。

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

2. ストリームがアクティブになるまで数分待ちます。describe-stream コマンドを使用して . プロパティをチェックすることで、ストリームがアクティブかどうかを確認できますStreamDescription。StreamStatus

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

出力例を次に示します。

```
{
    "StreamDescription": {
        "StreamStatus": "ACTIVE",
        "StreamName": "TestStream",
        "StreamARN": "arn:aws:kinesis:region:123456789012:stream/TestStream",
        "Shards": [
```

```
            {
                "ShardId": "shardId-000000000000",
                "HashKeyRange": {
                    "EndingHashKey": "EXAMPLE84634633746074317682114555",
                    "StartingHashKey": "0"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber":
                    "EXAMPLE6888184566795038319814587845913527021815726l0"
                }
            }
        ]
    }
}
```

3. ストリームにデータを配置するアクセス許可を CloudWatch Logs に付与する IAM ロールを作成します。まず、ファイル (~/TrustPolicyForCWL-Kinesis.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。

   このポリシーには、「混乱した代理」のセキュリティ上の問題を防止するための aws:SourceArn グローバル条件コンテキストキーが含まれています。詳細については、「混乱した代理の防止」を参照してください。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
        "StringLike": { "aws:SourceArn": "arn:aws:logs:region:123456789012:*" }
     }
   }
}
```

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-Kinesis.json
```

   次は出力の例です。

```
{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.amazonaws.com"
                },
                "Condition": {
                    "StringLike": {
                        "aws:SourceArn": { "arn:aws:logs:region:123456789012:*" }
                    }
                }
            }
        },
        "RoleId": "EXAMPLE450GAB4HC5F431",
        "CreateDate": "2023-05-29T13:46:29.431Z",
        "RoleName": "CWLtoKinesisRole",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisRole"
    }
}
```

5. アクセス許可ポリシーを作成して、Logs CloudWatch 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-Kinesis.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作成しないでください。

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

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-Kinesis.json
```

7. ストリームがアクティブ状態になり、IAM ロールを作成したら、 CloudWatch ログサブスクリプションフィルターポリシーを作成できます。ポリシーは、ストリームへのリアルタイムログデータのフローをすぐに開始します。この例では、 文字列を含むすべてのログイベントERRORがストリーミングされます。ただし、 LogGroupToExclude1および という名前のロググループ内のイベントは除きますLogGroupToExclude2。

```
aws logs put-account-policy \
    --policy-name "ExamplePolicy" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
    --policy-document '{"RoleArn":"arn:aws:iam::123456789012:role/
CWLtoKinesisRole", "DestinationArn":"arn:aws:kinesis:region:123456789012:stream/
TestStream", "FilterPattern": "Test", "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
 "LogGroupToExclude2"]' \
    --scope "ALL"
```

8. サブスクリプションフィルターを設定すると、 CloudWatch Logs はフィルターパターンと選択条件に一致するすべての受信ログイベントをストリームに転送します。

   selection-criteria フィールドはオプションですが、サブスクリプションフィルターから無限ログの再帰を引き起こす可能性のあるロググループを除外するために重要です。この問題と除外するロググループの決定の詳細については、「」を参照してくださいログの再帰防止。現在、 NOT IN は でサポートされている唯一の演算子ですselection-criteria。

   ログイベントのフローを確認するには、Kinesis Data Streams シャードイテレーターを使用し、Kinesis Data Streams get-records コマンドを使用していくつかの Kinesis Data Streams レコードを取得します。

```
aws kinesis get-shard-iterator --stream-name TestStream --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 Data Streams がデータを返す前に、このコマンドを数回使用する必要がある場合があります。

レコードの配列を含むレスポンスが表示されます。Kinesis データストリームレコードの 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 でデコードおよび解凍されたデータは、次の構造を使用して JSON としてフォーマットされます。

```
{
    "messageType": "DATA_MESSAGE",
    "owner": "123456789012",
    "logGroup": "Example1",
    "logStream": "logStream1",
    "subscriptionFilters": [
        "ExamplePolicy"
    ],
    "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\"}"
        }
    ],
    "policyLevel": "ACCOUNT_LEVEL_POLICY"
}
```

データ構造の主な要素は次のとおりです。

messageType

データメッセージは、"DATA_MESSAGE" 型を使用します。 CloudWatch Logs は、主に送信先に到達可能かどうかを確認するために、「CONTROL_MESSAGE」タイプの Kinesis Data Streams レコードを出力することがあります。

owner (オーナー)

発信元ログデータの 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。

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。

policyLevel

ポリシーが適用されたレベル。「ACCOUNT_LEVEL_POLICYpolicyLevel」は、アカウントレベルのサブスクリプションフィルターポリシーの です。

例 2: を使用したサブスクリプションフィルター AWS Lambda

この例では、ログデータを AWS Lambda 関数に送信する CloudWatch Logs アカウントレベルのサブスクリプションフィルターポリシーを作成します。

> **Warning**
>
> 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。関数が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。すべてのロググループまたはアカウントのロググループのサブセットのログイベントが送信先に転送されるため、スロットリングのリスクがあります。Lambda の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 のアカウントレベルのサブスクリプションフィルターポリシーを作成するには

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 デベロッパーガイドの「ステップ 2.2: IAM ロール (実行ロール) を作成する」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 = Buffer.from(input.awslogs.data, 'base64');
    zlib.gunzip(payload, function(e, result) {
        if (e) {
            context.fail(e);
        } else {
            result = JSON.parse(result.toString());
            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 fileb://file-path/helloWorld.zip \
    --role lambda-execution-role-arn \
    --handler helloWorld.handler \
    --runtime nodejs18.x
```

5. 関数を実行するアクセス許可を CloudWatch ログに付与します。次のコマンドを使用して、プレースホルダーアカウントを自分のアカウントに置き換えます。

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

6. 次のコマンドを使用してアカウントレベルのサブスクリプションフィルターポリシーを作成し、プレースホルダーアカウントを自分のアカウントに置き換えます。この例では、 文字列を含むすべてのログイベントERRORがストリーミングされます。ただし、 LogGroupToExclude1および という名前のロググループ内のイベントは除きますLogGroupToExclude2。

```
aws logs put-account-policy \
    --policy-name "ExamplePolicyLambda" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
    --policy-document
 '{"DestinationArn":"arn:aws:lambda:region:123456789012:function:helloWorld",
 "FilterPattern": "Test", "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
 "LogGroupToExclude2"]' \
    --scope "ALL"
```

サブスクリプションフィルターを設定すると、 CloudWatch Logs はフィルターパターンと選択条件に一致するすべての受信ログイベントをストリームに転送します。

selection-criteria フィールドはオプションですが、サブスクリプションフィルターから無限ログの再帰を引き起こす可能性のあるロググループを除外するために重要です。この問題

と除外するロググループの決定の詳細については、「」を参照してください[ログの再帰防止]。現在、 NOT IN は でサポートされている唯一の演算子ですselection-criteria。

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 Example1 --log-stream-name logStream1 --
   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] (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。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 でデコードおよび解凍されたデータは、次の構造を使用して JSON としてフォーマットされます。

   ```
   {
       "messageType": "DATA_MESSAGE",
       "owner": "123456789012",
       "logGroup": "Example1",
       "logStream": "logStream1",
       "subscriptionFilters": [
           "ExamplePolicyLambda"
       ],
       "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\"}"
        }
    ],
    "policyLevel": "ACCOUNT_LEVEL_POLICY"
}
```

> **Note**
>
> アカウントレベルのサブスクリプションフィルターは、送信先の Lambda 関数のロググループには適用されません。これは、取り込み請求の増加につながる可能性のある無限ログの再帰を防ぐためです。この問題の詳細については、ログの再帰防止「」を参照してください。

データ構造の主な要素は次のとおりです。

messageType

データメッセージは、"DATA_MESSAGE" 型を使用します。 CloudWatch Logs は、主に送信先に到達可能かどうかを確認するために、「CONTROL_MESSAGE」タイプの Kinesis Data Streams レコードを出力することがあります。

owner (オーナー)

発信元ログデータの 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。

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。

policyLevel

ポリシーが適用されたレベル。「ACCOUNT_LEVEL_POLICYpolicyLevel」は、アカウントレベルのサブスクリプションフィルターポリシーの です。

例 3: Amazon Data Firehose を使用したサブスクリプションフィルター

この例では、定義したフィルターに一致する受信 CloudWatch ログイベントを Amazon Data Firehose 配信ストリームに送信する Logs アカウントレベルのサブスクリプションフィルターポリシーを作成します。 CloudWatch Logs から Amazon Data Firehose に送信されるデータは、すでに gzip レベル 6 圧縮で圧縮されているため、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。その後、Firehose の解凍機能を使用して、ログを自動的に解凍できます。詳細については、 CloudWatch 「ログを使用した Kinesis Data Firehose への書き込み」を参照してください。

> **Warning**
>
> Firehose ストリームを作成する前に、生成され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 Firehose ストリームを必ず作成してください。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。Firehose ストリームのボリューム制限の詳細については、「Amazon Data Firehose データ制限」を参照してください。

Firehose のサブスクリプションフィルターを作成するには

1. Amazon Simple Storage Service (Amazon S3) バケットを作成します。 CloudWatch Logs 専用に作成され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. Amazon S3 バケットにデータを配置するアクセス許可を Amazon Data Firehose に付与する IAM ロールを作成します。

   詳細については、「Amazon Data Firehose デベロッパーガイド」の「Amazon Data Firehose によるアクセスの制御」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。

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

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 値を書き留めておきます。

```
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": "EXAMPLE50GAB4HC5F431",
        "CreateDate": "2023-05-29T13:46:29.431Z",
        "RoleName": "FirehosetoS3Role",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/FirehosetoS3Role"
    }
}
```

4. アクセス許可ポリシーを作成して、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. 次のように送信先の Firehose 配信ストリームを作成し、RoleARN と BucketARN のプレースホルダー値を、作成したロールとバケット ARNs。

```
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"}'
```

NFirehose は、配信された Amazon S3 オブジェクトに YYYY/MM/DD/HH UTC 時間形式のプレフィックスを自動的に使用します。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数分待ちます。Firehose describe-delivery-stream コマンドを使用して、DeliveryStreamDescription.DeliveryStreamStatus プロパティを確認できます。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. Firehose 配信ストリームにデータを配置するアクセス許可を CloudWatch Logs に付与する IAM ロールを作成します。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。

   このポリシーには、「混乱した代理」のセキュリティ上の問題を防止するための aws:SourceArn グローバル条件コンテキストキーが含まれています。詳細については、「混乱した代理の防止」を参照してください。

```
{
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Service": "logs.amazonaws.com" },
    "Action": "sts:AssumeRole",
    "Condition": {
         "StringLike": {
             "aws:SourceArn": "arn:aws:logs:region:123456789012:*"
         }
     }
  }
}
```

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 値を書き留めておきます。

```
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.amazonaws.com"
                },
                "Condition": {
                     "StringLike": {
```

```
                        "aws:SourceArn": "arn:aws:logs:region:123456789012:*"
                    }
                }
            }
        },
        "RoleId": "AAOIIAH450GAB4HC5F431",
        "CreateDate": "2015-05-29T13:46:29.431Z",
        "RoleName": "CWLtoKinesisFirehoseRole",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"
    }
}
```

10. アクセス許可ポリシーを作成して、Logs CloudWatch 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:PutRecord"],
        "Resource":[
            "arn:aws:firehose:region:account-id:deliverystream/delivery-stream-name"]
      }
    ]
}
```

11. 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 put-role-policy。

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

12. Amazon Data Firehose 配信ストリームがアクティブ状態になり、IAM ロールを作成したら、CloudWatch Logs アカウントレベルのサブスクリプションフィルターポリシーを作成できます。このポリシーは、選択したロググループから Amazon Data Firehose 配信ストリームへのリアルタイムログデータのフローを直ちに開始します。

```
aws logs put-account-policy \
    --policy-name "ExamplePolicyFirehose" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
```

```
    --policy-document '{"RoleArn":"arn:aws:iam::123456789012:role/
CWLtoKinesisFirehoseRole", "DestinationArn":"arn:aws:firehose:us-
east-1:123456789012:deliverystream/delivery-stream-name", "FilterPattern": "Test",
 "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
 "LogGroupToExclude2"]' \
    --scope "ALL"
```

13. サブスクリプションフィルターを設定すると、 CloudWatch Logs はフィルターパターンに一致する受信ログイベントを Amazon Data Firehose 配信ストリームに転送します。

    selection-criteria フィールドはオプションですが、サブスクリプションフィルターから無限ログの再帰を引き起こす可能性のあるロググループを除外するために重要です。この問題と除外するロググループの決定の詳細については、「」を参照してくださいログの再帰防止。現在、 NOT IN は でサポートされている唯一の演算子ですselection-criteria。

    Amazon Data Firehose 配信ストリームに設定されているバッファ間隔に基づいて、データが Amazon S3 に表示され始めます。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": "2023-10-29T00:01:25.000Z",
            "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/2023/10/29/00/my-delivery-stream-2023-10-29-00-35-40-
EXAMPLE-7e66-49bc-9fd4-fc9819cc8ed3",
            "Owner": {
```

```
                "DisplayName": "cloudwatch-logs",
                "ID": "EXAMPLE6be062b19584e0b7d84ecc19237f87b6"
            },
            "Size": 5752
        }
    ]
}
```

```
aws s3api get-object --bucket 'my-bucket' --key 'firehose/2023/10/29/00/my-
delivery-stream-2023-10-29-00-01-21-a188030a-62d2-49e6-b7c2-b11f1a7ba250'
 testfile.gz

{
    "AcceptRanges": "bytes",
    "ContentType": "application/octet-stream",
    "LastModified": "Thu, 29 Oct 2023 00:07:06 GMT",
    "ContentLength": 593,
    "Metadata": {}
}
```

Simple Storage Service (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
```

クロスアカウントクロスリージョンサブスクリプション

別の AWS アカウントの所有者と共同作業し、Amazon Kinesis や Amazon Data Firehose ストリームなどの AWS リソースでログイベントを受信できます (これはクロスアカウントデータ共有と呼ばれます)。例えば、このログイベントデータは、一元化された Kinesis Data Streams または Firehose ストリームから読み取って、カスタム処理と分析を実行できます。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 データストリーム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。

> **Note**
>
> ロググループと送信先は同じ AWS リージョンにある必要があります。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。以下のセクションの例では、リージョン固有のリソースはすべて米国東部 (バージニア北部)) に作成されます。

トピック

Kinesis Data Streams を使用したクロスアカウントクロスリージョンログデータ共有

クロスアカウントサブスクリプションを作成するときに、単一のアカウントまたは組織を送信者として指定できます。組織を指定した場合、この手順により組織内のすべてのアカウントがレシーバーアカウントにログを送信できるようになります。

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。

- ログデータ送信者 — 受信者から送信先情報を取得し、指定した送信先にログイベントを送信する準備ができていることを CloudWatch Logs に通知します。このセクションの残りの手順では、ログデータ送信者は架空の AWS アカウント番号 111111111111 で表示されます。

  1 つの組織で複数のアカウントを 1 つの受信者アカウントにログを送信する場合は、組織内のすべてのアカウントに対して受信者アカウントにログを送信するアクセス許可を付与するポリシーを作成することができます。引き続き、送信者アカウントごとに個別のサブスクリプションフィルターを設定する必要があります。
- ログデータ受信者 — Kinesis Data Streams ストリームをカプセル化する送信先を設定し、受信者がログデータを受信したいことを CloudWatch Logs に通知します。この後、受信者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、ログデータ受信者は架空の AWS アカウント番号 999999999999 で表示されます。

クロスアカウントユーザーからログイベントの受信を開始するには、まずログデータ受信者が CloudWatch ログ送信先を作成します。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リソースの Amazon AWS リソースネーム (ARN)。

ロールの ARN

選択したストリームにデータを配置するために必要なアクセス許可を CloudWatch ログに付与する 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) 。

> **Note**
>
> ロググループと送信先は同じ AWS リージョンにある必要があります。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。次のセクションの例では、リージョン固有のリソースはすべて米国東部 (バージニア北部) で作成されます。

トピック

新しいクロスアカウントサブスクリプションの設定

次のセクションの手順に従って、新しいクロスアカウントログサブスクリプションを設定します。

トピック

- ステップ 3: クロスアカウント宛先の IAM アクセス許可を追加/検証する
- ステップ 4: サブスクリプションフィルターを作成する
- ログイベントの送信を検証
- ランタイムの送信先のメンバーシップを変更

ステップ 1: 送信先を作成する

> **Important**
> 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、ログデータ受信者アカウントの AWS アカウント ID は 999999999999 で、ログデータ送信者 AWS アカウント ID は 111111111111 です。

この例では、 という名前の Kinesis Data Streams ストリームと RecipientStream、 CloudWatch Logs がデータを書き込めるようにするロールを使用して送信先を作成します。

送信先が作成されると、 CloudWatch Logs は受信者アカウントに代わって送信先にテストメッセージを送信します。サブスクリプションフィルターが後でアクティブになると、 CloudWatch Logs はソースアカウントに代わってログイベントを送信先に送信します。

送信先を作成するには

1. 受信者アカウントから、Kinesis データストリーム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. 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 プロパティを確認できます。さらに、StreamDescription.StreamARN 値は後で CloudWatch Logs に渡されるため、書き留めておきます。

```
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": "340282366920938463463374607431768EXAMPLE",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
 "49551135218688818456679503831981458784591352702 18EXAMPLE"
        }
      }
    ]
  }
}
```

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. ストリームにデータを配置するアクセス許可を 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 に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。

   このポリシーには、「混乱した代理」のセキュリティ上の問題を防止するための sourceAccountId が指定された aws:SourceArn グローバル条件コンテキストキーが含まれています。最初の呼び出しでソースアカウント ID が不明な場合は、送信元 ARN フィールドに送信先 ARN を指定することをお勧めします。後続の呼び出しでは、送信元 ARN を、最初の呼び出しで取得した実際の送信元 ARN に設定する必要があります。詳細については、「混乱した代理の防止」を参照してください。

```
{
    "Statement": {
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.amazonaws.com"
        },
        "Condition": {
            "StringLike": {
                "aws:SourceArn": [
                    "arn:aws:logs:region:sourceAccountId:*",
                    "arn:aws:logs:region:recipientAccountId:*"
```

```
                ]
            }
        },
        "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 値は後で CloudWatch ログに渡されるため、書き留めておきます。

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

{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Condition": {
                    "StringLike": {
                        "aws:SourceArn": [
                            "arn:aws:logs:region:sourceAccountId:*",
                            "arn:aws:logs:region:recipientAccountId:*"
                        ]
                    }
                },
                "Principal": {
                    "Service": "logs.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、Logs CloudWatch 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 ~/PermissionsForCWL.json でアクセス許可ポリシーを作成します。

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

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. ストリームがアクティブ状態になり、IAM ロールを作成したら、 CloudWatch ログ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DestinationArnされる を書き留めます。

```
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。このポリシーでは、ログを指定する必要があります。PutSubscriptionFilterアクション と は、送信先にアクセスするためのアクセス許可を送信者アカウントに付与します。

このポリシーは、ログを送信する AWS アカウントにアクセス許可を付与します。ポリシーの中で対象のアカウントを 1 つだけ指定してもよいですが、送信者アカウントが組織のメンバーのものである場合は組織 ID を指定することもできます。このように、ポリシーを 1 つ作成するだけで、1 つの組織内の複数のアカウントが送信先アカウントにログを送信できるように設定できます。

テキストエディタを使用して ~/AccessPolicy.json という名前のファイルを作成し、以下のいずれかのポリシーステートメントを使用します。

この最初の例のポリシーでは、組織内で o-1234567890 という ID を持つすべてのアカウントが、受信者アカウントにログを送信することを許可します。

```
{
    "Version" : "2012-10-17",
    "Statement" : [
        {
            "Sid" : "",
            "Effect" : "Allow",
            "Principal" : "*",
            "Action" : "logs:PutSubscriptionFilter",
            "Resource" :
 "arn:aws:logs:region:999999999999:destination:testDestination",
            "Condition": {
               "StringEquals" : {
                   "aws:PrincipalOrgID" : ["o-1234567890"]
                }
            }
        }
    ]
}
```

次の例では、ログデータの送信者アカウント (111111111111) がログデータの受信者アカウントにログを送信できるようにします。

```
{
```

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

c. 前のステップで作成したポリシーを送信先に添付します。

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

このアクセスポリシーにより、ID 111111111111 の AWS アカウントのユーザーは、ARN arn:aws:logs:*region* :999999999999:destination:testDestination を使用して送信先PutSubscriptionFilterに対して を呼び出すことができます。この送信先 PutSubscriptionFilter に対して他のユーザーが を呼び出そうとすると、拒否されます。

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

完了したら、クロスアカウントアクセス許可 AWS Organizations に を使用している場合は、「」のステップに従いますステップ 2: IAM ロールを作成する (組織を使用している場合のみ)。組織を使用せずに他のアカウントに直接アクセス許可を付与する場合は、そのステップを飛ばして「ステップ 4: サブスクリプションフィルターを作成する」に進みます。

ステップ 2: IAM ロールを作成する (組織を使用している場合のみ)

前のセクションで アカウント 111111111111 に直接アクセス許可を付与するのではなく、アカウント 111111111111 が属する組織にアクセス許可を付与するアクセスポリシーを使用することにより送信先を作成した場合は、このセクションのステップを実行します。それ以外の場合は、「ステップ 4: サブスクリプションフィルターを作成する」に進みます。

このセクションのステップでは、IAM ロールを作成します。これにより、送信者アカウントが受信者の送信先に対してサブスクリプションフィルターを作成するアクセス許可を持っているかどうかを引き受けて検証 CloudWatch できます。

このセクションの手順は、送信者アカウントで実行してください。ロールは送信者アカウントに存在する必要があり、このロールの ARN はサブスクリプションフィルターで指定します。この例では、送信者アカウントは 111111111111 です。

AWS Organizationsを使用してクロスアカウントのログサブスクリプションに必要な IAM ロールを作成する方法

1. 以下の信頼ポリシーを作成し、/TrustPolicyForCWLSubscriptionFilter.json という名前のテキストファイルに保存します。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.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

2. このポリシーを使用する IAM ロールを作成します。下記のコマンドが返す Arn の値は後ほど必要になるため、書き留めておきます。この例では、作成するロールに CWLtoSubscriptionFilterRole という名前を付けます。

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

3. アクセス許可ポリシーを作成して、Logs CloudWatch がアカウントで実行できるアクションを定義します。

   a. まず、テキストエディタを使用して、~/PermissionsForCWLSubscriptionFilter.json という名前のファイルに以下のようなアクセス許可ポリシーを作成します。

```
{
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:region:111111111111:log-
group:LogGroupOnWhichSubscriptionFilterIsCreated:*"
        }
    ]
}
```

b. 次のコマンドを入力して、先ほど作成したアクセス許可ポリシーを、ステップ 2 で作成したロールに関連付けます。

```
aws iam put-role-policy
    --role-name CWLtoSubscriptionFilterRole
    --policy-name Permissions-Policy-For-CWL-Subscription-filter
    --policy-document file://~/PermissionsForCWLSubscriptionFilter.json
```

終了したら、「ステップ 4: サブスクリプションフィルターを作成する」に進みます。

ステップ 3: クロスアカウント宛先の IAM アクセス許可を追加/検証する

AWS クロスアカウントポリシーの評価ロジックに従って、クロスアカウントリソース (サブスクリプションフィルターの送信先として使用される Kinesis または Firehose ストリームなど) にアクセスするには、クロスアカウントの送信先リソースへの明示的なアクセスを提供するアイデンティティベースのポリシーを送信側アカウントに含める必要があります。ポリシーの評価論理の詳細については、「クロスアカウントポリシーの評価論理」を参照してください。

ID ベースのポリシーは、サブスクリプションフィルターの作成に使用している IAM ロールまたは IAM ユーザーにアタッチできます。送信アカウントにこのポリシーが存在する必要があります。管理者ロールを使用してサブスクリプションフィルタを作成している場合は、このステップをスキップして ステップ 4: サブスクリプションフィルターを作成する に進んでください。

クロスアカウントに必要な IAM アクセス許可を追加または検証するには

1. 次のコマンドを入力して、 AWS ログコマンドの実行に使用されている IAM ロールまたは IAM ユーザーを確認します。

```
aws sts get-caller-identity
```

このコマンドにより、以下のような出力が返されます。

```
{
"UserId": "User ID",
"Account": "sending account id",
"Arn": "arn:aws:sending account id:role/user:RoleName/UserName"
}
```

RoleName または で表される値を書き留めます*UserName*。

2. 送信側アカウント AWS Management Console で にサインインし、ステップ 1 で入力したコマンドの出力で返された IAM ロールまたは IAM ユーザーを使用して、アタッチされたポリシーを検索します。
3. このロールまたはユーザーにアタッチされたポリシーが、クロスアカウントの宛先リソースで logs:PutSubscriptionFilter を呼び出すための明示的なアクセス許可が付与されていることを確認します。次の例で示すポリシーは、推奨されるアクセス許可を示しています。

   次のポリシーは、1 つの AWS アカウント、アカウント でのみ、任意の送信先リソースにサブスクリプションフィルターを作成するアクセス許可を提供します123456789012。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow subscription filters on any resource in one specific
 account",
            "Effect": "Allow",
            "Action": "logs:PutSubscriptionFilter",
            "Resource": [
                "arn:aws:logs:*:*:log-group:*",
                "arn:aws:logs:*:123456789012:destination:*"
            ]
        }
    ]
}
```

   次のポリシーは、単一のアカウント、 AWS アカウント sampleDestinationで という名前の特定の送信先リソースにのみサブスクリプションフィルターを作成するアクセス許可を提供します123456789012。

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

```
    "Statement": [
        {
            "Sid": "Allow subscription filters on one specific resource in one
 specific account",
            "Effect": "Allow",
            "Action": "logs:PutSubscriptionFilter",
            "Resource": [
                "arn:aws:logs:*:*:log-group:*",
                "arn:aws:logs:*:123456789012:destination:sampleDestination"
            ]
        }
    ]
}
```

ステップ 4: サブスクリプションフィルターを作成する

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。

> **Note**
>
> サブスクリプションフィルターのためのアクセス許可を組織全体に付与する際は、ステップ 2: IAM ロールを作成する (組織を使用している場合のみ) で作成した IAM ロールの ARN を使用する必要があります。

次の例では、送信アカウントにサブスクリプションフィルターが作成されます。フィルターは AWS CloudTrail イベントを含むロググループに関連付けられ、「ルート AWS 」認証情報によって行われたすべてのログアクティビティが、以前に作成した宛先に配信されます。その送信先は「」というストリームをカプセル化RecipientStreamします。

以下のセクションの残りのステップでは、「 AWS CloudTrail ユーザーガイド CloudTrail」の CloudWatch 「ログへのイベントの送信」の指示に従って、 CloudTrail イベントを含むロググループを作成していることを前提としています。そのステップでは、ロググループに `CloudTrail/logs` という名前を付けることになっています。

次のコマンドを入力するときは、必ず IAM ユーザーとしてサインインしているか、ステップ 3: クロスアカウント宛先の IAM アクセス許可を追加/検証する にポリシーを追加した IAM ロールを使用してサインインしていることを確認してください。

```
aws logs put-subscription-filter \
    --log-group-name "CloudTrail/logs" \
    --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 リージョンにある必要があります。ただし、送信先は、別のリージョンにある Kinesis Data Streams ストリームなどの AWS リソースを指すことができます。

ログイベントの送信を検証

サブスクリプションフィルターを作成すると、 CloudWatch Logs は、フィルターパターンに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にカプセル化された「」という名前のストリームに転送しますRecipientStream。送信先所有者は、aws kinesis get-shard-iterator コマンドを使用して Kinesis Data Streams 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 Data Streams レコードを取得することで、これが起こっ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 データストリーム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 データストリームレコードを含んでいるレスポンスが表示されます。Kinesis データストリームレコードのデータ属性は、gzip 形式で圧縮され、さらに base64 でエンコードされています。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 でデコードおよび解凍されたデータは、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/logs",
    "logStream": "111111111111_CloudTrail/logs_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 (オーナー)

発信元ログデータの 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、主に送信先に到達可能かどうかを確認するために、「CONTROL_MESSAGE」タイプの Kinesis Data Streams レコードを出力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。

ランタイムの送信先のメンバーシップを変更

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を使用して、送信先で `put-destination-policy` コマンドを使用できます。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. 現在送信先の testDestination に関連付けられているポリシーを取得し、 を書き留めますAccessPolicy。

```
aws logs describe-destinations \
    --destination-name-prefix "testDestination"
```

```
{
 "Destinations": [
   {
     "DestinationName": "testDestination",
     "RoleArn": "arn:aws:iam::999999999999:role/CWLtoKinesisRole",
     "DestinationArn":
 "arn:aws:logs:region:999999999999:destination:testDestination",
     "TargetArn": "arn:aws:kinesis:region:999999999999:stream/RecipientStream",
     "AccessPolicy": "{\"Version\": \"2012-10-17\", \"Statement\":
 [{\"Sid\": \"\", \"Effect\": \"Allow\", \"Principal\": {\"AWS\":
 \"111111111111\"}, \"Action\": \"logs:PutSubscriptionFilter\", \"Resource\":
 \"arn:aws:logs:region:999999999999: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 ~/NewAccessPolicy.json ファイルに入れます。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "222222222222"
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:region:999999999999:destination:testDestination"
    }
  ]
}
```

3. PutDestinationPolicy を呼び出して、NewAccessPolicy.json ファイルで定義されたポリシーを送信先と関連付けます。

```
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 の所有者がサブスクリプションフィルターを作成すると、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。

既存のクロスアカウントサブスクリプションの更新

送信先アカウントが特定の送信者アカウントにのみアクセス許可を付与しているクロスアカウントのログサブスクリプションがあり、このサブスクリプションを更新して送信先アカウントが組織内のすべてのアカウントにアクセスできるようにする場合は、このセクションのステップを実施します。

トピック

ステップ 1: サブスクリプションフィルターを更新する

> **Note**
>
> この手順は、AWS サービスからのログ記録を有効にする に記載されているサービスによって作成されたログのクロスアカウントのサブスクリプションにのみ必要です。これらのロググループのいずれかで作成されたログを操作していない場合は、 ステップ 2: 既存の送信先アクセスポリシーを更新する にスキップできます。

場合によっては、送信先アカウントにログを送信する、すべての送信者アカウントのサブスクリプションフィルターを更新する必要があります。この更新では、IAM ロールが追加されます。これにより、送信者アカウントが受信者アカウントにログを送信するアクセス許可を持っていることを引き受け、検証 CloudWatch できます。

すべての送信者アカウントについてクロスアカウントサブスクリプションのアクセス許可に組織 ID を使用するように更新するには、このセクションのステップを実施します。

このセクションの例では、2 つのアカウント 111111111111 と 222222222222 は、アカウント 999999999999 にログを送信するために作成されたサブスクリプションフィルターをすでに持っています。既存のサブスクリプションフィルター値は次のとおりです。

```
## Existing Subscription Filter parameter values
```

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

現在のサブスクリプションフィルターパラメータ値を見つける必要がある場合は、次のコマンドを入力します。

```
aws logs describe-subscription-filters
    \ --log-group-name "my-log-group-name"
```

サブスクリプションフィルターを更新して、クロスアカウントログの権限で組織 ID の使用をスタートする方法

1. 以下の信頼ポリシーを作成し、~/TrustPolicyForCWL.json という名前のテキストファイルに保存します。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.amazonaws.com" },
       "Action": "sts:AssumeRole"
     }
   }
   ```

2. このポリシーを使用する IAM ロールを作成します。下記のコマンドが返す Arn 値の Arn の値は後ほど必要になるため、書き留めておきます。この例では、作成するロールに CWLtoSubscriptionFilterRole という名前を付けます。

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

3. アクセス許可ポリシーを作成して、Logs CloudWatch がアカウントで実行できるアクションを定義します。

   a. まず、テキストエディタを使用して、/PermissionsForCWLSubscriptionFilter.json という名前のファイルに以下のようなアクセス許可ポリシーを作成します。

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:region:111111111111:log-
group:LogGroupOnWhichSubscriptionFilterIsCreated:*"
        }
    ]
}
```

b. 次のコマンドを入力して、先ほど作成したアクセス許可ポリシーを、ステップ 2 で作成したロールに関連付けます。

```
aws iam put-role-policy
    --role-name CWLtoSubscriptionFilterRole
    --policy-name Permissions-Policy-For-CWL-Subscription-filter
    --policy-document file://~/PermissionsForCWLSubscriptionFilter.json
```

4. 次のコマンドを入力して、サブスクリプションフィルターを更新します。

```
aws logs put-subscription-filter
    \ --log-group-name "my-log-group-name"
    \ --filter-name "RecipientStream"
    \ --filter-pattern "{$.userIdentity.type = Root}"
    \ --destination-arn
 "arn:aws:logs:region:999999999999:destination:testDestination"
    \ --role-arn "arn:aws:iam::111111111111:role/CWLtoSubscriptionFilterRole"
```

ステップ 2: 既存の送信先アクセスポリシーを更新する

すべての送信者アカウントのサブスクリプションフィルターを更新した後、受信者アカウントの送信先アクセスポリシーを更新できます。

以下の例では、受信者アカウントは 999999999999、送信先は testDestination となっています。

この更新により、ID o-1234567890 を持つ組織に属するすべてのアカウントが、受信者アカウントにログを送信できるようになりました。サブスクリプションフィルターが作成されたアカウントのみが、実際に受信者アカウントにログを送信します。

受信者アカウントの送信先アクセスポリシーを更新して、権限の組織 ID の使用をスタートする方法

1. 受信者アカウントで、テキストエディタを使用して、以下の内容の ~/AccessPolicy.json ファイルを作成します。

```
{
    "Version" : "2012-10-17",
    "Statement" : [
        {
            "Sid" : "",
            "Effect" : "Allow",
            "Principal" : "*",
            "Action" : "logs:PutSubscriptionFilter",
            "Resource" :
 "arn:aws:logs:region:999999999999:destination:testDestination",
            "Condition": {
               "StringEquals" : {
                   "aws:PrincipalOrgID" : ["o-1234567890"]
                }
            }
        }
    ]
}
```

2. 次のコマンドを入力して、先ほど作成したポリシーを既存の送信先にアタッチします。特定の AWS アカウント ID をリストにしたアクセスポリシーではなく、組織 ID を含むアクセスポリシーを使用するように送信先を更新するには、force パラメータを指定します。

> **Warning**
>
> に記載されている AWS のサービスによって送信されたログを使用している場合はAWS サービスからのログ記録を有効にする、このステップを実行する前に、 で説明されているように、すべての送信者アカウントのサブスクリプションフィルターを更新しておく必要がありますステップ 1: サブスクリプションフィルターを更新する。

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

Firehose を使用したクロスアカウントクロスリージョンログデータ共有

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。

- ログデータ送信者 — 受信者から送信先情報を取得し、指定した送信先にログイベントを送信する準備ができていることを CloudWatch Logs に通知します。このセクションの残りの手順では、ログデータ送信者は架空の AWS アカウント番号 111111111111 で表示されます。
- ログデータ受信者 — Kinesis Data Streams ストリームをカプセル化する送信先を設定し、受信者がログデータを受信したいことを CloudWatch Logs に通知します。この後、受信者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、ログデータ受信者は架空の AWS アカウント番号 222222222222 で表示されます。

このセクションの例では、Amazon S3 ストレージを備えた Firehose 配信ストリームを使用します。さまざまな設定で Firehose 配信ストリームを設定することもできます。詳細については、「Firehose 配信ストリームの作成」を参照してください。

> **Note**
>
> ロググループと送信先は同じ AWS リージョンにある必要があります。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。

> **Note**
>
> 同じアカウントとクロスリージョン配信ストリームの Firehose サブスクリプションフィルターがサポートされています。

トピック

- 実行時の送信先のメンバーシップの変更

ステップ 1: Firehose 配信ストリームを作成する

> **Important**
>
> 次の手順を完了する前に、Firehose が Amazon S3 バケットにアクセスできるように、アクセスポリシーを使用する必要があります。詳細については、「Amazon Data Firehose デベロッパーガイド」の「アクセスの制御」を参照してください。
> このセクションのすべての手順 (ステップ 1) は、ログデータの受取人アカウントで行われます。
> 次のサンプルコマンドでは、米国東部 (バージニア北部) が使用されています。このリージョンを、デプロイに適したリージョンに置き換えます。

送信先として使用する Firehose 配信ストリームを作成するには

1. Amazon S3 バケットの作成

```
aws s3api create-bucket --bucket firehose-test-bucket1 --create-bucket-
configuration LocationConstraint=us-east-1
```

2. Firehose にバケットにデータを配置するアクセス許可を付与する IAM ロールを作成します。

   a. まず、テキストエディタを使用して、ファイル ~/TrustPolicyForFirehose.json で信頼ポリシーを作成します。

```
{ "Statement": { "Effect": "Allow", "Principal": { "Service":
 "firehose.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition":
 { "StringEquals": { "sts:ExternalId":"222222222222" } } } }
```

   b. 作成したばかりの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。

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

   c. このコマンドの出力は、次のようになります。ロール名とロール ARN を書き留めます。

```
{
```

```
    "Role": {
        "Path": "/",
        "RoleName": "FirehosetoS3Role",
        "RoleId": "AROAR3BXASEKW7K635M53",
        "Arn": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
        "CreateDate": "2021-02-02T07:53:10+00:00",
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Effect": "Allow",
                "Principal": {
                    "Service": "firehose.amazonaws.com"
                },
                "Action": "sts:AssumeRole",
                "Condition": {
                    "StringEquals": {
                        "sts:ExternalId": "222222222222"
                    }
                }
            }
        }
    }
}
```

3. アクセス許可ポリシーを作成して、Firehose がアカウントで実行できるアクションを定義します。

   a. まず、テキストエディタを使用して、~/PermissionsForFirehose.json という名前のファイルに以下のようなアクセス許可ポリシーを作成します。ユースケースによっては、このファイルにさらにアクセス権限を追加する必要がある場合があります。

```
{
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:ListBucket"
        ],
        "Resource": [
            "arn:aws:s3:::firehose-test-bucket1",
            "arn:aws:s3:::firehose-test-bucket1/*"
        ]
    }]
```

```
}
```

b. 次のコマンドを入力して、先ほど作成したアクセス権限ポリシーを IAM ロールに関連付けます。

```
aws iam put-role-policy --role-name FirehosetoS3Role --policy-name
 Permissions-Policy-For-Firehose-To-S3 --policy-document file://~/
PermissionsForFirehose.json
```

4. 次のコマンドを入力して、Firehose 配信ストリームを作成します。*my-role-arn* と をデプロイに適した値*my-bucket-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::222222222222:role/FirehosetoS3Role", "BucketARN":
 "arn:aws:s3:::firehose-test-bucket1"}'
```

出力は次の例に類似したものになります:

```
{
    "DeliveryStreamARN": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream"
}
```

ステップ 2: 送信先を作成する

> **Important**
>
> 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。

送信先が作成されると、 CloudWatch Logs は受信者アカウントに代わって送信先にテストメッセージを送信します。サブスクリプションフィルターが後でアクティブになると、 CloudWatch Logs はソースアカウントに代わってログイベントを送信先に送信します。

送信先を作成するには

1. で作成した Firehose ストリームがアクティブステップ 1: Firehose 配信ストリームを作成するになるまで待ちます。次のコマンドを使用して、StreamDescription.StreamStatus プロパティを確認できます。

```
aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
```

さらに、DeliveryStreamDescription.DeliveryStreamARN 値は後のステップで使用する必要があるため、書き留めておきます。このコマンドの出力例:

```
{
    "DeliveryStreamDescription": {
        "DeliveryStreamName": "my-delivery-stream",
        "DeliveryStreamARN": "arn:aws:firehose:us-
east-1:222222222222:deliverystream/my-delivery-stream",
        "DeliveryStreamStatus": "ACTIVE",
        "DeliveryStreamEncryptionConfiguration": {
            "Status": "DISABLED"
        },
        "DeliveryStreamType": "DirectPut",
        "VersionId": "1",
        "CreateTimestamp": "2021-02-01T23:59:15.567000-08:00",
        "Destinations": [
            {
                "DestinationId": "destinationId-000000000001",
                "S3DestinationDescription": {
                    "RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
                    "BucketARN": "arn:aws:s3:::firehose-test-bucket1",
                    "BufferingHints": {
                        "SizeInMBs": 5,
                        "IntervalInSeconds": 300
                    },
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "CloudWatchLoggingOptions": {
                        "Enabled": false
                    }
                },
                "ExtendedS3DestinationDescription": {
```

```
                    "RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
                    "BucketARN": "arn:aws:s3:::firehose-test-bucket1",
                    "BufferingHints": {
                        "SizeInMBs": 5,
                        "IntervalInSeconds": 300
                    },
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "CloudWatchLoggingOptions": {
                        "Enabled": false
                    },
                    "S3BackupMode": "Disabled"
                }
            }
        ],
        "HasMoreDestinations": false
    }
}
```

配信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。

2. 配信ストリームがアクティブになったら、Firehose ストリームにデータを置くアクセス許可を 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 に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。 CloudWatch Logs エンドポイントの詳細については、「Amazon CloudWatch Logs エンドポイントとクォータ」を参照してください。

   このポリシーには、「混乱した代理」のセキュリティ上の問題を防止するための sourceAccountId が指定された aws:SourceArn グローバル条件コンテキストキーが含まれています。最初の呼び出しでソースアカウント ID が不明な場合は、送信元 ARN フィールドに送信先 ARN を指定することをお勧めします。後続の呼び出しでは、送信元 ARN を、最初の呼び出しで取得した実際の送信元 ARN に設定する必要があります。詳細については、「混乱した代理の防止」を参照してください。

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

```
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringLike": {
                "aws:SourceArn": [
                    "arn:aws:logs:region:sourceAccountId:*",
                    "arn:aws:logs:region:recipientAccountId:*"
                ]
            }
        }
    }
}
```

3. aws iam create-role コマンドを使用して、作成した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。

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

以下は出力例です。後のステップで使用する必要があるため、Role.Arn の戻り値を書き留めます。

```
{
    "Role": {
        "Path": "/",
        "RoleName": "CWLtoKinesisFirehoseRole",
        "RoleId": "AROAR3BXASEKYJYWF243H",
        "Arn": "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole",
        "CreateDate": "2021-02-02T08:10:43+00:00",
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.region.amazonaws.com"
                },
                "Action": "sts:AssumeRole",
                "Condition": {
                    "StringLike": {
                        "aws:SourceArn": [
                            "arn:aws:logs:region:sourceAccountId:*",
                            "arn:aws:logs:region:recipientAccountId:*"
                        ]
```

```
                    }
                }
            }
        }
    }
}
```

4. アクセス許可ポリシーを作成して、Logs CloudWatch 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 ~/PermissionsForCWL.json でアクセス許可ポリシーを作成します。

```
{
    "Statement":[
      {
        "Effect":"Allow",
        "Action":["firehose:*"],
        "Resource":["arn:aws:firehose:region:222222222222:*"]
      }
    ]
}
```

5. 次のコマンドを入力して、アクセス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
```

6. Firehose 配信ストリームがアクティブ状態になり、IAM ロールを作成したら、 CloudWatch ログ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。後のステップでこれを destination.arn として使用するため、ペイロードで返される新しい宛先の ARN を書き留めます。

```
aws logs put-destination \

    --destination-name "testFirehoseDestination" \
    --target-arn "arn:aws:firehose:us-east-1:222222222222:deliverystream/my-
delivery-stream" \
    --role-arn "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole"

{
```

```
    "destination": {
        "destinationName": "testFirehoseDestination",
        "targetArn": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream",
        "roleArn": "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole",
        "arn": "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"}
}
```

b. 前のステップが完了したら、ログデータ受取人アカウント (222222222222) で、アクセスポリシーを送信先に関連付けます。

このポリシーにより、ログデータの送信者アカウント (111111111111) に対し、ログデータの受信者アカウント (222222222222) にある送信先にアクセスすることを許可します。テキストエディタを使用して、このポリシーを ~/AccessPolicy.json ファイルに入れることができます。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "111111111111"
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
}
```

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 アクションを指定する必要があります。クロスアカウントユーザーは、 PutSubscriptionFilterアクションを使用してログイベントを送信先に送信します。

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

ステップ 3: クロスアカウント宛先の IAM アクセス許可を追加/検証する

AWS クロスアカウントポリシーの評価ロジックに従って、クロスアカウントリソース (サブスクリプションフィルターの送信先として使用される Kinesis または Firehose ストリームなど) にアクセスするには、クロスアカウントの送信先リソースへの明示的なアクセスを提供するアイデンティティベースのポリシーを送信側アカウントに含める必要があります。ポリシーの評価論理の詳細については、「クロスアカウントポリシーの評価論理」を参照してください。

ID ベースのポリシーは、サブスクリプションフィルターの作成に使用している IAM ロールまたは IAM ユーザーにアタッチできます。送信アカウントにこのポリシーが存在する必要があります。管理者ロールを使用してサブスクリプションフィルタを作成している場合は、このステップをスキップして ステップ 4: サブスクリプションフィルターを作成する に進んでください。

クロスアカウントに必要な IAM アクセス許可を追加または検証するには

1. 次のコマンドを入力して、 AWS ログコマンドの実行に使用されている IAM ロールまたは IAM ユーザーを確認します。

   ```
   aws sts get-caller-identity
   ```

   このコマンドにより、以下のような出力が返されます。

   ```
   {
   "UserId": "User ID",
   "Account": "sending account id",
   "Arn": "arn:aws:sending account id:role/user:RoleName/UserName"
   }
   ```

   RoleName または で表される値を書き留めます*UserName*。

2. 送信側アカウント AWS Management Console で にサインインし、ステップ 1 で入力したコマンドの出力で返された IAM ロールまたは IAM ユーザーを使用して、アタッチされたポリシーを検索します。

3. このロールまたはユーザーにアタッチされたポリシーが、クロスアカウントの宛先リソースで `logs:PutSubscriptionFilter` を呼び出すための明示的なアクセス許可が付与されていることを確認します。次の例で示すポリシーは、推奨されるアクセス許可を示しています。

   次のポリシーは、1 つの AWS アカウント、アカウント でのみ、任意の送信先リソースにサブスクリプションフィルターを作成するアクセス許可を提供します123456789012。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow subscription filters on any resource in one specific
 account",
            "Effect": "Allow",
            "Action": "logs:PutSubscriptionFilter",
            "Resource": [
                "arn:aws:logs:*:*:log-group:*",
                "arn:aws:logs:*:123456789012:destination:*"
            ]
        }
    ]
}
```

次のポリシーは、単一のアカウント、 AWS アカウント sampleDestinationで という名前の特定の送信先リソースにのみサブスクリプションフィルターを作成するアクセス許可を提供します123456789012。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow subscription filters on one specific resource in one
 specific account",
            "Effect": "Allow",
            "Action": "logs:PutSubscriptionFilter",
            "Resource": [
                "arn:aws:logs:*:*:log-group:*",
                "arn:aws:logs:*:123456789012:destination:sampleDestination"
            ]
        }
    ]
}
```

ステップ 4: サブスクリプションフィルターを作成する

送信側のアカウント (この例では 111111111111) に切り替えます。次に、送信側のアカウントにサブスクリプションフィルターを作成します。この例では、フィルターは AWS CloudTrail イベン

トを含むロググループに関連付けられ、「ルート AWS 」認証情報によって行われたすべてのログアクティビティが、以前に作成した送信先に配信されます。 CloudWatch ログに AWS CloudTrail イベントを送信する方法の詳細については、「 AWS CloudTrail ユーザーガイド CloudTrail」の CloudWatch 「 ログへのイベントの送信」を参照してください。

次のコマンドを入力するときは、必ず IAM ユーザーとしてサインインしているか、ステップ 3: クロスアカウント宛先の IAM アクセス許可を追加/検証する にポリシーを追加した IAM ロールを使用してサインインしていることを確認してください。

```
aws logs put-subscription-filter \
    --log-group-name "aws-cloudtrail-logs-111111111111-300a971e" \
    --filter-name "firehose_test" \
    --filter-pattern "{$.userIdentity.type = AssumedRole}" \
    --destination-arn "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
```

ロググループと送信先は同じ AWS リージョンにある必要があります。ただし、送信先は、別のリージョンにある Firehose ストリームなどの AWS リソースを指すことができます。

ログイベントの送信の検証

サブスクリプションフィルターを作成すると、 CloudWatch Logs はフィルターパターンに一致するすべての受信ログイベントを Firehose 配信ストリームに転送します。Firehose 配信ストリームに設定されている時間バッファ間隔に基づいて、データが Amazon S3 バケットに表示され始めます。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。バケットを確認するには、次のコマンドを入力します。

```
aws s3api list-objects --bucket 'firehose-test-bucket1'
```

そのコマンドの出力は、次のようになります。

```
{
    "Contents": [
        {
            "Key": "2021/02/02/08/my-delivery-
stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba",
            "LastModified": "2021-02-02T09:00:26+00:00",
            "ETag": "\"EXAMPLEa817fb88fc770b81c8f990d\"",
            "Size": 198,
            "StorageClass": "STANDARD",
```

```
            "Owner": {
                "DisplayName": "firehose+2test",
                "ID": "EXAMPLE27fd05889c665d2636218451970ef79400e3d2aecca3adb1930042e0"
            }
        }
    ]
}
```

その後、次のコマンドを入力して、バケットから特定のオブジェクトを取得できます。key の値を、前のコマンドで検索した値に置き換えます。

```
aws s3api get-object --bucket 'firehose-test-bucket1' --key '2021/02/02/08/my-delivery-stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba' testfile.gz
```

Simple Storage Service (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コマンドを使用して調べることができます。

Linux:

```
zcat testfile.gz
```

macOS:

```
zcat <testfile.gz
```

実行時の送信先のメンバーシップの変更

所有している送信先からログ送信者を追加または削除しなければならない状況が発生することがあります。新しいアクセスポリシーを使用して、送信先で 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 333333333333 が有効になります。

1. 現在送信先の testDestination に関連付けられているポリシーを取得し、 を書き留めますAccessPolicy。

   ```
   aws logs describe-destinations \
       --destination-name-prefix "testFirehoseDestination"

   {
       "destinations": [
   ```

```
        {
            "destinationName": "testFirehoseDestination",
            "targetArn": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream",
            "roleArn": "arn:aws:iam:: 222222222222:role/CWLtoKinesisFirehoseRole",
            "accessPolicy": "{\n  \"Version\" : \"2012-10-17\",\n  \"Statement
\" : [\n    {\n      \"Sid\" : \"\",\n      \"Effect\" : \"Allow\",\n
 \"Principal\" : {\n        \"AWS\" : \"111111111111 \"\n      },\n      \"Action
\" : \"logs:PutSubscriptionFilter\",\n      \"Resource\" : \"arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination\"\n    }\n  ]\n}\n\n",
            "arn": "arn:aws:logs:us-east-1:
 222222222222:destination:testFirehoseDestination",
            "creationTime": 1612256124430
        }
    ]
}
```

2. アカウント 111111111111 が停止したこととアカウント 333333333333 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 ~/NewAccessPolicy.json ファイルに入れます。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "333333333333 "
      },
      "Action" : "logs:PutSubscriptionFilter",
      "Resource" : "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
}
```

3. 次のコマンドを使用して、NewAccessPolicy.json ファイルで定義されたポリシーを送信先と関連付けます。

```
aws logs put-destination-policy \
    --destination-name "testFirehoseDestination" \
```

```
    --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 333333333333 の所有者がサブスクリプションフィルターを作成すると、すぐに 333333333333 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。

Kinesis Data Streams を使用したクロスアカウントクロスリージョンアカウントレベルのサブスクリプション

クロスアカウントサブスクリプションを作成するときに、単一のアカウントまたは組織を送信者として指定できます。組織を指定した場合、この手順により組織内のすべてのアカウントがレシーバーアカウントにログを送信できるようになります。

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。

- ログデータ送信者 — 受信者から送信先情報を取得し、ログイベントを指定された送信先に送信する準備ができていることを CloudWatch Logs に通知します。このセクションの残りの手順では、ログデータ送信者は架空の AWS アカウント番号 111111111111 で表示されます。

  1 つの組織で複数のアカウントを 1 つの受信者アカウントにログを送信する場合は、組織内のすべてのアカウントに対して受信者アカウントにログを送信するアクセス許可を付与するポリシーを作成することができます。引き続き、送信者アカウントごとに個別のサブスクリプションフィルターを設定する必要があります。
- ログデータ受信者 — Kinesis Data Streams ストリームをカプセル化する送信先を設定し、受信者がログデータを受信したいことを CloudWatch Logs に通知します。この後、受信者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、ログデータ受信者は架空の AWS アカウント番号 999999999999 で表示されます。

クロスアカウントユーザーからログイベントの受信を開始するには、まずログデータ受信者が CloudWatch ログ送信先を作成します。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リソースの Amazon AWS リソースネーム (ARN)。

ロールの ARN

選択したストリームにデータを配置するために必要なアクセス許可を CloudWatch ログに付与する 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) 。

> **Note**
>
> ロググループと送信先は同じ AWS リージョンにある必要があります。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。次のセクションの例では、リージョン固有のリソースはすべて米国東部 (バージニア北部) で作成されます。

トピック

新しいクロスアカウントサブスクリプションの設定

次のセクションの手順に従って、新しいクロスアカウントログサブスクリプションを設定します。

トピック

ステップ 1: 送信先を作成する

> **Important**
>
> 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、ログデータ受信者アカウントの AWS アカウント ID は 999999999999 で、ログデータ送信者 AWS アカウント ID は 111111111111 です。

この例では、 という名前の Kinesis Data Streams ストリームと RecipientStream、 CloudWatch Logs がデータを書き込めるようにするロールを使用して送信先を作成します。

送信先が作成されると、 CloudWatch Logs は受信者アカウントに代わって送信先にテストメッセージを送信します。サブスクリプションフィルターが後でアクティブになると、 CloudWatch Logs はソースアカウントに代わってログイベントを送信先に送信します。

送信先を作成するには

1. 受信者アカウントから、Kinesis データストリーム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. 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 プロパティを確認できます。さらに、StreamDescription.StreamARN 値は後で CloudWatch Logs に渡されるため、書き留めておきます。

   ```
   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": "340282366920938463463374607431768EXAMPLE",
             "StartingHashKey": "0"
   ```

```
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
 "49551135218688818456679503831981458784591352702 18EXAMPLE"
        }
      }
    ]
  }
}
```

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. ストリームにデータを配置するアクセス許可を 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 に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。

   このポリシーには、「混乱した代理」のセキュリティ上の問題を防止するための sourceAccountId が指定された aws:SourceArn グローバル条件コンテキストキーが含まれています。最初の呼び出しでソースアカウント ID が不明な場合は、送信元 ARN フィールドに送信先 ARN を指定することをお勧めします。後続の呼び出しでは、送信元 ARN を、最初の呼び出しで取得した実際の送信元 ARN に設定する必要があります。詳細については、「混乱した代理の防止」を参照してください。

```
{
    "Statement": {
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.amazonaws.com"
        },
        "Condition": {
            "StringLike": {
                "aws:SourceArn": [
                    "arn:aws:logs:region:sourceAccountId:*",
                    "arn:aws:logs:region:recipientAccountId:*"
                ]
            }
        },
        "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 値は後で CloudWatch ログに渡されるため、書き留めておきます。

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

{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Condition": {
                    "StringLike": {
                        "aws:SourceArn": [
                            "arn:aws:logs:region:sourceAccountId:*",
                            "arn:aws:logs:region:recipientAccountId:*"
                        ]
                    }
                },
                "Principal": {
                    "Service": "logs.amazonaws.com"
                }
            }
        },
        "RoleId": "AAOIIAH450GAB4HC5F431",
        "CreateDate": "2023-05-29T13:46:29.431Z",
        "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、Logs CloudWatch 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 ~/PermissionsForCWL.json でアクセス許可ポリシーを作成します。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
```

```
      "Resource": "arn:aws:kinesis:region:999999999999:stream/RecipientStream"
    }
  ]
}
```

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. ストリームがアクティブ状態になり、IAM ロールを作成したら、 CloudWatch ログ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DestinationArnされる を書き留めます。

```
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。このポリシーでは、ログを指定する必要があります。PutSubscriptionFilterアクション と は、送信先にアクセスするためのアクセス許可を送信者アカウントに付与します。

      このポリシーは、ログを送信する AWS アカウントにアクセス許可を付与します。ポリシーの中で対象のアカウントを 1 つだけ指定してもよいですが、送信者アカウントが組織のメンバーのものである場合は組織 ID を指定することもできます。このように、ポリシーを 1

つ作成するだけで、1 つの組織内の複数のアカウントが送信先アカウントにログを送信できるように設定できます。

テキストエディタを使用して ~/AccessPolicy.json という名前のファイルを作成し、以下のいずれかのポリシーステートメントを使用します。

この最初の例のポリシーでは、組織内で o-1234567890 という ID を持つすべてのアカウントが、受信者アカウントにログを送信することを許可します。

```
{
    "Version" : "2012-10-17",
    "Statement" : [
        {
            "Sid" : "",
            "Effect" : "Allow",
            "Principal" : "*",
            "Action" : ["logs:PutSubscriptionFilter","logs:PutAccountPolicy"],
            "Resource" :
 "arn:aws:logs:region:999999999999:destination:testDestination",
            "Condition": {
               "StringEquals" : {
                   "aws:PrincipalOrgID" : ["o-1234567890"]
                }
            }
        }
    ]
}
```

次の例では、ログデータの送信者アカウント (111111111111) がログデータの受信者アカウントにログを送信できるようにします。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "111111111111"
      },
      "Action" : ["logs:PutSubscriptionFilter","logs:PutAccountPolicy"],
```

```
      "Resource" :
 "arn:aws:logs:region:999999999999:destination:testDestination"
    }
  ]
}
```

c. 前のステップで作成したポリシーを送信先に添付します。

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

このアクセスポリシーにより、ID 111111111111 の AWS アカウントのユーザーは、ARN arn:aws:logs:*region* :999999999999:destination:testDestination を使用して送信先PutSubscriptionFilterに対して を呼び出すことができます。この送信先 PutSubscriptionFilter に対して他のユーザーが を呼び出そうとすると、拒否されます。

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

完了したら、クロスアカウントアクセス許可 AWS Organizations に を使用している場合は、「」のステップに従いますステップ 2: IAM ロールを作成する (組織を使用している場合のみ)。組織を使用せずに他のアカウントに直接アクセス許可を付与する場合は、そのステップを飛ばして「ステップ 3: アカウントレベルのサブスクリプションフィルターポリシーを作成する」に進みます。

ステップ 2: IAM ロールを作成する (組織を使用している場合のみ)

前のセクションで アカウント 111111111111 に直接アクセス許可を付与するのではなく、アカウント 111111111111 が属する組織にアクセス許可を付与するアクセスポリシーを使用することにより送信先を作成した場合は、このセクションのステップを実行します。それ以外の場合は、「ステップ 3: アカウントレベルのサブスクリプションフィルターポリシーを作成する」に進みます。

このセクションのステップでは、IAM ロールを作成します。これにより、送信者アカウントが受信者の送信先に対してサブスクリプションフィルターを作成するアクセス許可を持っているかどうかを引き受けて検証 CloudWatch できます。

このセクションの手順は、送信者アカウントで実行してください。ロールは送信者アカウントに存在する必要があり、このロールの ARN はサブスクリプションフィルターで指定します。この例では、送信者アカウントは 111111111111 です。

AWS Organizationsを使用してクロスアカウントのログサブスクリプションに必要な IAM ロールを作成する方法

1. 以下の信頼ポリシーを作成し、/TrustPolicyForCWLSubscriptionFilter.json という名前のテキストファイルに保存します。このポリシーの作成にはテキスト工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.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

2. このポリシーを使用する IAM ロールを作成します。下記のコマンドが返す Arn の値は後ほど必要になるため、書き留めておきます。この例では、作成するロールに CWLtoSubscriptionFilterRole という名前を付けます。

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

3. アクセス許可ポリシーを作成して、Logs CloudWatch がアカウントで実行できるアクションを定義します。

   a. まず、テキスト工ディタを使用して、~/PermissionsForCWLSubscriptionFilter.json という名前のファイルに以下のようなアクセス許可ポリシーを作成します。

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:region:111111111111:log-
group:LogGroupOnWhichSubscriptionFilterIsCreated:*"
        }
    ]
}
```

b. 次のコマンドを入力して、先ほど作成したアクセス許可ポリシーを、ステップ 2 で作成したロールに関連付けます。

```
aws iam put-role-policy
    --role-name CWLtoSubscriptionFilterRole
    --policy-name Permissions-Policy-For-CWL-Subscription-filter
    --policy-document file://~/PermissionsForCWLSubscriptionFilter.json
```

終了したら、「ステップ 3: アカウントレベルのサブスクリプションフィルターポリシーを作成する」に進みます。

ステップ 3: アカウントレベルのサブスクリプションフィルターポリシーを作成する

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。

> **Note**
>
> サブスクリプションフィルターのためのアクセス許可を組織全体に付与する際は、ステップ 2: IAM ロールを作成する (組織を使用している場合のみ) で作成した IAM ロールの ARN を使用する必要があります。

次の例では、アカウントレベルのサブスクリプションフィルターポリシーを送信側アカウントで作成します。フィルターは送信者アカウントに関連付けられている111111111111ため、フィルターと選択条件に一致するすべてのログイベントが、以前に作成した送信先に配信されます。その送信先は「」というストリームをカプセル化RecipientStreamします。

`selection-criteria` フィールドはオプションですが、サブスクリプションフィルターから無限ログの再帰を引き起こす可能性のあるロググループを除外するために重要です。この問題と除外するロググループの決定の詳細については、「」を参照してくださいログの再帰防止。現在、 NOT IN はでサポートされている唯一の演算子ですselection-criteria。

```
aws logs put-account-policy \
```

```
    --policy-name "CrossAccountStreamsExamplePolicy" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
    --policy-document
 '{"DestinationArn":"arn:aws:logs:region:999999999999:destination:testDestination",
 "FilterPattern": "", "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
 "LogGroupToExclude2"]' \
    --scope "ALL"
```

送信者アカウントのロググループと送信先は同じ AWS リージョンにある必要があります。ただし、送信先は、別のリージョンにある Kinesis Data Streams ストリームなどの AWS リソースを指すことができます。

ログイベントの送信を検証

アカウントレベルのサブスクリプションフィルターポリシーを作成すると、 CloudWatch Logs はフィルターパターンと選択条件に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にカプセル化された「」という名前のストリームに転送しますRecipientStream。送信先所有者は、aws kinesis get-shard-iterator コマンドを使用して Kinesis Data Streams 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 Data Streams レコードを取得することで、これが起こっ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**
>
> Kinesis Data Streams がデータを返す前に、`get-records`コマンドを数回再実行する必要がある場合があります。

一連の Kinesis データストリームレコードを含んでいるレスポンスが表示されます。Kinesis データストリームレコードのデータ属性は、gzip 形式で圧縮され、さらに base64 でエンコードされています。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 でデコードおよび解凍されたデータは、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/logs",
    "logStream": "111111111111_CloudTrail/logs_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
\"}"
```

```
        }
    ]
}
```

データ構造の主な要素は次のとおりです。

messageType

データメッセージは、"DATA_MESSAGE" 型を使用します。 CloudWatch Logs は、主に送信先に到達可能かどうかを確認するために、「CONTROL_MESSAGE」タイプの Kinesis Data Streams レコードを出力することがあります。

owner (オーナー)

発信元ログデータの 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。

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。

policyLevel

ポリシーが適用されたレベル。「ACCOUNT_LEVEL_POLICY`policyLevel`」は、アカウントレベルのサブスクリプションフィルターポリシーの です。

ランタイムの送信先のメンバーシップを変更

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を使用して、送信先で `put-destination-policy` コマンドを使用できます。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. 現在送信先の testDestination に関連付けられているポリシーを取得し、 を書き留めますAccessPolicy。

```
aws logs describe-destinations \
    --destination-name-prefix "testDestination"

{
 "Destinations": [
   {
     "DestinationName": "testDestination",
     "RoleArn": "arn:aws:iam::999999999999:role/CWLtoKinesisRole",
     "DestinationArn":
 "arn:aws:logs:region:999999999999:destination:testDestination",
     "TargetArn": "arn:aws:kinesis:region:999999999999:stream/RecipientStream",
     "AccessPolicy": "{\"Version\": \"2012-10-17\", \"Statement\":
 [{\"Sid\": \"\", \"Effect\": \"Allow\", \"Principal\": {\"AWS\":
 \"111111111111\"}, \"Action\": \"logs:PutSubscriptionFilter\", \"Resource\":
 \"arn:aws:logs:region:999999999999: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 ~/NewAccessPolicy.json ファイルに入れます。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "222222222222"
      },
      "Action" : ["logs:PutSubscriptionFilter","logs:PutAccountPolicy"],
      "Resource" : "arn:aws:logs:region:999999999999:destination:testDestination"
    }
  ]
}
```

3. PutDestinationPolicy を呼び出して、NewAccessPolicy.json ファイルで定義されたポリシーを送信先と関連付けます。

```
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 の所有者がサブスクリプションフィルターを作成すると、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。

既存のクロスアカウントサブスクリプションの更新

送信先アカウントが特定の送信者アカウントにのみアクセス許可を付与しているクロスアカウントのログサブスクリプションがあり、このサブスクリプションを更新して送信先アカウントが組織内のすべてのアカウントにアクセスできるようにする場合は、このセクションのステップを実施します。

トピック

- ステップ 1: サブスクリプションフィルターを更新する
- ステップ 2: 既存の送信先アクセスポリシーを更新する

ステップ 1: サブスクリプションフィルターを更新する

> **Note**
>
> この手順は、AWS サービスからのログ記録を有効にする に記載されているサービスによって作成されたログのクロスアカウントのサブスクリプションにのみ必要です。これらのロググループのいずれかで作成されたログを操作していない場合は、 ステップ 2: 既存の送信先アクセスポリシーを更新する にスキップできます。

場合によっては、送信先アカウントにログを送信する、すべての送信者アカウントのサブスクリプションフィルターを更新する必要があります。この更新では、IAM ロールが追加されます。これにより、送信者アカウントが受信者アカウントにログを送信するアクセス許可を持っていることを引き受け、検証 CloudWatch できます。

すべての送信者アカウントについてクロスアカウントサブスクリプションのアクセス許可に組織 ID を使用するように更新するには、このセクションのステップを実施します。

このセクションの例では、2 つのアカウント 111111111111 と 222222222222 は、アカウント 999999999999 にログを送信するために作成されたサブスクリプションフィルターをすでに持っています。既存のサブスクリプションフィルター値は次のとおりです。

```
## Existing Subscription Filter parameter values
{
    "DestinationArn": "arn:aws:logs:region:999999999999:destination:testDestination",
    "FilterPattern": "{$.userIdentity.type = Root}",
    "Distribution": "Random"
}
```

現在のサブスクリプションフィルターパラメータ値を見つける必要がある場合は、次のコマンドを入力します。

```
aws logs describe-account-policies \
--policy-type "SUBSCRIPTION_FILTER_POLICY" \
--policy-name "CrossAccountStreamsExamplePolicy"
```

サブスクリプションフィルターを更新して、クロスアカウントログの権限で組織 ID の使用をスタートする方法

1. 以下の信頼ポリシーを作成し、~/TrustPolicyForCWL.json という名前のテキストファイルに保存します。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.amazonaws.com" },
       "Action": "sts:AssumeRole"
     }
   }
   ```

2. このポリシーを使用する IAM ロールを作成します。下記のコマンドが返す Arn 値の Arn の値は後ほど必要になるため、書き留めておきます。この例では、作成するロールに CWLtoSubscriptionFilterRole という名前を付けます。

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

3. アクセス許可ポリシーを作成して、Logs CloudWatch がアカウントで実行できるアクションを定義します。

   a. まず、テキストエディタを使用して、*PermissionsForCWLSubscriptionFilter.json* という名前のファイルに以下のようなアクセス許可ポリシーを作成します。

   ```
   {
       "Statement": [
           {
               "Effect": "Allow",
               "Action": "logs:PutLogEvents",
               "Resource": "arn:aws:logs:region:111111111111:log-
   group:LogGroupOnWhichSubscriptionFilterIsCreated:*"
           }
       ]
   }
   ```

   b. 次のコマンドを入力して、先ほど作成したアクセス許可ポリシーを、ステップ 2 で作成したロールに関連付けます。

   ```
   aws iam put-role-policy
       --role-name CWLtoSubscriptionFilterRole
       --policy-name Permissions-Policy-For-CWL-Subscription-filter
       --policy-document file://~/PermissionsForCWLSubscriptionFilter.json
   ```

4. 次のコマンドを入力して、サブスクリプションフィルターポリシーを更新します。

```
aws logs put-account-policy \
    --policy-name "CrossAccountStreamsExamplePolicy" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
    --policy-document
 '{"DestinationArn":"arn:aws:logs:region:999999999999:destination:testDestination",
 "FilterPattern": "{$.userIdentity.type = Root}", "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
 "LogGroupToExclude2"]' \
    --scope "ALL"
```

ステップ 2: 既存の送信先アクセスポリシーを更新する

すべての送信者アカウントのサブスクリプションフィルターを更新した後、受信者アカウントの送信先アクセスポリシーを更新できます。

以下の例では、受信者アカウントは 999999999999、送信先は testDestination となっています。

この更新により、ID o-1234567890 を持つ組織に属するすべてのアカウントが、受信者アカウントにログを送信できるようになりました。サブスクリプションフィルターが作成されたアカウントのみが、実際に受信者アカウントにログを送信します。

受信者アカウントの送信先アクセスポリシーを更新して、権限の組織 ID の使用をスタートする方法

1. 受信者アカウントで、テキストエディタを使用して、以下の内容の ~/AccessPolicy.json ファイルを作成します。

```
{
"Version" : "2012-10-17",
    "Statement" : [
        {
"Sid" : "",
            "Effect" : "Allow",
            "Principal" : "*",
            "Action" : ["logs:PutSubscriptionFilter","logs:PutAccountPolicy"],
            "Resource" :
 "arn:aws:logs:region:999999999999:destination:testDestination",
            "Condition": {
"StringEquals" : {
"aws:PrincipalOrgID" : ["o-1234567890"]
                }
            }
        }
    ]
}
```

2. 次のコマンドを入力して、先ほど作成したポリシーを既存の送信先にアタッチします。特定の AWS アカウント ID をリストにしたアクセスポリシーではなく、組織 ID を含むアクセスポリシーを使用するように送信先を更新するには、force パラメータを指定します。

> **Warning**
>
> にリストされている AWS のサービスによって送信されたログを使用している場合はAWS サービスからのログ記録を有効にする、このステップを実行する前に、 で説明されているように、すべての送信者アカウントのサブスクリプションフィルターを更新しておく必要がありますステップ 1: サブスクリプションフィルターを更新する。

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

Firehose を使用したクロスアカウントクロスリージョンアカウントレベルのサブスクリプション

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。

- ログデータ送信者 — 受信者から送信先情報を取得し、指定した送信先にログイベントを送信する準備ができていることを CloudWatch Logs に通知します。このセクションの残りの手順では、ログデータ送信者は架空の AWS アカウント番号 111111111111 で表示されます。
- ログデータ受信者 — Kinesis Data Streams ストリームをカプセル化する送信先を設定し、受信者がログデータを受信したいことを CloudWatch Logs に通知します。この後、受信者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、ログデータ受信者は架空の AWS アカウント番号 222222222222 で表示されます。

このセクションの例では、Amazon S3 ストレージを備えた Firehose 配信ストリームを使用します。さまざまな設定で Firehose 配信ストリームを設定することもできます。詳細については、「Firehose 配信ストリームの作成」を参照してください。

> **Note**
>
> ロググループと送信先は同じ AWS リージョンにある必要があります。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。

> **Note**
>
> 同じアカウントとクロスリージョン配信ストリームの Firehose サブスクリプションフィルターがサポートされています。

トピック

ステップ 1: Firehose 配信ストリームを作成する

> **Important**
>
> 次の手順を完了する前に、Firehose が Amazon S3 バケットにアクセスできるように、アクセスポリシーを使用する必要があります。詳細については、「Amazon Data Firehose デベロッパーガイド」の「アクセスの制御」を参照してください。
> このセクションのすべての手順 (ステップ 1) は、ログデータの受取人アカウントで行われます。
> 次のサンプルコマンドでは、米国東部 (バージニア北部) が使用されています。このリージョンを、デプロイに適したリージョンに置き換えます。

送信先として使用する Firehose 配信ストリームを作成するには

1. Amazon S3 バケットの作成

```
aws s3api create-bucket --bucket firehose-test-bucket1 --create-bucket-
configuration LocationConstraint=us-east-1
```

2. Firehose にバケットにデータを配置するアクセス許可を付与する IAM ロールを作成します。

   a. まず、テキストエディタを使用して、ファイル ~/TrustPolicyForFirehose.json で信頼ポリシーを作成します。

```
{ "Statement": { "Effect": "Allow", "Principal": { "Service":
 "firehose.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition":
 { "StringEquals": { "sts:ExternalId":"222222222222" } } } }
```

b. 作成したばかりの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。

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

c. このコマンドの出力は、次のようになります。ロール名とロール ARN を書き留めます。

```
{
    "Role": {
        "Path": "/",
        "RoleName": "FirehosetoS3Role",
        "RoleId": "AROAR3BXASEKW7K635M53",
        "Arn": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
        "CreateDate": "2021-02-02T07:53:10+00:00",
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Effect": "Allow",
                "Principal": {
                    "Service": "firehose.amazonaws.com"
                },
                "Action": "sts:AssumeRole",
                "Condition": {
                    "StringEquals": {
                        "sts:ExternalId": "222222222222"
                    }
                }
            }
        }
    }
}
```

3. アクセス許可ポリシーを作成して、Firehose がアカウントで実行できるアクションを定義します。

a. まず、テキストエディタを使用して、~/PermissionsForFirehose.json という名前のファイルに以下のようなアクセス許可ポリシーを作成します。ユースケースによっては、このファイルにさらにアクセス権限を追加する必要がある場合があります。

```
{
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:ListBucket"
        ],
        "Resource": [
            "arn:aws:s3:::firehose-test-bucket1",
            "arn:aws:s3:::firehose-test-bucket1/*"
        ]
    }]
}
```

b. 次のコマンドを入力して、先ほど作成したアクセス権限ポリシーを IAM ロールに関連付けます。

```
aws iam put-role-policy --role-name FirehosetoS3Role --policy-name
 Permissions-Policy-For-Firehose-To-S3 --policy-document file://~/
PermissionsForFirehose.json
```

4. 次のコマンドを入力して、Firehose 配信ストリームを作成します。*my-role-arn* と をデプロイに適した値*my-bucket-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::222222222222:role/FirehosetoS3Role", "BucketARN":
 "arn:aws:s3:::firehose-test-bucket1"}'
```

出力は次の例に類似したものになります:

```
{
    "DeliveryStreamARN": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream"
}
```

ステップ 2: 送信先を作成する

> **Important**
>
> 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。

送信先が作成されると、 CloudWatch Logs は受信者アカウントに代わって送信先にテストメッセージを送信します。サブスクリプションフィルターが後でアクティブになると、 CloudWatch Logs はソースアカウントに代わってログイベントを送信先に送信します。

送信先を作成するには

1. で作成した Firehose ストリームがアクティブ[ステップ 1: Firehose 配信ストリームを作成する](#)になるまで待ちます。次のコマンドを使用して、StreamDescription.StreamStatus プロパティを確認できます。

   ```
   aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
   ```

   さらに、DeliveryStreamDescription.DeliveryStreamARN 値は後のステップで使用する必要があるため、書き留めておきます。このコマンドの出力例:

   ```
   {
       "DeliveryStreamDescription": {
           "DeliveryStreamName": "my-delivery-stream",
           "DeliveryStreamARN": "arn:aws:firehose:us-
   east-1:222222222222:deliverystream/my-delivery-stream",
           "DeliveryStreamStatus": "ACTIVE",
           "DeliveryStreamEncryptionConfiguration": {
               "Status": "DISABLED"
           },
           "DeliveryStreamType": "DirectPut",
           "VersionId": "1",
           "CreateTimestamp": "2021-02-01T23:59:15.567000-08:00",
           "Destinations": [
               {
                   "DestinationId": "destinationId-000000000001",
                   "S3DestinationDescription": {
                       "RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
                       "BucketARN": "arn:aws:s3:::firehose-test-bucket1",
                       "BufferingHints": {
   ```

```
                        "SizeInMBs": 5,
                        "IntervalInSeconds": 300
                    },
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "CloudWatchLoggingOptions": {
                        "Enabled": false
                    }
                },
                "ExtendedS3DestinationDescription": {
                    "RoleARN": "arn:aws:iam::222222222222:role/FirehosetoS3Role",
                    "BucketARN": "arn:aws:s3:::firehose-test-bucket1",
                    "BufferingHints": {
                        "SizeInMBs": 5,
                        "IntervalInSeconds": 300
                    },
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "CloudWatchLoggingOptions": {
                        "Enabled": false
                    },
                    "S3BackupMode": "Disabled"
                }
            }
        ],
        "HasMoreDestinations": false
    }
}
```

配信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。

2. 配信ストリームがアクティブになったら、Firehose ストリームにデータを置くアクセス許可を 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 に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。 CloudWatch Logs エンドポイントの詳細については、「Amazon CloudWatch Logs エンドポイントとクォータ」を参照してください。

   このポリシーには、「混乱した代理」のセキュリティ上の問題を防止するための sourceAccountId が指定された aws:SourceArn グローバル条件コンテキストキーが含まれ

ています。最初の呼び出しでソースアカウント ID が不明な場合は、送信元 ARN フィールドに送信先 ARN を指定することをお勧めします。後続の呼び出しでは、送信元 ARN を、最初の呼び出しで取得した実際の送信元 ARN に設定する必要があります。詳細については、「混乱した代理の防止」を参照してください。

```
{
    "Statement": {
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringLike": {
                "aws:SourceArn": [
                    "arn:aws:logs:region:sourceAccountId:*",
                    "arn:aws:logs:region:recipientAccountId:*"
                ]
            }
        }
     }
}
```

3. aws iam create-role コマンドを使用して、作成した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。

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

以下は出力例です。後のステップで使用する必要があるため、Role.Arn の戻り値を書き留めます。

```
{
    "Role": {
        "Path": "/",
        "RoleName": "CWLtoKinesisFirehoseRole",
        "RoleId": "AROAR3BXASEKYJYWF243H",
        "Arn": "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole",
        "CreateDate": "2023-02-02T08:10:43+00:00",
        "AssumeRolePolicyDocument": {
```

```
            "Statement": {
                "Effect": "Allow",
                "Principal": {
                    "Service": "logs.amazonaws.com"
                },
                "Action": "sts:AssumeRole",
                "Condition": {
                    "StringLike": {
                        "aws:SourceArn": [
                            "arn:aws:logs:region:sourceAccountId:*",
                            "arn:aws:logs:region:recipientAccountId:*"
                        ]
                    }
                }
            }
        }
    }
```

4. アクセス許可ポリシーを作成して、Logs CloudWatch 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 ~/PermissionsForCWL.json でアクセス許可ポリシーを作成します。

```
{
    "Statement":[
      {
        "Effect":"Allow",
        "Action":["firehose:*"],
        "Resource":["arn:aws:firehose:region:222222222222:*"]
      }
    ]
}
```

5. 次のコマンドを入力して、アクセス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
```

6. Firehose 配信ストリームがアクティブ状態になり、IAM ロールを作成したら、 CloudWatch ログ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。後のステップでこれを destination.arn として使用するため、ペイロードで返される新しい宛先の ARN を書き留めます。

```
aws logs put-destination \

    --destination-name "testFirehoseDestination" \
    --target-arn "arn:aws:firehose:us-east-1:222222222222:deliverystream/my-
delivery-stream" \
    --role-arn "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole"

{
    "destination": {
        "destinationName": "testFirehoseDestination",
        "targetArn": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream",
        "roleArn": "arn:aws:iam::222222222222:role/CWLtoKinesisFirehoseRole",
        "arn": "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"}
}
```

b. 前のステップが完了したら、ログデータ受取人アカウント (222222222222) で、アクセスポリシーを送信先に関連付けます。このポリシーにより、ログデータの送信者アカウント (111111111111) に対し、ログデータの受信者アカウント (222222222222) にある送信先にアクセスすることを許可します。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","logs:PutAccountPolicy"],
      "Resource" : "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
}
```

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 および logs:PutAccountPolicyアクションを指定する必要があります。クロスアカウントユーザーは、 PutSubscriptionFilter および PutAccountPolicyアクションを使用してログイベントを送信先に送信します。

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

ステップ 3: アカウントレベルのサブスクリプションフィルターポリシーを作成する

送信側のアカウント (この例では 111111111111) に切り替えます。次に、送信側アカウントでアカウントレベルのサブスクリプションフィルターポリシーを作成します。この例では、フィルターにより、2 つのロググループを除くERRORすべてのロググループの文字列を含むすべてのログイベントが、以前に作成した送信先に配信されます。

```
aws logs put-account-policy \
    --policy-name "CrossAccountFirehoseExamplePolicy" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
    --policy-document '{"DestinationArn":"arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination", "FilterPattern":
 "{$.userIdentity.type = AssumedRole}", "Distribution": "Random"}' \
    --selection-criteria 'LogGroupName NOT IN ["LogGroupToExclude1",
 "LogGroupToExclude2"]' \
    --scope "ALL"
```

送信側アカウントのロググループと送信先は同じ AWS リージョンにある必要があります。ただし、送信先は、別のリージョンにある Firehose ストリームなどの AWS リソースを指すことができます。

ログイベントの送信の検証

サブスクリプションフィルターを作成すると、 CloudWatch Logs はフィルターパターンと選択条件に一致するすべての受信ログイベントを Firehose 配信ストリームに転送します。Firehose 配信ストリームに設定されている時間バッファ間隔に基づいて、データが Amazon S3 バケットに表示され始めます。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。バケットを確認するには、次のコマンドを入力します。

```
aws s3api list-objects --bucket 'firehose-test-bucket1'
```

そのコマンドの出力は、次のようになります。

```
{
    "Contents": [
        {
            "Key": "2021/02/02/08/my-delivery-
stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba",
            "LastModified": "2023-02-02T09:00:26+00:00",
            "ETag": "\"EXAMPLEa817fb88fc770b81c8f990d\"",
            "Size": 198,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "firehose+2test",
                "ID": "EXAMPLE27fd05889c665d2636218451970ef79400e3d2aecca3adb1930042e0"
            }
        }
    ]
}
```

その後、次のコマンドを入力して、バケットから特定のオブジェクトを取得できます。key の値を、前のコマンドで検索した値に置き換えます。

```
aws s3api get-object --bucket 'firehose-test-bucket1' --key '2021/02/02/08/my-delivery-
stream-1-2021-02-02-08-55-24-5e6dc317-071b-45ba-a9d3-4805ba39c2ba' testfile.gz
```

Simple Storage Service (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コマンドを使用して調べることができます。

Linux:

```
zcat testfile.gz
```

macOS:

```
zcat <testfile.gz
```

実行時の送信先のメンバーシップの変更

所有している送信先からログ送信者を追加または削除しなければならない状況が発生することがあります。新しいアクセスポリシーを使用して、送信先の PutDestinationPolicyおよびPutAccountPolicy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 333333333333 が有効になります。

1. 現在送信先の testDestination に関連付けられているポリシーを取得し、 を書き留めますAccessPolicy。

```
aws logs describe-destinations \
    --destination-name-prefix "testFirehoseDestination"
```

返されるデータは次のようになります。

```
{
    "destinations": [
        {
            "destinationName": "testFirehoseDestination",
            "targetArn": "arn:aws:firehose:us-east-1:222222222222:deliverystream/
my-delivery-stream",
            "roleArn": "arn:aws:iam:: 222222222222:role/CWLtoKinesisFirehoseRole",
            "accessPolicy": "{\n  \"Version\" : \"2012-10-17\",\n  \"Statement
\" : [\n    {\n      \"Sid\" : \"\",\n      \"Effect\" : \"Allow\",\n
 \"Principal\" : {\n        \"AWS\" : \"111111111111 \"\n      },\n      \"Action
\" : \"logs:PutSubscriptionFilter\",\n      \"Resource\" : \"arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination\"\n    }\n  ]\n}\n\n",
            "arn": "arn:aws:logs:us-east-1:
 222222222222:destination:testFirehoseDestination",
            "creationTime": 1612256124430
        }
    ]
}
```

2. アカウント 111111111111 が停止したこととアカウント 333333333333 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 ~/NewAccessPolicy.json ファイルに入れます。

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

```
    {
      "Sid" : "",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "333333333333 "
      },
      "Action" : ["logs:PutSubscriptionFilter","logs:PutAccountPolicy"],
      "Resource" : "arn:aws:logs:us-
east-1:222222222222:destination:testFirehoseDestination"
    }
  ]
}
```

3. 次のコマンドを使用して、NewAccessPolicy.json ファイルで定義されたポリシーを送信先と関連付けます。

```
aws logs put-destination-policy \
    --destination-name "testFirehoseDestination" \

    --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 333333333333 の所有者がサブスクリプションフィルターを作成すると、すぐに 333333333333 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。

混乱した代理の防止

混乱した代理問題は、アクションを実行するためのアクセス許可を持たないエンティティが、より特権のあるエンティティにアクションの実行を強制できてしまう場合に生じる、セキュリティ上の問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。これを防ぐために、 は、アカウント内のリソースへのアクセスが許可されているサービスプリンシパルを持つすべてのサービスのデータを保護するのに役立つツール AWS を提供します。

リソースポリシーで aws:SourceArn、aws:SourceAccount、、および aws:SourceOrgPaths グローバル条件コンテキストキーを使用してaws:SourceOrgID、 が別のサービスに付与するアク

セス許可をリソースに制限することをお勧めします。1 つのリソースだけをクロスサービスのアクセスに関連付ける場合は、aws:SourceArn を使用します。アカウント内の任意のリソースをクロスサービスの使用に関連付ける場合は、aws:SourceAccount を使用します。組織内の任意のアカウントの任意のリソースをクロスサービスの使用に関連付ける場合は、aws:SourceOrgID を使用します。 AWS Organizations パス内の任意のアカウントのリソースをクロスサービスの使用に関連付ける場合は、aws:SourceOrgPaths を使用します。パスの使用と理解の詳細については、AWS Organizations 「エンティティパスの理解」を参照してください。

混乱した代理問題から保護するための最も効果的な方法は、リソースの完全な ARN を指定して、aws:SourceArn グローバル条件コンテキストキーを使用することです。リソースの完全な ARN が不明な場合や、複数のリソースを指定する場合には、グローバルコンテキスト条件キー aws:SourceArn で、ARN の未知部分を示すためにワイルドカード文字 (*) を使用します。例えば、arn:aws:*servicename*:*:*123456789012*:* です。

aws:SourceArn の値に Amazon S3 バケット ARN などのアカウント ID が含まれていない場合は、両方の aws:SourceAccount と aws:SourceArn を使用して、アクセス許可を制限する必要があります。

混乱した代理問題から保護するために、リソースベースポリシー内のリソースの組織 ID または組織パスを指定しながら、aws:SourceOrgID または aws:SourceOrgPaths のグローバル条件コンテキストキーを使用してください。aws:SourceOrgID または aws:SourceOrgPaths キーを含むポリシーには正しいアカウントが自動的に組み込まれるため、組織のアカウントを追加、削除、移動する際には手動で更新する必要はありません。

で Kinesis Data Streams と Firehose にデータを書き込むための CloudWatch ログへのアクセスを許可するためのポリシーステップ 1: 送信先を作成すると、混乱した代理問題を防ぐために aws:SourceArn global 条件コンテキストキーを使用する方法ステップ 2: 送信先を作成するを示すポリシー。

ログの再帰防止

サブスクリプションフィルターで無限のログ再帰が発生するリスクがあり、防止しないと、CloudWatch ログと送信先の両方で取り込み料金が大幅に増加する可能性があります。これは、サブスクリプション配信ワークフローの結果としてログイベントを受信するロググループにサブスクリプションフィルターが関連付けられている場合に発生する可能性があります。ロググループに取り込まれたログは送信先に配信され、ロググループがより多くのログを取り込んで送信先に転送され、再帰ループが作成されます。

例えば、送信先が Firehose であるサブスクリプションフィルターを考えてみましょう。これにより、ログイベントが Amazon S3 に配信されます。さらに、Amazon S3 に配信される新しいイベントを処理し、いくつかのログ自体を生成する Lambda 関数もあります。サブスクリプションフィルターが Lambda 関数のロググループに適用されると、関数によって生成されたログイベントが送信先で Firehose と Amazon S3 に転送され、その後、関数が再度呼び出され、より多くのログが生成されて Firehose と Amazon S3 に転送され、関数が再度呼び出されます。これは無限ループで発生し、ログの取り込み、Firehose、および Amazon S3 の料金が予想外に増加します。

Lambda 関数が CloudWatch Logs でフローログが有効になっている VPC にアタッチされている場合、VPC のロググループもログの再帰を引き起こす可能性があります。

サブスクリプション配信ワークフローの一部であるロググループには、サブスクリプションフィルターを適用しないことをお勧めします。アカウントレベルのサブスクリプションフィルターの場合は、 `PutAccountPolicy` API の `selectionCriteria`パラメータを使用して、これらのロググループをポリシーから除外します。

ロググループを除外する場合は、ログを生成する以下の AWS サービスを検討してください。これはサブスクリプション配信ワークフローの一部である可能性があります。

- Fargate を使用した Amazon EC2
- Lambda
- AWS Step Functions
- Logs で有効になっている Amazon VPC CloudWatch フローログ

> **Note**
>
> Lambda 送信先のロググループによって生成されたログイベントは、アカウントレベルのサブスクリプションフィルターポリシーの Lambda 関数に転送されません。この場合、アカウントサブスクリプションポリシーには、 を使用する送信先 Lambda 関数のロググループを除外`selectionCriteria`する必要はありません。

メトリクスフィルター、サブスクリプションフィルター、フィルターログイベント、およびライブテールのフィルターパターン構文

> **Note**
>
> Amazon CloudWatch Logs Insights クエリ言語でロググループをクエリする方法については、「」を参照してくださいCloudWatch Logs Insights クエリ構文。

CloudWatch Logs を使用すると、メトリクスフィルターを使用してログデータを実用的なメトリクスに変換したり、サブスクリプションフィルターを使用してログイベントを他の AWS サービスにルーティングしたり、ログイベントをフィルタリングしてログイベントを検索したり、Live Tail を使用してログを取り込み時にインタラクティブにリアルタイムで表示したりできます。

フィルターパターンは、メトリクスフィルター、サブスクリプションフィルター、フィルターログイベント、ライブテールがログイベントの語句を照合するために使用する構文を構成します。語句には、単語、正確なフレーズ、または数値を指定できます。正規表現 (regex) は、スタンドアロンのフィルターパターンの作成に使用するか、JSON やスペース区切りのフィルターパターンに組み込むことができます。

照合する語句を使用してフィルターパターンを作成します。フィルターパターンは、定義する語句を含むログイベントのみを返します。 CloudWatch コンソールでフィルターパターンをテストできます。

トピック

- サポートされている正規表現 (regex) 構文
- フィルターパターンを使用した正規表現 (regex) の語句の一致
- フィルターパターンを使用した非構造化ログイベントの語句の一致
- フィルターパターンを使用した JSON ログイベントの語句の一致
- フィルターパターンを使用したスペース区切りのログイベントでの語句の一致

サポートされている正規表現 (regex) 構文

サポートされている regex 構文

regex を使用してログデータを検索とフィルタリングする際は、その式を % で囲む必要があります。

regex を使ったフィルターパターンには、次のものしか含めることができません

- 英数字 - 英数字とは、文字 (A〜Z または a〜z) または数字 (0 ~ 9) を指します。
- サポートされている記号文字 - これには、「_」、「#」、「=」、「@」、「/」、「;」、「,」、「-」が含まれます。たとえば、「!」はサポートされていないため、%something!% は拒否されます。
- サポートされている演算子 - これには、「^」、「$」、「?」、「[」、「]」、「{」、「}」、「|」、「\」、「*」、「+」、「.」が含まれます。

(と) 演算子はサポートされていません。括弧を使用してサブパターンを定義することはできません。

マルチバイト文字はサポートされていません。

> **Note**
>
> クォータ
> メトリックスフィルターまたはサブスクリプションフィルターを作成するとき、ロググループごとに regex を含むフィルターパターンが最大 5 つあります。
> メトリックスフィルターとサブスクリプションフィルターの区切りまたは JSON フィルターパターンを作成するとき、またはログイベントまたはライブテールをフィルタリングするとき、フィルターパターンごとに 2 つの regex の制限があります。

[サポートされている演算子の使い方]

- ^:文字列の先頭の一致。たとえば、%^[hc]at% は「hat」と「cat」を一致とみなしますが、文字列の先頭でのみ適用されます。
- $: 文字列の末尾の一致。たとえば、%[hc]at$% は「hat」と「cat」を一致とみなしますが、文字列の末尾でのみ適用されます。
- ?: 前の期間の 0 個以上のインスタンスに一致します。たとえば、%colou?r% は「color」と「colour」を一致とみなします。

- []: 文字クラスを定義します。括弧内の文字リストまたは文字範囲との一致。たとえば、%[abc]% は「a」、「b」、「c」を一致とみなします。%[a-z]% は「a」から「z」までのすべての小文字を一致とみなします。%[abcx-z]% は「a」、「b」、「c」、「x」、「y」、「z」を一致とみなします。
- {m, n}: m 以上の前の語句と一致し、n 回を超えることはありません。たとえば、%a{3,5}% は「aaa」、「aaaa」、「aaaaa」 のみを一致とみなします。

> **Note**
>
> 最小値または最大値を定義しない場合、m と n のいずれかを省略できます。

- |: 垂直バーのどちら側の語句と一致するブール値「Or」。たとえば、%gra|ey% は「gray」または「grey」を一致とみなします。

> **Note**
>
> 語句とは、?、*、+、{n,m} のいずれかの演算子を使用する単一文字または繰り返される文字クラスです。

- \: 演算子の特殊な意味ではなく、文字通りの意味を使用できるようにする エスケープ文字。たとえば、「[a]」、「[b]」、「[7]」、「[@]」、「[]]」、「[ ]」などのように、括弧が エスケープされるため、%\[.\]%は「[」と「]」で囲まれたすべての 1 文字を一致とみなします。

> **Note**
>
> %10\.10\.0\.1% は、IP アドレス 10.10.0.1 を一致とみなす regex を作成する正しい方法です。

- *: 前の期間の 0 個以上のインスタンスに一致します。たとえば、 %ab*c% は「ac」、「abc」、「abbbc」と一致できます。%ab[0-9]*% は「ab」、「ab0」、「ab129」を一致とみなします。
- +: 前述の期間の 1 つ以上のインスタンスに一致します。たとえば、%ab+c%は「abc」、「abbc」、「abbbc」を一致とみなしますが、「ac」を一致とみなしません。
- .: すべての 1 文字と一致します。たとえば、「hat」、「cat」、「bat」、「4at」、「#at」、「 at」 (先頭にスペース)を含め、%.at% は「at」で終わるすべての 3 文字の文字列を一致とみなします。

> **Note**
>
> IP アドレスと一致させる regex を作成するとき、. 演算子からエスケープすることが重要です。たとえば、%10.10.0.1% は「10010,051」を一致とみなしますが、これは表現の本来の用途とは異なる場合があります。

- \d、\D : 数字または数字以外の文字を一致とみなします。たとえば、%\d% は %[0-9]% と同等であり、%\D% は %[^0-9]% と同等です。

> **Note**
>
> 大文字の演算子は、対応する小文字の逆を表します。

- \s、\S: 空白文字または非空白文字を一致とみなします。

> **Note**
>
> 大文字の演算子は、対応する小文字の逆を表します。空白文字にはタブ (\t)、スペース ( )、改行 (\n)文字が含まれます。

- \w、\W: 英数字または非英数字と一致します。たとえば、%\w% は %[a-zA-Z_0-9]% と同等であり、%\W% は %[^a-zA-Z_0-9]% と同等です。

> **Note**
>
> 大文字の演算子は、対応する小文字の逆を表します。

- \xhh: 2 桁の 16 進文字の ASCII マッピングと一致します。\xは、次の文字が ASCII の 16 進値を表すことを示すエスケープ シーケンスです。hhは、ASCII 表の文字を指す 2 つの 16 進数字(0 ~ 9 と A ~ F)を指定します。

> **Note**
>
> \xhh を使用してフィルターパターンでサポートされていない記号文字を一致とみなすことができます。たとえば、%\x3A% は : を一致とみなし、%\x28% は (を一致とみなします。

フィルターパターンを使用した正規表現 (regex) の語句の一致

regex を使用した語句の一致

% (regex パターン前後のパーセント記号)で囲まれた regex パターンを使用し、ログイベントの語句を一致とみなすことができます。次のコードスニペットでは、[許可された]キーワードで構成されているすべてのログイベントを返すフィルターパターンの例が示されています。

サポートされている正規表現のリストについては、「サポートされている正規表現」を参照してください。

```
%AUTHORIZED%
```

このフィルターパターンは、次のようなログイベントメッセージを返します。

- `[ERROR 401] UNAUTHORIZED REQUEST`
- `[SUCCESS 200] AUTHORIZED REQUEST`

フィルターパターンを使用した非構造化ログイベントの語句の一致

非構造化ログイベントの語句の一致

次の例には、フィルターパターンを使用して非構造化ログイベントで語句をマッチさせる方法について示すコードスニペットが含まれています。

> **Note**
>
> フィルターパターンでは大文字と小文字が区別されます。英数字以外の文字を含む正確なフレーズと語句を、二重引用符 ([""]) で囲みます。

Example: Match a single term

次のコードスニペットは、メッセージに [ERROR] という単語が含まれるすべてのログイベントを返す単一の語句のフィルターパターンの例を示しています。

```
ERROR
```

このフィルターパターンは、次のようなログイベントメッセージを一致とみなします。

- [ERROR 400] BAD REQUEST
- [ERROR 401] UNAUTHORIZED REQUEST
- [ERROR 419] MISSING ARGUMENTS
- [ERROR 420] INVALID ARGUMENTS

Example: Match multiple terms

次のコードスニペットは、メッセージに ERROR と ARGUMENTS という単語が含まれるすべてのログイベントを返す複数の語句のフィルターパターンの例を示しています。

```
ERROR ARGUMENTS
```

フィルターは、次のようなログイベントメッセージを返します。

- [ERROR 419] MISSING ARGUMENTS
- [ERROR 420] INVALID ARGUMENTS

次のログイベントメッセージにはフィルターパターンで指定された語句が両方とも含まれないため、このフィルターパターンでは返されません。

- [ERROR 400] BAD REQUEST
- [ERROR 401] UNAUTHORIZED REQUEST

Example: Match optional terms

パターン一致を使用し、オプション語句を含むログイベントを返すフィルターパターンを作成できます。照合する語句の前に疑問符 (「?」) を配置します。次のコードスニペットは、メッセージに ERROR または ARGUMENTS という単語が含まれるすべてのログイベントを返すフィルターパターンの例を示しています。

```
```

```
?ERROR ?ARGUMENTS
```

このフィルターパターンは、次のようなログイベントメッセージを一致とみなします。

- [ERROR 400] BAD REQUEST
- [ERROR 401] UNAUTHORIZED REQUEST
- [ERROR 419] MISSING ARGUMENTS
- [ERROR 420] INVALID ARGUMENTS

> **Note**
>
> 疑問符 (「?」) を他のフィルターパターン (「含む」や「除外」の条件など) 組み合わせることはできません 。「?」を他のフィルターパターンと組み合わせると、その疑問符 (「?」) は無視されます。
> 例えば、次のフィルターパターンは REQUEST という単語を含むすべてのイベントにマッチしますが、疑問符 (「?」) は無視され、何ら影響力を持ちません。
>
> ```
> ?ERROR ?ARGUMENTS REQUEST
> ```
>
> ログイベントのマッチ
>
> - [INFO] REQUEST FAILED
> - [WARN] UNAUTHORIZED REQUEST
> - [ERROR] 400 BAD REQUEST

Example: Match exact phrases

次のコードスニペットは、メッセージに INTERNAL SERVER ERROR という正確なフレーズが含まれるログイベントを返すフィルターパターンの例を示しています。

```
"INTERNAL SERVER ERROR"
```

このフィルターパターンは、次のログイベントメッセージを返します

- [ERROR 500] INTERNAL SERVER ERROR

Example: Include and exclude terms

メッセージにいくつかの語句が含まれるログイベントを返し、他の語句が除外されるフィルターパターンを作成できます。除外する語句の前にマイナス記号 ([「-」]) を配置します。次のコードスニペットは、メッセージに ERROR が含まれるログイベントを返し、ARGUMENTS という語句が除外されるフィルターパターンの例を示しています。

```
ERROR -ARGUMENTS
```

このフィルターパターンは、次のようなログイベントメッセージを返します。

- [ERROR 400] BAD REQUEST
- [ERROR 401] UNAUTHORIZED REQUEST

次のログイベントメッセージには [引数] という単語が含まれているため、このフィルターパターンでは返されません。

- [ERROR 419] MISSING ARGUMENTS
- [ERROR 420] INVALID ARGUMENTS

Example: Match everything

二重引用符で囲むことで、ログイベント内の完全一致を照合することができます。次のコードスニペットは、すべてのログイベントを返すフィルターパターンの例を示しています。

```
" "
```

フィルターパターンを使用した JSON ログイベントの語句の一致

JSON ログイベントのフィルターパターンの式

次の内容では、文字列と数値を含む JSON 語句と一致するフィルターパターンの構文を式する方法について説明します。

Writing filter patterns that match strings

JSON ログイベントで文字列とマッチさせるフィルターパターンを作成できます。次のコードスニペットには、文字列ベースのフィルターパターンの構文例が示されています。

```
{ PropertySelector EqualityOperator String }
```

フィルターパターンを中括弧(「{}」)で囲みます。文字列ベースのフィルターパターンには、次の部分が含まれている必要があります。

- [Property selector] (プロパティセレクタ)

  ドル記号の後にピリオド (「$.」) が付いたプロパティセレクタをオフに設定します。プロパティセレクタは英数字の文字列であり、ハイフン (「-」) およびアンダースコア (「_」) をサポートします。文字列は科学表記をサポートしていません。プロパティセレクタは、JSON ログイベントの値ノードを指します。値ノードには、文字列または数値を指定できます。プロパティセレクタの後に配列を配置します。配列内の要素は 0 から始まる番号付けシステムに従います。つまり、配列の最初の要素は要素 0、2 番目の要素は要素 1 というようになります。要素を角かっこ (「[]」) で囲みます。プロパティセレクターが配列またはオブジェクトを指定している場合、フィルターパターンはログ形式を一致とみなしません。JSON プロパティにピリオド (".") が含まれている場合は、そのプロパティを選択するためにブラケット表記を使用できます。

  > **Note**
  >
  > [ワイルドカードセレクター]
  > JSON ワイルドカードを使用し、任意の配列要素または JSON オブジェクトフィールドを選択できます。
  > クォータ

プロパティセレクターでは 1 つのワイルドカードセレクターしか使用できません。

- 等値演算子

  等しい (「=」) または等しくない (「!=」) の記号のいずれかを使用して、等価演算子を区切ります。等値演算子は、ブール値 (true または false) を返します。

- 文字列

  文字列は、二重引用符 ("") で囲むことができます。英数字とアンダースコア記号以外の種類を含む文字列は、二重引用符で囲む必要があります。アスタリスク (「*」) をワイルドカードとして使用して、テキストを照合します。

> **Note**
>
> JSON ログイベントの語句と一致するフィルターパターンを作成するとき、任意の条件付き正規表現を使用できます。サポートされている正規表現のリストについては、「サポートされている正規表現」を参照してください。

次のコードスニペットには、文字列を持った JSON 語句とマッチさせるためにフィルターパターンをフォーマットする方法を示すフィルターパターンの例が含まれています。

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

Writing filter patterns that match numeric values

JSON ログイベントの数値と一致するフィルターパターンを作成できます。次のコードスニペットには、数値とマッチさせるフィルターパターンの構文例が示されています。

```
{ PropertySelector NumericOperator Number }
```

フィルターパターンを中括弧(「{}」)で囲みます。数値と一致するフィルターパターンには、次の部分が含まれている必要があります。

- [Property selector] (プロパティセレクタ)

ドル記号の後にピリオド (「$.」) が付いたプロパティセレクタをオフに設定します。プロパティセレクタは英数字の文字列であり、ハイフン (「-」) およびアンダースコア (「_」) をサポートします。文字列は科学表記をサポートしていません。プロパティセレクタは、JSON ログイベントの値ノードを指します。値ノードには、文字列または数値を指定できます。プロパティセレクタの後に配列を配置します。配列内の要素は 0 から始まる番号付けシステムに従います。つまり、配列の最初の要素は要素 0、2 番目の要素は要素 1 というようになります。要素を角かっこ (「[]」) で囲みます。プロパティセレクターが配列またはオブジェクトを指定している場合、フィルターパターンはログ形式を一致とみなしません。JSON プロパティにピリオド (".") が含まれている場合は、そのプロパティを選択するためにブラケット表記を使用できます。

> **Note**
>
> [ワイルドカードセレクター]
> JSON ワイルドカードを使用し、任意の配列要素または JSON オブジェクトフィールドを選択できます。
> クォータ
> プロパティセレクターでは 1 つのワイルドカードセレクターしか使用できません。

- 数値演算子

  より大きい (「>」)、より小さい (「<」)、等しい (「=」)、等しくない (「! =」)、以上 (「>=」)、または以下 (「<=」) のいずれかの記号を使用して、数値演算子を区切ります。

- 数値

  プラス (「+」) またはマイナス (「-」) 記号を含む整数を使用し、科学表記に従うことができます。アスタリスク (「*」) をワイルドカードとして使用して、数値を照合します。

次のコードス ニペットには、JSON 語句を数値をマッチさせるためにフィルターパターンをフォーマットする方法を示す例が含まれています。

```
// Filter pattern with greater than symbol
{ $.bandwidth > 75 }
// Filter pattern with less than symbol
{ $.latency < 50 }
// Filter pattern with greater than or equal to symbol
{ $.refreshRate >= 60 }
// Filter pattern with less than or equal to symbol
```

```
{ $.responseTime <= 5 }
// Filter pattern with equal sign
{ $.errorCode = 400}
// Filter pattern with not equal sign
{ $.errorCode != 500 }
// Filter pattern with scientific notation and plus symbol
{ $.number[0] = 1e-3 }
// Filter pattern with scientific notation and minus symbol
{ $.number[0] != 1e+3 }
```

簡単な表現を使用して JSON ログイベントで語句の一致

次の例には、フィルターパターンが JSON ログイベントの語句をマッチさせる方法について示すコードスニペットが含まれています。

> **Note**
>
> 例の JSON ログイベントを使用して例のフィルターパターンをテストする場合、例の JSON ログを 1 行で入力する必要があります。

[JSON ログイベント]

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

```
      "cluster.name": "c"
}
```

Example: Filter pattern that matches string values

このフィルターパターンは、プロパティ "eventType" の文字列 "UpdateTrail" と一致します。

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

Example: Filter pattern that matches string values (IP address)

このフィルターパターンは、プレフィックス "123.123." が付いた数字が含まれていないため、ワイルドカードが含んでおり、プロパティ"sourceIPAddress" を一致とみなします。

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

Example: Filter pattern that matches a specific array element with a string value

このフィルターパターンは、配列 "arrayKey" の要素 "value" と一致します。

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

Example: Filter pattern that matches a string using regex

このフィルターパターンは、プロパティ "eventType" の文字列 "Trail" と一致します。

```
{ $.eventType = %Trail% }
```

Example: Filter pattern that uses a wildcard to match values of any element in the array using regex

フィルターパターンには、配列 "arrayKey" の要素 "value" と一致する regex が含まれています。

```
{ $.arrayKey[*] = %val.{2}% }
```

Example: Filter pattern that uses a wildcard to match values of any element with a specific prefix and subnet using regex (IP address)

このフィルターパターンには、プロパティ "sourceIPAddress" の要素 "111.111.111.111" と一致する regex が含まれています。

```
{ $.* = %111\.111\.111\.1[0-9]{1,2}% }
```

> **Note**
>
> クォータ
> プロパティセレクターでは 1 つのワイルドカードセレクターしか使用できません。

Example: Filter pattern that matches a JSON property with a period (.) in the key

```
{ $.['cluster.name'] = "c" }
```

Example: Filter pattern that matches JSON logs using IS

IS 変数で JSON ログのフィールドと一致するフィルターパターンを作成できます。IS 変数は、値 NULL、TRUE または FALSE を含むフィールドと一致させることができます。次のフィルターパターンでは、SomeObject の値が NULL の場合、JSON ログが返されます。

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

Example: Filter pattern that matches JSON logs using NOT EXISTS

NOT EXISTS 変数を使用してフィルターパターンを作成し、ログデータに特定のフィールドを含まない JSON ログを返すことができます。次のフィルターパターンでは、NOT EXISTS を使用してフィールド SomeOtherObject を含まない JSON ログを返します。

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

> **Note**
>
> 変数 IS NOT および EXISTS は現在サポートされていません。

複合式を使用した JSON オブジェクトの語句の一致

論理演算子 AND (「&&」) と OR (「||」) をフィルターパターンで使用し、2 つ以上の条件が真であるログイベントをマッチさせる複合式を作成できます。複合式では、かっこ (「()」) の使用と、次の標準的な演算順序 (() > && > ||) がサポートされます。次の例には、JSON オブジェクトの語句とマッチさせるための複合式を持ったフィルターパターンを使用する方法について示すコードスニペットが含まれています。

[JSON オブジェクト]

```
{
    "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]
    ]
}
```

Example: Expression that matches using AND (&&)

このフィルターパターンには、"user" の "id" を 1 の数値とマッチし、文字列 "John.Doe@example.com" を使用して "users" 配列の最初の要素にある "email" とマッチする複合式が含まれています。

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

Example: Expression that matches using OR (||)

このフィルターパターンには、"user" の "email" を文字列 "John.Stiles@example.com" とマッチさせる複合式が含まれています。

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

Example: Expression that doesn't match using AND (&&)

このフィルターパターンには、式が "actions" の第 3 アクションとマッチしないため、一致が見つからない複合式が含まれています。

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

> **Note**
>
> クォータ
> プロパティセレクターではワイルドカードセレクターを 1 つしか使用できません。また、複合式を含むフィルターパターンでは最大 3 つのワイルドカードセレクターを使用することができます。

Example: Expression that doesn't match using OR (||)

このフィルターパターンには、式が "users" の最初のプロパティまたは"actions"の第 3 アクションと一致しないため、一致が見つからない複合式が含まれています。

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

フィルターパターンを使用したスペース区切りのログイベントでの語句の一致

スペース区切りのログイベントのフィルターパターン式

フィルターパターンを作成し、スペース区切りのログイベントで語句を一致させることができます。次の内容では、スペース区切りのログイベントの例を示し、スペース区切りのログイベントで語句を一致するフィルターパターンの構文を式する方法について説明します。

> **Note**
>
> スペース区切りのログイベントで語句を一致するフィルターパターンを作成するとき、任意の条件付正規表現を使用できます。サポートされている正規表現のリストについては、「サポートされている正規表現」を参照してください。

Example: Space-delimited log event

次のコードスニペットは、7 つのフィールド (ip、user、username、timestamp、request、status_code、および bytes) を含むスペース区切りログイベントを示しています。

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

> **Note**
>
> 角かっこ (「[]」) と二重引用符 ("") の間の文字は、単一フィールドと見なされます。

Writing filter patterns that match terms in a space-delimited log event

スペース区切りのログイベントで語句を一致するフィルターパターンを作成するには、フィルターパターンを括弧(「[]」)で囲み、カンマ(「,」)で区切られた名前でフィールドを指定します。次のフィルターパターンは 7 つのフィールドを解析します。

```
[ip=%127\.0\.0\.[1-9]%, user, username, timestamp, request =*.html*, status_code =
 4*, bytes]
```

数値演算子 (>、<、=、!=、>=、<=) とアスタリスク (*) をワイルド カードまたは regex として使用し、フィルターパターン条件を指定できます。フィルターパターンの例では、ip は 127.0.0.1 ~ 127.0.0.9 の IP アドレス範囲に一致する regex を使用し、request は .html の値を抽出する必要があることを示すワイルドカードが含まれ、status_code は 4 で始まる値を抽出する必要があることを示すワイルドカードが含まれています。

スペース区切りログイベントで解析するフィールドの数がわからない場合は、省略記号 (...) を使用して名前のないフィールドを参照できます。省略記号を使用すると、必要な数のフィールドを

参照できます。次の例には、前のフィルターパターンの例で示された最初の 4 つの無名フィールドを表す省略記号を使用するフィルターパターンが示されています。

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

論理演算子 AND (&&) と OR (||) を使用して複合式を作成することもできます。次のフィルターパターンには、status_code の値が 404 または 410 である必要があることを示す複合式が含まれています。

```
[ip, user, username, timestamp, request =*.html*, status_code = 404 || status_code =
 410, bytes]
```

パターン マッチングを使用したスペース区切りのログイベントの語句との一致

パターンマッチングを使用し、特定の順序で語句を一致するスペース区切りのフィルターパターンを作成できます。インジケーターを使用して語句の順序を指定します。[w1] を使用して最初の語句を表し、次に [w2] などを使用して、その後の語句の順序を表します。語句の間にカンマ (「,」) を入力します。次の例には、スペース区切りのフィルターパターンでパターンマッチングを使用する方法について示すコードスニペットが含まれています。

> **Note**
>
> スペース区切りのログイベントで語句を一致するフィルターパターンを作成するとき、任意の条件付正規表現を使用できます。サポートされている正規表現のリストについては、「サポートされている正規表現」を参照してください。

[スペース区切りのログイベント]

```
INFO 09/25/2014 12:00:00 GET /service/resource/67 1200
INFO 09/25/2014 12:00:01 POST /service/resource/67/part/111 1310
WARNING 09/25/2014 12:00:02 Invalid user request
ERROR 09/25/2014 12:00:02 Failed to process request
```

Example: Match terms in order

次のスペース区切りのフィルターパターンは、ログイベントの最初の単語が[エラー] であるログイベントを返します。

```
[w1=ERROR, w2]
```

> **Note**
>
> パターンマッチングを使用するスペース区切りのフィルターパターンを作成するとき、語句の順序を指定した後に空白のインジケーターを含める必要があります。たとえば、最初の単語が [エラー] であるログイベントを返すフィルターパターンを作成する場合、[w1] 語句の後に空白の [w2] インジケーターを含めます。

Example: Match terms with AND (&&) and OR (||)

論理演算子 AND (「&&」) と OR (「||」) を使用し、条件を含むスペース区切りのフィルターパターンを作成できます。次のフィルターパターンは、イベントの最初の単語が[エラー] または [警告] であるログイベントを返します。

```
[w1=ERROR || w1=WARNING, w2]
```

Example: Exclude terms from matches

1 つ以上の語句を除外するログイベントを返すスペース区切りのフィルターパターンを作成できます。除外する語句の前に等しくないの記号 (「!=」) を配置します。次のコードスニペットは、最初の単語が [エラー] と [警告] ではないログイベントを返すフィルターパターンの例を示しています。

```
[w1!=ERROR && w1!=WARNING, w2]
```

Example: Match the top level item in a resource URI

次のコードスニペットは、regex を使用してリソース URI の最上位の項目を一致するフィルターパターンの例を示しています。

```
[logLevel, date, time, method, url=%/service/resource/[0-9]+$%, response_time]
```

Example: Match the child level item in a resource URI

次のコードスニペットは、regex を使用してリソース URI の子レベルの項目を一致するフィルターパターンの例を示しています。

```
[logLevel, date, time, method, url=%/service/resource/[0-9]+/part/[0-9]+$%,
 response_time]
```

AWS サービスからのログ記録を有効にする

多くの サービスはログのみを CloudWatch Logs に発行しますが、一部の AWS サービスはログを Amazon Simple Storage Service または Amazon Data Firehose に直接発行できます。ログの主な要件が、これらのサービスのいずれかでのストレージまたは処理である場合は、ログを生成するサービスが、追加のセットアップを行わずに Amazon S3 または Firehose に直接ログを送信するように簡単にできます。

ログが Amazon S3 または Firehose に直接公開された場合でも、料金が適用されます。詳細については、「Amazon 料金表」の「ログ」タブの「販売されたログ」を参照してください。 CloudWatch

一部の AWS サービスでは、共通のインフラストラクチャを使用してログを送信します。これらのサービスからのロギングを有効にするには、特定の権限を持つユーザーとしてログインする必要があります。さらに、ログの送信を有効にする AWS には、 にアクセス許可を付与する必要があります。

これらのアクセス許可を必要とするサービスの場合、必要なアクセス許可には 2 つのバージョンがあります。これらの追加のアクセス許可を必要とするサービスは、表に [サポートあり [V1 アクセス許可]] および [サポートあり [V2 アクセス許可]] と表示されます。これらの必要な権限については、表の後のセクションを参照してください。

ログタイプ	CloudWatch Logs	Amazon S3	Firehose
Amazon API Gateway アクセスログ	サポートあり [V1 アクセス許可]		
AWS AppSync ログ	サポート		
Amazon Aurora MySQL ログ	サポート		
Amazon Bedrock ナレッジベースのログ記録	サポートあり [V2 アクセス許可]	サポートあり [V2 アクセス許可]	サポートあり [V2 アクセス許可]

ログタイプ	CloudWatch Logs	Amazon S3	Firehose
Amazon Chime のメディア品質メトリクスログと SIP メッセージログ	サポートあり [V1 アクセス許可]		
CloudFront: アクセスログ		サポートあり [V1 アクセス許可]	
AWS CloudHSM 監査ログ	サポート		
CloudWatch Evidently 評価イベントログ	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]	
CloudWatch Internet Monitor ログ		サポートあり [V1 アクセス許可]	
CloudTrail ログ	サポート		
AWS CodeBuild ログ	サポート		
Amazon CodeWhisperer イベントログ	サポートあり [V2 アクセス許可]	サポートあり [V2 アクセス許可]	サポートあり [V2 アクセス許可]
Amazon Cognito ログ	サポートあり [V1 アクセス許可]		
Amazon Connect のログ	サポート		
AWS DataSync ログ	サポート		

ログタイプ	CloudWatch Logs	Amazon S3	Firehose
Amazon ElastiCache for Redis ログ	サポートあり [V1 アクセス許可]		サポートあり [V1 アクセス許可]
AWS Elastic Beanstalk ログ	サポート		
Amazon Elastic Container Service のログ	サポート		
Amazon Elastic Kubernetes Service コントロールプレーンのログ	サポート		
Amazon EventBridge パイプのログ記録	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]
AWS Fargate ログ	サポート		
AWS Fault Injection Service 実験ログ		サポートあり [V1 アクセス許可]	
Amazon FinSpace	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]
AWS Global Accelerator フローログ		サポートあり [V1 アクセス許可]	
AWS Glue ジョブログ	サポート		
IAM Identity Center エラーログ	サポートあり [V2 アクセス許可]	サポートあり [V2 アクセス許可]	サポートあり [V2 アクセス許可]

ログタイプ	CloudWatch Logs	Amazon S3	Firehose
Amazon Interactive Video Service チャット ログ	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]
AWS IoT ログ	サポート		
AWS IoT FleetWise ログ	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]
AWS Lambda ログ	サポート		
Amazon Macie のログ	サポート		
AWS Mainframe Modernization	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]
Amazon Managed Service for Prometheus のログ	サポートあり [V1 アクセス許可]		
Amazon MSK ブローカーログ	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]
Amazon MSK Connect ログ	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]
Amazon MQ の一般ログと監査ログ	サポート		
AWS Network Firewall ログ	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]

ログタイプ	CloudWatch Logs	Amazon S3	Firehose
Network Load Balancer アクセスログ		サポートあり [V1 アクセス許可]	
OpenSearch ログ	サポート		
Amazon OpenSearch Service の取り込みログ	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]
AWS OpsWorks ログ	サポート		
Amazon Relational Database ServicePostgreSQL ログ	サポート		
AWS RoboMaker ログ	サポート		
Amazon Route 53 パブリック DNS クエリログ	サポート		
Amazon Route 53 Resolver クエリログ	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]	
Amazon SageMaker イベント	サポートあり [V1 アクセス許可]		
Amazon SageMaker ワーカーイベント	サポートあり [V1 アクセス許可]		
AWS Site-to_Site VPN ログ	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]
Amazon Simple Notification Service のログ	サポート		

ログタイプ	CloudWatch Logs	Amazon S3	Firehose
Amazon Simple Notification Service のデータ保護ポリシーログ	サポート		
EC2 スポットインスタンスのデータフィードファイル		サポートあり [V1 アクセス許可]	
AWS Step Functions Express ワークフローと標準ワークフローのログ	サポートあり [V1 アクセス許可]		
Storage Gateway 監査ログとヘルスログ	サポートあり [V1 アクセス許可]		
AWS Transfer Family ログ	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]
AWS Verified Access ログ	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]
Amazon Virtual Private Cloud フローログ	サポート	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]
Amazon VPC Lattice アクセスログ	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]
AWS WAF ログ	サポートあり [V1 アクセス許可]	サポートあり [V1 アクセス許可]	サポート

ログタイプ	CloudWatch Logs	Amazon S3	Firehose
Amazon WorkMail ログ	サポートあり [V2 アクセス許可]	サポートあり [V2 アクセス許可]	サポートあり [V2 アクセス許可]

追加のアクセス許可が必要なロギング [V1]

一部の AWS サービスでは、共通のインフラストラクチャを使用してログを CloudWatch Logs、Amazon S3、または Firehose に送信します。以下の表にリストされている AWS のサービスがこれらの宛先にログを送信できるようにするには、特定のアクセス許可を持つユーザーとしてログインする必要があります。

さらに、ログの送信を有効にする AWS には、 にアクセス許可を付与する必要があります。 は、ログの設定時にそれらのアクセス許可を自動的に作成 AWS できます。または、ログ記録を設定する前に最初に自分で作成することもできます。クロスアカウント配信の場合は、アクセス許可ポリシーを手動で作成する必要があります。

自分または組織の誰かが最初にログの送信を設定するときに、 で必要なアクセス許可とリソースポリシー AWS を自動的に設定することを選択した場合、このセクションで後述するように、ログの送信を設定するユーザーには特定のアクセス許可が必要です。または、リソースポリシーをユーザーが独自に作成することもできます。そうすると、ログの送信を設定するユーザーがそれほど多くのアクセス許可を持つ必要がなくなります。

次の表は、このセクションの情報が適用されるログの種類とログの送信先の概要です。

以下のセクションでは、これらの各送信先について詳しく説明します。

ログに送信された CloudWatch ログ

> **Important**
>
> 次のリストのログタイプを CloudWatch Logs に送信するように設定すると、 は必要に応じてログを受け取るロググループに関連付けられたリソースポリシー AWS を作成または変更します。詳細については、このセクションを続けてお読みください。

このセクションは、前のセクションの表にリストされているログのタイプが CloudWatch Logs に送信される場合に適用されます。

ユーザーアクセス許可

これらのタイプのログを初めて CloudWatch Logs に送信できるように設定するには、次のアクセス許可を持つアカウントにログインする必要があります。

- `logs:CreateLogDelivery`
- `logs:PutResourcePolicy`
- `logs:DescribeResourcePolicies`
- `logs:DescribeLogGroups`

> **Note**
>
> `logs:DescribeLogGroups`、`logs:DescribeResourcePolicies`、または アクセス`logs:PutResourcePolicy`許可を指定する場合は、1 つのロググループ名のみを指定するのではなく、そのResource行の ARN を*ワイルドカードを使用するように設定してください。例えば、次のようになります: `"Resource": "arn:aws:logs:us-east-1:111122223333:log-group:*"`

これらのタイプのログのいずれかが CloudWatch Logs のロググループにすでに送信されている場合、同じロググループへの別のタイプのログの送信を設定するには、 アクセス`logs:CreateLogDelivery`許可のみが必要です。

ロググループのリソースポリシー

ログが送信されているロググループには、特定のアクセス許可が含まれるリソースポリシーが必要です。ロググループに現在リソースポリシーがなく、ログ記録を設定するユーザーがロググループの `logs:PutResourcePolicy`、`logs:DescribeResourcePolicies`、および アクセス`logs:DescribeLogGroups`許可を持っている場合、 はログの CloudWatch ログへの送信を開始すると、次のポリシー AWS を自動的に作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite20150319",
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "delivery.logs.amazonaws.com"
        ]
      },
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:0123456789:log-group:my-log-group:log-stream:*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": ["0123456789"]
        },
        "ArnLike": {
          "aws:SourceArn": ["arn:aws:logs:us-east-1:0123456789:*"]
        }
      }
    }
  ]
}
```

ロググループにリソースポリシーがあるが、上記のポリシーにある文がそのポリシーに含まれておらず、ロギングをセットアップしているユーザーがロググループに対する `logs:PutResourcePolicy`、`logs:DescribeResourcePolicies`、および `logs:DescribeLogGroups` 許可を持っているという場合は、その文がロググループのリソースポリシーに追加されます。

ロググループリソースポリシーのサイズ制限に関する考慮事項

これらのサービスは、リソースポリシーでログを送信する各ロググループをリストする必要があります。また、 CloudWatch ログリソースポリシーは 5120 文字に制限されています。多数のロググループにログを送信するサービスは、この制限に達する可能性があります。

これを軽減するために、 CloudWatch Logs はログを送信しているサービスが使用するリソースポリシーのサイズをモニタリングし、ポリシーが 5120 文字のサイズ制限に近づいたことを検出した場合、そのサービスのリソースポリシー`/aws/vendedlogs/*`で CloudWatch を自動的に有効にします。その後、`/aws/vendedlogs/` で始まる名前のロググループをこれらのサービスからのログの送信先として使用し始めることができます。

Amazon S3 に送信されたログ

Amazon S3 にログを送信するように設定すると、 は、必要に応じてログを受信している S3 バケットに関連付けられたリソースポリシー AWS を作成または変更します。

Amazon S3 に直接発行されたログは、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。

ログを Amazon S3 バケットに初めて配信する場合、ログを配信するサービスはバケットの所有者を記録し、ログがこのアカウントに属するバケットにのみ配信されるようにします。その結果、Amazon S3 バケット所有者を変更するには、元のサービスでログサブスクリプションを再作成または更新する必要があります。

> **Note**
>
> CloudFront は、提供されたログを S3 に送信する他の サービスとは異なるアクセス許可モデルを使用します。詳細については、「標準ログ記録の設定およびログファイルへのアクセスに必要なアクセス許可」を参照してください。
> さらに、 CloudFront アクセスログと別のログソースに同じ S3 バケットを使用する場合、のバケットで ACL を有効にすると、このバケットを使用する他のすべてのログソース CloudFront にもアクセス許可が付与されます。

ユーザーアクセス許可

これらのタイプのログの Amazon S3 への送信を初めてセットアップするには、以下のアクセス許可でアカウントにログインする必要があります。

- `logs:CreateLogDelivery`
- `S3:GetBucketPolicy`
- `S3:PutBucketPolicy`

これらのタイプのログのいずれかが Amazon S3 バケットにすでに送信されている場合、これらの中の別のログを同じバケットに送信するためのセットアップに必要となるのは `logs:CreateLogDelivery` アクセス許可のみです。

S3 バケットのリソースポリシー

ログが送信されている S3 バケットには、特定のアクセス許可が含まれるリソースポリシーが必要です。バケットに現在リソースポリシーがなく、ログ記録を設定するユーザーがバケットの S3:GetBucketPolicyおよび アクセスS3:PutBucketPolicy許可を持っている場合、 は Amazon S3 へのログの送信を開始すると、そのバケットに対して次のポリシー AWS を自動的に作成します。

```
{
    "Version": "2012-10-17",
    "Id": "AWSLogDeliveryWrite20150319",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryAclCheck",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
                },
            "Action": "s3:GetBucketAcl",
            "Resource": "arn:aws:s3:::my-bucket",
            "Condition": {
                "StringEquals": {
                "aws:SourceAccount": ["0123456789"]
                },
                "ArnLike": {
                "aws:SourceArn": ["arn:aws:logs:us-east-1:0123456789:*"]
                }
            }
        },
        {
            "Sid": "AWSLogDeliveryWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-bucket/AWSLogs/account-ID/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": ["0123456789"]
                },
                "ArnLike": {
                    "aws:SourceArn": ["arn:aws:logs:us-east-1:0123456789:*"]
                }
```

```
            }
        }
    ]
}
```

前のポリシーでは、aws:SourceAccount にはこのバケットにログが配信されるアカウント ID のリストを指定します。aws:SourceArn には、ログを生成するリソースの ARN のリストを arn:aws:logs:*source-region*:*source-account-id*:* の形式で指定します。

バケットにリソースポリシーがあるが、上記のポリシーにあるステートメントがそのポリシーに含まれておらず、ロギングをセットアップしているユーザーがバケットに対する S3:GetBucketPolicy および S3:PutBucketPolicy アクセス許可を持っているという場合は、そのステートメントがバケットのリソースポリシーに追加されます。

> **Note**
>
> アクセスs3:ListBucket許可 AWS CloudTrail が に付与されていない場合、にAccessDeniedエラーが表示されることがありますdelivery.logs.amazonaws.com。これらのエラーが CloudTrail ログに記録されないようにするには、 に アクセスs3:ListBucket許可を付与delivery.logs.amazonaws.comし、前述のバケットポリシーの アクセスs3:GetBucketAcl許可セットで表示されるConditionパラメータを含める必要があります。これを簡単にするには、新しい Statement を作成する代わりに、AWSLogDeliveryAclCheck を “Action”: [“s3:GetBucketAcl”, “s3:ListBucket”] であるように直接更新することができます

Amazon S3 バケットのサーバー側の暗号化

Amazon S3 バケット内のデータを保護するには、Amazon S3 S3-managedキーによるサーバー側の暗号化 (SSE-S3) または に保存されている AWS KMS キーによるサーバー側の暗号化 AWS Key Management Service (SSE-KMS) のいずれかを有効にします。詳細については、「サーバー側の暗号化を使用したデータの保護」を参照してください。

SSE-S3 を選択した場合、追加の設定は必要ありません。Amazon S3 が暗号化キーを処理します。

> **Warning**
>
> SSE-KMS を選択した場合は、カスタマーマネージドキーを使用する必要があります。このシナリオでは、 AWS マネージドキーの使用はサポートされていないためです。 AWS マ

ネージドキーを使用して暗号化を設定すると、ログは読み取り不可能な形式で配信されます。

カスタマーマネージド AWS KMS キーを使用する場合、バケット暗号化を有効にするときに、カスタマーマネージドキーの Amazon リソースネーム (ARN) を指定できます。ログデリバリーアカウントが S3 バケットに書き込めるように、カスタマーマネージドキーのキーポリシー (S3 バケットのバケットポリシーではありません) に次を追加する必要があります。

SSE-KMS を選択した場合は、カスタマーマネージドキーを使用する必要があります。このシナリオでは、 AWS マネージドキーの使用はサポートされていないためです。カスタマーマネージド AWS KMS キーを使用する場合、バケット暗号化を有効にするときに、カスタマーマネージドキーの Amazon リソースネーム (ARN) を指定できます。ログデリバリーアカウントが S3 バケットに書き込めるように、カスタマーマネージドキーのキーポリシー (S3 バケットのバケットポリシーではありません) に次を追加する必要があります。

```
{
    "Sid": "Allow Logs Delivery to use the key",
    "Effect": "Allow",
    "Principal": {
        "Service": [ "delivery.logs.amazonaws.com" ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:SourceAccount": ["0123456789"]
        },
        "ArnLike": {
            "aws:SourceArn": ["arn:aws:logs:us-east-1:0123456789:*"]
        }
        }
}
```

aws:SourceAccount には、このバケットにログが配信されるアカウント ID のリストを指定します。aws:SourceArn には、ログを生成するリソースの ARN のリストを arn:aws:logs:*source-region*:*source-account-id*:* の形式で指定します。

Firehose に送信されたログ

このセクションは、前のセクションの表にリストされているログのタイプが Firehose に送信される場合に適用されます。

ユーザーアクセス許可

これらのタイプのログを Firehose に初めて送信するように設定するには、次のアクセス許可を持つアカウントにログインする必要があります。

- logs:CreateLogDelivery
- firehose:TagDeliveryStream
- iam:CreateServiceLinkedRole

これらのタイプのログのいずれかが Firehose にすでに送信されている場合は、これらのタイプの別のログの Firehose への送信をセットアップするには、 logs:CreateLogDeliveryとのfirehose:TagDeliveryStreamアクセス許可のみが必要です。

アクセス許可のために使用される IAM ロール

Firehose はリソースポリシーを使用しないため、これらのログを Firehose に送信するように設定するときに IAM ロール AWS を使用します。 という名前のサービスにリンクされたロール AWS を作成しますAWSServiceRoleForLogDelivery。このサービスリンクロールには、以下のアクセス許可が含まれます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "firehose:PutRecord",
                "firehose:PutRecordBatch",
                "firehose:ListTagsForDeliveryStream"
            ],
            "Resource": "*",
```

```
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/LogDeliveryEnabled": "true"
                }
            },
            "Effect": "Allow"
        }
    ]
}
```

このサービスにリンクされたロールは、 LogDeliveryEnabled タグが に設定されたすべての Firehose 配信ストリームに対するアクセス許可を付与しますtrue。ログ記録を設定するときに、このタグを宛先配信ストリームに AWS 付与します。

このサービスリンクロールには、delivery.logs.amazonaws.com サービスプリンシパルが必要なサービスリンクロールを引き受けることを可能にする信頼ポリシーもあります。以下がその信頼ポリシーです。

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

追加のアクセス許可が必要なロギング [V2]

一部の AWS サービスでは、新しい方法を使用してログを送信します。これは、これらのサービスからログ、Amazon S3 CloudWatch 、または Firehose の 1 つ以上の宛先へのログ配信を設定できる柔軟な方法です。

作業ログ配信は、次の 3 つの要素で構成されます。

- はDeliverySource、ログを実際に送信するリソースを表す論理オブジェクトです。

- はDeliveryDestination、実際の配信先を表す論理オブジェクトです。
- 配信元を配信先Deliveryに接続する 。

サポートされている AWS サービスと送信先間のログ配信を設定するには、以下を実行する必要があります。

- を使用して配信ソースを作成しますPutDeliverySource。
- を使用して配信先を作成しますPutDeliveryDestination。
- クロスアカウントでログを配信する場合は、送信先アカウント PutDeliveryDestinationPolicyで を使用して、送信先に IAM ポリシーを割り当てる必要があります。このポリシーは、アカウント A の配信元からアカウント B の配信先への配信の作成を許可します。クロスアカウント配信の場合は、アクセス許可ポリシーを手動で作成する必要があります。
- を使用して、1 つの配信元と 1 CreateDeliveryつの配信先をペアリングして配信を作成します。

以下のセクションでは、V2 プロセスを使用して各タイプの宛先へのログ配信を設定するためにサインインしたときに必要なアクセス許可の詳細について説明します。これらのアクセス許可は、サインインに使用する IAM ロールに付与できます。

> **Important**
>
> ログ生成リソースを削除した後、ログ配信リソースを削除するのはユーザーの責任です。そのためには、以下の手順に従います。
>
> 1. DeleteDelivery オペレーションDeliveryを使用して を削除します。
> 2. DeleteDeliverySource オペレーションDeliverySourceを使用して を削除します。
> 3. 削除DeliverySourceした DeliveryDestinationに関連付けられた がこの特定の にのみ使用される場合はDeliverySource、 DeleteDeliveryDestinationsオペレーションを使用して削除できます。

目次

- サービス固有のアクセス許可
- コンソール固有のアクセス許可

ログに送信された CloudWatch ログ

ユーザーアクセス許可

CloudWatch Logs へのログの送信を有効にするには、次のアクセス許可でサインインする必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:GetDelivery",
                "logs:GetDeliverySource",
                "logs:PutDeliveryDestination",
                "logs:GetDeliveryDestinationPolicy",
                "logs:DeleteDeliverySource",
                "logs:PutDeliveryDestinationPolicy",
                "logs:CreateDelivery",
                "logs:GetDeliveryDestination",
                "logs:PutDeliverySource",
                "logs:DeleteDeliveryDestination",
                "logs:DeleteDeliveryDestinationPolicy",
                "logs:DeleteDelivery"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:delivery:*",
                "arn:aws:logs:region:account-id:delivery-source:*",
                "arn:aws:logs:region:account-id:delivery-destination:*"
            ]
        },
        {
            "Sid": "ListAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeDeliveryDestinations",
                "logs:DescribeDeliverySources",
```

```
                "logs:DescribeDeliveries"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUpdatesToResourcePolicyCWL",
            "Effect": "Allow",
            "Action": [
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:*"
            ]
        }
    ]
}
```

ロググループのリソースポリシー

ログが送信されているロググループには、特定のアクセス許可が含まれるリソースポリシーが必要です。ロググループに現在リソースポリシーがなく、ログ記録を設定するユーザーがロググループの `logs:PutResourcePolicy`、`logs:DescribeResourcePolicies`、および アクセス`logs:DescribeLogGroups`許可を持っている場合、 はログの CloudWatch ログへの送信を開始すると、次のポリシー AWS を自動的に作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite20150319",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "delivery.logs.amazonaws.com"
        ]
      },
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
```

```
        "arn:aws:logs:us-east-1:0123456789:log-group:my-log-group:log-stream:*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": ["0123456789"]
        },
        "ArnLike": {
          "aws:SourceArn": ["arn:aws:logs:us-east-1:0123456789:*"]
        }
      }
    }
  ]
}
```

ロググループリソースポリシーのサイズ制限に関する考慮事項

これらのサービスは、リソースポリシーでログを送信する各ロググループをリストする必要があります。また、 CloudWatch ログリソースポリシーは 5120 文字に制限されています。このため、多数のロググループにログを送信するサービスは、この上限に到達する可能性があります。

これを軽減するために、 CloudWatch Logs はログを送信しているサービスが使用するリソースポリシーのサイズをモニタリングし、ポリシーが 5120 文字のサイズ制限に近づいたことを検出した場合、そのサービスのリソースポリシー/aws/vendedlogs/*で CloudWatch を自動的に有効にします。その後、/aws/vendedlogs/ で始まる名前のロググループをこれらのサービスからのログの送信先として使用し始めることができます。

Amazon S3 に送信されたログ

ユーザーアクセス許可

Amazon S3 へのログ送信を有効にするには、次のアクセス許可でサインインする必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:GetDelivery",
                "logs:GetDeliverySource",
                "logs:PutDeliveryDestination",
```

```
                "logs:GetDeliveryDestinationPolicy",
                "logs:DeleteDeliverySource",
                "logs:PutDeliveryDestinationPolicy",
                "logs:CreateDelivery",
                "logs:GetDeliveryDestination",
                "logs:PutDeliverySource",
                "logs:DeleteDeliveryDestination",
                "logs:DeleteDeliveryDestinationPolicy",
                "logs:DeleteDelivery"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:delivery:*",
                "arn:aws:logs:region:account-id:delivery-source:*",
                "arn:aws:logs:region:account-id:delivery-destination:*"
            ]
        },
        {
            "Sid": "ListAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeDeliveryDestinations",
                "logs:DescribeDeliverySources",
                "logs:DescribeDeliveries"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUpdatesToResourcePolicyS3",
            "Effect": "Allow",
            "Action": [
                "s3:PutBucketPolicy",
                "s3:GetBucketPolicy"
            ],
            "Resource": "arn:aws:s3:::bucket-name"
        }
    ]
}
```

ログが送信されている S3 バケットには、特定のアクセス許可が含まれるリソースポリシーが必要です。バケットに現在リソースポリシーがなく、ログ記録を設定するユーザーがバケットの S3:GetBucketPolicyおよび アクセスS3:PutBucketPolicy許可を持っている場合、 は Amazon S3 へのログの送信を開始すると、そのバケットに対して次のポリシー AWS を自動的に作成します。

```
{
    "Version": "2012-10-17",
    "Id": "AWSLogDeliveryWrite20150319",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryAclCheck",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
                },
            "Action": "s3:GetBucketAcl",
            "Resource": "arn:aws:s3:::my-bucket",
            "Condition": {
                "StringEquals": {
                "aws:SourceAccount": ["0123456789"]
                },
                "ArnLike": {
                "aws:SourceArn": ["arn:aws:logs:us-east-1:0123456789:delivery-source*"]
                }
            }
        },
        {
            "Sid": "AWSLogDeliveryWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-bucket/AWSLogs/account-ID/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": ["0123456789"]
                },
                "ArnLike": {
                    "aws:SourceArn": ["arn:aws:logs:us-east-1:0123456789:delivery-
source:*"]
                }
            }
        }
    ]
}
```

前のポリシーでは、aws:SourceAccount にはこのバケットにログが配信されるアカウント ID のリストを指定します。aws:SourceArn には、ログを生成するリソースの ARN のリストを arn:aws:logs:*source-region*:*source-account-id*:* の形式で指定します。

バケットにリソースポリシーがあるが、上記のポリシーにあるステートメントがそのポリシーに含まれておらず、ロギングをセットアップしているユーザーがバケットに対する S3:GetBucketPolicy および S3:PutBucketPolicy アクセス許可を持っているという場合は、そのステートメントがバケットのリソースポリシーに追加されます。

> **Note**
>
> アクセスs3:ListBucket許可 AWS CloudTrail が に付与されていない場合、にAccessDeniedエラーが表示されることがありますdelivery.logs.amazonaws.com。これらのエラーが CloudTrail ログに記録されないようにするには、 に アクセスs3:ListBucket許可を付与delivery.logs.amazonaws.comし、前述のバケットポリシーの アクセスs3:GetBucketAcl許可セットで表示されるConditionパラメータを含める必要があります。これを簡単にするには、新しい Statement を作成する代わりに、AWSLogDeliveryAclCheck を “Action”: [“s3:GetBucketAcl”, “s3:ListBucket”] であるように直接更新することができます

Amazon S3 バケットのサーバー側の暗号化

Amazon S3 バケット内のデータを保護するには、Amazon S3 S3-managedキーによるサーバー側の暗号化 (SSE-S3) または に保存されている AWS KMS キーによるサーバー側の暗号化 AWS Key Management Service (SSE-KMS) のいずれかを有効にします。詳細については、「サーバー側の暗号化を使用したデータの保護」を参照してください。

SSE-S3 を選択した場合、追加の設定は必要ありません。Amazon S3 が暗号化キーを処理します。

> **Warning**
>
> SSE-KMS を選択した場合は、カスタマーマネージドキーを使用する必要があります。このシナリオでは、 AWS マネージドキーの使用はサポートされていないためです。 AWS マネージドキーを使用して暗号化を設定すると、ログは読み取り不可能な形式で配信されます。

カスタマーマネージド AWS KMS キーを使用する場合、バケット暗号化を有効にするときに、カスタマーマネージドキーの Amazon リソースネーム (ARN) を指定できます。ログデリバリーアカウントが S3 バケットに書き込めるように、カスタマーマネージドキーのキーポリシー (S3 バケットのバケットポリシーではありません) に次を追加する必要があります。

SSE-KMS を選択した場合は、カスタマーマネージドキーを使用する必要があります。このシナリオでは、 AWS マネージドキーの使用はサポートされていないためです。カスタマーマネージド AWS KMS キーを使用する場合、バケット暗号化を有効にするときに、カスタマーマネージドキーの Amazon リソースネーム (ARN) を指定できます。ログデリバリーアカウントが S3 バケットに書き込めるように、カスタマーマネージドキーのキーポリシー (S3 バケットのバケットポリシーではありません) に次を追加する必要があります。

```
{
    "Sid": "Allow Logs Delivery to use the key",
    "Effect": "Allow",
    "Principal": {
        "Service": [ "delivery.logs.amazonaws.com" ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:SourceAccount": ["0123456789"]
        },
        "ArnLike": {
            "aws:SourceArn": ["arn:aws:logs:us-east-1:0123456789:delivery-source:*"]
        }
        }
}
```

`aws:SourceAccount` には、このバケットにログが配信されるアカウント ID のリストを指定します。`aws:SourceArn` には、ログを生成するリソースの ARN のリストを `arn:aws:logs:source-region:source-account-id:*` の形式で指定します。

Firehose に送信されたログ

ユーザーアクセス許可

Firehose へのログの送信を有効にするには、次のアクセス許可でサインインする必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:GetDelivery",
                "logs:GetDeliverySource",
                "logs:PutDeliveryDestination",
                "logs:GetDeliveryDestinationPolicy",
                "logs:DeleteDeliverySource",
                "logs:PutDeliveryDestinationPolicy",
                "logs:CreateDelivery",
                "logs:GetDeliveryDestination",
                "logs:PutDeliverySource",
                "logs:DeleteDeliveryDestination",
                "logs:DeleteDeliveryDestinationPolicy",
                "logs:DeleteDelivery"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:delivery:*",
                "arn:aws:logs:region:account-id:delivery-source:*",
                "arn:aws:logs:region:account-id:delivery-destination:*"
            ]
        },
        {
            "Sid": "ListAccessForLogDeliveryActions",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeDeliveryDestinations",
                "logs:DescribeDeliverySources",
                "logs:DescribeDeliveries"
            ],
            "Resource": "*"
        },
        {
```

```
            "Sid": "AllowUpdatesToResourcePolicyFH",
            "Effect": "Allow",
            "Action": [
                "firehose:TagDeliveryStream"
            ],
            "Resource": [
                "arn:aws:firehose:region:account-id:deliverystream/*"
            ]
        },
        {
            "Sid": "CreateServiceLinkedRole",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::account-id:role/aws-service-role/
delivery.logs.amazonaws.com/AWSServiceRoleForLogDelivery"
        }
    ]
}
```

リソースのアクセス許可のために使用される IAM ロール

Firehose はリソースポリシーを使用しないため、これらのログを Firehose に送信するように設定するときに IAM ロール AWS を使用します。 という名前のサービスにリンクされたロール AWS を作成しますAWSServiceRoleForLogDelivery。このサービスリンクロールには、以下のアクセス許可が含まれます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "firehose:PutRecord",
                "firehose:PutRecordBatch",
                "firehose:ListTagsForDeliveryStream"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/LogDeliveryEnabled": "true"
                }
            },
```

```
            "Effect": "Allow"
        }
    ]
}
```

このサービスにリンクされたロールは、 LogDeliveryEnabled タグが に設定されたすべての Firehose 配信ストリームに対するアクセス許可を付与しますtrue。ログ記録を設定するときに、このタグを宛先配信ストリームに AWS 付与します。

このサービスリンクロールには、delivery.logs.amazonaws.com サービスプリンシパルが必要なサービスリンクロールを引き受けることを可能にする信頼ポリシーもあります。以下がその信頼ポリシーです。

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

サービス固有のアクセス許可

前のセクションに記載されている送信先固有のアクセス許可に加えて、一部のサービスでは、セキュリティの追加レイヤーとして、お客様が リソースからログを送信することを許可する明示的な認可が必要です。これは、そのサービス内でログを供給したリソースの AllowVendedLogDeliveryForResource アクションを承認します。これらのサービスでは、次のポリシーを使用し、*service* と *resource-type* を適切な値に置き換えます。これらのフィールドのサービス固有の値については、これらのサービスのドキュメントページを参照してください。

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

```
            "Effect": "Allow",
            "Action": [
                "service:AllowVendedLogDeliveryForResource"
            ],
            "Resource": "arn:aws:service:region:account-id:resource-type/*"
        }
    ]
}
```

コンソール固有のアクセス許可

前のセクションで説明したアクセス許可に加えて、APIs ではなくコンソールを使用してログ配信を設定する場合は、次の追加のアクセス許可も必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowLogDeliveryActionsConsoleCWL",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:*"
            ]
        },
        {
            "Sid": "AllowLogDeliveryActionsConsoleS3",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        },
        {
            "Sid": "AllowLogDeliveryActionsConsoleFH",
            "Effect": "Allow",
            "Action": [
```

```
                "firehose:ListDeliveryStreams",
                "firehose:DescribeDeliveryStream"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

サービス間での不分別な代理処理の防止

混乱した代理問題は、アクションを実行するためのアクセス許可を持たないエンティティが、より特権のあるエンティティにアクションの実行を強制できてしまう場合に生じる、セキュリティ上の問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。これを防ぐため、 AWS では、アカウント内のリソースへのアクセス権が付与されたサービスプリンシパルですべてのサービスのデータを保護するために役立つツールを提供しています。

CloudWatch Logs が別のサービスに付与するアクセス許可をリソースポリシーで制限するには`aws:SourceArn`、`aws:SourceAccount`、`aws:SourceOrgID`、および `aws:SourceOrgPaths` グローバル条件コンテキストキーを使用することをお勧めします。1 つのリソースだけをクロスサービスのアクセスに関連付ける場合は、`aws:SourceArn` を使用します。アカウント内の任意のリソースをクロスサービスの使用に関連付ける場合は、`aws:SourceAccount` を使用します。組織内の任意のアカウントの任意のリソースをクロスサービスの使用に関連付ける場合は、`aws:SourceOrgID` を使用します。 AWS Organizations パス内の任意のアカウントのリソースをクロスサービスの使用に関連付ける場合は、`aws:SourceOrgPaths` を使用します。パスの使用と理解の詳細については、AWS Organizations 「エンティティパスの理解」を参照してください。

混乱した代理問題から保護するための最も効果的な方法は、リソースの完全な ARN を指定して、`aws:SourceArn` グローバル条件コンテキストキーを使用することです。リソースの完全な ARN が不明な場合や、複数のリソースを指定する場合には、グローバルコンテキスト条件キー `aws:SourceArn` で、ARN の未知部分を示すためにワイルドカード文字 (*) を使用します。例えば、`arn:aws:`*`servicename`*`:*:`*`123456789012`*`:*` です。

aws:SourceArn の値に Amazon S3 バケット ARN などのアカウント ID が含まれていない場合は、両方の aws:SourceAccount と aws:SourceArn を使用して、アクセス許可を制限する必要があります。

混乱した代理問題から保護するために、リソースベースポリシー内のリソースの組織 ID または組織パスを指定しながら、aws:SourceOrgID または aws:SourceOrgPaths のグローバル条件コンテキストキーを使用してください。aws:SourceOrgID または aws:SourceOrgPaths キーを含むポリシーには正しいアカウントが自動的に組み込まれるため、組織のアカウントを追加、削除、移動する際には手動で更新する必要はありません。

このページの前のセクションにあるポリシーでは、aws:SourceArn と aws:SourceAccount グローバル条件コンテキストキーを使って、混乱した代理問題を防ぐ方法を示しています。

CloudWatch AWS 管理ポリシーの更新をログに記録します。

このサービスがこれらの変更の追跡を開始した以降の CloudWatch ログの AWS マネージドポリシーの更新に関する詳細を表示します。このページの変更に関する自動通知を受け取るには、ログドキュメントの履歴ページの RSS CloudWatch フィードにサブスクライブしてください。

変更	説明	日付
AWSServiceRoleForLogDelivery サービスにリンクされたロールポリシー — 既存のポリシーへの更新	CloudWatch ログは、AWSServiceRoleForLogDeliveryサービスにリンクされたロールに関連付けられた IAM ポリシーのアクセス許可を変更しました。以下の変更が行われました。 • firehose:ResourceTag/LogDeliveryEnabled": "true" 条件キーが aws:ResourceTag/LogDeliveryEnabled": "true" に変更されました。	2021 年 7 月 15 日

変更	説明	日付
CloudWatch ログが変更の追跡を開始しました	CloudWatch ログが AWS マネージドポリシーの変更の追跡を開始しました。	2021 年 6 月 10 日

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、別のシステムに読み込んだりします。同じアカウントまたは別のアカウントのバケットにエクスポートできます。

以下の操作を行うことができます。

- AWS Key Management Service（ AWS KMS) で SSE-KMS によって暗号化された S3 バケットにログデータをエクスポートする
- 保持期間が設定されている S3 オブジェクトロックが有効になっている S3 バケットに、ログデータをエクスポートする

> **Note**
>
> Amazon S3 へのエクスポートは、標準ログクラスのロググループでのみサポートされます。ログクラスの詳細については、「」を参照してくださいログクラス。

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を S3 バケットに保存し、Amazon S3 ライフサイクルルールを定義すると、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ことができます。

AES-256 または SSE-KMS で暗号化された S3 バケットへのエクスポートが可能です。DSSE-KMS で暗号化されたバケットへのエクスポートはサポートされていません。

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。

> **Note**
>
> エクスポートされたファイル内のログデータのチャンクに対する時間ベースのソートは保証されません。Linux ユーティリティを使用して、エクスポートされたログフィールドデータをソートできます。例えば、次のユーティリティコマンドは、1 つのフォルダー内のすべての .gz ファイルのイベントをソートします。

```
find . -exec zcat {} + | sed -r 's/^[0-9]+/\x0&/' | sort -z
```

次のユーティリティコマンドは、複数のサブフォルダにある.gz ファイルをソートします。

```
find ./*/ -type f -exec zcat {} + | sed -r 's/^[0-9]+/\x0&/' | sort -z
```

さらに、別の stdout コマンドを使用して、ソートされた出力を別のファイルにパイプして保存することもできます。

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。エクスポートタスクは 24 時間後にタイムアウトします。エクスポートタスクがタイムアウトする場合は、エクスポートタスクを作成するときの時間範囲を短くしてください。

ほぼリアルタイムのログデータ分析については、CloudWatch Logs Insights を使用したログデータの分析 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 を参照してください。

内容

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 S3 バケットにエクスポートされます。

開始日 (タイムスタンプ)

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。この時刻以降に取り込まれたロググループ内のすべてのログイベントがエクスポートされます。

終了日 (タイムスタンプ)

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ている 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 Amazon 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てログデータを Amazon S3 にエクスポートする

次の例では、Amazon CloudWatch コンソールを使用して、 という名前の Amazon CloudWatch Logs ロググループから という名前の Amazon S3 バケット`my-log-group`にすべてのデータをエクスポートします`my-exported-logs`。

SSE-KMS によって暗号化された S3 バケットへのログデータのエクスポートは、サポートされています。DSSE-KMS で暗号化されたバケットへのエクスポートはサポートされていません。

エクスポートの設定方法の詳細は、エクスポート先の Amazon S3 バケットがエクスポート対象のログと同じアカウントにあるか、別のアカウントにあるかによって異なります。

トピック

- 同一アカウントへのエクスポート
- クロスアカウントでのエクスポート

同一アカウントへのエクスポート

Amazon S3 バケットがエクスポート対象のログと同じアカウントにある場合は、このセクションの手順を使用してください。

トピック

ステップ 1: Amazon S3 バケットを作成する

CloudWatch Logs 専用に作成され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**
>
> 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する必要があります。CloudWatch Logs は、別のリージョンの 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。

S3 バケットを作成するには

1. https://console.aws.amazon.com/s3/で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 ログが存在するリージョンを選択します。
3. [バケットを作成] 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 ログデータが存在するリージョンを選択します。
6. [作成] を選択します。

ステップ 2: アクセス許可を設定する

ステップ 5 でエクスポートタスクを作成するには、AmazonS3ReadOnlyAccess IAM ロールと以下のアクセス許可でサインオンする必要があります。

- logs:CreateExportTask
- logs:CancelExportTask
- logs:DescribeExportTasks
- logs:DescribeLogStreams
- logs:DescribeLogGroups

アクセス権限を付与するには、ユーザー、グループ、またはロールにアクセス許可を追加します。

- のユーザーとグループ AWS IAM Identity Center：

  アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」の手順に従ってください。
- IAM 内で、ID プロバイダーによって管理されているユーザー:

  ID フェデレーションのロールを作成します。詳細については、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールの作成」を参照してください。
- IAM ユーザー:
  - ユーザーが担当できるロールを作成します。手順については、「IAM ユーザーガイド」の「IAM ユーザー用ロールの作成」を参照してください。
  - (お奨めできない方法) ポリシーをユーザーに直接アタッチするか、ユーザーをユーザーグループに追加する。詳細については、「IAM ユーザーガイド」の「ユーザー (コンソール) へのアクセス権限の追加」を参照してください。

ステップ 3: S3 バケットのアクセス許可を設定する

すべての S3 バケットとオブジェクトは、デフォルト状態でプライベートに設定されます。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。

> **Important**
>
> S3 バケットへのエクスポートをより安全にするために、ログデータを S3 バケットにエクスポートできるソースアカウントのリストの指定が必要になりました。
> 次の例では、`aws:SourceAccount`キー内のアカウント IDs のリストは、ユーザーがログデータを S3 バケットにエクスポートできるアカウントになります。`aws:SourceArn` キーは、アクションが実行される対象のリソースです。これを特定のロググループに制限することも、この例のようにワイルドカードを使用することもできます。
> S3 バケットが作成されたアカウントのアカウント ID も含めることで、エクスポートを同じアカウント内で行えるようにすることをお勧めします。

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. [Bucket Policy Editor] (バケットポリシーエディタ) で、以下のポリシーを追加します。`my-exported-logs` を Amazon S3 バケットの名前に変更します。[プリンシパル] に `us-west-1` などの正しいリージョンエンドポイントを指定してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs",
          "Principal": { "Service": "logs.Region.amazonaws.com" },
          "Condition": {
            "StringEquals": {
                "aws:SourceAccount": [
                    "AccountId1",
                    "AccountId2",
                    ...
```

```
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:Region:AccountId1:log-group:*",
                        "arn:aws:logs:Region:AccountId2:log-group:*",
                        ...
                     ]
            }
          }
      },
      {
          "Action": "s3:PutObject" ,
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs/*",
          "Principal": { "Service": "logs.Region.amazonaws.com" },
          "Condition": {
            "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control",
                "aws:SourceAccount": [
                    "AccountId1",
                    "AccountId2",
                    ...
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:Region:AccountId1:log-group:*",
                        "arn:aws:logs:Region:AccountId2:log-group:*",
                        ...
                    ]
            }
          }
      }
    ]
}
```

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 ログはログデータを S3 バケットにエクスポートできます。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**
>
> 既存のバケットにすでに 1 つ以上のポリシーがアタッチされている場合は、そのポリシーへの 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。

(オプション) ステップ 4: SSE-KMS で暗号化されたバケットへのエクスポート

このステップは、 でサーバー側の暗号化を使用する S3 バケットにエクスポートする場合にのみ必要です AWS KMS keys。この暗号化は SSE-KMS と呼ばれます。

SSE-KMS で暗号化されたバケットにエクスポートするには

1. https://console.aws.amazon.com/kms で AWS KMS コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクタを使用します。
3. 左のナビゲーションバーで、[Customer managed keys] (カスタマーマネージドキー) を選択します。

   [Create Key] (キーを作成) を選択します。
4. [キーの種類] で、[対称] を選択します。
5. [Key usage] (キーの使用) で、[Encrypt and decrypt] (暗号化および復号化) 、[Next] (次へ) の順に選択します。
6. [Add labels] (ラベルを追加) で、キーのエイリアスを入力し、オプションで説明またはタグを追加します。次いで、[次へ] を選択します。
7. [Key administrators] (キー管理者) で、このキーを管理できるユーザーを選択した後、[Next] (次へ) を選択します。
8. [Define key usage permissions] (キーの使用アクセス許可を定義) の設定は変更せずに、[Next] (次へ) を選択します。
9. 設定した内容を確認し、[Finish] (終了) を選択します。
10. [Customer managed keys] (カスタマーマネージドキー) ページに戻り、この前に作成したキーの名前を選択します。
11. [Key policy] (キーポリシー) タブを表示し、次に [Switch to policy view] (ポリシービューへの切り替え) を選択します。

12. [Key policy] (キーポリシー) セクションで、[Edit] (編集) を選択します。
13. キーポリシーステートメントのリストに、次のステートメントを追加します。これを行う際、*Region* は実際のログのリージョンに置き換え、*account-ARN* は KMS キーを所有するアカウントの ARN に置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow CWL Service Principal usage",
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.Region.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "account-ARN"
            },
            "Action": [
                "kms:GetKeyPolicy*",
                "kms:PutKeyPolicy*",
                "kms:DescribeKey*",
                "kms:CreateAlias*",
                "kms:ScheduleKeyDeletion*",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

14. [変更を保存] を選択します。
15. https://console.aws.amazon.com/s3/でAmazon S3 コンソールを開きます。
16. ステップ 1: S3 バケットを作成する。 で作成したバケットを検索し、その名前を選択します。

17. [プロパティ] タブを選択します。[Default Encryption] (デフォルトの暗号化) で、[Edit] (編集) を選択します。
18. [Server-side Encryption] (サーバー側の暗号化) で、[Enable] (有効化) を選択します。
19. 暗号キーで、AWS Key Management Service キー (SSE-KMS) を選択します。
20. AWS KMS キーから選択を選択し、作成したキーを見つけます。
21. [Bucket key] (バケットキー) で、[Enable] (有効化) を選択します。
22. [変更を保存] を選択します。

ステップ 5: 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: アクセス許可を設定する に記載されているように、十分なアクセス許可を使用してサインインします。
2. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
3. ナビゲーションペインで、[ロググループ] を選択します。
4. [ロググループ] 画面で、ロググループの名前を選択します。
5. [Actions (アクション)]、[Export data to Amazon S3 (データを Amazon S3 にエクスポート)] の順に選択します。
6. [Export data to Amazon S3 (データを Amazon S3 にエクスポート)] 画面の [Define data export (データエクスポートを定義)] で、[From (開始)] と [To (終了)] 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。
7. 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。
8. [Choose S3 bucket] (S3 バケットの選択) で、S3 バケットに関連付けられたアカウントを選択します。
9. [S3 bucket name] (S3 バケット名) で、 バケットを選択します。
10. [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。
11. [Export (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。

12. 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、[Actions (アクション)]、[View all exports to Amazon S3 (Amazon S3 へのすべてのエクスポートを表示)] を選択します。

クロスアカウントでのエクスポート

Amazon S3 バケットがエクスポート対象のログとは別のアカウントにある場合は、このセクションの手順を使用してください。

トピック

ステップ 1: Amazon S3 バケットを作成する

CloudWatch Logs 専用に作成され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**
>
> 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する必要があります。CloudWatch Logs は、別のリージョンの 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。

S3 バケットを作成するには

1. https://console.aws.amazon.com/s3/で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 ログが存在するリージョンを選択します。
3. [バケットを作成] 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 ログデータが存在するリージョンを選択します。

6. [作成] を選択します。

ステップ 2: アクセス許可を設定する

まず、新しい IAM ポリシーを作成して、送信先アカウントの送信先 Amazon S3 バケットに対する`s3:PutObject`アクセス許可を CloudWatch Logs に付与できるようにする必要があります。Amazon S3

作成するポリシーは、レプリケート先バケットが AWS KMS 暗号化を使用するかどうかによって異なります。

Amazon S3 バケットにログをエクスポートする IAM ポリシーを作成するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左側のナビゲーションペインで、[ポリシー] を選択します。
3. [ポリシーの作成] を選択します。
4. [ポリシーエディター] セクションで、[JSON] を選択します。
5. 送信先バケットが AWS KMS 暗号化を使用しない場合は、次のポリシーをエディタに貼り付けます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-exported-logs/*"
        }
    ]
}
```

レプリケート先バケットが AWS KMS 暗号化を使用している場合は、次のポリシーをエディタに貼り付けます。

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

```
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::my-exported-logs/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "ARN_OF_KMS_KEY"
    }
  ]
}
```

6. [次へ] をクリックします。
7. ポリシー名を入力します。この名前を使用して、ポリシーを IAM ロールにアタッチします。
8. 次に、[ポリシーの作成] を選択してポリシーを保存します。

ステップ 5 でエクスポートタスクを作成するために、AmazonS3ReadOnlyAccess IAM ロールでサインオンする必要があります。また、作成したばかりの IAM ポリシーと以下のアクセス許可でもサインオンする必要があります。

- logs:CreateExportTask
- logs:CancelExportTask
- logs:DescribeExportTasks
- logs:DescribeLogStreams
- logs:DescribeLogGroups

アクセス権限を付与するには、ユーザー、グループ、またはロールにアクセス許可を追加します。

- のユーザーとグループ AWS IAM Identity Center :

  アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」の手順に従ってください。
- IAM 内で、ID プロバイダーによって管理されているユーザー:

  ID フェデレーションのロールを作成します。詳細については、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールの作成」を参照してください。

- IAM ユーザー:
  - ユーザーが担当できるロールを作成します。手順については、「IAM ユーザーガイド」の「IAM ユーザー用ロールの作成」を参照してください。
  - (お奨めできない方法) ポリシーをユーザーに直接アタッチするか、ユーザーをユーザーグループに追加する。詳細については、「IAM ユーザーガイド」の「ユーザー (コンソール) へのアクセス権限の追加」を参照してください。

ステップ 3: S3 バケットのアクセス許可を設定する

すべての S3 バケットとオブジェクトは、デフォルト状態でプライベートに設定されます。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。

> **Important**
>
> S3 バケットへのエクスポートをより安全にするために、ログデータを S3 バケットにエクスポートできるソースアカウントのリストの指定が必要になりました。
> 次の例では、`aws:SourceAccount`キー内のアカウント IDs のリストは、ユーザーがログデータを S3 バケットにエクスポートできるアカウントになります。`aws:SourceArn` キーは、アクションが実行される対象のリソースです。これを特定のロググループに制限することも、この例のようにワイルドカードを使用することもできます。
> S3 バケットが作成されたアカウントのアカウント ID も含めることで、エクスポートを同じアカウント内で行えるようにすることをお勧めします。

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. [Bucket Policy Editor] (バケットポリシーエディタ) で、以下のポリシーを追加します。`my-exported-logs` を Amazon S3 バケットの名前に変更します。[プリンシパル] に `us-west-1` などの正しいリージョンエンドポイントを指定してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs",
          "Principal": { "Service": "logs.Region.amazonaws.com" },
          "Condition": {
            "StringEquals": {
                "aws:SourceAccount": [
                    "AccountId1",
                    "AccountId2",
                    ...
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:Region:AccountId1:log-group:*",
                        "arn:aws:logs:Region:AccountId2:log-group:*",
                        ...
                     ]
            }
          }
      },
      {
          "Action": "s3:PutObject" ,
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs/*",
          "Principal": { "Service": "logs.Region.amazonaws.com" },
          "Condition": {
            "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control",
                "aws:SourceAccount": [
                    "AccountId1",
                    "AccountId2",
                    ...
                ]
            },
```

```
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:Region:AccountId1:log-group:*",
                        "arn:aws:logs:Region:AccountId2:log-group:*",
                        ...
                    ]
            }
        }
      },
      {
          "Effect": "Allow",
          "Principal": {
            "AWS": "arn:aws:iam::create_export_task_caller_account:role/role_name"
          },
          "Action": "s3:PutObject",
          "Resource": "arn:aws:s3:::my-exported-logs/*",
          "Condition": {
            "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control"
            }
          }
      }
    ]
}
```

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 ログはログデータを S3 バケットにエクスポートできます。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**
>
> 既存のバケットにすでに 1 つ以上のポリシーがアタッチされている場合は、そのポリシーへの 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。

(オプション) ステップ 4: SSE-KMS で暗号化されたバケットへのエクスポート

このステップは、 でサーバー側の暗号化を使用する S3 バケットにエクスポートする場合にのみ必要です AWS KMS keys。この暗号化は SSE-KMS と呼ばれます。

SSE-KMS で暗号化されたバケットにエクスポートするには

1. https://console.aws.amazon.com/kms で AWS KMS コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクタを使用します。
3. 左のナビゲーションバーで、[Customer managed keys] (カスタマーマネージドキー) を選択します。

   [Create Key] (キーを作成) を選択します。
4. [キーの種類] で、[対称] を選択します。
5. [Key usage] (キーの使用) で、[Encrypt and decrypt] (暗号化および復号化) 、[Next] (次へ) の順に選択します。
6. [Add labels] (ラベルを追加) で、キーのエイリアスを入力し、オプションで説明またはタグを追加します。次いで、[次へ] を選択します。
7. [Key administrators] (キー管理者) で、このキーを管理できるユーザーを選択した後、[Next] (次へ) を選択します。
8. [Define key usage permissions] (キーの使用アクセス許可を定義) の設定は変更せずに、[Next] (次へ) を選択します。
9. 設定した内容を確認し、[Finish] (終了) を選択します。
10. [Customer managed keys] (カスタマーマネージドキー) ページに戻り、この前に作成したキーの名前を選択します。
11. [Key policy] (キーポリシー) タブを表示し、次に [Switch to policy view] (ポリシービューへの切り替え) を選択します。
12. [Key policy] (キーポリシー) セクションで、[Edit] (編集) を選択します。
13. キーポリシーステートメントのリストに、次のステートメントを追加します。これを行う際、*Region* は実際のログのリージョンに置き換え、*account-ARN* は KMS キーを所有するアカウントの ARN に置き換えます。

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

```
            "Sid": "Allow CWL Service Principal usage",
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.Region.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "account-ARN"
            },
            "Action": [
                "kms:GetKeyPolicy*",
                "kms:PutKeyPolicy*",
                "kms:DescribeKey*",
                "kms:CreateAlias*",
                "kms:ScheduleKeyDeletion*",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM Role Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS":
 "arn:aws:iam::create_export_task_caller_account:role/role_name"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "ARN_OF_KMS_KEY"
        }
    ]
}
```

14. [変更を保存] を選択します。

15. https://console.aws.amazon.com/s3/でAmazon S3 コンソールを開きます。
16. ステップ 1: S3 バケットを作成する。 で作成したバケットを検索し、その名前を選択します。
17. [プロパティ] タブを選択します。[Default Encryption] (デフォルトの暗号化) で、[Edit] (編集) を選択します。
18. [Server-side Encryption] (サーバー側の暗号化) で、[Enable] (有効化) を選択します。
19. 暗号キーで、AWS Key Management Service キー (SSE-KMS) を選択します。
20. AWS KMS キーから選択を選択し、作成したキーを見つけます。
21. [Bucket key] (バケットキー) で、[Enable] (有効化) を選択します。
22. [変更を保存] を選択します。

ステップ 5: 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: アクセス許可を設定する に記載されているように、十分なアクセス許可を使用してサインインします。
2. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
3. ナビゲーションペインで、[ロググループ] を選択します。
4. [ロググループ] 画面で、ロググループの名前を選択します。
5. [Actions (アクション)]、[Export data to Amazon S3 (データを Amazon S3 にエクスポート)] の順に選択します。
6. [Export data to Amazon S3 (データを Amazon S3 にエクスポート)] 画面の [Define data export (データエクスポートを定義)] で、[From (開始)] と [To (終了)] 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。
7. 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。
8. [Choose S3 bucket] (S3 バケットの選択) で、S3 バケットに関連付けられたアカウントを選択します。
9. [S3 bucket name] (S3 バケット名) で、 バケットを選択します。

10. [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。
11. [Export (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。
12. 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、[Actions (アクション)]、[View all exports to Amazon S3 (Amazon S3 へのすべてのエクスポートを表示)] を選択します。

を使用してログデータを Amazon S3 にエクスポートする AWS CLI

次の例では、エクスポートタスクを使用して、 という名前の CloudWatch ロググループから という名前の Amazon S3 バケットmy-log-groupにすべてのデータをエクスポートしますmy-exported-logs。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。

で暗号化された S3 バケットへのログデータのエクスポート AWS KMS がサポートされています。DSSE-KMS で暗号化されたバケットへのエクスポートはサポートされていません。

エクスポートの設定方法の詳細は、エクスポート先の Amazon S3 バケットがエクスポート対象のログと同じアカウントにあるか、別のアカウントにあるかによって異なります。

トピック

- 同一アカウントへのエクスポート
- クロスアカウントでのエクスポート

同一アカウントへのエクスポート

Amazon S3 バケットがエクスポート対象のログと同じアカウントにある場合は、このセクションの手順を使用してください。

トピック

- ステップ 1: S3 バケットを作成する。
- ステップ 2: アクセス許可を設定する
- ステップ 3: S3 バケットのアクセス許可を設定する
- (オプション) ステップ 4: SSE-KMS で暗号化されたバケットへのエクスポート

- ステップ 5: エクスポートタスクを作成する

ステップ 1: S3 バケットを作成する。

CloudWatch Logs 専用に作成され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**
>
> 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する必要があります。CloudWatch Logs は、別のリージョンの 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。

を使用して S3 バケットを作成するには AWS CLI

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
```

以下は出力例です。

```
{
    "Location": "/my-exported-logs"
}
```

ステップ 2: アクセス許可を設定する

ステップ 5 でエクスポートタスクを作成するには、AmazonS3ReadOnlyAccess IAM ロールと以下のアクセス許可でサインオンする必要があります。

- logs:CreateExportTask
- logs:CancelExportTask
- logs:DescribeExportTasks
- logs:DescribeLogStreams

- `logs:DescribeLogGroups`

アクセス権限を付与するには、ユーザー、グループ、またはロールにアクセス許可を追加します。

- のユーザーとグループ AWS IAM Identity Center :

  アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」の手順に従ってください。
- IAM 内で、ID プロバイダーによって管理されているユーザー:

  ID フェデレーションのロールを作成します。詳細については、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールの作成」を参照してください。
- IAM ユーザー:
  - ユーザーが担当できるロールを作成します。手順については、「IAM ユーザーガイド」の「IAM ユーザー用ロールの作成」を参照してください。
  - (お奨めできない方法) ポリシーをユーザーに直接アタッチするか、ユーザーをユーザーグループに追加する。詳細については、「IAM ユーザーガイド」の「ユーザー (コンソール) へのアクセス権限の追加」を参照してください。

ステップ 3: S3 バケットのアクセス許可を設定する

すべての S3 バケットとオブジェクトは、デフォルト状態でプライベートに設定されます。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。

> **Important**
>
> S3 バケットへのエクスポートをより安全にするために、ログデータを S3 バケットにエクスポートできるソースアカウントのリストの指定が必要になりました。
> 次の例では、`aws:SourceAccount`キー内のアカウント IDs のリストは、ユーザーがログデータを S3 バケットにエクスポートできるアカウントになります。`aws:SourceArn` キーは、アクションが実行される対象のリソースです。これを特定のロググループに制限することも、この例のようにワイルドカードを使用することもできます。
> S3 バケットが作成されたアカウントのアカウント ID も含めることで、エクスポートを同じアカウント内で行えるようにすることをお勧めします。

S3 バケットでアクセス許可を設定するには

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、my-exported-logs 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 (us-west-1 など) 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 コンソールを使用しないでください。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs",
          "Principal": { "Service": "logs.Region.amazonaws.com" },
          "Condition": {
            "StringEquals": {
                "aws:SourceAccount": [
                    "AccountId1",
                    "AccountId2",
                    ...
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:Region:AccountId1:log-group:*",
                        "arn:aws:logs:Region:AccountId2:log-group:*",
                        ...
                     ]
            }
          }
      },
      {
          "Action": "s3:PutObject" ,
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs/*",
          "Principal": { "Service": "logs.Region.amazonaws.com" },
          "Condition": {
            "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control",
                "aws:SourceAccount": [
                    "AccountId1",
                    "AccountId2",
```

```
                    ...
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:Region:AccountId1:log-group:*",
                        "arn:aws:logs:Region:AccountId2:log-group:*",
                        ...
                    ]
            }
          }
      }
    ]
}
```

2. put-bucket-policy コマンドを使用して、バケットに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 ログはログデータを S3 バケットにエクスポートできます。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**
>
> 既存のバケットにすでに 1 つ以上のポリシーがアタッチされている場合は、そのポリシーへの 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。

(オプション) ステップ 4: SSE-KMS で暗号化されたバケットへのエクスポート

このステップは、 でサーバー側の暗号化を使用する S3 バケットにエクスポートする場合にのみ必要です AWS KMS keys。この暗号化は SSE-KMS と呼ばれます。

SSE-KMS で暗号化されたバケットにエクスポートするには

1. テキストエディタを使用して key_policy.json という名前のファイルを作成し、以下のアクセスポリシーを追加します。ポリシーを追加する際、以下の点を変更します。

- *Region* を、実際のログのリージョンに置き換えます。
- *account-ARN* を、KMS キーを所有するアカウントの ARN に置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow CWL Service Principal usage",
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.Region.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "account-ARN"
            },
            "Action": [
                "kms:GetKeyPolicy*",
                "kms:PutKeyPolicy*",
                "kms:DescribeKey*",
                "kms:CreateAlias*",
                "kms:ScheduleKeyDeletion*",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

2. 次のコマンドを入力します。

```
aws kms create-key --policy file://key_policy.json
```

以下は、このコマンドに対する出力例です。

```
{
    "KeyMetadata": {
        "AWSAccountId": "account_id",
        "KeyId": "key_id",
        "Arn": "arn:aws:kms:us-east-2:account_id:key/key_id",
        "CreationDate": "time",
        "Enabled": true,
        "Description": "",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": false
    }
```

3. テキストエディタを使用して、bucketencryption.json という名前のファイルを作成し、次の内容を記述します。

```
{
  "Rules": [
    {
      "ApplyServerSideEncryptionByDefault": {
        "SSEAlgorithm": "aws:kms",
        "KMSMasterKeyID": "{KMS Key ARN}"
      },
      "BucketKeyEnabled": true
    }
  ]
}
```

4. 次のコマンドを実行します。その際、*bucket-name* を、ログをエクスポートするバケットの名前に置き換えます。

```
aws s3api put-bucket-encryption --bucket bucket-name --server-side-encryption-
configuration file://bucketencryption.json
```

コマンドがエラーを返さなければ、このプロセスは成功しています。

ステップ 5: エクスポートタスクを作成する

次の コマンドを使用してエクスポートタスクを作成します。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。

を使用して Amazon S3 にデータをエクスポートするには AWS CLI

1. ステップ 2: アクセス許可を設定する に記載されているように、十分なアクセス許可を使用してサインインします。
2. 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"
```

以下は出力例です。

```
{
    "taskId": "cda45419-90ea-4db5-9833-aade86253e66"
}
```

クロスアカウントでのエクスポート

Amazon S3 バケットがエクスポート対象のログとは別のアカウントにある場合は、このセクションの手順を使用してください。

トピック

- ステップ 1: S3 バケットを作成する。
- ステップ 2: アクセス許可を設定する

- ステップ 3: S3 バケットのアクセス許可を設定する
- (オプション) ステップ 4: SSE-KMS で暗号化されたバケットへのエクスポート
- ステップ 5: エクスポートタスクを作成する

ステップ 1: S3 バケットを作成する。

CloudWatch Logs 専用に作成され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**
>
> 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する必要があります。CloudWatch Logs は、別のリージョンの 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。

を使用して S3 バケットを作成するには AWS CLI

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
```

以下は出力例です。

```
{
    "Location": "/my-exported-logs"
}
```

ステップ 2: アクセス許可を設定する

まず、新しい IAM ポリシーを作成して、 CloudWatch ログが送信先 Amazon S3 バケットに対するアクセスs3:PutObject許可を持つようにする必要があります。

ステップ 5 でエクスポートタスクを作成するには、AmazonS3ReadOnlyAccess IAM ロールとその他の特定のアクセス許可でサインオンする必要があります。その他の必要なアクセス許可の一部を含むポリシーを作成できます。

作成するポリシーは、レプリケート先バケットが AWS KMS 暗号化を使用するかどうかによって異なります。 AWS KMS 暗号化を使用しない場合は、次の内容のポリシーを作成します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-exported-logs/*"
        }
    ]
}
```

レプリケート先バケットが AWS KMS 暗号化を使用している場合は、次の内容のポリシーを作成します。

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-exported-logs/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "ARN_OF_KMS_KEY"
        }
    ]
}
```

ステップ 5 でエクスポートタスクを作成するには、AmazonS3ReadOnlyAccess IAM ロール、先ほど作成した IAM ポリシー、および次のアクセス許可でサインオンする必要があります。

- logs:CreateExportTask
- logs:CancelExportTask
- logs:DescribeExportTasks

- `logs:DescribeLogStreams`
- `logs:DescribeLogGroups`

アクセス権限を付与するには、ユーザー、グループ、またはロールにアクセス許可を追加します。

- のユーザーとグループ AWS IAM Identity Center :

  アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」の手順に従ってください。

- IAM 内で、ID プロバイダーによって管理されているユーザー:

  ID フェデレーションのロールを作成します。詳細については、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールの作成」を参照してください。

- IAM ユーザー:
  - ユーザーが担当できるロールを作成します。手順については、「IAM ユーザーガイド」の「IAM ユーザー用ロールの作成」を参照してください。
  - (お奨めできない方法) ポリシーをユーザーに直接アタッチするか、ユーザーをユーザーグループに追加する。詳細については、「IAM ユーザーガイド」の「ユーザー (コンソール) へのアクセス権限の追加」を参照してください。

ステップ 3: S3 バケットのアクセス許可を設定する

すべての S3 バケットとオブジェクトは、デフォルト状態でプライベートに設定されます。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。

> **Important**
>
> S3 バケットへのエクスポートをより安全にするために、ログデータを S3 バケットにエクスポートできるソースアカウントのリストの指定が必要になりました。
> 次の例では、`aws:SourceAccount`キー内のアカウント IDs のリストは、ユーザーがログデータを S3 バケットにエクスポートできるアカウントになります。`aws:SourceArn` キーは、アクションが実行される対象のリソースです。これを特定のロググループに制限することも、この例のようにワイルドカードを使用することもできます。

S3 バケットが作成されたアカウントのアカウント ID も含めることで、エクスポートを同じアカウント内で行えるようにすることをお勧めします。

S3 バケットでアクセス許可を設定するには

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、my-exported-logs 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 (us-west-1 など) 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 コンソールを使用しないでください。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
          "Action": "s3:GetBucketAcl",
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs",
          "Principal": { "Service": "logs.Region.amazonaws.com" },
          "Condition": {
            "StringEquals": {
                "aws:SourceAccount": [
                    "AccountId1",
                    "AccountId2",
                    ...
                ]
            },
            "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:Region:AccountId1:log-group:*",
                        "arn:aws:logs:Region:AccountId2:log-group:*",
                        ...
                     ]
            }
          }
      },
      {
          "Action": "s3:PutObject" ,
          "Effect": "Allow",
          "Resource": "arn:aws:s3:::my-exported-logs/*",
          "Principal": { "Service": "logs.Region.amazonaws.com" },
```

```
            "Condition": {
              "StringEquals": {
                  "s3:x-amz-acl": "bucket-owner-full-control",
                  "aws:SourceAccount": [
                      "AccountId1",
                      "AccountId2",
                      ...
                  ]
              },
              "ArnLike": {
                      "aws:SourceArn": [
                          "arn:aws:logs:Region:AccountId1:log-group:*",
                          "arn:aws:logs:Region:AccountId2:log-group:*",
                          ...
                      ]
              }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
              "AWS": "arn:aws:iam::create_export_task_caller_account:role/role_name"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-exported-logs/*",
            "Condition": {
              "StringEquals": {
                  "s3:x-amz-acl": "bucket-owner-full-control"
              }
            }
        }
    ]
}
```

2. put-bucket-policy コマンドを使用して、バケットに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 ログはログデータを S3 バケットにエクスポートできます。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**
>
> 既存のバケットにすでに 1 つ以上のポリシーがアタッチされている場合は、そのポリシーへの 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。

(オプション) ステップ 4: SSE-KMS で暗号化されたバケットへのエクスポート

このステップは、 でサーバー側の暗号化を使用する S3 バケットにエクスポートする場合にのみ必要です AWS KMS keys。この暗号化は SSE-KMS と呼ばれます。

SSE-KMS で暗号化されたバケットにエクスポートするには

1. テキストエディタを使用して key_policy.json という名前のファイルを作成し、以下のアクセスポリシーを追加します。ポリシーを追加する際、以下の点を変更します。

   - *Region* を、実際のログのリージョンに置き換えます。
   - *account-ARN* を、KMS キーを所有するアカウントの ARN に置き換えます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow CWL Service Principal usage",
            "Effect": "Allow",
            "Principal": {
                "Service": "logs.Region.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "account-ARN"
            },
            "Action": [
                "kms:GetKeyPolicy*",
                "kms:PutKeyPolicy*",
                "kms:DescribeKey*",
                "kms:CreateAlias*",
                "kms:ScheduleKeyDeletion*",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Enable IAM Role Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS":
 "arn:aws:iam::create_export_task_caller_account:role/role_name"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "ARN_OF_KMS_KEY"
        }
    ]
}
```

2. 次のコマンドを入力します。

```
aws kms create-key --policy file://key_policy.json
```

以下は、このコマンドに対する出力例です。

```
{
    "KeyMetadata": {
        "AWSAccountId": "account_id",
        "KeyId": "key_id",
        "Arn": "arn:aws:kms:us-east-2:account_id:key/key_id",
        "CreationDate": "time",
        "Enabled": true,
        "Description": "",
```

```
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": false
    }
```

3. テキストエディタを使用して、bucketencryption.json という名前のファイルを作成し、次の内容を記述します。

```
{
  "Rules": [
    {
      "ApplyServerSideEncryptionByDefault": {
        "SSEAlgorithm": "aws:kms",
        "KMSMasterKeyID": "{KMS Key ARN}"
      },
      "BucketKeyEnabled": true
    }
  ]
}
```

4. 次のコマンドを実行します。その際、*bucket-name* を、ログをエクスポートするバケットの名前に置き換えます。

```
aws s3api put-bucket-encryption --bucket bucket-name --server-side-encryption-
configuration file://bucketencryption.json
```

コマンドがエラーを返さなければ、このプロセスは成功しています。

ステップ 5: エクスポートタスクを作成する

次の コマンドを使用してエクスポートタスクを作成します。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。

を使用して Amazon S3 にデータをエクスポートするには AWS CLI

1. ステップ 2: アクセス許可を設定する に記載されているように、十分なアクセス許可を使用してサインインします。
2. 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"
```

以下は出力例です。

```
{
    "taskId": "cda45419-90ea-4db5-9833-aade86253e66"
}
```

エクスポートタスクの記述

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。

を使用してエクスポートタスクを記述するには AWS CLI

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"
```

以下は出力例です。

```
{
   "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 でフィルタリング – 指定された ID のエクスポートタスクが存在する場合に、それらがリストされます。
- タスクステータスによるフィルタリング – 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。

例えば、FAILED ステータスでフィルタリングするには、次のコマンドを使用します。

```
aws logs --profile CWLExportUser describe-export-tasks --status-code "FAILED"
```

以下は出力例です。

```
{
   "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
   }]
}
```

エクスポートタスクのキャンセル

エクスポートタスクが PENDING または RUNNING の状態にある場合、そのタスクをキャンセルできます。

を使用してエクスポートタスクをキャンセルするには AWS CLI

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ます。

Amazon OpenSearch Service への CloudWatch ログデータのストリーミング

CloudWatch Logs サブスクリプションを使用して、受信したデータをほぼリアルタイムで Amazon OpenSearch Service クラスターにストリーミングするように Logs CloudWatch ロググループを設定できます。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」を参照してください。

> **Note**
>
> OpenSearch サービスへのストリーミングは、標準ログクラスのロググループでのみサポートされます。ログクラスの詳細については、「」を参照してくださいログクラス。

ストリーミングされるログデータの量によっては、関数に対して関数レベルの同時実行数の制限を設定することもできます。詳細については、「Lambda 関数のスケーリング」を参照してください。

> **Note**
>
> 大量の CloudWatch Logs データを OpenSearch サービスにストリーミングすると、使用料が高くなる可能性があります。 AWS Billing and Cost Management コンソールで Budget を作成することをお勧めします。詳細については、「AWS Budgets によるコスト管理」を参照してください。

前提条件

開始する前に、 OpenSearch サービスドメインを作成します。ドメインにはパブリックアクセスまたは VPC アクセスのいずれかを設定できますが、その場合、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 OpenSearch サービスドメインの設定を確認し、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もできます。ドメインを作成する手順については、 OpenSearch 「サービスドメインの作成」を参照してください。

OpenSearch サービスの詳細については、「Amazon OpenSearch Service デベロッパーガイド」を参照してください。

ロググループを OpenSearch サービスにサブスクライブする

CloudWatch コンソールを使用して、ロググループを OpenSearch Service にサブスクライブできます。

ロググループを OpenSearch Service にサブスクライブするには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、[ロググループ] を選択します。
3. ロググループの名前を選択します。
4. アクション、サブスクリプションフィルター、Amazon OpenSearch Service サブスクリプションフィルターの作成 を選択します。
5. このアカウントのクラスターにストリーミングするか、別のアカウントにストリーミングするかを選択します。
   - このアカウントを選択した場合は、前のステップで作成したドメインを選択します。
   - 別のアカウントを選択した場合は、ドメイン ARN とエンドポイントを指定します。
6. Lambda IAM 実行ロール では、 への呼び出しを実行するときに Lambda が使用する IAM ロールを選択します OpenSearch。

   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 OpenSearch サービスドメインが 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。この Amazon 管理ポリシーは、Lambda にお客様の VPC へのアクセスを許可し、Lambda が VPC 内の OpenSearch エンドポイントに書き込むことを可能にします。

7. [Log format] (ログの形式) で、ログの形式を選択します。
8. [Subscription filter pattern] (サブスクリプションフィルターのパターン) 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、関心のあるデータのみを OpenSearch クラスターに送信できます。詳細については、「フィルターを使用したログイベントからのメトリクスの作成」を参照してください。
9. (オプション) [Select log data to test] (テストするログデータの選択) を開き、ログストリームを選択してから、[Test pattern] (パターンをテスト) を選択して、期待通りの結果が出ることを確認します。
10. [Start streaming] (ストリーミングの開始) を選択します。

AWS SDKs を使用した CloudWatch ログのコード例

次のコード例は、 AWS Software Development Kit (SDK) で CloudWatch Logs を使用する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

クロスサービスの例は、複数の AWS のサービスで動作するサンプルアプリケーションです。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

コードの例

- AWS SDKs を使用した CloudWatch ログのアクション
  - AWS SDK または CLI AssociateKmsKeyで を使用する
  - AWS SDK または CLI CancelExportTaskで を使用する
  - AWS SDK または CLI CreateExportTaskで を使用する
  - AWS SDK または CLI CreateLogGroupで を使用する
  - AWS SDK または CLI CreateLogStreamで を使用する
  - AWS SDK または CLI DeleteLogGroupで を使用する
  - AWS SDK または CLI DeleteSubscriptionFilterで を使用する
  - AWS SDK または CLI DescribeExportTasksで を使用する
  - AWS SDK または CLI DescribeLogGroupsで を使用する
  - AWS SDK または CLI DescribeSubscriptionFiltersで を使用する
  - AWS SDK または CLI GetQueryResultsで を使用する
  - AWS SDK または CLI PutSubscriptionFilterで を使用する
  - AWS SDK または CLI StartLiveTailで を使用する
  - AWS SDK または CLI StartQueryで を使用する
- AWS SDKs を使用した CloudWatch ログのシナリオ

AWS SDKs を使用した CloudWatch ログのアクション

次のコード例は、 AWS SDKs で個々の CloudWatch Logs アクションを実行する方法を示しています。これらの抜粋は CloudWatch Logs API を呼び出し、コンテキスト内で実行する必要がある大規模なプログラムからのコードの抜粋です。各例には へのリンクが含まれており GitHub、コードの設定と実行の手順を確認できます。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細なリストについては、「Amazon CloudWatch Logs API リファレンス」を参照してください。

例

- AWS SDK または CLI AssociateKmsKeyで を使用する
- AWS SDK または CLI CancelExportTaskで を使用する
- AWS SDK または CLI CreateExportTaskで を使用する
- AWS SDK または CLI CreateLogGroupで を使用する
- AWS SDK または CLI CreateLogStreamで を使用する
- AWS SDK または CLI DeleteLogGroupで を使用する
- AWS SDK または CLI DeleteSubscriptionFilterで を使用する
- AWS SDK または CLI DescribeExportTasksで を使用する
- AWS SDK または CLI DescribeLogGroupsで を使用する
- AWS SDK または CLI DescribeSubscriptionFiltersで を使用する
- AWS SDK または CLI GetQueryResultsで を使用する
- AWS SDK または CLI PutSubscriptionFilterで を使用する
- AWS SDK または CLI StartLiveTailで を使用する
- AWS SDK または CLI StartQueryで を使用する

AWS SDK または CLI **`AssociateKmsKey`**で を使用する

次の例は、`AssociateKmsKey` を使用する方法を説明しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;

    /// <summary>
    /// Shows how to associate an AWS Key Management Service (AWS KMS) key with
    /// an Amazon CloudWatch Logs log group.
    /// </summary>
    public class AssociateKmsKey
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();

            string kmsKeyId = "arn:aws:kms:us-west-2:<account-
number>:key/7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";
            string groupName = "cloudwatchlogs-example-loggroup";

            var request = new AssociateKmsKeyRequest
            {
                KmsKeyId = kmsKeyId,
                LogGroupName = groupName,
            };

            var response = await client.AssociateKmsKeyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
```

```
            {
                Console.WriteLine($"Successfully associated KMS key ID:
{kmsKeyId} with log group: {groupName}.");
            }
            else
            {
                Console.WriteLine("Could not make the association between:
{kmsKeyId} and {groupName}.");
            }
        }
    }
```

- API の詳細については、「 API リファレンスAssociateKmsKey」の「」を参照してください。 AWS SDK for .NET

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CancelExportTask**で を使用する

次の例は、CancelExportTask を使用する方法を説明しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
```

```
    /// <summary>
    /// Shows how to cancel an Amazon CloudWatch Logs export task.
    /// </summary>
    public class CancelExportTask
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();
            string taskId = "exampleTaskId";

            var request = new CancelExportTaskRequest
            {
                TaskId = taskId,
            };

            var response = await client.CancelExportTaskAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{taskId} successfully canceled.");
            }
            else
            {
                Console.WriteLine($"{taskId} could not be canceled.");
            }
        }
    }
```

- API の詳細については、「 API リファレンスCancelExportTask」の「」を参照してください。 AWS SDK for .NET

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CreateExportTask**で を使用する

次の例は、CreateExportTask を使用する方法を説明しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;

    /// <summary>
    /// Shows how to create an Export Task to export the contents of the Amazon
    /// CloudWatch Logs to the specified Amazon Simple Storage Service (Amazon
S3)
    /// bucket.
    /// </summary>
    public class CreateExportTask
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();
            string taskName = "export-task-example";
            string logGroupName = "cloudwatchlogs-example-loggroup";
            string destination = "doc-example-bucket";
            var fromTime = 1437584472382;
            var toTime = 1437584472833;

            var request = new CreateExportTaskRequest
            {
```

```
                From = fromTime,
                To = toTime,
                TaskName = taskName,
                LogGroupName = logGroupName,
                Destination = destination,
            };

            var response = await client.CreateExportTaskAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"The task, {taskName} with ID: " +
                                  $"{response.TaskId} has been created
successfully.");
            }
        }
    }
```

- API の詳細については、「 API リファレンスCreateExportTask」の「」を参照してください。 AWS SDK for .NET

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CreateLogGroup**で を使用する

以下のコード例は、CreateLogGroup の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;

    /// <summary>
    /// Shows how to create an Amazon CloudWatch Logs log group.
    /// </summary>
    public class CreateLogGroup
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();

            string logGroupName = "cloudwatchlogs-example-loggroup";

            var request = new CreateLogGroupRequest
            {
                LogGroupName = logGroupName,
            };

            var response = await client.CreateLogGroupAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully create log group with ID:
{logGroupName}.");
            }
            else
            {
                Console.WriteLine("Could not create log group.");
            }
        }
    }
```

- API の詳細については、「 API リファレンスCreateLogGroup」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

次のコマンドは、my-logs という名前のロググループを作成します。

```
aws logs create-log-group --log-group-name my-logs
```

- API の詳細については、「 コマンドリファレンスCreateLogGroup」の「」を参照してください。 AWS CLI

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import { CreateLogGroupCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new CreateLogGroupCommand({
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- API の詳細については、「 API リファレンスCreateLogGroup」の「」を参照してください。 AWS SDK for JavaScript

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CreateLogStream**で を使用する

以下のコード例は、CreateLogStream の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to create an Amazon CloudWatch Logs stream for a CloudWatch
/// log group.
/// </summary>
public class CreateLogStream
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();
        string logGroupName = "cloudwatchlogs-example-loggroup";
        string logStreamName = "cloudwatchlogs-example-logstream";
```

```
            var request = new CreateLogStreamRequest
            {
                LogGroupName = logGroupName,
                LogStreamName = logStreamName,
            };

            var response = await client.CreateLogStreamAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{logStreamName} successfully created for
{logGroupName}.");
            }
            else
            {
                Console.WriteLine("Could not create stream.");
            }
        }
    }
```

- API の詳細については、「 API リファレンスCreateLogStream」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

次のコマンドは、ロググループ my-logs に 20150601 という名前のログストリームを作成します。

```
aws logs create-log-stream --log-group-name my-logs --log-stream-name 20150601
```

- API の詳細については、「 コマンドリファレンスCreateLogStream」の「」を参照してください。 AWS CLI

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteLogGroup**で を使用する

以下のコード例は、DeleteLogGroup の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Uses the Amazon CloudWatch Logs Service to delete an existing
/// CloudWatch Logs log group.
/// </summary>
public class DeleteLogGroup
{
    public static async Task Main()
    {
        var client = new AmazonCloudWatchLogsClient();
        string logGroupName = "cloudwatchlogs-example-loggroup";

        var request = new DeleteLogGroupRequest
        {
            LogGroupName = logGroupName,
        };

        var response = await client.DeleteLogGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully deleted CloudWatch log group,
{logGroupName}.");
        }
```

```
        }
    }
```

- API の詳細については、「 API リファレンスDeleteLogGroup」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

以下のコマンドは、my-logs という名前のロググループを削除します。

```
aws logs delete-log-group --log-group-name my-logs
```

- API の詳細については、「 コマンドリファレンスDeleteLogGroup」の「」を参照してください。 AWS CLI

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import { DeleteLogGroupCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteLogGroupCommand({
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
```

```
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- API の詳細については、「 API リファレンスDeleteLogGroup」の「」を参照してください。 AWS SDK for JavaScript

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteSubscriptionFilter**で を使用する

以下のコード例は、DeleteSubscriptionFilter の使用方法を示しています。

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DeleteSubscriptionFilterRequest.h>
#include <iostream>
```

サブスクリプションフィルターを削除します。

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::DeleteSubscriptionFilterRequest request;
        request.SetFilterName(filter_name);
        request.SetLogGroupName(log_group);

        auto outcome = cwl.DeleteSubscriptionFilter(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Failed to delete CloudWatch log subscription filter "
                << filter_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        } else {
            std::cout << "Successfully deleted CloudWatch logs subscription " <<
                "filter " << filter_name << std::endl;
        }
```

- API の詳細については、「 API リファレンスDeleteSubscriptionFilter」の「」を参照してください。 AWS SDK for C++

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DeleteSubscriptionFilterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <filter> <logGroup>

                Where:
                  filter - The name of the subscription filter (for example,
 MyFilter).
                  logGroup - The name of the log group. (for example, testgroup).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String logGroup = args[1];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
                .build();

        deleteSubFilter(logs, filter, logGroup);
        logs.close();
    }

    public static void deleteSubFilter(CloudWatchLogsClient logs, String filter,
 String logGroup) {
        try {
            DeleteSubscriptionFilterRequest request =
 DeleteSubscriptionFilterRequest.builder()
                    .filterName(filter)
                    .logGroupName(logGroup)
                    .build();

            logs.deleteSubscriptionFilter(request);
            System.out.printf("Successfully deleted CloudWatch logs subscription
 filter %s", filter);

        } catch (CloudWatchException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「 API リファレンスDeleteSubscriptionFilter」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import { DeleteSubscriptionFilterCommand } from "@aws-sdk/client-cloudwatch-
logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteSubscriptionFilterCommand({
    // The name of the filter.
    filterName: process.env.CLOUDWATCH_LOGS_FILTER_NAME,
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- API の詳細については、「 API リファレンスDeleteSubscriptionFilter」の「」を参照してください。 AWS SDK for JavaScript

SDK for JavaScript (v2)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  filterName: "FILTER",
  logGroupName: "LOG_GROUP",
};

cwl.deleteSubscriptionFilter(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスDeleteSubscriptionFilter」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun deleteSubFilter(
    filter: String?,
    logGroup: String?,
) {
    val request =
        DeleteSubscriptionFilterRequest {
            filterName = filter
            logGroupName = logGroup
        }

    CloudWatchLogsClient { region = "us-west-2" }.use { logs ->
        logs.deleteSubscriptionFilter(request)
        println("Successfully deleted CloudWatch logs subscription filter named
 $filter")
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスDeleteSubscriptionFilterの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeExportTasks**で を使用する

次の例は、DescribeExportTasks を使用する方法を説明しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to retrieve a list of information about Amazon CloudWatch
/// Logs export tasks.
/// </summary>
public class DescribeExportTasks
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();

        var request = new DescribeExportTasksRequest
        {
            Limit = 5,
        };

        var response = new DescribeExportTasksResponse();

        do
        {
            response = await client.DescribeExportTasksAsync(request);
            response.ExportTasks.ForEach(t =>
            {
```

```
                    Console.WriteLine($"{t.TaskName} with ID: {t.TaskId} has
status: {t.Status}");
                });
            }
            while (response.NextToken is not null);
        }
    }
```

- API の詳細については、「 API リファレンスDescribeExportTasks」の「」を参照してください。 AWS SDK for .NET

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeLogGroups**で を使用する

以下のコード例は、DescribeLogGroups の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Retrieves information about existing Amazon CloudWatch Logs log groups
/// and displays the information on the console.
/// </summary>
public class DescribeLogGroups
```

```
    {
        public static async Task Main()
        {
            // Creates a CloudWatch Logs client using the default
            // user. If you need to work with resources in another
            // AWS Region than the one defined for the default user,
            // pass the AWS Region as a parameter to the client constructor.
            var client = new AmazonCloudWatchLogsClient();

            bool done = false;
            string newToken = null;

            var request = new DescribeLogGroupsRequest
            {
                Limit = 5,
            };

            DescribeLogGroupsResponse response;

            do
            {
                if (newToken is not null)
                {
                    request.NextToken = newToken;
                }

                response = await client.DescribeLogGroupsAsync(request);

                response.LogGroups.ForEach(lg =>
                {
                    Console.WriteLine($"{lg.LogGroupName} is associated with the
key: {lg.KmsKeyId}.");
                    Console.WriteLine($"Created on:
{lg.CreationTime.Date.Date}");
                    Console.WriteLine($"Date for this group will be stored for:
{lg.RetentionInDays} days.\n");
                });

                if (response.NextToken is null)
                {
                    done = true;
                }
                else
                {
```

```
                    newToken = response.NextToken;
                }
            }
            while (!done);
        }
    }
```

- API の詳細については、「 API リファレンスDescribeLogGroups」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

次のコマンドは、my-logs という名前のロググループを記述します。

```
aws logs describe-log-groups --log-group-name-prefix my-logs
```

出力:

```
{
    "logGroups": [
        {
            "storedBytes": 0,
            "metricFilterCount": 0,
            "creationTime": 1433189500783,
            "logGroupName": "my-logs",
            "retentionInDays": 5,
            "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:*"
        }
    ]
}
```

- API の詳細については、「 コマンドリファレンスDescribeLogGroups」の「」を参照してください。 AWS CLI

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import {
  paginateDescribeLogGroups,
  CloudWatchLogsClient,
} from "@aws-sdk/client-cloudwatch-logs";

const client = new CloudWatchLogsClient({});

export const main = async () => {
  const paginatedLogGroups = paginateDescribeLogGroups({ client }, {});
  const logGroups = [];

  for await (const page of paginatedLogGroups) {
    if (page.logGroups && page.logGroups.every((lg) => !!lg)) {
      logGroups.push(...page.logGroups);
    }
  }

  console.log(logGroups);
  return logGroups;
};
```

- API の詳細については、「 API リファレンスDescribeLogGroups」の「」を参照してください。 AWS SDK for JavaScript

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeSubscriptionFilters**で を使用する

以下のコード例は、DescribeSubscriptionFilters の使用方法を示しています。

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DescribeSubscriptionFiltersRequest.h>
#include <aws/logs/model/DescribeSubscriptionFiltersResult.h>
#include <iostream>
#include <iomanip>
```

サブスクリプションフィルターを一覧表示します。

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::DescribeSubscriptionFiltersRequest request;
        request.SetLogGroupName(log_group);
        request.SetLimit(1);

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = cwl.DescribeSubscriptionFilters(
                    request);
            if (!outcome.IsSuccess()) {
                std::cout << "Failed to describe CloudWatch subscription filters
 "
                    << "for log group " << log_group << ": " <<
```

```
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header) {
                std::cout << std::left << std::setw(32) << "Name" <<
                    std::setw(64) << "FilterPattern" << std::setw(64) <<
                    "DestinationArn" << std::endl;
                header = true;
            }

            const auto &filters = outcome.GetResult().GetSubscriptionFilters();
            for (const auto &filter : filters) {
                std::cout << std::left << std::setw(32) <<
                    filter.GetFilterName() << std::setw(64) <<
                    filter.GetFilterPattern() << std::setw(64) <<
                    filter.GetDestinationArn() << std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- API の詳細については、「 API リファレンスDescribeSubscriptionFilters」の「」を参照してください。 AWS SDK for C++

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
```

```
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersResponse
import software.amazon.awssdk.services.cloudwatchlogs.model.SubscriptionFilter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeSubscriptionFilters {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                  <logGroup>

                Where:
                  logGroup - A log group name (for example, myloggroup).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String logGroup = args[0];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        describeFilters(logs, logGroup);
        logs.close();
    }

    public static void describeFilters(CloudWatchLogsClient logs, String
 logGroup) {
        try {
```

```
            boolean done = false;
            String newToken = null;

            while (!done) {
                DescribeSubscriptionFiltersResponse response;
                if (newToken == null) {
                    DescribeSubscriptionFiltersRequest request =
 DescribeSubscriptionFiltersRequest.builder()
                            .logGroupName(logGroup)
                            .limit(1).build();

                    response = logs.describeSubscriptionFilters(request);
                } else {
                    DescribeSubscriptionFiltersRequest request =
 DescribeSubscriptionFiltersRequest.builder()
                            .nextToken(newToken)
                            .logGroupName(logGroup)
                            .limit(1).build();
                    response = logs.describeSubscriptionFilters(request);
                }

                for (SubscriptionFilter filter : response.subscriptionFilters())
 {
                    System.out.printf("Retrieved filter with name %s, " +
 "pattern %s " + "and destination arn %s",
                            filter.filterName(),
                            filter.filterPattern(),
                            filter.destinationArn());
                }

                if (response.nextToken() == null) {
                    done = true;
                } else {
                    newToken = response.nextToken();
                }
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.printf("Done");
    }
}
```

- API の詳細については、「 API リファレンスDescribeSubscriptionFilters」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import { DescribeSubscriptionFiltersCommand } from "@aws-sdk/client-cloudwatch-
logs";
import { client } from "../libs/client.js";

const run = async () => {
  // This will return a list of all subscription filters in your account
  // matching the log group name.
  const command = new DescribeSubscriptionFiltersCommand({
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
    limit: 1,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- API の詳細については、「 API リファレンスDescribeSubscriptionFilters」の「」を参照してください。 AWS SDK for JavaScript

SDK for JavaScript (v2)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  logGroupName: "GROUP_NAME",
  limit: 5,
};

cwl.describeSubscriptionFilters(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.subscriptionFilters);
  }
});
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスDescribeSubscriptionFilters」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeFilters(logGroup: String) {
    val request =
        DescribeSubscriptionFiltersRequest {
            logGroupName = logGroup
            limit = 1
        }

    CloudWatchLogsClient { region = "us-west-2" }.use { cwlClient ->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name  ${filter.filterName} pattern
 ${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスDescribeSubscriptionFiltersの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **GetQueryResults**で を使用する

以下のコード例は、GetQueryResults の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 大規模なクエリを実行する

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Simple wrapper for the GetQueryResultsCommand.
 * @param {string} queryId
 */
_getQueryResults(queryId) {
  return this.client.send(new GetQueryResultsCommand({ queryId }));
}
```

- API の詳細については、「 API リファレンスGetQueryResults」の「」を参照してください。 AWS SDK for JavaScript

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    def _wait_for_query_results(self, client, query_id):
        """
        Waits for the query to complete and retrieves the results.

        :param query_id: The ID of the initiated query.
```

```
        :type query_id: str
        :return: A list containing the results of the query.
        :rtype: list
        """
        while True:
            time.sleep(1)
            results = client.get_query_results(queryId=query_id)
            if results["status"] in [
                "Complete",
                "Failed",
                "Cancelled",
                "Timeout",
                "Unknown",
            ]:
                return results.get("results", [])
```

- API の詳細については、 GetQueryResults AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **PutSubscriptionFilter**で を使用する

以下のコード例は、PutSubscriptionFilter の使用方法を示しています。

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

```
#include <aws/core/Aws.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/PutSubscriptionFilterRequest.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

サブスクリプションフィルターを作成します。

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::PutSubscriptionFilterRequest request;
        request.SetFilterName(filter_name);
        request.SetFilterPattern(filter_pattern);
        request.SetLogGroupName(log_group);
        request.SetDestinationArn(dest_arn);
        auto outcome = cwl.PutSubscriptionFilter(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch logs subscription filter "
                << filter_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch logs subscription " <<
                "filter " << filter_name << std::endl;
        }
```

- API の詳細については、「 API リファレンスPutSubscriptionFilter」の「」を参照してください。 AWS SDK for C++

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.PutSubscriptionFilterRequest;

/**
 * Before running this code example, you need to grant permission to CloudWatch
 * Logs the right to execute your Lambda function.
 * To perform this task, you can use this CLI command:
 *
 * aws lambda add-permission --function-name "lamda1" --statement-id "lamda1"
 * --principal "logs.us-west-2.amazonaws.com" --action "lambda:InvokeFunction"
 * --source-arn "arn:aws:logs:us-west-2:111111111111:log-group:testgroup:*"
 * --source-account "111111111111"
 *
 * Make sure you replace the function name with your function name and replace
 * '111111111111' with your account details.
 * For more information, see "Subscription Filters with AWS Lambda" in the
 * Amazon CloudWatch Logs Guide.
 *
 *
 * Also, before running this Java V2 code example,set up your development
 * environment,including your credentials.
 *
 * For more information,see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 */

public class PutSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <filter> <pattern> <logGroup> <functionArn>\s

                Where:
                  filter - A filter name (for example, myfilter).
                  pattern - A filter pattern (for example, ERROR).
```

```
                logGroup - A log group name (testgroup).
                functionArn - An AWS Lambda function ARN (for example,
arn:aws:lambda:us-west-2:111111111111:function:lambda1) .
               """;

       if (args.length != 4) {
           System.out.println(usage);
           System.exit(1);
       }

       String filter = args[0];
       String pattern = args[1];
       String logGroup = args[2];
       String functionArn = args[3];
       Region region = Region.US_WEST_2;
       CloudWatchLogsClient cwl = CloudWatchLogsClient.builder()
               .region(region)
               .build();

       putSubFilters(cwl, filter, pattern, logGroup, functionArn);
       cwl.close();
   }

   public static void putSubFilters(CloudWatchLogsClient cwl,
           String filter,
           String pattern,
           String logGroup,
           String functionArn) {

       try {
           PutSubscriptionFilterRequest request =
PutSubscriptionFilterRequest.builder()
                   .filterName(filter)
                   .filterPattern(pattern)
                   .logGroupName(logGroup)
                   .destinationArn(functionArn)
                   .build();

           cwl.putSubscriptionFilter(request);
           System.out.printf(
                   "Successfully created CloudWatch logs subscription filter
%s",
                   filter);
```

```
        } catch (CloudWatchLogsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、「 API リファレンスPutSubscriptionFilter」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import { PutSubscriptionFilterCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new PutSubscriptionFilterCommand({
    // An ARN of a same-account Kinesis stream, Kinesis Firehose
    // delivery stream, or Lambda function.
    // https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/
SubscriptionFilters.html
    destinationArn: process.env.CLOUDWATCH_LOGS_DESTINATION_ARN,

    // A name for the filter.
    filterName: process.env.CLOUDWATCH_LOGS_FILTER_NAME,

    // A filter pattern for subscribing to a filtered stream of log events.
    // https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/
FilterAndPatternSyntax.html
    filterPattern: process.env.CLOUDWATCH_LOGS_FILTER_PATTERN,
```

```
    // The name of the log group. Messages in this group matching the filter
 pattern
    // will be sent to the destination ARN.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- API の詳細については、「 API リファレンスPutSubscriptionFilter」の「」を参照してください。 AWS SDK for JavaScript

SDK for JavaScript (v2)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コードサンプルリポジトリでの設定と実行の方法を確認してください。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  destinationArn: "LAMBDA_FUNCTION_ARN",
  filterName: "FILTER_NAME",
  filterPattern: "ERROR",
  logGroupName: "LOG_GROUP",
};

cwl.putSubscriptionFilter(params, function (err, data) {
```

```
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスPutSubscriptionFilter」の「」を参照してください。 AWS SDK for JavaScript

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **StartLiveTail**で を使用する

以下のコード例は、StartLiveTail の使用方法を示しています。

.NET

AWS SDK for .NET

必要なファイルを含めます。

```
using Amazon;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
```

Live Tail セッションを開始します。

```
            var client = new AmazonCloudWatchLogsClient();
            var request = new StartLiveTailRequest
            {
                LogGroupIdentifiers = logGroupIdentifiers,
                LogStreamNames = logStreamNames,
                LogEventFilterPattern = filterPattern,
            };
```

```
            var response = await client.StartLiveTailAsync(request);

            // Catch if request fails
            if (response.HttpStatusCode != System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine("Failed to start live tail session");
                return;
            }
```

Live Tail セッションのイベントは 2 つの方法で処理できます。

```
            /* Method 1
            * 1). Asynchronously loop through the event stream
            * 2). Set a timer to dispose the stream and stop the Live Tail
session at the end.
            */
            var eventStream = response.ResponseStream;
            var task = Task.Run(() =>
            {
                foreach (var item in eventStream)
                {
                    if (item is LiveTailSessionUpdate liveTailSessionUpdate)
                    {
                        foreach (var sessionResult in
liveTailSessionUpdate.SessionResults)
                        {
                            Console.WriteLine("Message : {0}",
sessionResult.Message);
                        }
                    }
                    if (item is LiveTailSessionStart)
                    {
                        Console.WriteLine("Live Tail session started");
                    }
                    // On-stream exceptions are processed here
                    if (item is CloudWatchLogsEventStreamException)
                    {
                        Console.WriteLine($"ERROR: {item}");
                    }
                }
            });
            // Close the stream to stop the session after a timeout
```

```
            if (!task.Wait(TimeSpan.FromSeconds(10))){
                eventStream.Dispose();
                Console.WriteLine("End of line");
            }
```

```
            /* Method 2
            * 1). Add event handlers to each event variable
            * 2). Start processing the stream and wait for a timeout using
AutoResetEvent
            */
            AutoResetEvent endEvent = new AutoResetEvent(false);
            var eventStream = response.ResponseStream;
            using (eventStream) // automatically disposes the stream to stop the
session after execution finishes
            {
                eventStream.SessionStartReceived += (sender, e) =>
                {
                    Console.WriteLine("LiveTail session started");
                };
                eventStream.SessionUpdateReceived += (sender, e) =>
                {
                    foreach (LiveTailSessionLogEvent logEvent in
e.EventStreamEvent.SessionResults){
                        Console.WriteLine("Message: {0}", logEvent.Message);
                    }
                };
                // On-stream exceptions are captured here
                eventStream.ExceptionReceived += (sender, e) =>
                {
                    Console.WriteLine($"ERROR:
{e.EventStreamException.Message}");
                };

                eventStream.StartProcessing();
                // Stream events for this amount of time.
                endEvent.WaitOne(TimeSpan.FromSeconds(10));
                Console.WriteLine("End of line");
            }
```

- API の詳細については、「 API リファレンスStartLiveTail」の「」を参照してください。 AWS SDK for .NET

Go

SDK for Go V2

必要なファイルを含めます。

```
import (
 "context"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)
```

Live Tail セッションのイベントを処理します。

```
func handleEventStreamAsync(stream *cloudwatchlogs.StartLiveTailEventStream) {
 eventsChan := stream.Events()
 for {
  event := <-eventsChan
  switch e := event.(type) {
  case *types.StartLiveTailResponseStreamMemberSessionStart:
   log.Println("Received SessionStart event")
  case *types.StartLiveTailResponseStreamMemberSessionUpdate:
   for _, logEvent := range e.Value.SessionResults {
    log.Println(*logEvent.Message)
   }
  default:
   // Handle on-stream exceptions
   if err := stream.Err(); err != nil {
    log.Fatalf("Error occured during streaming: %v", err)
   } else if event == nil {
    log.Println("Stream is Closed")
    return
   } else {
    log.Fatalf("Unknown event type: %T", e)
   }
  }
 }
}
```

Live Tail セッションを開始します。

```
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
 panic("configuration error, " + err.Error())
}
client := cloudwatchlogs.NewFromConfig(cfg)

request := &cloudwatchlogs.StartLiveTailInput{
 LogGroupIdentifiers:   logGroupIdentifiers,
 LogStreamNames:        logStreamNames,
 LogEventFilterPattern: logEventFilterPattern,
}

response, err := client.StartLiveTail(context.TODO(), request)
// Handle pre-stream Exceptions
if err != nil {
 log.Fatalf("Failed to start streaming: %v", err)
}

// Start a Goroutine to handle events over stream
stream := response.GetStream()
go handleEventStreamAsync(stream)
```

一定時間が経過したら Live Tail セッションを停止します。

```
// Close the stream (which ends the session) after a timeout
time.Sleep(10 * time.Second)
stream.Close()
log.Println("Event stream closed")
```

- API の詳細については、「 API リファレンスStartLiveTail」の「」を参照してください。 AWS SDK for Go

Java

SDK for Java 2.x

必要なファイルを含めます。

```
import io.reactivex.FlowableSubscriber;
import io.reactivex.annotations.NonNull;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsAsyncClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionStart;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionUpdate;
import software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseHandler;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseStream;

import java.util.Date;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;
```

Live Tail セッションのイベントを処理します。

```
    private static StartLiveTailResponseHandler
 getStartLiveTailResponseStreamHandler(
            AtomicReference<Subscription> subscriptionAtomicReference) {
        return StartLiveTailResponseHandler.builder()
            .onResponse(r -> System.out.println("Received initial response"))
            .onError(throwable -> {
                CloudWatchLogsException e = (CloudWatchLogsException)
 throwable.getCause();
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
            })
            .subscriber(() -> new FlowableSubscriber<>() {
```

```
                @Override
                public void onSubscribe(@NonNull Subscription s) {
                    subscriptionAtomicReference.set(s);
                    s.request(Long.MAX_VALUE);
                }

                @Override
                public void onNext(StartLiveTailResponseStream event) {
                    if (event instanceof LiveTailSessionStart) {
                        LiveTailSessionStart sessionStart =
(LiveTailSessionStart) event;
                        System.out.println(sessionStart);
                    } else if (event instanceof LiveTailSessionUpdate) {
                        LiveTailSessionUpdate sessionUpdate =
(LiveTailSessionUpdate) event;
                        List<LiveTailSessionLogEvent> logEvents =
sessionUpdate.sessionResults();
                        logEvents.forEach(e -> {
                            long timestamp = e.timestamp();
                            Date date = new Date(timestamp);
                            System.out.println("[" + date + "] " + e.message());
                        });
                    } else {
                        throw CloudWatchLogsException.builder().message("Unknown
event type").build();
                    }
                }

                @Override
                public void onError(Throwable throwable) {
                    System.out.println(throwable.getMessage());
                    System.exit(1);
                }

                @Override
                public void onComplete() {
                    System.out.println("Completed Streaming Session");
                }
            })
            .build();
    }
```

Live Tail セッションを開始します。

```
        CloudWatchLogsAsyncClient cloudWatchLogsAsyncClient =
                CloudWatchLogsAsyncClient.builder()
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();

        StartLiveTailRequest request =
                StartLiveTailRequest.builder()
                    .logGroupIdentifiers(logGroupIdentifiers)
                    .logStreamNames(logStreamNames)
                    .logEventFilterPattern(logEventFilterPattern)
                    .build();

        /* Create a reference to store the subscription */
        final AtomicReference<Subscription> subscriptionAtomicReference = new
AtomicReference<>(null);

        cloudWatchLogsAsyncClient.startLiveTail(request,
getStartLiveTailResponseStreamHandler(subscriptionAtomicReference));
```

一定時間が経過したら Live Tail セッションを停止します。

```
        /* Set a timeout for the session and cancel the subscription. This will:
         * 1). Close the stream
         * 2). Stop the Live Tail session
         */
        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        if (subscriptionAtomicReference.get() != null) {
            subscriptionAtomicReference.get().cancel();
            System.out.println("Subscription to stream closed");
        }
```

- API の詳細については、「 API リファレンスStartLiveTail」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

必要なファイルを含めます。

```
import { CloudWatchLogsClient, StartLiveTailCommand } from "@aws-sdk/client-
cloudwatch-logs";
```

Live Tail セッションのイベントを処理します。

```
async function handleResponseAsync(response) {
    try {
      for await (const event of response.responseStream) {
        if (event.sessionStart !== undefined) {
          console.log(event.sessionStart);
        } else if (event.sessionUpdate !== undefined) {
          for (const logEvent of event.sessionUpdate.sessionResults) {
            const timestamp = logEvent.timestamp;
            const date = new Date(timestamp);
            console.log("[" + date + "] " + logEvent.message);
          }
        } else {
            console.error("Unknown event type");
        }
      }
    } catch (err) {
        // On-stream exceptions are captured here
        console.error(err)
    }
}
```

Live Tail セッションを開始します。

```
    const client = new CloudWatchLogsClient();

    const command = new StartLiveTailCommand({
        logGroupIdentifiers: logGroupIdentifiers,
        logStreamNames: logStreamNames,
        logEventFilterPattern: filterPattern
    });
```

```
    try{
        const response = await client.send(command);
        handleResponseAsync(response);
    } catch (err){
        // Pre-stream exceptions are captured here
        console.log(err);
    }
```

一定時間が経過したら Live Tail セッションを停止します。

```
    /* Set a timeout to close the client. This will stop the Live Tail session.
*/
    setTimeout(function() {
        console.log("Client timeout");
        client.destroy();
      }, 10000);
```

- API の詳細については、「 API リファレンスStartLiveTail」の「」を参照してください。 AWS SDK for JavaScript

Kotlin

SDK for Kotlin

必要なファイルを含めます。

```
import aws.sdk.kotlin.services.cloudwatchlogs.CloudWatchLogsClient
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailRequest
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailResponseStream
import kotlinx.coroutines.flow.takeWhile
```

Live Tail セッションを開始します。

```
    val client = CloudWatchLogsClient.fromEnvironment()

    val request = StartLiveTailRequest {
        logGroupIdentifiers = logGroupIdentifiersVal
        logStreamNames = logStreamNamesVal
        logEventFilterPattern = logEventFilterPatternVal
```

```
    }

    val startTime = System.currentTimeMillis()

    try {
        client.startLiveTail(request) { response ->
            val stream = response.responseStream
            if (stream != null) {
                /* Set a timeout to unsubcribe from the flow. This will:
                * 1). Close the stream
                * 2). Stop the Live Tail session
                */
                stream.takeWhile { System.currentTimeMillis() - startTime <
10000 }.collect { value ->
                    if (value is StartLiveTailResponseStream.SessionStart) {
                        println(value.asSessionStart())
                    } else if (value is
StartLiveTailResponseStream.SessionUpdate) {
                        for (e in value.asSessionUpdate().sessionResults!!) {
                            println(e)
                        }
                    } else {
                        throw IllegalArgumentException("Unknown event type")
                    }
                }
            } else {
                throw IllegalArgumentException("No response stream")
            }
        }
    } catch (e: Exception) {
        println("Exception occurred during StartLiveTail: $e")
        System.exit(1)
    }
```

- API の詳細については、 AWS SDK for Kotlin API リファレンスStartLiveTailの「」を参照してください。

Python

SDK for Python (Boto3)

必要なファイルを含めます。

```
import boto3
import time
from datetime import datetime
```

Live Tail セッションを開始します。

```
    # Initialize the client
    client = boto3.client('logs')

    start_time = time.time()

    try:
        response = client.start_live_tail(
            logGroupIdentifiers=log_group_identifiers,
            logStreamNames=log_streams,
            logEventFilterPattern=filter_pattern
        )
        event_stream = response['responseStream']
        # Handle the events streamed back in the response
        for event in event_stream:
            # Set a timeout to close the stream.
            # This will end the Live Tail session.
            if (time.time() - start_time >= 10):
                event_stream.close()
                break
            # Handle when session is started
            if 'sessionStart' in event:
                session_start_event = event['sessionStart']
                print(session_start_event)
            # Handle when log event is given in a session update
            elif 'sessionUpdate' in event:
                log_events = event['sessionUpdate']['sessionResults']
                for log_event in log_events:
                    print('[{date}]
 {log}'.format(date=datetime.fromtimestamp(log_event['timestamp']/1000),log=log_event['me
            else:
                # On-stream exceptions are captured here
                raise RuntimeError(str(event))
    except Exception as e:
        print(e)
```

- API の詳細については、 StartLiveTail AWS SDK for Python (Boto3) API リファレンス の「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI `StartQuery`で を使用する

以下のコード例は、`StartQuery` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- 大規模なクエリを実行する

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Wrapper for the StartQueryCommand. Uses a static query string
 * for consistency.
 * @param {[Date, Date]} dateRange
 * @param {number} maxLogs
 * @returns {Promise<{ queryId: string }>}
 */
async _startQuery([startDate, endDate], maxLogs = 10000) {
  try {
    return await this.client.send(
      new StartQueryCommand({
        logGroupNames: this.logGroupNames,
        queryString: "fields @timestamp, @message | sort @timestamp asc",
        startTime: startDate.valueOf(),
```

```
          endTime: endDate.valueOf(),
          limit: maxLogs,
        }),
      );
    } catch (err) {
      /** @type {string} */
      const message = err.message;
      if (message.startsWith("Query's end date and time")) {
        // This error indicates that the query's start or end date occur
        // before the log group was created.
        throw new DateOutOfBoundsError(message);
      }

      throw err;
    }
  }
```

- API の詳細については、「 API リファレンスStartQuery」の「」を参照してください。 AWS SDK for JavaScript

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    def perform_query(self, date_range):
        """
        Performs the actual CloudWatch log query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :return: A list containing the query results.
        :rtype: list
        """
```

```
        client = boto3.client("logs")
        try:
            try:
                start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
                )
                end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
                )
                response = client.start_query(
                    logGroupName=self.log_groups,
                    startTime=start_time,
                    endTime=end_time,
                    queryString="fields @timestamp, @message | sort @timestamp
asc",
                    limit=self.limit,
                )
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
        except DateOutOfBoundsError:
            return []

    def _initiate_query(self, client, date_range, max_logs):
        """
        Initiates the CloudWatch logs query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :param max_logs: The maximum number of logs to retrieve.
```

```
        :type max_logs: int
        :return: The query ID as a string.
        :rtype: str
        """
        try:
            start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_groups,
                startTime=start_time,
                endTime=end_time,
                queryString="fields @timestamp, @message | sort @timestamp asc",
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")
```

- API の詳細については、 StartQuery AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDKs を使用した CloudWatch ログのシナリオ

次のコード例は、 AWS SDKs を使用した CloudWatch ログ で一般的なシナリオを実装する方法を示しています。これらのシナリオは、 CloudWatch ログ内で複数の関数を呼び出して特定のタスクを実行する方法を示しています。各シナリオには GitHub、コードの設定と実行の手順を示す へのリンクが含まれています。

例

- CloudWatch ログを使用して大きなクエリを実行する

CloudWatch ログを使用して大きなクエリを実行する

次のコード例は、 CloudWatch ログを使用して 10,000 を超えるレコードをクエリする方法を示しています。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

これはエントリポイントです。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
import { CloudWatchQuery } from "./cloud-watch-query.js";

console.log("Starting a recursive query...");

if (!process.env.QUERY_START_DATE || !process.env.QUERY_END_DATE) {
  throw new Error(
    "QUERY_START_DATE and QUERY_END_DATE environment variables are required.",
  );
}

const cloudWatchQuery = new CloudWatchQuery(new CloudWatchLogsClient({}), {
  logGroupNames: ["/workflows/cloudwatch-logs/large-query"],
  dateRange: [
    new Date(parseInt(process.env.QUERY_START_DATE)),
    new Date(parseInt(process.env.QUERY_END_DATE)),
  ],
});

await cloudWatchQuery.run();
```

```
console.log(
  `Queries finished in ${cloudWatchQuery.secondsElapsed} seconds.\nTotal logs
 found: ${cloudWatchQuery.results.length}`,
);
```

これは、必要に応じてクエリを複数のステップに分割するクラスです。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  StartQueryCommand,
  GetQueryResultsCommand,
} from "@aws-sdk/client-cloudwatch-logs";
import { splitDateRange } from "@aws-doc-sdk-examples/lib/utils/util-date.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

class DateOutOfBoundsError extends Error {}

export class CloudWatchQuery {
  /**
   * Run a query for all CloudWatch Logs within a certain date range.
   * CloudWatch logs return a max of 10,000 results. This class
   * performs a binary search across all of the logs in the provided
   * date range if a query returns the maximum number of results.
   *
   * @param {import('@aws-sdk/client-cloudwatch-logs').CloudWatchLogsClient}
 client
   * @param {{ logGroupNames: string[], dateRange: [Date, Date], queryConfig:
 { limit: number } }} config
   */
  constructor(client, { logGroupNames, dateRange, queryConfig }) {
    this.client = client;
    /**
     * All log groups are queried.
     */
    this.logGroupNames = logGroupNames;

    /**
     * The inclusive date range that is queried.
     */
    this.dateRange = dateRange;
```

```
    /**
     * CloudWatch Logs never returns more than 10,000 logs.
     */
    this.limit = queryConfig?.limit ?? 10000;

    /**
     * @type {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]}
     */
    this.results = [];
  }

  /**
   * Run the query.
   */
  async run() {
    this.secondsElapsed = 0;
    const start = new Date();
    this.results = await this._largeQuery(this.dateRange);
    const end = new Date();
    this.secondsElapsed = (end - start) / 1000;
    return this.results;
  }

  /**
   * Recursively query for logs.
   * @param {[Date, Date]} dateRange
   * @returns {Promise<import("@aws-sdk/client-cloudwatch-logs").ResultField[]
[]>}
   */
  async _largeQuery(dateRange) {
    const logs = await this._query(dateRange, this.limit);

    console.log(
      `Query date range: ${dateRange
        .map((d) => d.toISOString())
        .join(" to ")}. Found ${logs.length} logs.`,
    );

    if (logs.length < this.limit) {
      return logs;
    }

    const lastLogDate = this._getLastLogDate(logs);
```

```
    const offsetLastLogDate = new Date(lastLogDate);
    offsetLastLogDate.setMilliseconds(lastLogDate.getMilliseconds() + 1);
    const subDateRange = [offsetLastLogDate, dateRange[1]];
    const [r1, r2] = splitDateRange(subDateRange);
    const results = await Promise.all([
      this._largeQuery(r1),
      this._largeQuery(r2),
    ]);
    return [logs, ...results].flat();
  }

  /**
   * Find the most recent log in a list of logs.
   * @param {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]} logs
   */
  _getLastLogDate(logs) {
    const timestamps = logs
      .map(
        (log) =>
          log.find((fieldMeta) => fieldMeta.field === "@timestamp")?.value,
      )
      .filter((t) => !!t)
      .map((t) => `${t}Z`)
      .sort();

    if (!timestamps.length) {
      throw new Error("No timestamp found in logs.");
    }

    return new Date(timestamps[timestamps.length - 1]);
  }

  // snippet-start:[javascript.v3.cloudwatch-logs.actions.GetQueryResults]
  /**
   * Simple wrapper for the GetQueryResultsCommand.
   * @param {string} queryId
   */
  _getQueryResults(queryId) {
    return this.client.send(new GetQueryResultsCommand({ queryId }));
  }
  // snippet-end:[javascript.v3.cloudwatch-logs.actions.GetQueryResults]

  /**
   * Starts a query and waits for it to complete.
```

```
   * @param {[Date, Date]} dateRange
   * @param {number} maxLogs
   */
  async _query(dateRange, maxLogs) {
    try {
      const { queryId } = await this._startQuery(dateRange, maxLogs);
      const { results } = await this._waitUntilQueryDone(queryId);
      return results ?? [];
    } catch (err) {
      /**
       * This error is thrown when StartQuery returns an error indicating
       * that the query's start or end date occur before the log group was
       * created.
       */
      if (err instanceof DateOutOfBoundsError) {
        return [];
      } else {
        throw err;
      }
    }
  }

  // snippet-start:[javascript.v3.cloudwatch-logs.actions.StartQuery]
  /**
   * Wrapper for the StartQueryCommand. Uses a static query string
   * for consistency.
   * @param {[Date, Date]} dateRange
   * @param {number} maxLogs
   * @returns {Promise<{ queryId: string }>}
   */
  async _startQuery([startDate, endDate], maxLogs = 10000) {
    try {
      return await this.client.send(
        new StartQueryCommand({
          logGroupNames: this.logGroupNames,
          queryString: "fields @timestamp, @message | sort @timestamp asc",
          startTime: startDate.valueOf(),
          endTime: endDate.valueOf(),
          limit: maxLogs,
        }),
      );
    } catch (err) {
      /** @type {string} */
      const message = err.message;
```

```
      if (message.startsWith("Query's end date and time")) {
        // This error indicates that the query's start or end date occur
        // before the log group was created.
        throw new DateOutOfBoundsError(message);
      }

      throw err;
    }
  }
  // snippet-end:[javascript.v3.cloudwatch-logs.actions.StartQuery]

  /**
   * Call GetQueryResultsCommand until the query is done.
   * @param {string} queryId
   */
  _waitUntilQueryDone(queryId) {
    const getResults = async () => {
      const results = await this._getQueryResults(queryId);
      const queryDone = [
        "Complete",
        "Failed",
        "Cancelled",
        "Timeout",
        "Unknown",
      ].includes(results.status);

      return { queryDone, results };
    };

    return retry(
      { intervalInMs: 1000, maxRetries: 60, quiet: true },
      async () => {
        const { queryDone, results } = await getResults();
        if (!queryDone) {
          throw new Error("Query not done.");
        }

        return results;
      },
    );
  }
}
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の以下のトピックを参照してください。
  - GetQueryResults
  - StartQuery

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

このファイルは、10,000 件 CloudWatch を超えるクエリを管理するためのサンプルモジュールを呼び出します。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import logging
import os
import sys

import boto3
from botocore.config import Config

from cloudwatch_query import CloudWatchQuery
from date_utilities import DateUtilities

# Configure logging at the module level.
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(filename)s:%(lineno)d - %(message)s",
)


class CloudWatchLogsQueryRunner:
    def __init__(self):
        """
```

```
        Initializes the CloudWatchLogsQueryRunner class by setting up date
utilities
        and creating a CloudWatch Logs client with retry configuration.
        """
        self.date_utilities = DateUtilities()
        self.cloudwatch_logs_client = self.create_cloudwatch_logs_client()

    def create_cloudwatch_logs_client(self):
        """
        Creates and returns a CloudWatch Logs client with a specified retry
configuration.

        :return: A CloudWatch Logs client instance.
        :rtype: boto3.client
        """
        try:
            return boto3.client("logs", config=Config(retries={"max_attempts":
10}))
        except Exception as e:
            logging.error(f"Failed to create CloudWatch Logs client: {e}")
            sys.exit(1)

    def fetch_environment_variables(self):
        """
        Fetches and validates required environment variables for query start and
end dates.

        :return: Tuple of query start date and end date as integers.
        :rtype: tuple
        :raises SystemExit: If required environment variables are missing or
invalid.
        """
        try:
            query_start_date = int(os.environ["QUERY_START_DATE"])
            query_end_date = int(os.environ["QUERY_END_DATE"])
        except KeyError:
            logging.error(
                "Both QUERY_START_DATE and QUERY_END_DATE environment variables
are required."
            )
            sys.exit(1)
        except ValueError as e:
            logging.error(f"Error parsing date environment variables: {e}")
            sys.exit(1)
```

```
        return query_start_date, query_end_date

    def convert_dates_to_iso8601(self, start_date, end_date):
        """
        Converts UNIX timestamp dates to ISO 8601 format using DateUtilities.

        :param start_date: The start date in UNIX timestamp.
        :type start_date: int
        :param end_date: The end date in UNIX timestamp.
        :type end_date: int
        :return: Start and end dates in ISO 8601 format.
        :rtype: tuple
        """
        start_date_iso8601 =
 self.date_utilities.convert_unix_timestamp_to_iso8601(
            start_date
        )
        end_date_iso8601 = self.date_utilities.convert_unix_timestamp_to_iso8601(
            end_date
        )
        return start_date_iso8601, end_date_iso8601

    def execute_query(
        self,
        start_date_iso8601,
        end_date_iso8601,
        log_group="/workflows/cloudwatch-logs/large-query",
    ):
        """
        Creates a CloudWatchQuery instance and executes the query with provided
 date range.

        :param start_date_iso8601: The start date in ISO 8601 format.
        :type start_date_iso8601: str
        :param end_date_iso8601: The end date in ISO 8601 format.
        :type end_date_iso8601: str
        :param log_group: Log group to search: "/workflows/cloudwatch-logs/large-
query"
        :type log_group: str
        """
        cloudwatch_query = CloudWatchQuery(
            [start_date_iso8601, end_date_iso8601],
        )
```

```
        cloudwatch_query.query_logs((start_date_iso8601, end_date_iso8601))
        logging.info("Query executed successfully.")
        logging.info(
            f"Queries completed in {cloudwatch_query.query_duration} seconds.
 Total logs found: {len(cloudwatch_query.query_results)}"
        )


def main():
    """
    Main function to start a recursive CloudWatch logs query.
    Fetches required environment variables, converts dates, and executes the
 query.
    """
    logging.info("Starting a recursive CloudWatch logs query...")
    runner = CloudWatchLogsQueryRunner()
    query_start_date, query_end_date = runner.fetch_environment_variables()
    start_date_iso8601 = DateUtilities.convert_unix_timestamp_to_iso8601(
        query_start_date
    )
    end_date_iso8601 =
 DateUtilities.convert_unix_timestamp_to_iso8601(query_end_date)
    runner.execute_query(start_date_iso8601, end_date_iso8601)


if __name__ == "__main__":
    main()
```

このモジュールは、10,000 件を超える CloudWatch クエリを処理します。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import logging
import time
from datetime import datetime
import threading
import boto3

from date_utilities import DateUtilities


class DateOutOfBoundsError(Exception):
```

```
    """Exception raised when the date range for a query is out of bounds."""

    pass


class CloudWatchQuery:
    """
    A class to query AWS CloudWatch logs within a specified date range.

    :ivar date_range: Start and end datetime for the query.
    :vartype date_range: tuple
    :ivar limit: Maximum number of log entries to return.
    :vartype limit: int
    """

    def __init__(self, date_range):
        self.lock = threading.Lock()
        self.log_groups = "/workflows/cloudwatch-logs/large-query"
        self.query_results = []
        self.date_range = date_range
        self.query_duration = None
        self.datetime_format = "%Y-%m-%d %H:%M:%S.%f"
        self.date_utilities = DateUtilities()
        self.limit = 10000

    def query_logs(self, date_range):
        """
        Executes a CloudWatch logs query for a specified date range and
 calculates the execution time of the query.

        :return: A batch of logs retrieved from the CloudWatch logs query.
        :rtype: list
        """
        start_time = datetime.now()

        start_date, end_date = self.date_utilities.normalize_date_range_format(
            date_range, from_format="unix_timestamp", to_format="datetime"
        )

        logging.info(
            f"Original query:"
            f"\n       START:    {start_date}"
            f"\n       END:      {end_date}"
        )
```

```
        self.recursive_query((start_date, end_date))
        end_time = datetime.now()
        self.query_duration = (end_time - start_time).total_seconds()

    def recursive_query(self, date_range):
        """
        Processes logs within a given date range, fetching batches of logs
recursively if necessary.

        :param date_range: The date range to fetch logs for, specified as a tuple
(start_timestamp, end_timestamp).
        :type date_range: tuple
        :return: None if the recursive fetching is continued or stops when the
final batch of logs is processed.
                 Although it doesn't explicitly return the query results, this
method accumulates all fetched logs
                 in the `self.query_results` attribute.
        :rtype: None
        """
        batch_of_logs = self.perform_query(date_range)
        # Add the batch to the accumulated logs
        with self.lock:
            self.query_results.extend(batch_of_logs)
        if len(batch_of_logs) == self.limit:
            logging.info(f"Fetched {self.limit}, checking for more...")
            most_recent_log = self.find_most_recent_log(batch_of_logs)
            most_recent_log_timestamp = next(
                item["value"]
                for item in most_recent_log
                if item["field"] == "@timestamp"
            )
            new_range = (most_recent_log_timestamp, date_range[1])
            midpoint = self.date_utilities.find_middle_time(new_range)

            first_half_thread = threading.Thread(
                target=self.recursive_query,
                args=((most_recent_log_timestamp, midpoint),),
            )
            second_half_thread = threading.Thread(
                target=self.recursive_query, args=((midpoint, date_range[1]),)
            )

            first_half_thread.start()
            second_half_thread.start()
```

```
            first_half_thread.join()
            second_half_thread.join()

    def find_most_recent_log(self, logs):
        """
        Search a list of log items and return most recent log entry.
        :param logs: A list of logs to analyze.
        :return: log
        :type :return List containing log item details
        """
        most_recent_log = None
        most_recent_date = "1970-01-01 00:00:00.000"

        for log in logs:
            for item in log:
                if item["field"] == "@timestamp":
                    logging.debug(f"Compared: {item['value']} to
{most_recent_date}")
                    if (
                        self.date_utilities.compare_dates(
                            item["value"], most_recent_date
                        )
                        == item["value"]
                    ):
                        logging.debug(f"New most recent: {item['value']}")
                        most_recent_date = item["value"]
                        most_recent_log = log
        logging.info(f"Most recent log date of batch: {most_recent_date}")
        return most_recent_log

    # snippet-start:[python.example_code.cloudwatch_logs.start_query]
    def perform_query(self, date_range):
        """
        Performs the actual CloudWatch log query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :return: A list containing the query results.
        :rtype: list
        """
        client = boto3.client("logs")
        try:
```

```
            try:
                start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
                )
                end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
                )
                response = client.start_query(
                    logGroupName=self.log_groups,
                    startTime=start_time,
                    endTime=end_time,
                    queryString="fields @timestamp, @message | sort @timestamp
asc",
                    limit=self.limit,
                )
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
        except DateOutOfBoundsError:
            return []

    def _initiate_query(self, client, date_range, max_logs):
        """
        Initiates the CloudWatch logs query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :param max_logs: The maximum number of logs to retrieve.
        :type max_logs: int
        :return: The query ID as a string.
```

```
        :rtype: str
        """
        try:
            start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_groups,
                startTime=start_time,
                endTime=end_time,
                queryString="fields @timestamp, @message | sort @timestamp asc",
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")

    # snippet-end:[python.example_code.cloudwatch_logs.start_query]

    # snippet-start:[python.example_code.cloudwatch_logs.get_query_results]
    def _wait_for_query_results(self, client, query_id):
        """
        Waits for the query to complete and retrieves the results.

        :param query_id: The ID of the initiated query.
        :type query_id: str
        :return: A list containing the results of the query.
        :rtype: list
        """
        while True:
            time.sleep(1)
            results = client.get_query_results(queryId=query_id)
            if results["status"] in [
                "Complete",
                "Failed",
                "Cancelled",
                "Timeout",
                "Unknown",
            ]:
```

```
                return results.get("results", [])

    # snippet-end:[python.example_code.cloudwatch_logs.get_query_results]
```

- API の詳細については、『AWS SDK for Python (Boto3) API リファレンス』の以下のトピックを参照してください。
  - GetQueryResults
  - StartQuery

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDKs を使用した CloudWatch ログのクロスサービスの例

次のサンプルアプリケーションでは AWS SDKs を使用して CloudWatch ログを他の と組み合わせます AWS のサービス。各例には GitHub、アプリケーションのセットアップと実行の手順を示す へのリンクが含まれています。

例

- スケジュールされたイベントを使用した Lambda 関数の呼び出し

スケジュールされたイベントを使用した Lambda 関数の呼び出し

次のコード例は、Amazon EventBridge スケジュールされたイベントによって呼び出される AWS Lambda 関数を作成する方法を示しています。

Python

SDK for Python (Boto3)

この例では、スケジュールされた Amazon EventBridge イベントのターゲットとして AWS Lambda 関数を登録する方法を示します。Lambda ハンドラーは、わかりやすいメッセージと完全なイベントデータを Amazon CloudWatch Logs に書き込み、後で取得できるようにします。

- Lambda 関数をデプロイします。

- EventBridge スケジュールされたイベントを作成し、Lambda 関数をターゲットにします。
- Lambda 関数 EventBridge を呼び出す許可を に付与します。
- CloudWatch ログから最新のデータを出力して、スケジュールされた呼び出しの結果を表示します。
- デモ中に作成されたすべてのリソースをクリーンアップします。

この例は、 で最もよく表示されます GitHub。完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- CloudWatch ログ
- EventBridge
- Lambda

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での CloudWatch ログの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon CloudWatch Logs のセキュリティ

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

セキュリティは、 AWS とユーザーの間で共有される責任です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

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

このドキュメントは、Amazon CloudWatch Logs を使用する際の責任共有モデルの適用方法を理解するのに役立ちます。ここでは、セキュリティおよびコンプライアンスの目的を達成するように Amazon CloudWatch Logs を設定する方法を示します。また、 CloudWatch ログリソースのモニタリングや保護に役立つ他の AWS のサービスの使用方法についても説明します。

内容

Amazon CloudWatch Logs でのデータ保護

> **Note**
>
> の一般的なデータ保護に関する以下の情報に加えて AWS、 CloudWatch Logs では、ログイベントの機密データをマスキングして保護することもできます。詳細については、「機密性の高いログデータをマスキングで保護する」を参照してください。

責任 AWS 共有モデル、Amazon CloudWatch Logs でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。お客様は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。また、使用する AWS のサービス のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「データプライバシーのよくある質問」を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された記事「AWS 責任共有モデルおよび GDPR」を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須であり TLS 1.3 がお勧めです。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、コンソール、API、または AWS CLI SDK を使用して CloudWatch ログまたは他の AWS のサービス を操作する場合も同様

です。 AWS SDKs 名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

保管中の暗号化

CloudWatch ログは、暗号化を使用して保管中のデータを保護します。すべてのロググループは暗号化されます。デフォルトでは、 CloudWatch Logs サービスはサーバー側の暗号化キーを管理します。

ログの暗号化と復号に使用するキーを管理する場合は、 AWS KMS キーを使用します。詳細については、「を使用して Logs CloudWatch のログデータを暗号化する AWS Key Management Service」を参照してください。

転送中の暗号化

CloudWatch ログは転送中のデータの end-to-end 暗号化を使用します。 CloudWatch Logs サービスは、サーバー側の暗号化キーを管理します。

Amazon CloudWatch Logs の Identity and Access Management

Amazon CloudWatch Logs にアクセスするには AWS 、 がリクエストの認証に使用できる認証情報が必要です。これらの認証情報には、クラウド AWS リソースに関する CloudWatch ログデータを取得するなど、 リソースへのアクセス許可が必要です。以下のセクションでは、 AWS Identity and Access Management (IAM) と CloudWatch ログを使用して、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保護する方法について詳しく説明します。

- 認証
- アクセスコントロール

認証

アクセス権限を付与するには、ユーザー、グループ、またはロールにアクセス許可を追加します。

- のユーザーとグループ AWS IAM Identity Center :

  アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」の手順に従ってください。

- IAM 内で、ID プロバイダーによって管理されているユーザー:

  ID フェデレーションのロールを作成します。詳細については、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールの作成」を参照してください。
- IAM ユーザー:
  - ユーザーが担当できるロールを作成します。手順については、「IAM ユーザーガイド」の「IAM ユーザー用ロールの作成」を参照してください。
  - (お奨めできない方法) ポリシーをユーザーに直接アタッチするか、ユーザーをユーザーグループに追加する。詳細については、「IAM ユーザーガイド」の「ユーザー (コンソール) へのアクセス権限の追加」を参照してください。

アクセスコントロール

リクエストを認証するための有効な認証情報を持つことができますが、アクセス許可がない限り、CloudWatch ログリソースを作成またはアクセスすることはできません。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 ログ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
- CloudWatch ログでのアイデンティティベースのポリシー (IAM ポリシー) の使用
- CloudWatch Logs アクセス許可リファレンス

CloudWatch Logs リソースへのアクセス許可の管理の概要

アクセス権限を付与するには、ユーザー、グループ、またはロールにアクセス許可を追加します。

- のユーザーとグループ AWS IAM Identity Center :

  アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」の手順に従ってください。
- IAM 内で、ID プロバイダーによって管理されているユーザー:

  ID フェデレーションのロールを作成します。詳細については、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールの作成」を参照してください。
- IAM ユーザー:

- ユーザーが担当できるロールを作成します。手順については、「IAM ユーザーガイド」の「IAM ユーザー用ロールの作成」を参照してください。
- (お奨めできない方法) ポリシーをユーザーに直接アタッチするか、ユーザーをユーザーグループに追加する。詳細については、「IAM ユーザーガイド」の「ユーザー (コンソール) へのアクセス権限の追加」を参照してください。

トピック

CloudWatch リソースとオペレーションを記録します。

CloudWatch ログでは、プライマリリソースはロググループ、ログストリーム、および送信先です。CloudWatch ログはサブリソース (プライマリリソースで使用するその他のリソース) をサポートしていません。

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	次の 2 つの形式が使用されます。2 つ目は、:*最後に があり、 CLI コマンドと DescribeLogGroups API `describe-log-groups` によって返されるものです。 arn:aws:logs:*region*:*account-id* :log-group:*log_group_name* arn:aws:logs:*region*:*account-id* :log-group:*log_group_name* :*

リソースタイプ	ARN 形式
	次の状況:*では、末尾の なしで最初のバージョンを使用します。 • 多くの CloudWatch Logs APIs `logGroupIdentifier` の入力フィールド 。 • API のタグ付けの `resourceArn` フィールドで APIs • IAM ポリシーでは、、TagResource、UntagResourceおよび のアクセス許可を指定する場合ListTagsForResource。 他のすべての API アクションの IAM ポリシーでアクセス許可を指定するときは:*、末尾ので 2 番目のバージョンを使用して ARN を参照します。
ログストリーム	arn:aws:logs:*region* : *account-id* :log-group:*log_group_name* :log-stream:*log-stream-name*
デスティネーション	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 ログ ARNs「」の「Amazon リソースネーム (ARNs」を参照してくださいAmazon Web Services 全般のリファレンス。CloudWatch ログを対象とするポリシーの例については、「」を参照してください CloudWatch ログでのアイデンティティベースのポリシー (IAM ポリシー) の使用。

CloudWatch Logs は、Logs CloudWatch リソースを操作するための一連のオペレーションを提供します。使用可能なオペレーションのリストについては、「CloudWatch Logs アクセス許可リファレンス」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、リソースを作成したユーザーに関係なく、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、リソース作成リクエスト AWS 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、ユーザー、または IAM ロール) のアカウントです。次の例は、この仕組みを示しています。

- AWS アカウントのルート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 Logs CloudWatch リソースの所有者です。
- AWS アカウントにユーザーを作成し、そのユーザーに CloudWatch Logs リソースを作成するアクセス許可を付与すると、そのユーザーは Logs CloudWatch 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 Logs CloudWatch リソースを所有します。
- 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 CloudWatch ロールを作成すると、ロールを引き受けることのできるいずれのユーザーも Logs CloudWatch リソースを作成できます。ロールが属する AWS アカウントが Logs CloudWatch リソースを所有します。

リソースへのアクセスの管理

アクセス権限ポリシー では、誰が何にアクセスできるかを記述します。以下のセクションで、アクセス許可ポリシーを作成するために使用可能なオプションについて説明します。

> **Note**
>
> このセクションでは、 CloudWatch ログのコンテキストでの IAM の使用について説明します。これは、IAM サービスに関する詳細情報を取得できません。IAM に関する詳細なドキュメントについては、「IAM ユーザーガイド」の「What is IAM?」(IAM とは？) を参照してください。IAM ポリシー構文の詳細と説明については、「IAM ユーザーガイド」の「 IAM ポリシーリファレンス」を参照してください。

IAM 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 はアイデンティティベースのポリシーと、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、「クロスアカウントクロスリージョンサブスクリプション」を参照してください。

トピック

- ロググループの許可と Contributor Insights
- リソースベースのポリシー

ロググループの許可と Contributor Insights

Contributor Insights は、ロググループのデータを分析して、コントリビューターデータを表示する時系列を作成 CloudWatch できる の機能です。トップ N コントリビューター、一意のコントリビューターの合計数、およびそれらの使用状況に関するメトリクスを確認できます。詳細については、「Contributor Insights を使用した高カーディナリティデータの分析」を参照してください。

ユーザーに `cloudwatch:PutInsightRule`および アクセス`cloudwatch:GetInsightRuleReport`許可を付与すると、そのユーザーは CloudWatch Logs でロググループを評価し、結果を表示するルールを作成できます。結果には、これらのロググループのコントリビューターデータを含めることができます。これらのアクセス許可は、このデータを表示できるように設定したいユーザーのみに付与してください。

リソースベースのポリシー

CloudWatch ログは、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でき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、「ステップ 1: 送信先を作成する」を参照してください。送信先は PutDestination API を使用して作成でき、PutDestinationPolicyAPI 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、 AWS アカウント ID 111122223333 の別のアカウントが、ロググループを送信先 にサブスクライブすることを許可します`arn:aws:logs:us-east-1:123456789012:destination:testDestination`。

```
{
  "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 (アクション、効果、プリンシパル) の指定

CloudWatch Logs リソースごとに、サービスは一連の API オペレーションを定義します。これらの API オペレーションのアクセス許可を付与するために、 CloudWatch Logs はポリシーで指定できる一連のアクションを定義します。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 リソースとオペレーションを記録します。」および「CloudWatch Logs アクセス許可リファレンス」を参照してください。

以下は、基本的なポリシーの要素です。

- リソース - Amazon リソースネーム (ARN) を使用して、ポリシーを適用するリソースを識別します。詳細については、「CloudWatch リソースとオペレーションを記録します。」を参照してください。
- [Action] (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 – ID ベースのポリシー (IAM ポリシー) で、ポリシーがアタッチされているユーザーが黙示的なプリンシパルとなります。リソースベースのポリシーでは、アクセス許可を受け取るユーザー、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 (リソースベースのポリシーにのみ適用されます)。 CloudWatch ログは、送信先のリソースベースのポリシーをサポートします。

IAM ポリシーの構文と記述の詳細については、「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。

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を示す表については、「」を参照してくださいCloudWatch Logs アクセス許可リファレンス。

ポリシーでの条件の指定

アクセス権限を付与するとき、アクセスポリシー言語を使用して、ポリシーが有効になる必要がある条件を指定できます。例えば、特定の日付の後にのみ適用されるポリシーが必要になる場合がありま

す。ポリシー言語での条件の指定の詳細については、「IAM ユーザーガイド」の「条件」を参照してください。

条件を表すには、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ているコンテキストキーのリストと AWS全体のポリシーキーのリストについては、「 AWS のサービスおよびグローバル条件コンテキストキーのアクション、リソース、および条件キー」を参照してください。 AWS

> **Note**
>
> タグを使用して、 CloudWatch ロググループや送信先などの Logs リソースへのアクセスを制御できます。ロググループとログストリームの間には階層的な関係があるため、ログストリームへのアクセスはロググループレベルで制御されます。リソースへのアクセスを制御するタグの使用の詳細については、タグを使用した Amazon Web Services のリソースへのアクセスの制御を参照してください。

CloudWatch ログでのアイデンティティベースのポリシー (IAM ポリシー) の使用

このトピックでは、アカウント管理者が 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**
>
> まず、 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確認する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」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許可
- AWS CloudWatch ログの マネージド (事前定義) ポリシー
- カスタマーマネージドポリシーの例

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の詳細については、IAM ユーザーガイドの「IAM JSON ポリシーの要素のリファレンス」を参照してください。すべての CloudWatch ログアクションを示すリストについては、「」を参照してくださいCloudWatch Logs アクセス許可リファレンス。

CloudWatch コンソールを使用するために必要なアクセス許可

ユーザーが CloudWatch コンソールで CloudWatch ログを操作するには、そのユーザーに、アカウント AWS 内の他の AWS リソースを記述できる最小限のアクセス許可のセットが必要です。CloudWatch コンソールで CloudWatch ログを使用するには、次のサービスからのアクセス許可が必要です。

- CloudWatch
- CloudWatch ログ

- OpenSearch サービス
- IAM
- Kinesis
- Lambda
- Amazon S3

これらの最小限必要なアクセス許可よりも制限された IAM 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。これらのユーザーが CloudWatch 引き続きコンソールを使用できるようにするには、「」で説明されているように、 `CloudWatchReadOnlyAccess`管理ポリシーもユーザーにアタッチしますAWS CloudWatch ログの マネージド (事前定義) ポリシー。

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許可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サブスクリプションを管理していないユーザーに対してコンソールを操作するために必要なアクセス許可の完全なセットは次のとおりです。

- cloudwatch:GetMetricData
- cloudwatch:ListMetrics
- ログ：CancelExportTask
- ログ：CreateExportTask
- ログ：CreateLogGroup
- ログ：CreateLogStream
- ログ：DeleteLogGroup
- ログ：DeleteLogStream
- ログ：DeleteMetricFilter
- ログ：DeleteQueryDefinition
- ログ：DeleteRetentionPolicy
- ログ：DeleteSubscriptionFilter
- ログ：DescribeExportTasks
- ログ：DescribeLogGroups
- ログ：DescribeLogStreams

- ログ：DescribeMetricFilters
- ログ：DescribeQueryDefinitions
- ログ：DescribeQueries
- ログ：DescribeSubscriptionFilters
- ログ：FilterLogEvents
- ログ：GetLogEvents
- ログ：GetLogGroupFields
- ログ：GetLogRecord
- ログ：GetQueryResults
- ログ：PutMetricFilter
- ログ：PutQueryDefinition
- ログ：PutRetentionPolicy
- ログ：StartQuery
- ログ：StopQuery
- ログ：PutSubscriptionFilter
- ログ：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

AWS CloudWatch ログの マネージド (事前定義) ポリシー

AWS は、 によって作成および管理されるスタンドアロン IAM ポリシーを提供することで、多くの一般的なユースケースに対処します AWS。マネージドポリシーは、一般的ユースケースに必要な許可を付与することで、どの許可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 マネージドポリシー」を参照してください。

アカウントのユーザーとロールにアタッチできる次の AWS マネージドポリシーは、 CloudWatch ログに固有です。

- CloudWatchLogsFullAccess – CloudWatch ログへのフルアクセスを許可します。
- CloudWatchLogsReadOnlyAccess – CloudWatch Logs への読み取り専用アクセスを許可します。

CloudWatchLogsFullAccess

このCloudWatchLogsFullAccessポリシーは、 CloudWatch ログへのフルアクセスを許可します。ポリシーには アクセス`cloudwatch:GenerateQuery`許可が含まれているため、このポリシーを持つユーザーは自然言語プロンプトから CloudWatch Logs Insights クエリ文字列を生成できます。内容は次のとおりです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:*",
                "cloudwatch:GenerateQuery"
```

```
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

CloudWatchLogsReadOnlyAccess

このCloudWatchLogsReadOnlyAccessポリシーは、 CloudWatch ログへの読み取り専用アクセスを許可します。これには アクセスcloudwatch:GenerateQuery許可が含まれており、このポリシーを持つユーザーは自然言語プロンプトから CloudWatch Logs Insights クエリ文字列を生成できます。内容は次のとおりです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:Describe*",
                "logs:Get*",
                "logs:List*",
                "logs:StartQuery",
                "logs:StopQuery",
                "logs:TestMetricFilter",
                "logs:FilterLogEvents",
                "logs:StartLiveTail",
                "logs:StopLiveTail",
                "cloudwatch:GenerateQuery"
            ],
            "Resource": "*"
        }
    ]
}
```

CloudWatchLogsCrossAccountSharingConfiguration

このCloudWatchLogsCrossAccountSharingConfigurationポリシーは、アカウント間で CloudWatch Logs リソースを共有するための Observability Access Manager リンクを作成、管理、表示するためのアクセスを許可します。詳細については、CloudWatch 「クロスアカウントオブザーバビリティ」を参照してください。

内容は次のとおりです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:Link",
                "oam:ListLinks"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:DeleteLink",
                "oam:GetLink",
                "oam:TagResource"
            ],
            "Resource": "arn:aws:oam:*:*:link/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:CreateLink",
                "oam:UpdateLink"
            ],
            "Resource": [
                "arn:aws:oam:*:*:link/*",
                "arn:aws:oam:*:*:sink/*"
            ]
        }
    ]
}
```

CloudWatch AWS 管理ポリシーの更新をログに記録します。

このサービスがこれらの変更の追跡を開始してからの CloudWatch 、 ログの AWS マネージドポリシーの更新に関する詳細を表示します。このページの変更に関する自動アラートについては、ログドキュメントの履歴ページの RSS CloudWatch フィードをサブスクライブしてください。

変更	説明	日付
CloudWatchLogsFullAccess – 既存ポリシーへの更新。	CloudWatch ログが にアクセス許可を追加しましたCloudWatchLogsFullAccess。 このポリシーを持つユーザーが自然言語プロンプトから CloudWatch Logs Insights クエリ文字列を生成できるように、 アクセス`cloudwatch:GenerateQuery` 許可が追加されました。	2023 年 11 月 27 日
CloudWatchLogsReadOnlyAccess – 既存ポリシーへの更新。	CloudWatch が にアクセス許可を追加しましたCloudWatchLogsReadOnlyAccess。 このポリシーを持つユーザーが自然言語プロンプトから CloudWatch Logs Insights クエリ文字列を生成できるように、 アクセス`cloudwatch:GenerateQuery` 許可が追加されました。	2023 年 11 月 27 日
CloudWatchLogsReadOnlyAccess – 既存ポリシーへの更新	CloudWatch に追加されたアクセス許可をログに記録しますCloudWatchLogsReadOnlyAccess。 このポリシーを持つユーザーがコンソールを使用して CloudWatch Logs ライブテールセッションを開始および停	2023 年 6 月 6 日

変更	説明	日付
	止できるように、 `logs:StartLiveTail` および アクセス`logs:StopLiveTail` 許可が追加されました。詳細については、「ライブテールを使用してほぼリアルタイムでログを表示する」を参照してください。	
CloudWatchLogsCrossAccountSharingConfiguration - 新しいポリシー	CloudWatch Logs に、CloudWatch Logs ロググループを共有する CloudWatch クロスアカウントオブザーバビリティリンクを管理できるようにする新しいポリシーが追加されました。 詳細については、CloudWatch「クロスアカウントオブザーバビリティ」を参照してください。	2022 年 11 月 27 日

変更	説明	日付
CloudWatchLogsReadOnlyAccess – 既存ポリシーへの更新	CloudWatch に追加されたアクセス許可を CloudWatch Logs に記録しますCloudWatchLogsReadOnlyAccess。 このポリシーを持つユーザーがコンソールを使用して、ソースアカウントから共有されたデータを CloudWatch クロスアカウントオブザーバビリティで表示できるように、`oam:ListSinks` および アクセス`oam:ListAttachedLinks` 許可が追加されました。	2022 年 11 月 27 日

カスタマーマネージドポリシーの例

独自のカスタム IAM ポリシーを作成して、 CloudWatch Logs アクションとリソースのアクセス許可を許可できます。こうしたカスタムポリシーは、該当するアクセス許可が必要なユーザーまたはグループにアタッチできます。

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許可を付与するユーザーポリシーの例を示します。これらのポリシーは、 CloudWatch Logs API、 AWS SDKs、または を使用している場合に機能します AWS CLI。

例

- 例 1: CloudWatch ログへのフルアクセスを許可する
- 例 2: CloudWatch ログへの読み取り専用アクセスを許可する
- 例 3: 1 つのロググループへのアクセスを許可する

例 1: CloudWatch ログへのフルアクセスを許可する

次のポリシーでは、ユーザーはすべての CloudWatch Logs アクションにアクセスできます。

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

例 2: CloudWatch ログへの読み取り専用アクセスを許可する

AWS は、 CloudWatch ログデータへの読み取り専用アクセスを有効にするCloudWatchLogsReadOnlyAccessポリシーを提供します。このポリシーには、以下のアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:Describe*",
                "logs:Get*",
                "logs:List*",
                "logs:StartQuery",
                "logs:StopQuery",
                "logs:TestMetricFilter",
                "logs:FilterLogEvents",
                "logs:StartLiveTail",
                "logs:StopLiveTail",
                "cloudwatch:GenerateQuery"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

例 3: 1 つのロググループへのアクセスを許可する

次のポリシーでは、指定した 1 つのロググループのログイベントの読み取りと書き込みをユーザーに許可します。

> **Important**
>
> Resource 行のロググループ名の末尾にある :* は、ポリシーがこのロググループのすべてのログストリームに適用されることを示すために必要です。:* を省略すると、ポリシーは適用されません。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
      "Action": [
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:us-west-2:123456789012:log-group:SampleLogGroupName:*"
      }
   ]
}
```

ロググループレベルでのコントロールのためにタグ付けと IAM ポリシーを使用する

他のロググループへのアクセスを防止しながら、特定のロググループへのアクセスをユーザーに許可することができます。これを行うには、ロググループにタグを付け、IAM ポリシーを使用してそれらのタグを参照します。タグをロググループに適用するには、logs:TagResource または logs:TagLogGroup のアクセス許可が必要です。これは、作成時にロググループにタグを割り当てる場合と、後で割り当てる場合の両方に当てはまります。

ロググループのタグ付けの詳細については、「Amazon CloudWatch Logs のロググループにタグを付ける」を参照してください。

ロググループにタグを付けるときは、特定のタグを持つロググループのみにアクセスを許可する IAM ポリシーをユーザーに付与できます。たとえば、以下のポリシーステートメントでは、タグキー Green の値が Team のロググループにのみアクセス権が付与されます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:*"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/Team": "Green"
                }
            }
        }
    ]
}
```

StopQuery および StopLiveTail API オペレーションは、従来の意味では AWS リソースとやり取りしません。何らかの方法でデータを返したり、データを入力したり、リソースを変更したりすることはありません。代わりに、特定のライブテールセッションまたはリソースに分類されていない特定の CloudWatch Logs Insights クエリでのみ動作します。そのため、これらの操作の IAM ポリシーで Resource フィールドを指定するとき、次の例のように、Resource フィールドの値を * として設定する必要があります。

```
{
    "Version": "2012-10-17",
    "Statement":
        [ {
            "Effect": "Allow",
            "Action": [
                "logs:StopQuery",
                "logs:StopLiveTail"
            ],
            "Resource": "*"
            }
        ]
```

```
}
```

IAM ポリシーステートメントの詳細については、『IAM ユーザーガイド』の「ポリシーを使用したアクセス制御」を参照してください。

CloudWatch Logs アクセス許可リファレンス

アクセスコントロール をセットアップし、IAM ID (ID ベースのポリシー) にアタッチできるアクセス許可ポリシーを作成する際、次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 オペレーションと、アクションを実行するためのアクセス許可を付与できる対応するアクションが一覧表示されています。アクションは、ポリシーの `Action` フィールドで指定します。`Resource` フィールドでは、ロググループまたはログストリームの ARN を指定するか、を指定*してすべての CloudWatch Logs リソースを表すことができます。

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できます。 AWS全体のキーの完全なリストについては、AWS 「IAM ユーザーガイド」の「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`、または `logs:*` (すべての CloudWatch ログアクションの場合) 。

CloudWatch アクションの API オペレーションと必要なアクセス許可を記録します。

CloudWatch API オペレーションのログ記録	必要なアクセス許可 (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`

CloudWatch API オペレーションのログ記録	必要なアクセス許可 (API アクション)
	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` 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。
DeleteQueryDefinition	`logs:DeleteQueryDefinition` CloudWatch Logs Insights で保存されたクエリ定義を削除するために必要です。
DeleteResourcePolicy	`logs:DeleteResourcePolicy` CloudWatch Logs リソースポリシーを削除するために必要です。

CloudWatch API オペレーションのログ記録	必要なアクセス許可 (API アクション)
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。
DescribeQueryDefinitions	`logs:DescribeQueryDefinitions` CloudWatch Logs Insights に保存されているクエリ定義のリストを表示するために必要です。

CloudWatch API オペレーションのログ記録	必要なアクセス許可 (API アクション)
DescribeQueries	`logs:DescribeQueries` スケジュール、実行、または最近超過した CloudWatch Logs Insights クエリのリストを表示するために必要です。
DescribeResourcePolicies	`logs:DescribeResourcePolicies` CloudWatch Logs リソースポリシーのリストを表示するために必要です。
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。
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。
GetLogGroupFields	`logs:GetLogGroupFields` ロググループのログイベントに含まれるフィールドのリストを取得するために必要です。
GetLogRecord	`logs:GetLogRecord` 1 つのログイベントから詳細を取得するために必要です。

CloudWatch API オペレーションのログ記録	必要なアクセス許可 (API アクション)
GetQueryResults	`logs:GetQueryResults` CloudWatch Logs Insights クエリの結果を取得するために必要です。
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。
PutQueryDefinition	`logs:PutQueryDefinition` CloudWatch Logs Insights にクエリを保存するために必要です。
PutResourcePolicy	`logs:PutResourcePolicy` CloudWatch Logs リソースポリシーの作成に必要です。

CloudWatch API オペレーションのログ記録	必要なアクセス許可 (API アクション)
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。
StartQuery	`logs:StartQuery` CloudWatch Logs Insights クエリを開始するために必要です。
StopQuery	`logs:StopQuery` 進行中の CloudWatch Logs Insights クエリを停止するために必要です。
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 のサービスにリンクされたロールの使用

Amazon CloudWatch Logs は AWS Identity and Access Management 、 (IAM) サービスにリンクされたロール を使用します。サービスにリンクされたロールは、 CloudWatch ログに直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは Logs CloudWatch によって事前

定義されており、ユーザーに代わってサービスから他の AWS のサービスを呼び出すために必要なすべてのアクセス許可が含まれています。

サービスにリンクされたロールを使用すると、必要なアクセス許可を手動で追加する必要がないため、 CloudWatch ログの設定がより効率的になります。 CloudWatch Logs はサービスにリンクされたロールのアクセス許可を定義します。特に定義されている場合を除き、これらのロールを引き受けることができるのは CloudWatch Logs のみです。定義された許可には、信頼ポリシーと許可ポリシーが含まれます。アクセス権限ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスにリンクされたロールをサポートしているその他のサービスの詳細については、「IAM と連携するAWS のサービス」を参照してください。サービスにリンクされたロール列が「はい」になっているサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

CloudWatch Logs のサービスにリンクされたロールのアクセス許可

CloudWatch ログは、 という名前のサービスにリンクされたロールを使用しますAWSServiceRoleForLogDelivery。CloudWatch Logs は、このサービスにリンクされたロールを使用して、Firehose に直接ログを書き込みます。詳細については、「AWS サービスからのログ記録を有効にする」を参照してください。

AWSServiceRoleForLogDelivery サービスにリンクされたロールは、ロールの引き受けについて以下のサービスを信頼します。

- `logs.amazonaws.com`

ロールのアクセス許可ポリシーは、指定されたリソースに対して以下のアクションを実行することを CloudWatch Logs に許可します。

- アクション: 値が のLogDeliveryEnabledキーを持つタグを持つすべての Firehose ストリームfirehose:PutRecordBatchで firehose:PutRecordおよび True。このタグは、Firehose にログを配信するサブスクリプションを作成すると、Firehose ストリームに自動的にアタッチされます。

IAM エンティティがサービスにリンクされたロールを作成、編集、削除できるようにするには、アクセス許可を設定する必要があります。このエンティティは、ユーザー、グループ、またはロールです。詳細については、IAM ユーザーガイドの「サービスにリンクされたロールのアクセス許可」を参照してください。

CloudWatch Logs のサービスにリンクされたロールの作成

サービスにリンクされたロールを手動で作成する必要はありません。、 AWS Management Console、 AWS CLIまたは AWS API で Firehose ストリームに直接送信されるログを設定すると、CloudWatch Logs によってサービスにリンクされたロールが作成されます。

このサービスリンクロールを削除した後で再度作成する必要が生じた場合は、同じ方法でアカウントにロールを再作成できます。Firehose ストリームに直接送信されるログを再度設定すると、CloudWatch Logs によってサービスにリンクされたロールが再度作成されます。

CloudWatch Logs のサービスにリンクされたロールの編集

CloudWatch ログではAWSServiceRoleForLogDelivery、作成後に または他のサービスにリンクされたロールを編集することはできません。さまざまなエンティティがロールを参照する可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロールの説明の編集はできます。詳細については、「IAM ユーザーガイド」の「サービスリンクロールの編集」を参照してください。

CloudWatch Logs のサービスにリンクされたロールの削除

サービスリンクロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、積極的にモニタリングまたは保守されていない未使用のエンティティを排除できます。ただし、手動で削除する前に、サービスリンクロールのリソースをクリーンアップする必要があります。

> **Note**
>
> リソースを削除しようとしたときに CloudWatch Logs サービスがロールを使用している場合、削除が失敗する可能性があります。失敗した場合は、数分待ってから操作を再試行してください。

AWSServiceRoleForLogDelivery サービスにリンクされたロールが使用する CloudWatch Logs リソースを削除するには

- Firehose ストリームへのログの直接送信を停止します。

サービスにリンクされたロールを IAM で手動削除するには

IAM コンソール、、または AWS API を使用して AWS CLI、AWSServiceRoleForLogDeliveryサービスにリンクされたロールを削除します。詳細については、「サービスにリンクされたロールの削除」を参照してください。

CloudWatch Logs サービスにリンクされたロールでサポートされているリージョン

CloudWatch ログは、サービスが利用可能なすべての AWS リージョンで、サービスにリンクされたロールの使用をサポートします。詳細については、CloudWatch 「ログリージョンとエンドポイント」を参照してください。

Amazon CloudWatch Logs のコンプライアンス検証

サードパーティーの監査者は、さまざまな コンプライアンスプログラムの一環として Amazon CloudWatch Logs のセキュリティと AWS コンプライアンスを評価します。これらのプログラムには、SOC、PCI、FedRAMP、HIPAA などがあります。

特定のコンプライアンスプログラムの対象となる AWS のサービスのリストについては、「コンプライアンスプログラムAWS による対象範囲内の のサービス」「コンプライアンスプログラム」を参照してください。一般的な情報については、「AWS コンプライアンスプログラム」を参照してください。

サードパーティーの監査レポートは、 を使用してダウンロードできます AWS Artifact。詳細については、「 でのレポートのダウンロード AWS Artifact」の」を参照してください。

Amazon CloudWatch Logs を使用する際のお客様のコンプライアンス責任は、データの機密性、企業のコンプライアンス目的、適用法規によって決まります。 AWS では、コンプライアンスに役立つ以下のリソースを提供しています。

- 「セキュリティ＆コンプライアンスクイックリファレンスガイド」 – これらのデプロイガイドには、アーキテクチャ上の考慮事項の説明と、 AWSでセキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイするための手順が記載されています。
- Architecting for HIPAA Security and Compliance on Amazon Web Services – このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。
- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や場所に適用される場合があります。
- 「 デベロッパーガイド」の「ルールによるリソースの評価」 – AWS Configリソース設定が社内プラクティス、業界ガイドライン、規制にどの程度準拠しているかを評価します。 AWS Config

- AWS Security Hub — この AWS サービスは、 内のセキュリティ状態を包括的に把握 AWS し、セキュリティ業界標準およびベストプラクティスへの準拠を確認するのに役立ちます。

Amazon CloudWatch Logs の耐障害性

AWS のグローバルインフラストラクチャは AWS リージョンとアベイラビリティーゾーンを中心として構築されます。リージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立および隔離されたアベイラビリティーゾーンがあります。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、および拡張性が優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、「AWS グローバルインフラストラクチャ」を参照してください。

Amazon CloudWatch Logs のインフラストラクチャセキュリティ

マネージドサービスである Amazon CloudWatch Logs は AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「セキュリティの柱 AWS Well-Architected Framework」の「インフラストラクチャ保護」を参照してください。

が AWS 公開した API コールを使用して、ネットワーク経由で CloudWatch Logs にアクセスします。クライアントは以下をサポートする必要があります:

- Transport Layer Security (TLS)。TLS 1.2、できれば TLS 1.3 が必要です。
- DHE (Ephemeral Diffie-Hellman) や ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) などの Perfect Forward Secrecy (PFS) を使用した暗号スイート。これらのモードは、Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token ServiceAWS STS を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

インターフェイス VPC エンドポイントでの CloudWatch ログの使用

Amazon Virtual Private Cloud (Amazon VPC) を使用して AWS リソースをホストする場合、VPC と CloudWatch Logs の間にプライベート接続を確立できます。この接続を使用すると、インターネット経由でログを送信せずにログを CloudWatch Logs に送信できます。

Amazon VPC は、定義した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 のサービスです。VPC を使用することで、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 の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 接続を必要とせずに、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、Elastic Network Interface とプライベート IP アドレスを使用して AWS サービス間のプライベート通信を可能にする AWS テクノロジーである を利用しています。詳細については、「New – for AWS Services AWS PrivateLink」を参照してください。

以下の手順は、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。

可用性

CloudWatch ログは現在、 AWS リージョンを含むすべての AWS GovCloud (US) リージョンで VPC エンドポイントをサポートしています。

CloudWatch Logs 用の 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、Logs 用のインターフェイス VPC CloudWatch エンドポイントを作成します。選択するサービスは、[com.amazonaws.***Region***.logs] です。CloudWatch ログの設定は変更する必要はありません。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。

VPC と CloudWatch Logs 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 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 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 1 つ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。

CloudWatch Logs VPC エンドポイントへのアクセスの制御

VPC エンドポイントポリシーは、エンドポイントの作成時または変更時にエンドポイントに加える国際機械技術者協会 (IAM) のリソースポリシーです。エンドポイントの作成時にポリシーをアタッチしない場合、サービスへのフルアクセスを許可するデフォルトのポリシーがアタッチされます。エンドポイントポリシーは、IAM ポリシーやサービス固有のポリシーを上書き、または置き換えたりするものではありません。これは、評価項目から指定されたサービスへのアクセスを制御するための別のポリシーです。

評価項目のポリシーは、JSON形式で記載する必要があります。

詳細については、「Amazon VPCユーザーガイド」の「VPC評価項目によるサービスのアクセス制御」を参照してください。

CloudWatch ログのエンドポイントポリシーの例を次に示します。このポリシーにより、VPC 経由で CloudWatch Logs に接続するユーザーはログストリームを作成し、ログを CloudWatch Logs に送信できます。また、他の CloudWatch Logs アクションを実行することはできません。

```
{
  "Statement": [
    {
      "Sid": "PutOnly",
      "Principal": "*",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

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

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

VPC コンテキストキーのサポート

CloudWatch ログは、特定の VPCsまたは特定の VPC エンドポイントへのアクセスを制限できる `aws:SourceVpc`および `aws:SourceVpce`コンテキストキーをサポートします。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、「IAM ユーザーガイド」の「一部のサービスに使用可能なキー」を参照してください。

での CloudWatch Logs API およびコンソールオペレーションのログ記録 AWS CloudTrail

Amazon CloudWatch Logs は と統合されています。これは AWS CloudTrail、ユーザー、ロール、または サービスによって実行されたアクションの記録を Logs. CloudTrail captures API CloudWatch コールで提供する AWS サービスです AWS 。キャプチャされた呼び出しには、 CloudWatch コンソールからの呼び出しと Logs API CloudWatch 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 ログの CloudTrailイベントなど、Amazon S3 バケットへの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 コンソールのイベント履歴 で最新のイベントを表示できます。によって収集された情報を使用して CloudTrail、 CloudWatch ログに対して行われたリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

の設定と有効化の方法など CloudTrail、 の詳細については、AWS CloudTrail 「 ユーザーガイド」を参照してください。

トピック

- CloudWatch で情報を口グに記録する CloudTrail
- でのクエリ生成情報 CloudTrail
- ログファイルエントリについて

CloudWatch で情報を口グに記録する CloudTrail

CloudTrail AWS アカウントを作成すると、 がアカウントで有効になります。サポートされているイベントアクティビティが CloudWatch Logs で発生すると、そのアクティビティは CloudTrail イベント履歴 の他の AWS サービスイベントとともにイベントに記録されます。 AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「イベント履歴を使用した CloudTrail イベントの表示」を参照してください。

CloudWatch ログのイベントなど、 AWS アカウント内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、コンソールで証跡を作成すると、証跡はすべての AWS リージョンに適用されます。証跡は、 AWS パーティション内のすべてのリージョンからのイベントを口グに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、 CloudTrail ログで収集されたイベン

トデータをより詳細に分析し、それに基づく対応を行うように他の AWS サービスを設定できます。詳細については、次を参照してください:

- 証跡の作成のための概要
- CloudTrail サポートされているサービスと統合
- の Amazon SNS 通知の設定 CloudTrail
- 複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからのログファイルの受信 CloudTrail

CloudWatch ログは、次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
- PutResourcePolicy
- PutRetentionPolicy
- PutSubscriptionFilter
- StartQuery
- StopQuery
- TestMetricFilter

これらの CloudWatch Logs API アクション CloudTrail には、リクエスト要素のみがログインされます。

- DescribeDestinations
- DescribeExportTasks
- DescribeLogGroups
- DescribeLogStreams
- DescribeMetricFilters
- DescribeQueries
- DescribeResourcePolicies
- DescribeSubscriptionFilters
- FilterLogEvents
- GetLogEvents
- GetLogGroupFields
- GetLogRecord
- GetQueryResults

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。同一性情報は次の判断に役立ちます。

- リクエストが、ルートと IAM ユーザー認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーションユーザーの一時的なセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

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

でのクエリ生成情報 CloudTrail

CloudTrail クエリジェネレーターコンソールイベントのログ記録もサポートされています。クエリジェネレーターは現在、 CloudWatch Logs Insights と Metric Insights CloudWatch でサポートされています。これらの CloudTrail イベントでは、 は `eventSource`で すmonitoring.amazonaws.com。

次の例は、 CloudTrail Logs Insights の GenerateQueryアクションを示す CloudWatch ログエントリを示しています。


```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::111222333444:role/Administrator",
                "accountId": "123456789012",
                "userName": "SAMPLE_NAME"
            },
            "attributes": {
                "creationDate": "2020-04-08T21:43:24Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2020-04-08T23:06:30Z",
    "eventSource": "monitoring.amazonaws.com",
    "eventName": "GenerateQuery",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "exampleUserAgent",
    "requestParameters": {
        "query_ask": "***",
        "query_type": "LogsInsights",
        "logs_insights": {
            "fields": "***",
            "log_group_names": ["yourloggroup"]
        },
        "include_description": true
    },
    "responseElements": null,
    "requestID": "2f56318c-cfbd-4b60-9d93-1234567890",
    "eventID": "52723fd9-4a54-478c-ac55-1234567890",
```

```
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

ログファイルエントリについて

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ようにする設定です。 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。イベントは任意のソース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、リクエストパラメータなどに関する情報が含まれます。 CloudTrail ログファイルはパブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次のログファイルエントリは、ユーザーが CloudWatch ログCreateExportTaskアクションを呼び出したことを示しています。

```
{
        "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 エージェントリファレンス

> **Important**
>
> このリファレンスは、廃止された古い CloudWatch Logs エージェント用です。インスタンスメタデータサービスバージョン 2 (IMDSv2) を使用する場合は、新しい統合 CloudWatch エージェントを使用する必要があります。IMDSv2 を使用していない場合でも、古いログ CloudWatch エージェントではなく、新しい統合エージェントを使用することを強くお勧めします。新しい統合エージェントの詳細については、CloudWatch 「エージェント を使用した Amazon EC2 インスタンスとオンプレミスサーバーからのメトリクスとログの収集」を参照してください。
>
> 古い CloudWatch Logs エージェントから統合エージェントへの移行については、「ウィザードを使用して 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。

CloudWatch Logs エージェントは、Amazon EC2 インスタンスから CloudWatch ログにログデータを自動的に送信する方法を提供します。エージェントには以下のコンポーネントが含まれます。

- ログデータを CloudWatch Logs にプッシュ AWS CLI する へのプラグイン。
- ログにデータをプッシュするプロセスを開始するスクリプト (デーモン ） CloudWatch 。
- 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には、 CloudWatch Logs エージェントに必要な情報が記述されています。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。ファイルは Python 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、gzip http コンテンツエンコーディングが圧縮ペイロードを CloudWatch Logs に送信できるようになります。これにより、CPU 使用率が低下し、 が低下し NetworkOut、配置レイテンシーが減少します。この機能を無効にするには、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 CloudWatch を追加し、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 ログにプッシュされ

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。フィンガープリント行は、指定された行がすべて使用可能でない限り、 CloudWatch ログに送信されません。

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

CloudWatch Logs エージェントは HTTP プロキシで使用できます。

> **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. 次のいずれかを行います。

   a. CloudWatch Logs エージェントを新規インストールするには、次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、「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。Amazon EC2

`http-proxy` および `https-proxy` の値で、URL 全体を指定します。

b. CloudWatch Logs エージェントの既存のインストールでは、/var/awslogs/etc/proxy.conf を編集し、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分割

awscli-cwlogs 1.3.3 以降で awslogs-agent-setup.py バージョン 1.3.8 以降を使用している場合は、/var/awslogs/etc/config/ ディレクトリ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異なるストリーム設定を個別にインポートできます。 CloudWatch Logs エージェントを起動すると、これらの追加設定ファイルにストリーム設定が含まれます。[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 ログ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

Logs エージェントのインストールにセットアップスクリプトを使用した場合は、/var/awslogs/bin/awslogs-version.sh CloudWatch を使用して、使用しているエージェントのバージョンを確認できます。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 を使用して CloudWatch Logs エージェントをインストールした場合は、「yum info awslogs」と「yum

info aws-cli-plugin-cloudwatch-logs」を使用して CloudWatch 、Logs エージェントとプラグインのバージョンを確認できます。

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。例えば、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。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。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ると、 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、停止した場所から開始し、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 エージェントがロググループまたはログストリームを自動的に作成しないようにします。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 (DescribeLogStreams、PutLogEvents) のみに制限できます。

エージェントから CreateLogGroup および CreateLogStream 権限を取り消す前に、エージェントが使用するロググループとログストリームの両方を作成してください。ログエージェントは、CreateLogGroup および CreateLogStream 権限の両方がない限り、作成されたロググループにログストリームを作成できません。

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 メトリクスによるモニタリング

CloudWatch ログは 1 分 CloudWatch ごとにメトリクスを Amazon に送信します。

CloudWatch ログメトリクス

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
CallCount	アカウントで実行された指定された API オペレーションの数。 CallCount は Logs CloudWatch サービス使用状況メトリクスです。詳細については、「CloudWatch サービス使用状況メトリクスのログ記録」を参照してください。 有効なディメンション: クラス、リソース、サービス、タイプ 有効な統計: Sum 単位: なし
DeliveryErrors	サブスクリプションの送信先にデータを転送するときに CloudWatch Logs がエラーを受信したログイベントの数。送信先サービスがスロットリング例外や再試行可能なサービス例外 (HTTP 5xx など) などの再試行可能なエラーを返した場合、 CloudWatch ログは最大 24 時間配信を再試行し続けます。エラーが AccessDeniedException や などの再試行不可能なエラーである場合、 CloudWatch ログは再配信を試みませんResourceNotFoundException 。 有効なディメンション： LogGroupName、 DestinationType、 FilterName、 PolicyLevel 有効な統計: Sum 単位: なし
DeliveryThrottling	サブスクリプションの送信先にデータを転送するときに CloudWatch Logs がスロットリングされたログイベントの数。

メトリクス	説明
	送信先サービスがスロットリング例外や再試行可能なサービス例外 (HTTP 5xx など) などの再試行可能なエラーを返した場合、 CloudWatch ログは最大 24 時間配信を再試行し続けます。エラーが `AccessDeniedException` や などの再試行不可能なエラーである場合、CloudWatch ログは再配信を試みません`ResourceNotFoundException` 。 有効なディメンション： LogGroupName、 DestinationType、 FilterName、 PolicyLevel 有効な統計: Sum 単位: なし
`EMFParsingErrors`	埋め込みメトリクスフォーマットログの処理中に発生した解析エラーの数。このようなエラーは、埋め込みメトリクス形式として識別されたログが、適切な形式に従っていない場合に発生します。埋め込みメトリクス形式の詳細については、「仕様: 埋め込みメトリクスフォーマット」を参照してください。 有効なディメンション: `LogGroupName` 有効な統計: Sum 単位: なし

メトリクス	説明
EMFValidationErrors	埋め込みメトリクス形式ログの処理中に発生した検証エラーの数。これらのエラーは、埋め込みメトリクス形式ログ内のメトリクスの定義が、埋め込みメトリクス形式と MetricDatum の仕様に準拠していない場合に発生します。 CloudWatch 埋め込みメトリクス形式の詳細については、「仕様: 埋め込みメトリクス形式」を参照してください。データ型の詳細についてはMetricDatum 、「Amazon CloudWatch API リファレンスMetricDatum」の「」を参照してください。 Note 特定の検証エラーにより、EMF ログ内の複数のメトリクスが公開されないことがあります。例えば、無効な名前空間で設定されたメトリクスはすべて削除されます。 有効なディメンション: LogGroupName 有効な統計: Sum 単位: なし
ErrorCount	アカウントで実行され、エラーが発生した API オペレーションの数。 ErrorCount は Logs CloudWatch サービス使用状況メトリクスです。詳細については、「CloudWatch サービス使用状況メトリクスのログ記録」を参照してください。 有効なディメンション: クラス、リソース、サービス、タイプ 有効な統計: Sum 単位: なし

メトリクス	説明
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、 PolicyLevel 有効な統計: Sum 単位: なし
IncomingBytes	Logs にアップロードされた非圧縮バイト単位の CloudWatch ログイベントの量。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	Logs にアップロードされた CloudWatch 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

メトリクス	説明
LogEvents WithFindings	Logs データ保護機能を使用して監査するデータ文字列に一致した CloudWatch ログイベントの数。詳細については、「機密性の高いログデータをマスキングで保護する」を参照してください。 有効なディメンション: なし 有効な統計: Sum 単位: なし
ThrottleCount	アカウントで実行され、使用クォータによって調整された API オペレーションの数。 ThrottleCount は Logs CloudWatch サービス使用状況メトリクスです。詳細については、「CloudWatch サービス使用状況メトリクスのログ記録」を参照してください。 有効なディメンション: クラス、リソース、サービス、タイプ 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次の表に示します。

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 Data Firehose です。
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ー名は によっ

ディメンション	説明
	て ASCII CloudWatch に自動的に変換され、サポートされていない文字はすべて疑問符 (?) に置き換えられます。

アカウントレベルのサブスクリプションフィルターに関連するメトリクスのディメンションを次の表に示します。

ディメンション	説明
`PolicyLevel`	ポリシーが適用されるレベル。現在、このディメンションに有効な値は のみです。 `AccountPolicy`
`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 Data Firehose です。
`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ー名は によって ASCII CloudWatch に自動的に変換され、サポートされていない文字はすべて疑問符 (?) に置き換えられます。

CloudWatch サービス使用状況メトリクスのログ記録

CloudWatch Logs は、Logs API CloudWatch オペレーションの使用状況を追跡 CloudWatch するメトリクスを に送信します。これらのメトリクスは AWS Service Quotas に対応しています。これらのメトリクスを追跡することで、クォータを積極的に管理できます。詳細については、「Service Quotas の統合と使用状況メトリクス」を参照してください。

例えば、`ThrottleCount` メトリクスを追跡したり、そのメトリクスにアラームを設定したりできます。このメトリクスの値が上昇した場合は、スロットリングされた API オペレーションのためにクォータの引き上げをリクエストすることを検討してください。 CloudWatch Logs サービスクォータの詳細については、「」を参照してくださいCloudWatch ログクォータ。

CloudWatch ログは、 `AWS/Usage`および `AWS/Logs`名前空間の両方で、1 分ごとにサービスクォータ使用状況メトリクスを発行します。

次の表に、 CloudWatch ログによって発行されたサービス使用状況メトリクスを示します。これらのメトリクスには、指定された単位がありません。これらのメトリクスの最も有用な統計は SUM です。これは、1 分間の合計オペレーション数を表します。

これらのメトリクスは、Service、Class、Type、Resource のすべてのディメンションの値とともに発行されます。また、Account Metrics と呼ばれる 1 つのディメンションで発行されます。アカウント内のすべての API オペレーションのメトリクスの合計を確認するには、Account Metrics ディメンションを使用します。特定の API のメトリクスを検索するには、他のディメンションを使用し、Resource ディメンションの API オペレーションの名前を指定します。

メトリクス

メトリクス	説明
CallCount	アカウントで実行された指定されたオペレーションの数。 CallCount は、AWS/Usage と AWS/Logs の両方の名前空間で発行されます。
ErrorCount	アカウントで実行され、エラーが発生した API オペレーションの数。 ErrorCount は AWS/Logs にのみ発行されます。
ThrottleCount	アカウントで実行され、使用クォータによって調整された API オペレーションの数。 ThrottleCount は AWS/Logs にのみ発行されます。

ディメンション

ディメンション	説明
Account metrics	このディメンションを使用して、すべての CloudWatch Logs APIs の メトリクスの合計を取得します。 特定の API のメトリクスを表示するには、この表に示されている他のディメンションを使用して、API 名を Resource の値として指定します。

ディメンション	説明
Service	リソースを含む AWS サービスの名前。 CloudWatch Logs 使用状況メトリクスの場合、このディメンションの値は ですLogs。
Class	追跡されるリソースのクラス。 CloudWatch Logs API 使用状況メトリクスは、このディメンションを の値で使用しますNone。
Type	追跡されるリソースのタイプ。現在、Service ディメンションが Logs である場合、Type の有効な値は API のみです。
Resource	API オペレーションの名前。有効な値には、[アクション] にリストされているすべての API オペレーション名が含まれます。例えば、PutLogEvents

CloudWatch ログクォータ

次の表は、 AWS アカウントの CloudWatch ログの制限とも呼ばれるデフォルトのサービスクォータを示しています。これらのサービスクォータのほとんどは、すべてではありませんが、Service Quotas コンソールの Amazon CloudWatch Logs 名前空間に一覧表示されます。これらのクォータに対するクォータの引き上げをリクエストするには、このセクションの後半にある手順を参照してください。

リソース	デフォルトのクォータ
アカウントレベルのポリシー	アカウントごとに 1 つのアカウントレベルのサブスクリプションフィルターポリシー。 アカウントごとに 1 つのアカウントレベルのデータ保護ポリシー。 これらのクォータは変更できません。
異常ディテクター	アカウントあたり 10 個の異常ディテクター。このクォータは変更できません。
バッチサイズ	最大バッチサイズは 1,048,576 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。このクォータは変更できません。
データアーカイブ	データアーカイブは 5GB まで無料です。このクォータは変更できません。
CreateLogGroup	1 秒あたり 10 件のトランザクション (TPS/アカウント/リージョン) 。その後、トランザクションはスロットリングされます。クォータは、引き上げをリクエストすることができます。
CreateLogStream	50 件のトランザクション/秒 (TPS/アカウント/リージョン)。その後、トランザクションが調整されます。クォータは、引き上げをリクエストすることができます。

リソース	デフォルトのクォータ
カスタムデータ識別子	各データ保護ポリシーには、最大 10 個のカスタムデータ識別子を含めることができます。クォータは、引き上げをリクエストすることができます。 カスタムデータ識別子を定義する各正規表現には、最大 200 文字を含めることができます。このクォータは変更できません。
DeleteLogGroup	1 秒あたり 10 件のトランザクション (TPS/アカウント/リージョン) 。その後、トランザクションはスロットリングされます。クォータは、引き上げをリクエストすることができます。
DeleteLogStream	1 秒あたり 15 件のトランザクション (TPS/アカウント/リージョン) 。その後、トランザクションはスロットリングされます。クォータは、引き上げをリクエストすることができます。
DescribeLogGroups	1 秒あたり 10 件のトランザクション (TPS/アカウント/リージョン) 。クォータは、引き上げをリクエストすることができます。
DescribeLogStreams	1 秒あたり 25 件のトランザクション (TPS/アカウント/リージョン) 。クォータは、引き上げをリクエストすることができます。
検出されるログフィールド	CloudWatch Logs Insights は、ロググループ内で最大 1000 個のログイベントフィールドを検出できます。このクォータは変更できません。 詳細については、「サポートされるログと検出されるフィールド」を参照してください。

リソース	デフォルトのクォータ
抽出された JSON ログのフィールド	CloudWatch Logs Insights は、JSON ログから最大 200 個のログイベントフィールドを抽出できます。このクォータは変更できません。 詳細については、「サポートされるログと検出されるフィールド」を参照してください。
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。このクォータは変更できません。
FilterLogEvents	米国東部 (バージニア北部) で、1 秒あたり 25 件のリクエスト。 次のリージョンで 1 秒あたり 5 リクエスト： • アジアパシフィック (ジャカルタ) • アジアパシフィック (大阪) • 欧州 (フランクフルト) • カナダ西部 (カルガリー) • イスラエル (テルアビブ) 他のリージョンでは 1 秒あたり 10 リクエスト。 このクォータは変更できません。

リソース	デフォルトのクォータ
GetLogEvents	欧州 (パリ) で、1 秒あたり 30 件のリクエスト。 次のリージョンで、1 秒あたり 10 件のリクエスト: • 米国西部 (オレゴン) • アジアパシフィック (ジャカルタ) • アジアパシフィック (大阪) • カナダ西部 (カルガリー) • 欧州 (アイルランド) • 欧州 (フランクフルト) • イスラエル (テルアビブ) 他のすべてのリージョンで 1 秒あたり 25 リクエスト。 このクォータは変更できません。 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。このクォータは変更できません。
Live Tail の同時セッション。	15 の同時セッション。クォータは、引き上げをリクエストすることができます。
Live Tail: 1 回のセッションで検索されたロググループ。	1 回の Live Tail セッションでスキャンされるロググループの最大数は 10 です。このクォータは変更できません。
ログイベントサイズ	256 KB (最大)。このクォータは変更できません。

リソース	デフォルトのクォータ
ロググループ	1 アカウント、1 リージョンあたり 1,000,000 ロググループ。クォータは、引き上げをリクエスト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クォータはありません。
メトリクスフィルター	1 ロググループあたり 100。このクォータは変更できません。
組み込みメトリクス形式のメトリクス	ログイベントあたり 100 のメトリクスとメトリクスあたり 30 のディメンション。埋め込みメトリクス形式の詳細については、Amazon ユーザーガイドの「仕様: 埋め込みメトリクス形式 CloudWatch 」を参照してください。
PutLogEvents	PutLogEvents リクエストの最大バッチサイズは 1MB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。 1 秒、1 アカウント、1 リージョンあたり 5000 件のトランザクション Service Quotas サービスを使用して、1 秒あたりのスロットリングクォータの引き上げをリクエストできます。
クエリ実行タイムアウト	CloudWatch Logs Insights のクエリは 60 分後にタイムアウトします。この制限時間は変更できません。
クエリされたロググループ	1 つの Logs Insights クエリで最大 50 CloudWatch のロググループをクエリできます。このクォータは変更できません。

リソース	デフォルトのクォータ
クエリの同時実行数	標準クラスのロググループの場合、ダッシュボードに追加されたクエリを含め、最大 30 の同時 CloudWatch Logs Insights クエリ。 低頻度アクセスクラスのロググループの場合、ダッシュボードに追加されたクエリを含め、最大 5 つの CloudWatch 同時 Logs Insights クエリ。 これらのクォータは変更できません。
自然言語から生成されたクエリ	最大 5 つの同時自然言語生成クエリリクエスト。
クエリの可用性	コンソールで作成されたクエリは、[履歴] コマンドを使用して 30 日間使用できます。この使用可能期間は変更できません。 を使用して作成されたクエリ定義PutQueryDefinitionは期限切れになりません。
クエリ結果の使用可能期間	クエリの結果は 7 日間取得できます。この使用可能期間は変更できません。
コンソールに表示されるクエリ結果	デフォルトでは、最大 1000 行のクエリ結果がコンソールに表示されます。クエリで **`limit`** コマンドを使用すると、これを 10,000 行まで増やすことができます。詳細については、「CloudWatch Logs Insights クエリ構文」を参照してください。
正規表現	メトリックスフィルターまたはサブスクリプションフィルターを作成するとき、ロググループごとに正規表現を含む最大 5 つのフィルターパターン。このクォータは変更できません。 メトリックスフィルターとサブスクリプションフィルターの区切りまたは JSON フィルターパターンを作成するとき、またはログイベントをフィルタリングするとき、フィルターパターンごとに最大 2 つの正規表現。

リソース	デフォルトのクォータ
リソースポリシー	アカウントごとにリージョンごとに最大 10 CloudWatch Logs リソースポリシー。このクォータは変更できません。
保存されたクエリ	アカウントごとにリージョンごとに最大 1000 CloudWatch Logs Insights クエリを保存できます。このクォータは変更できません。
サブスクリプションフィルター	1 ロググループあたり 2。このクォータは変更できません。

CloudWatch Logs サービスクォータの管理

CloudWatch Logs は Service Quotas と統合されています。Service Quotas は、クォータを一元的に表示および管理できる AWS サービスです。詳細については、「Service Quotas ユーザーガイド」の「Service Quotas とは」を参照してください。

Service Quotas を使用すると、 CloudWatch Logs サービスクォータの値を簡単に検索できます。

AWS Management Console

コンソールを使用して CloudWatch Logs サービスクォータを表示するには

1. https://console.aws.amazon.com/servicequotas/ で Service Quotas コンソールを開きます。
2. ナビゲーションペインで、[AWS サービス] を選択します。
3. AWS サービスリストから Amazon CloudWatch Logs を検索して選択します。

   [Service Quotas] の一覧には、サービスクォータ名、適用された値 (使用可能な場合)、 AWS デフォルトのクォータ、クォータ値が調整可能かどうかが表示されます。
4. 説明など、Service Quotas に関する追加情報を表示するには、クォータ名を選択します。
5. (オプション) クォータの引き上げをリクエストするには、[Request quota increase (クォータ引き上げリクエスト)] を選択、または必要な情報を入力または選択して、[Request (リクエスト)] を選択します。

コンソールを使用してさらにサービスクォータの操作を行うには、Service Quotas ユーザーガイドを参照してください。クォータの引き上げをリクエストするには、「Service Quotas ユーザーガイド」の「クォータ引き上げリクエスト」を参照してください。

AWS CLI

を使用して CloudWatch Logs サービスクォータを表示するには AWS CLI

次のコマンドを実行して、デフォルトの CloudWatch Logs クォータを表示します。

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code logs \
    --output table
```

を使用してサービスクォータをさらに操作するには AWS CLI、Service Quotas AWS CLI コマンドリファレンス」を参照してください。クォータの引き上げをリクエストするには、「AWS CLI コマンド リファレンス」で `request-service-quota-increase` コマンドを参照してください。

ドキュメント履歴

次の表は、2018 年 6 月以降の CloudWatch 「 ログユーザーガイド」の各リリースにおける重要な変更点を示しています。このドキュメントの更新に関する通知を受け取るには、RSS フィードにサブスクライブできます。

変更	説明	日付
CloudWatch Logs Insights による自然言語クエリ生成のサポートは一般利用可能です	CloudWatch Logs Insights は、クエリを生成および更新するための自然言語をサポートしています。詳細については、「自然言語を使用して CloudWatch Logs Insights クエリを生成および更新する」を参照してください。	2024 年 6 月 20 日
CloudWatchLogsReadOnlyAccess ポリシーが更新されました	CloudWatch Logs はCloudWatchLogsReadOnlyAccess、このポリシーを持つユーザーが自然言語プロンプトから CloudWatch Logs Insights クエリ文字列を生成できるように、 にアクセス`cloudwatch:GenerateQuery` 許可を追加しました。	2023 年 11 月 26 日
CloudWatchLogsFullAccess ポリシーが更新されました	CloudWatch Logs はCloudWatchLogsFullAccess、このポリシーを持つユーザーが自然言語プロンプトから CloudWatch Logs Insights クエリ文字列を生成できるように、 にアクセス`cloudwatch:Generat`	2023 年 11 月 26 日

	eQuery 許可を追加しました。	
CloudWatch ログがログパターン分析を追加	CloudWatch ログは、Logs Insights クエリを実行するたびに CloudWatch ログイベントのパターンをスキャンするようになりました。詳細については、「パターン分析」を参照してください。	2023 年 11 月 26 日
CloudWatch ログがログ異常検出を追加する	ロググループのログ異常ディテクターを作成できます。異常ディテクターは、ロググループに取り込まれたログイベントをスキャンし、ログデータで異常を検出します。詳細については、「ログ異常検出」を参照してください。	2023 年 11 月 26 日
CloudWatch ログで比較機能を追加	CloudWatch Logs Insights を使用して、ログイベントの変更を経時的に比較できるようになりました。詳細については、「以前の時間範囲 と比較 (差分)」を参照してください。	2023 年 11 月 26 日

CloudWatch ログが新しいログクラスを追加する	CloudWatch Logs は 2 つのクラスのロググループをサポートしているため、アクセス頻度の低いログに対して費用対効果の高いオプションを使用でき、リアルタイムモニタリングやその他の機能を必要とするログに対してフル機能のオプションも使用できます。詳細については、「ログクラス」を参照してください。	2023 年 11 月 26 日
CloudWatch Logs Insights が自然言語クエリ生成をサポート	CloudWatch Logs Insights は、クエリを生成および更新するための自然言語をサポートしています。詳細については、「自然言語を使用して CloudWatch Logs Insights クエリを生成および更新する」を参照してください。	2023 年 11 月 26 日
CloudWatch ログに Live Tail の正規表現フィルターパターン構文のサポートが追加されました	ライブテールフィルターパターンで柔軟な正規表現を使用して、検索と一致操作をニーズに合わせてさらにカスタマイズできるようになりました。詳細については、「Amazon Logs ユーザーガイド」の「フィルターパターン構文」を参照してください。CloudWatch	2023 年 11 月 13 日

CloudWatch ログは、メトリクスフィルター、サブスクリプションフィルター、フィルターログイベントの正規表現フィルターパターン構文のサポートを追加します。	フィルターパターンで柔軟な正規表現を使用して、検索と一致操作をニーズに合わせてさらにカスタマイズできるようになりました。詳細については、「Amazon Logs ユーザーガイド」の「フィルターパターン構文」を参照してください。 CloudWatch	2023 年 9 月 5 日
CloudWatch Logs Insights がパターンコマンドを追加する	CloudWatch Logs Insights クエリで パターンを使用して、ログデータをパターンに自動的にクラスター化できるようになりました。パターンは、ログフィールド間で繰り返される共有テキスト構造です。詳細については、「Amazon Logs ユーザーガイド」の「パターン」を参照してください。 CloudWatch	2023 年 7 月 17 日
CloudWatch Logs Insights が重複排除コマンドを追加する	CloudWatch Logs Insights クエリで重複排除を使用して、指定したフィールドの特定の値に基づいて重複した結果を削除できるようになりました。詳細については、「Amazon Logs ユーザーガイドhttps://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/CWL_QuerySyntax-Dedup.html」の「重複排除」を参照してください。 CloudWatch	2023 年 6 月 20 日

アカウントレベルのデータ保護ポリシー	データ保護ポリシーをアカウントレベルで設定できるようになりました。アカウントレベルのポリシーでは、アカウント内のすべてのロググループの、ログイベントの機密情報を監査およびマスキングできます。詳細については、「Amazon CloudWatch Logs ユーザーガイド」の「マスキングによる機密ログデータの保護」を参照してください。	2023 年 6 月 8 日
Live Tail 機能が追加	CloudWatch ログに Live Tail 機能が追加されました。ログを取り込み中にスキャンしてトラブルシューティングに役立てることができます。また、指定した用語に基づいて、表示されるログイベントのストリームをフィルタリングしたり、指定した用語を含むログイベントを強調表示したりすることもできます。詳細については、「Use Live Tail to view logs in near real time」を参照してください。	2023 年 6 月 6 日

CloudWatchLogsReadOnlyAccess ポリシーが更新されました	CloudWatch に追加されたアクセス許可をログに記録しますCloudWatchLogsReadOnlyAccess。このポリシーを持つユーザーがコンソールを使用して CloudWatch Logs ライブテールセッションを開始および停止できるように、 `logs:StartLiveTail` および アクセス`logs:StopLiveTail` 許可が追加されました。詳細については、「ライブテールを使用してほぼリアルタイムでログを表示する」を参照してください。	2023 年 6 月 6 日
CloudWatch Logs Insights がリリースされました	CloudWatch Logs Insights を使用して、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、「Amazon Logs ユーザーガイド」の CloudWatch「Logs Insights を使用したログデータの分析」を参照してください。 CloudWatch	2018 年 11 月 27 日
Amazon VPC エンドポイントのサポート	これで、VPC と CloudWatch ログの間に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 ユーザーガイド」の「インターフェイス VPC エンドポイントでのログの使用」を参照してください。 CloudWatch	2018 年 6 月 28 日

次の表は、「Amazon CloudWatch Logs ユーザーガイド」の重要な変更点を示しています。

変更	説明	リリース日
インターフェイス VPC エンドポイント	リージョンによっては、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 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することができます。詳細については、「インターフェイス VPC エンドポイントでの CloudWatch ログの使用」を参照してください。	2018 年 3 月 7 日
Route 53 DNS クエリログ	CloudWatch ログを使用して、Route 53 が受信した DNS クエリに関するログを保存できます。詳細については、「Amazon CloudWatch Logs とは」または Amazon Route 53 デベロッパーガイドの「パブリック DNS クエリのログ記録」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 のロググループにタグを付ける」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、「メトリクスからログへのピボット」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、「Logs に送信された CloudWatch ログデータを表示する」を参照してください。	2016 年 8 月 29 日
Amazon CloudWatch Logs と新しい Logs CloudWatc	CloudWatch ログ AWS CloudTrail のサポートを追加しました。詳細については、「での CloudWatch Logs API およびコンソールオペレーションのログ記録 AWS CloudTrail」を参照してください。	2016 年 3 月 10 日

変更	説明	リリース日
h メトリクス AWS CloudTrail のサポートを追加		
Amazon S3 への CloudWatch ログのエクスポートのサポートを追加	Logs データを Amazon CloudWatch S3 にエクスポートするためのサポートが追加されました。Amazon S3 詳細については、「Amazon S3 へのログデータのエクスポート」を参照してください。	2015 年 12 月 7 日
Amazon CloudWatch Logs でログに AWS CloudTrail 記録されたイベントのサポートを追加	でアラームを作成し CloudWatch 、 によってキャプチャされた特定の API アクティビティの通知を受信 CloudTrail し、通知を使用してトラブルシューティングを実行できます。	2014 年 11 月 10 日
Amazon CloudWatch Logs のサポートを追加	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からシステム、アプリケーション、カスタムログファイルをモニタリング、保存、およびアクセスできます。その後、Amazon CloudWatch コンソール、 の CloudWatch Logs コマンド、または CloudWatch Logs SDK を使用して AWS CLI、関連するログデータを CloudWatch Logs から取得できます。詳細については、「Amazon CloudWatch Logs とは」を参照してください。	2014 年 7 月 10 日

AWS 用語集

最新の AWS 用語については、「 AWS の用語集 リファレンス」のAWS 「 用語集」を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。