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 Logs 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 サービスからのログを、スケーラブルな単一のサービスに一元化できます。これにより、ログ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 では、ソースにかかわらずすべてのログをイベントの 1 つの一貫した流れとして時間順に見ることができます。

CloudWatch Logs 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、フィルターや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ています。

CloudWatch Logs は 2 つのログクラスをサポートしています。CloudWatch Logs 標準ログクラスのロググループは、すべての CloudWatch Logs 機能をサポートしています。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

- 柔軟性を高める 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おり、アクセス頻度の低いログに対してコスト効率の高いオプションを利用できます。また、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。CloudWatch Logs Insights には、いくつかのシンプルで強力なコマンドを備え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。
- フィールドインデックスを作成してクエリをより効率的にする – ログイベントでフィールドのフィールドインデックスを作成できます。次に、CloudWatch Logs Insights クエリでフィールドイ

ンデックスを使用すると、クエリはインデックス付きフィールドを含まないことがわかっているログイベントの処理をスキップしようとします。このクエリはクエリのスキャン量を減らし、結果をより迅速に返すことができます。開始するには、「フィールドインデックスを作成してクエリのパフォーマンスを向上させ、スキャンボリュームを削減する」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、「CloudWatch Logs Live Tail を使用したトラブルシューティング」を参照してください。
- Amazon EC2 インスタンスのログをモニタリングする – CloudWatch Logs を使って、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 では、アプリケーションログに存在するエラーの数がトラッキングされ、エラー率が指定のしきい値を超えたときに管理者に通知が送信されます。お客様のログが CloudWatch Logs によるモニタリングに使用されるので、コードの変更は不要です。たとえば、アプリケーションログの特定のリテラルターム (例:「NullReferenceException」) をモニタリングしたり、ログデータの特定の場所でリテラルターム (例: Apache アクセスログの「404」ステータスコード) の発生数をカウントしたりできます。検索した語句が見つかると、CloudWatch Logs は指定された CloudWatch メトリクスにデータをレポートします。ログデータは、転送時や保管時に暗号化されます。開始するには、「CloudWatch Logs の開始方法」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 – CloudWatch でアラームを作成し、CloudTrail 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 Logs への CloudTrail Events 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。
- ログデータをアーカイブする – CloudWatch Logs を使用して高い耐久性のストレージにログデータを保存できます。CloudWatch Logs エージェントにより、ローテーションするログデータも

ローテーションしないログデータも、ホストからログサービスに簡単にすばやく送信できます。その後は、必要なときに生のログデータにアクセスできます。

- Route 53 DNS クエリのログ – CloudWatch Logs を使用して、Route 53 が受け取る DNS クエリに関するログ情報を使用できます。詳細については、Amazon Route 53 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

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

- AWS CloudTrail は、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、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 ログ記録がオンになると、CloudTrail は、API 呼び出しをアカウントにキャプチャし、指定する Amazon S3 バケットにログファイルを送信します。リクエストを満たすためにアクションをいくつ実行する必要があったかに応じて、各ログファイルには 1 個以上のレコードが含まれる可能性があります。詳細については AWS CloudTrail、「 AWS CloudTrail ユーザーガイド」の「 AWS CloudTrailとは」を参照してください。CloudWatch が CloudTrail のログファイルに書き込むデータのタイプの例については、「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。
- 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 を使用して、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 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。

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 Logs および CloudWatch のコストと使用状況を分析する方法、およびコストを削減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。

Amazon CloudWatch Logs の概念

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

ログクラス

CloudWatch Logs 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や頻繁にアクセスするログ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 2 つのプロパティがあります。イベント発生時のタイムスタンプおよび生のイベントメッセージです。イベントメッセージは UTF-8 でエンコードされている必要があります。

ログストリーム

ログストリームは、同じソースを共有する一連のログイベントです。より具体的には、ログストリームは一般的に、モニタリングされているアプリケーションインスタンスやリソースから送信された順序でイベントを表すものです。たとえば、ログストリームは特定のホストの Apache ア

クセスログと関連付けられる場合があります。ログストリームを必要としなくなった場合、aws logs delete-log-stream コマンドを使用して、それを削除できます。

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を使用して、取り込まれたイベントからメトリクスの監視データを抽出し、CloudWatch メトリクスのデータポイントに変換できます。メトリクスフィルタ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

保持設定は、CloudWatch Logs にログイベントを保持する期間を指定するために使用できます。期限切れのログイベントは自動的に削除されます。メトリクスフィルターと同様に、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。

にサインアップすると AWS、 無料AWS 利用枠を使用して 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 Logs には、2 つのクラスのロググループがあります。

- CloudWatch Logs 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や頻繁にアクセスするログに対するフル機能のオプションです。
- CloudWatch Logs 低頻度アクセスログクラスは、ログをコスト効率良く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、マネージド取り込み、ストレージ、クロスアカウントログ分析、暗号化など、CloudWatch Logs 機能のサブセットを提供し、GB あたりの取り込み料金は低くなります。低頻度アクセスログクラスは、アクセス頻度の低いログのアドホッククエリや事後フォレンジック分析に最適です。

> **Note**
>
> 標準ログクラスと低頻度アクセスログクラスの料金は、取り込みコストのみが異なります。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 の料金の詳細については、「Amazon CloudWatch Logs 料金表」をご覧ください。

> **Important**
>
> ロググループの作成後に、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。

機能	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	はい ✓	はい ✓	
クロスアカウント機能	はい ✓	はい ✓	
による暗号化 AWS KMS	はい ✓	はい ✓	

機能	規格	低頻度アクセス	
CloudWatch Logs Insights クエリコマンド	はい ✓	はい ✓ (ほとんどのコマンド – 「ログクラスでサポートされている Logs Insights QL コマンド」を参照)	
CloudWatch Logs Insights 検出フィールド	はい ✓	はい ✓	
OpenSearch PPL または OpenSearch SQL を使用して CloudWatch Logs Insights でクエリを実行する	はい ✓	いいえ	
自然言語クエリアシスト	はい ✓	いいえ	
CloudWatch Logs 異常検出	はい ✓	いいえ	
Live Tail	はい ✓	いいえ	
フィールドインデックス作成	はい ✓	いいえ	
以前の時間範囲と比較する	はい ✓	いいえ	
サブスクリプションフィルター	はい ✓	いいえ	
Amazon S3 へのエクスポート	はい ✓	いいえ	

機能	規格	低頻度アクセス	
GetLogEvents および FilterLogEvents API オペレーション	はい ✓	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。	
メトリクスフィルター	はい ✓	いいえ	
Container Insights ログの取り込み	はい ✓	いいえ	
Lambda Insights ログの取り込み	はい ✓	いいえ	
機密性の高いデータをマスキングで保護する	はい ✓	いいえ	
埋め込みメトリクス形式	はい ✓	いいえ	

> **Note**
>
> これら 2 つのログクラスに加えて、`Delivery`ログクラスがあります。`Delivery` ログクラスは、 Amazon S3 または に保存する AWS Lambda ログの配信にのみ使用します Amazon Data Firehose。配信クラスのロググループのログイベントは、1 CloudWatch Logs 日だけ保持されます。このログクラスは、 CloudWatch Logs Insights クエリなどの豊富な CloudWatch Logs 機能を提供しません。

CloudWatch Logs 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 CloudWatch Logs にログを収集するには、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 メトリクスを収集する場合、ゲスト内の可視性のために、追加のシステムメトリクスを収集できます。また、`StatsD` または `collectd` を使用して、カスタムメトリクスを収集することもできます。

詳細については、Amazon CloudWatch ユーザーガイドの「CloudWatch エージェントのインストール」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収集のみをサポートする古い CloudWatch Logs 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 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、次の手順を実行して作成します。

にサインアップするには AWS アカウント

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

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

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

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

管理アクセスを持つユーザーを作成する

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

を保護する 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、「 AWS IAM Identity Center ユーザーガイド」の「デフォルトを使用してユーザーアクセスを設定する IAM アイデンティティセンターディレクトリ」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、「 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。 AWS サインイン

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「グループの結合」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 CloudWatch Logs オペレーション AWS CLI 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」を参照してください。 AWS Command Line Interface

統合された CloudWatch エージェントを使用して CloudWatch Logs を使用する

統合された CloudWatch エージェントを使用して CloudWatch Logs を開始する方法については、Amazon CloudWatch ユーザーガイドの「CloudWatch エージェントを使用した Amazon EC2 Instances 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。このセクションに記載されている手順を実行してエージェントのインストールと設定を行い、開始します。エージェントを使用して CloudWatch メトリクスを収集していない場合、メトリクスを参照するセクションはすべて無視できます。

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

前の CloudWatch ログエージェントを使用して CloudWatch ログを開始する方法

> **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。
> ログのみの古い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に、その使用方法について説明します。

CloudWatch Logs エージェントを使用して、Linux または Windows Server を実行する Amazon EC2 インスタンスのログデータおよび AWS CloudTrailから記録されたイベントを発行できます。代わりに CloudWatch 統合エージェントを使用してログデータを発行することをお勧めします。新しいエージェントの詳細については、Amazon CloudWatch User Guideの「CloudWatch エージェントを使用

した Amazon EC2 Instances 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口グの収集」を参照してください。

内容

CloudWatch Logs エージェントの前提条件

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

- Amazon Linux バージョン 2014.03.02 以降。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廃止する予定です。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ックスの両方を収集できる統合エージェントが含まれています。詳細については、「CloudWatch Logs 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 エージェントでは、インスタンスメタデータサービスバージョン 2 (IMDSv2) がサポートされていません。サーバーで 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に、その使用方法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 エージェントをすでに使用しなくなった場合は、新しい統合 CloudWatch エージェントを使用することをお勧めします。詳細については、「CloudWatch Logs の開始方法」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。サーバーで 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 CloudWatch Logs API を使用して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を使用して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、または独自のログ発行アプリケーションを構築してログデータを発行する場合に適しています。

ステップ 1: CloudWatch Logs で使用する IAM 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。

IAM ロールまたは CloudWatch Logs 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 Roles (ロール) を選択してください。
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 Logs をインストールして設定する

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

既存の 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 から、awslogs パッケージを使用した RPM のインストールに CloudWatch Logs エージェントが使用可能です。それ以前のバージョンの Amazon Linux は、`sudo yum update -y` コマンドを使用してインスタンスを更新することで awslogs パッケージにアクセスできます。CloudWatch Logs インストーラを使用して ではなく awslogs パッケージを RPM 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ます。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。設定に問題が発生して CloudWatch Logs エージェントが CloudWatch にログを送信できなくなる恐れがあります。

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

   接続問題の詳細については、「Amazon EC2 ユーザーガイド」「Amazon EC2 Linux インスタンスへの接続に関する問題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 エージェントのリファレンス」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. エージェントが実行されてしばらくしたら、CloudWatch コンソールに新しく作成されたロググループとログストリームを確認してください。

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

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

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

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

   接続問題の詳細については、「Amazon EC2 ユーザーガイド」「Amazon EC2 Linux インスタンスへの接続に関する問題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サポートされていません。

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

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

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

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

前のコマンドが機能しない場合は、以下を試してください。

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

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

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

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

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

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

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

項目	説明
AWS アクセスキー ID	IAM ロールを使用する場合は Enter キーを押します。それ以外の場合は、 AWS アクセスキー ID を入力します。
AWS シークレットアクセスキー	IAM ロールを使用する場合は Enter キーを押します。それ以外の場合は、 AWS シークレットアクセスキーを入力します。
デフォルトリージョン名	Enter キーを押します。デフォルトは 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 エージェントのリファレンス」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. エージェントが実行されてしばらくしたら、CloudWatch コンソールに新しく作成されたロググループとログストリームを確認してください。

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

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

> **Tip**
>
> このセクションで説明している古い 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、CloudWatch エージェントのインストールを参照してください。
> このセクションの残りの部分では、古い CloudWatch Logs エージェントの使用について説明します。

起動時における EC2 Linux インスタンスへの古い CloudWatch Logs エージェントのインストール

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

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

前提条件

ロググループとログストリームをすべて記述したエージェントの設定ファイルを作成します。これは、モニタリングするログファイルとそのアップロード先のロググループおよびログストリームが記述されているテキストファイルです。エージェントはこの設定ファイルを使用して記述されたすべてのログファイルのモニタリングおよびアップロードを開始します。エージェント設定ファイルの設定の詳細については、「CloudWatch Logs エージェントのリファレンス」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。カスタムポリシー作成の詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 のアイデンティティベースのポリシー」を参照してください。
4. [Review Policy] ページで、[Policy Name] にポリシーの名前を入力します。
5. [Policy Document] に、次のポリシーをコピーして貼り付けます。

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

6. [ポリシーを作成] 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 Logs 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. [インスタンスの作成] を選択してください。

    詳細については、「Amazon EC2 ユーザーガイド」の「Launching an Instance」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 用の AMIs には cloud-init が既に含まれていますが、 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://amzn-s3-demo-bucket/my-config-file
    ```
6. そのほかインスタンスへの必要な変更を行い、起動設定を確認して [Launch] を選択します。

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

   詳細については、「CloudWatch Logs に送信されたログデータを表示する」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 インスタンスで CloudWatch Logs エージェントを使用してログを CloudWatch Logs に送信できるようにする

> **Tip**
>
> CloudWatch には、EC2 インスタンスとオンプレミスサーバーからログとメトリクスの両方を収集できる新しい統合エージェントが含まれています。新しい統合 CloudWatch エージェントを使用することをお勧めします。詳細については、「CloudWatch Logs の開始方法」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 インスタンスで、古い CloudWatch Logs エージェントを使用して CloudWatch Logs にログを送信できるようにする

Windows Server 2016 を実行しているインスタンスで CloudWatch Logs にログを送信するには、複数の方法を使用できます。このセクションのステップでは、Systems Manager Run Command を使用します。他の可能な方法の詳細については、「Amazon CloudWatch へのログ、イベント、パフォーマンスカウンターの送信」を参照してください。

ステップ

- サンプル設定ファイルをダウンロードする
- CloudWatch の JSON ファイルを設定する
- 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 Logs を有効化する

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。

CloudWatch の JSON ファイルを設定する

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

ステップ

- ステップ 1: CloudWatch Logs を有効化する
- ステップ 2: CloudWatch 設定を構成する
- 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 Logs を有効化する

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

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

ステップ 2: CloudWatch 設定を構成する

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

ログデータを CloudWatch Logs に送信するように設定するには

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

```
{
    "Id": "CloudWatchLogs",
```

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

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

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

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

ステップ 3: 送信するデータを設定する

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

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

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

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

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

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

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

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

セキュリティログデータを CloudWatch Logs に送信するには

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

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

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

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

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

```
{
    "Id": "SystemEventLog",
```

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

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

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

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

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

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

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

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

   a. イベントビューワーを開きます。

   b. ナビゲーションペインで、[Applications and Services Logs] を選択します。

c. ログに移動し、[Actions]、[Properties] を選択します。

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

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

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

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

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

1. JSON ファイルで、ETW セクションを見つけます。

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

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

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

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

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

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

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

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

> **Important**
>
> ソースログファイルには、各ログ行の先頭にタイムスタンプがあり、タイムスタンプの後にスペースがある必要があります。

4. `Encoding` には、使用するファイルエンコード (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。サポートされる値の詳細については、MSDN の「FileSystemWatcherFilter プロパティ」を参照してください。
6. (オプション) CultureName には、タイムスタンプが記録されているロケールを入力します。CultureName が空の場合、Windows インスタンスにより現在使用されているのと同じロケールがデフォルトになります。詳細については、MSDN の「Product Behavior」トピックの表で、Language tag 列を参照してください。

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

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

IIS ログデータを CloudWatch Logs に送信するには

1. JSON ファイルで、IISLog セクションを見つけます。

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

```
        "TimeZoneKind": "UTC",
        "LineCount": "5"
    }
},
```

2. LogDirectoryPath には、個々のサイト (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。サポートされる値の詳細については、MSDN の「FileSystemWatcherFilter プロパティ」を参照してください。
6. (オプション) CultureName には、タイムスタンプが記録されているロケールを入力します。CultureName が空の場合、Windows インスタンスにより現在使用されているのと同じロケールがデフォルトになります。サポートされる値の詳細については、MSDN の「Product Behavior」トピックの表で、Language tag 列を参照してください。

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

7. (オプション) TimeZoneKind には、Local または UTC を入力します。これを設定すると、ログのタイムスタンプにタイムゾーン情報が含まれていない場合にタイムゾーン情報を提供できます。このパラメータが空になっていて、タイムスタンプにタイムゾーン情報が含まれていない場

合、CloudWatch Logs ではデフォルトでローカルタイムゾーンが使用されます。タイムスタンプに既にタイムゾーン情報が含まれている場合、このパラメータは無視されます。

8. (オプション) LineCount には、ログファイルを識別するためのヘッダーの行数を入力します。たとえば、IIS のログファイルのヘッダーはほぼ同じです。「**5**」と入力すると、ログファイルのヘッダーの最初の 5 行が読み取られ、ログファイルを識別できます。IIS ログファイルで、3 番目の行は日時のスタンプですが、タイムスタンプは常にログファイル間で異なるという保証はありません。そのため、ログファイルの一意のフィンガープリントを作成するには、実際のログデータの少なくとも 1 行を含めることをお勧めします。

ステップ 4: フロー制御を設定する

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

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

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

フロー制御を設定するには

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

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

```
    ]
}
```

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

ステップ 5: JSON コンテンツを保存する

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

Systems Manager 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への IAM ロールのアタッチについては、「Amazon EC2 ユーザーガイド」の「Attaching an IAM Role to an Instance」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 Logs との統合を設定する前に、インスタンスが最低要件を満たしていることを確認してください。詳細については、AWS Systems Manager ユーザーガイドの「Systems Manager の前提条件」を参照してください。

インターネットアクセスを確認する

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

Systems Manager Run Command を使用して CloudWatch Logs を有効化する

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 Logs 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. [Command document] で、[AWS-ConfigureCloudWatch] を選択します。
6. [Target instances (ターゲットインスタンス)] で、CloudWatch Logs と統合するインスタンスを選択します。このリストに表示されていないインスタンスは、Run Command 用に設定されていない場合があります。詳細については、「Amazon EC2 ユーザーガイド」の「Systems Manager Prerequisites」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 インスタンスで、CloudWatch Logs へのログの送信を有効にする

> **Tip**
>
> CloudWatch には、EC2 インスタンスとオンプレミスサーバーからログとメトリクスの両方を収集できる新しい統合エージェントが含まれています。新しい統合 CloudWatch エージェントを使用することをお勧めします。詳細については、「CloudWatch Logs の開始方法」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、CloudWatch Logs へのログの送信を有効にする

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

サンプル設定ファイル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。

CloudWatch の JSON ファイルを設定する

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

ステップ

- ステップ 1: CloudWatch Logs を有効化する
- ステップ 2: CloudWatch 設定を構成する
- 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 Logs を有効化する

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

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

ステップ 2: CloudWatch 設定を構成する

認証情報、リージョン、ロググループ名、およびログストリーム名前空間を指定します。これにより、インスタンスがログデータを CloudWatch Logs に送信できます。同じログデータを複数の異な

る宛先に送信する場合は、一意の ID (たとえば「CloudWatchLogs2」および「CloudWatchLogs3」) および各 ID に異なるリージョンを付加してセクションを追加できます。

ログデータを CloudWatch Logs に送信するように設定するには

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

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

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

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

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

ステップ 3: 送信するデータを設定する

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

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

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

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

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

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

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

セキュリティログデータを CloudWatch Logs に送信するには

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

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

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

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

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

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

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

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

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

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

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

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

2. Id には、アップロードするログの名前を入力します (たとえば、**WindowsBackup**)。

3. LogName には、アップロードするログの名前を入力します。ログの名前を次のように確認できます。

   a. イベントビューワーを開きます。

   b. ナビゲーションペインで、[Applications and Services Logs] を選択します。

   c. ログに移動し、[Actions]、[Properties] を選択します。

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

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

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

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

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

1. JSON ファイルで、ETW セクションを見つけます。

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

2. LogName には、アップロードするログの名前を入力します。

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

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

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

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

1. JSON ファイルで、CustomLogs セクションを見つけます。

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

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

> **Important**
>
> ソースログファイルには、各ログ行の先頭にタイムスタンプがあり、タイムスタンプの後にスペースがある必要があります。

4. `Encoding` には、使用するファイルエンコード (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。サポートされる値の詳細については、MSDN の「FileSystemWatcherFilter プロパティ」を参照してください。

6. (オプション) `CultureName` には、タイムスタンプが記録されているロケールを入力します。`CultureName` が空の場合、Windows インスタンスにより現在使用されているのと同じロケールがデフォルトになります。サポートされる値の詳細については、MSDN の「Product Behavior」トピックの表で、`Language tag` 列を参照してください。

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

7. (オプション) `TimeZoneKind` には、`Local` または `UTC` を入力します。これを設定すると、ログのタイムスタンプにタイムゾーン情報が含まれていない場合にタイムゾーン情報を提供できます。このパラメータが空になっていて、タイムスタンプにタイムゾーン情報が含まれていない場合、CloudWatch Logs ではデフォルトでローカルタイムゾーンが使用されます。タイムスタンプに既にタイムゾーン情報が含まれている場合、このパラメータは無視されます。

8. (オプション) `LineCount` には、ログファイルを識別するためのヘッダーの行数を入力します。たとえば、IIS のログファイルのヘッダーはほぼ同じです。「**5**」と入力すると、ログファイルのヘッダーの最初の 3 行が読み取られ、ログファイルを識別できます。IIS ログファイルで、3 番目の行は日時のスタンプですが、タイムスタンプは常にログファイル間で異なるという保証はありません。そのため、ログファイルの一意のフィンガープリントを作成するには、実際のログデータの少なくとも 1 行を含めることをお勧めします。

IIS ログデータを CloudWatch Logs に送信するには

1. JSON ファイルで、`IISLog` セクションを見つけます。

```
{
    "Id": "IISLogs",
```

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

2. LogDirectoryPath には、個々のサイト (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。サポートされる値の詳細については、MSDN の「FileSystemWatcherFilter プロパティ」を参照してください。
6. (オプション) CultureName には、タイムスタンプが記録されているロケールを入力します。CultureName が空の場合、Windows インスタンスにより現在使用されているのと同じロケールがデフォルトになります。サポートされる値の詳細については、MSDN の「Product Behavior」トピックの表で、Language tag 列を参照してください。

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

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

ステップ 4: フロー制御を設定する

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

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

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

フロー制御を設定するには

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

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

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

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

エージェントを起動する

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 Logs にログを送信できるようにするには、EC2 Config サービス (EC2Config.exe)) を使用します。インスタンスには EC2Config 4.0 以降が必要であり、この手順を使用できます。

EC2 Config 4.x を使用して CloudWatch を設定するには

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

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

SSM エージェントは、再起動後に設定ファイルを検出し、CloudWatch 統合のためにインスタンスを設定します。ローカル設定ファイルのパラメータと設定を変更する場合は、変更を反映するために

SSM エージェントを再起動する必要があります。インスタンスで CloudWatch 統合を無効にする場合は、`IsEnabled` を `false` に変更して、設定ファイルで変更を保存します。

クイックスタート: 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。クックブックの例は (CloudWatchLogs-Cookbooks.zip) からもダウンロードできます。

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 Logs 設定ファイルを作成するには、

```
#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 エージェントのリファレンス」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. AWS OpsWorks コンソールを https://console.aws.amazon.com/opsworks/://https://www..com で開きます。
2. OpsWorks Dashboard で、スタックの追加を選択して 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 Logs 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。カスタムポリシー作成の詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 のアイデンティティベースのポリシー」を参照してください。
4. [Review Policy] ページで、[Policy Name] にポリシーの名前を入力します。
5. [Policy Document] に、次のポリシーをコピーして貼り付けます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
```

```
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
    ],
    "Resource": [
      "arn:aws:logs:*:*:*"
    ]
  }
 ]
}
```

6. [ポリシーを作成] 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に 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. AWS OpsWorks コンソールを https://console.aws.amazon.com/opsworks/:///https://https://https://https://https
2. ナビゲーションペインで [Layers] を選択します。
3. コンテンツペインでレイヤーを選択し、[Add layer] を選択します。
4. [OpsWorks] タブの [Layer type] で [Custom] を選択します。
5. [Name] および [Short name] フィールドにレイヤーの長い名前と短い名前を入力し、[Add layer] を選択します。

6. レシピタブのカスタム Chef レシピには、ライフサイクル AWS OpsWorks イベントに対応する設定、設定、デプロイ、デプロイ解除、シャットダウンの見出しがいくつかあります。 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. AWS OpsWorks コンソールを https://console.aws.amazon.com/opsworks/://www.com で開きます。
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 Logs を設定します。準備ができると、インスタンスのステータスがオンラインに変更されます。

ステップ 6: ログを表示する

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

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

CloudWatch Logs エージェントの状態を報告する

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

エージェントのステータスをレポートするには

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

   接続問題の詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 Linux インスタンスへの接続に関する問題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 エージェントのエラー、警告、問題を確認してください。

CloudWatch Logs エージェントを起動する

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

エージェントを開始するには

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

   接続問題の詳細については、「Amazon EC2 ユーザーガイド」「Amazon EC2 Linux インスタンスへの接続に関する問題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、以下の手順を使用します。

エージェントを停止するには

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

   接続問題の詳細については、「Amazon EC2 ユーザーガイド」「Amazon EC2 Linux インスタンスへの接続に関する問題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 Logs 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場合は 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 Logs 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	Tools for PowerShell のコード例
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 Logs の固有の例については、「SDK を使用した CloudWatch Logs のコード例 AWS SDKs」を参照してください。

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 のログデータをインタラクティブに検索し分析することができます。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、CloudWatch Logs Insights 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することができます。

CloudWatch Logs Insights は、クエリに使用できる 3 つのクエリ言語をサポートしています。

- 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 Logs Insights クエリ言語 (Logs Insights QL)。
- （新規) OpenSearch Service Piped Processing Language (PPL)。OpenSearch PPL を使用すると、パイプ (|) で区切られた一連のコマンドを使用してログを分析できます。

  OpenSearch PPL では、パイプ処理されたコマンドを使用してデータを取得、クエリ、分析できるため、複雑なクエリの理解と作成が容易になります。構文を使用すると、コマンドの連鎖によってデータを変換して処理できます。PPL を使用すると、データをフィルタリングして集計し、豊富な数学、文字列、日付、条件付き、その他の関数のセットを分析に使用できます。
- （新規) OpenSearch Service 構造化クエリ言語 (SQL)。OpenSearch SQL クエリを使用すると、宣言的な方法でログを分析できます。SELECT、FROM、WHERE、GROUP BY、HAVING などのコマンドや、SQL で使用できるその他のさまざまなコマンドや関数を使用できます。ロググループ間で JOINs を実行し、サブクエリを使用してログ間でデータを関連付け、豊富な JSON、数学、文字列、条件付き、その他の SQL 関数を使用してログに対して強力な分析を実行できます。

> **Important**
>
> CloudWatch Logs Insights クエリのセキュリティを強化するには、2025 年 7 月 31 日以降、CloudWatch コンソールでクエリを実行できるようにするには、 `logs:StartQuery` との両方の`logs:GetQueryResults`アクセス許可でユーザーがサインオンする必要があります。この日以降、これらのアクセス許可の両方を持たないユーザーは、コンソールでクエリ結果を表示できず、代わりにコンソールでクエリ結果を表示しようとするとバナーメッセージが表示されます。
> 変更後、必要なコンソールエクスペリエンスのアクセス許可は、SDK および StartQuery API と GetQueryResults API を使用する AWS CLI ユーザーに現在必要なアクセス許可と一致します。 APIs

IAM ポリシーを作成する方法、または既存のポリシーにアクセス許可を追加する方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーを使用したカスタム IAM アクセス許可の定義」および「IAM ポリシーの編集」を参照してください。

CloudWatch Logs Insights には、任意のクエリ言語で使用できる以下の機能があります。

- Amazon Route 53、、Amazon VPC などの AWS サービスからのログのログフィールド、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の自動検出。 AWS Lambda AWS CloudTrail
- フィールドインデックスを作成してコストを削減し、特に多数のロググループやログイベントのクエリで結果を高速化します。ログイベントで一般的なフィールドのフィールドインデックスを作成したら、クエリで で使用できます。クエリは、インデックス付きフィールドを含まないことが知られているログイベントの処理をスキップし、より少ないデータを処理します。

  > **Note**
  >
  > `filterIndex` コマンドは Logs Insights QL でのみ使用できます。

- ログイベントのパターンの検出と分析。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。
- クエリの保存、クエリ履歴の表示、保存されたクエリの再実行。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。
- ダッシュボードへのクエリの追加。
- によるクエリ結果の暗号化 AWS Key Management Service。

次の CloudWatch Logs Insights 機能は、Logs Insights QL を使用する場合にのみサポートされます。

- 自然言語を使用したクエリ生成。
- 低頻度アクセスログクラスのログのクエリ。
- ロググループのログイベントと以前の期間のログイベントを比較する比較クエリ。
- *`filterIndex` コマンド*。指定したフィールドインデックスを含むログイベントのみをクエリがスキャンするように強制します。

> **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はアクセスすることができません。

CloudWatch の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、このモニタリングアカウントにリンクされているソースアカウントのロググループで 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 の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。

Logs Insights QL を使用してクエリを作成する場合、自然言語を使用して CloudWatch Logs Insights 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。AI 支援機能が働いて、プロンプトに基づいてクエリが生成され、クエリの仕組みを説明した文が 1 行ずつ表示されます。詳細については、「Use natural language to generate and update CloudWatch Logs Insights queries」を参照してください。

サポートされているクエリ言語のいずれかを使用したクエリは、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 日間使用できます。

CloudWatch Logs Insights クエリには、クエリ言語に関係なく、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 料金表」をご覧ください。

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**
>
> ネットワークセキュリティチームがウェブソケットの使用を許可しない場合は、現在 CloudWatch 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。API を使用して CloudWatch Logs Insights のクエリ機能を使用できます。詳細については、Amazon CloudWatch Logs API リファレンスの「StartQuery」を参照してください。

内容

- フィールドインデックスを作成してクエリのパフォーマンスを向上させ、スキャンボリュームを削減する
- パターン分析
- CloudWatch Logs Insights クエリの保存と再実行
- 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
- 実行中のクエリまたはクエリ履歴を表示する
- でクエリ結果を暗号化する AWS Key Management Service

サポートされているクエリ言語

以下のセクションでは、各クエリ言語でサポートされているコマンドを一覧表示します。また、構文形式について説明し、サンプルクエリを提供します。

トピック

- CloudWatch Logs Insights クエリ言語 (Logs Insights QL)
- OpenSearch PPL 言語
- OpenSearch SQL 言語

CloudWatch Logs Insights クエリ言語 (Logs Insights QL)

このセクションには、Logs Insights QL コマンドと関数の完全なドキュメントが含まれています。また、この言語のサンプルクエリも含まれています。

トピック

- CloudWatch Logs Insights 言語クエリ構文
- Logs Insights QL の開始方法: クエリチュートリアル
- サンプルクエリ
- (diff) を以前の時間範囲と比較する
- グラフでログデータを視覚化する
- 自然言語を使用した CloudWatch Logs Insights クエリの生成と更新

CloudWatch Logs Insights 言語クエリ構文

このセクションでは、Logs Insights QL の詳細について説明します。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。

Important

大規模なクエリを実行して過剰な料金が発生するのを防ぐため、次のベストプラクティス留意してください。

- 各クエリに必要なロググループのみを選択します。
- クエリには、常にできるだけ狭い時間範囲を指定します。
- コンソールを使用してクエリを実行する場合は、CloudWatch Logs Insights コンソールページを閉じる前にすべてのクエリをキャンセルします。それ以外の場合、クエリは完了するまで実行され続けます。
- CloudWatch Logs Insights ウィジェットをダッシュボードに追加するときは、更新ごとに新しいクエリが開始されるため、ダッシュボードが高頻度で更新されないようにしてください。

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で検出し、@ 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「サポートされるログと検出されるフィールド」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 Logs Insights QL 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 Logs Insights QL クエリコマンドをサポートしますunmask。

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。
filterIndex	フィールドインデックスに記載されているフィールドでインデックス作成され、そのフィールドインデックスの値も含まれているロググループのみをスキャンするようにクエリを強制します。これにより、このフィールドインデックスのクエリで指定された値を含むこれらのロググループのログイベントのみをスキャンしようとすることで、スキャンされたボリュームが減少します。 このコマンドは、低頻度アクセスログクラスのロググループではサポートされていません。
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、ログイベントで検出され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検出されたログイベントと、以前の期間と同じ長さのログイベントを比較して、傾向を検索し、特定のログイベントが新しいかどうかを確認することができます。

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。
SOURCE	をクエリSOURCEに含めることは、クエリに含めるロググループ名のプレフィックス、アカウント識別子、およびロググループクラスに基づいて大量のロググループを指定するのに役立ちます。このコマンドは、CloudWatch コンソールではなく、 AWS CLI またはプログラムでクエリを作成する場合にのみサポートされます。
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。
unnest	入力として取得したリストをフラット化して、リスト内の各要素に対して 1 つのレコードを持つ複数のレコードを生成します。
その他の演算と関数	CloudWatch Logs Insightsは、比較、計算、日時、数値、文字列、IP アドレス、一般的な関数と演算も多数サポートしています。

以下のセクションでは、CloudWatch Logs Insights のクエリコマンドについてさらに詳しく説明します。

トピック

- ログクラスでサポートされている Logs Insights QL コマンド
- display

- fields
- フィルター
- filterIndex
- SOURCE
- pattern
- diff
- parse
- sort
- stats
- 制限
- 重複排除
- マスクを外す
- ネスト解除
- 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

ログクラスでサポートされている Logs Insights QL コマンド

すべての Logs Insights QL 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`、`filterIndex`、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。

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
```

インデックス付きフィールドとフィルターコマンド

ロググループのフィールドインデックスを作成した場合は、それらのフィールドインデックスを活用してfilterクエリをより効率的にし、スキャンされたボリュームを減らすことができます。たとえば、 のフィールドインデックスを作成したとしますrequestId。次に、 を含む、filter requestId = *value*または を含むロググループの CloudWatch Logs Insights クエリfilter requestId IN [*value*, *value*, ...]は、インデックス付きフィールドを含まないことがわかっているログイベントの処理をスキップしようとします。そのインデックス付きフィールドが含まれていることがわかっているログイベントのみをスキャンしようとすると、スキャンボリュームを減らすことができ、クエリが高速になります。

フィールドインデックスとその作成方法の詳細については、「」を参照してくださいフィールドインデックスを作成してクエリのパフォーマンスを向上させ、スキャンボリュームを削減する。

> **Important**
>
> フィールドインデックスの改善の恩恵を受けるのfilter *fieldName* IN...は、 filter *fieldName* =...と を持つクエリのみです。を使用したクエリではインデックスは使用filter *fieldName* likeされず、選択したロググループのすべてのログイベントを常にスキャンします。

例: インデックスを使用して、特定のリクエスト ID に関連するログイベントを検索する

この例では、 でフィールドインデックスを作成していることを前提としていますrequestId。このフィールドインデックスを使用するロググループの場合、クエリはフィールドインデックスを活用して、ログイベントの最小量をスキャンして、値が のイベントを見つけようとrequestIdします。123456

```
fields @timestamp, @message
| filter requestId = "1234656"
| 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ロ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、次のコードスニペットは、フィールド logGroup が完全な文字列 example_group であるロ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ニ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/
```

filterIndex

クエリで指定したフィールドでインデックスが作成されたロググループのみをスキャンするようにクエリに強制することで、インデックスが作成されたデータのみをfilterIndex返すには、 を使用します。このフィールドでインデックスが作成されたこれらのロググループの場合、インデックスが作成されたフィールドのクエリで指定された フィールドを含むログイベントがないロググループをスキップすることで、クエリをさらに最適化します。このフィールドインデックスのクエリで指定された値と一致するロググループのログイベントのみをスキャンしようとすることで、スキャンされたボリュームをさらに削減します。フィールドインデックスとその作成方法の詳細については、「」を参照してくださいフィールドインデックスを作成してクエリのパフォーマンスを向上させ、スキャンボリュームを削減する。

インデックス付きフィールドfilterIndexで を使用すると、フィールドインデックスを持つロググループとログイベントに実際の検索スペースを制限することで、ペタバイトのログデータを含むロググループを効率的にクエリできます。

例えば、アカウントの一部のロググループIPaddressで のフィールドインデックスを作成したとします。その後、次のクエリを作成し、アカウント内のすべてのロググループをクエリして、IPaddress フィールド198.51.100.0の値を含むログイベントを検索できます。

```
fields @timestamp, @message
| filterIndex IPaddress = "198.51.100.0"
| limit 20
```

filterIndex コマンドにより、このクエリは のインデックスが作成されていないすべてのロググループをスキップしようとしますIPaddress。さらに、インデックスが作成されたロググループ内では、クエリは IPaddressフィールドを持つが、そのフィールドの値198.51.100.0として観測されないログイベントをスキップします。

IN 演算子を使用して、インデックス付きフィールドの複数の値のいずれかに結果を展開します。次の の例では、 IPaddress フィールド198.51.100.1で 値198.51.100.0または を含むログイベントを検索します。

```
fields @timestamp, @message
| filterIndex IPaddress in ["198.51.100.0", "198.51.100.1"]
| limit 20
```

filterIndex と filter の比較

filterIndex と の違いを説明するためにfilter、次のクエリ例を検討してください。のフィールドインデックスを 4 つのロググループに対して作成したがIPaddress、5 番目のロググループに対しては作成していないとします。次の を使用したクエリfilterIndexでは、 フィールドのインデックスが作成されていないロググループのスキャンはスキップされます。インデックス付きロググループごとに、インデックス付きフィールドを持つログイベントのみをスキャンしようとします。また、フィールドインデックスの作成後に からの結果のみを返します。

```
fields @timestamp, @message
| filterIndex IPaddress = "198.51.100.0"
| limit 20
```

対照的に、同じ 5 つのロググループのクエリfilterfilterIndexの代わりに を使用する場合、クエリはインデックス付きロググループの値を含むログイベントだけでなく、インデックス化されていない 5 番目のロググループもスキャンし、その 5 番目のロググループ内のすべてのログイベントをスキャンします。

```
fields @timestamp, @message
| filter IPaddress = "198.51.100.0"
| limit 20
```

SOURCE

クエリSOURCEに を含めることは、 AWS CLI または API を使用してクエリを作成するときに、クエリに含めるロググループを指定するのに役立ちます。SOURCE コマンドは、CloudWatch コンソールではなく、 AWS CLI および API でのみサポートされています。CloudWatch コンソールを使用してクエリを開始するときは、コンソールインターフェイスを使用してロググループを指定します。

SOURCE を使用してクエリするロググループを指定するには、次のキーワードを使用できます。

- namePrefix は、指定した文字列で始まる名前を持つロググループに対してクエリを実行します。これを省略すると、すべてのロググループがクエリされます。

  リストには最大 5 つのプレフィックスを含めることができます。
- accountIdentifiers は、指定された AWS アカウントのロググループに対してクエリを実行します。これは、モニタリングアカウントでクエリを実行する場合にのみ機能します。これを省略した場合、デフォルトでは、リンクされたすべてのソースアカウントと現在のモニタリングアカウントをクエリします。クロスアカウントオブザーバビリティの詳細については、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。

  リストには最大 20 個のアカウント識別子を含めることができます。
- logGroupClass は、標準アクセスまたは低頻度アクセスのいずれかで、指定されたログクラスにあるロググループに対してクエリを実行します。これを省略すると、デフォルトの標準ログクラスが使用されます。ログクラスの詳細については、「ログクラス」を参照してください。

この方法でクエリを実行するために多数のロググループを指定できるため、作成したフィールドインデックスを利用するクエリSOURCEでのみ を使用することをお勧めします。ロググループのフィールドのインデックス作成の詳細については、「」を参照してください。 フィールドインデックスを作成してクエリのパフォーマンスを向上させ、スキャンボリュームを削減する

次の の例では、アカウント内のすべてのロググループを選択します。これがモニタリングアカウントの場合、モニタリング全体のロググループとすべてのソースアカウントが選択されます。ロググループの総数が 10,000 を超える場合は、別のロググループ選択方法を使用してロググループの数を減らすように求めるエラーが表示されます。

```
SOURCE logGroups()
```

次の の例では、111122223333ソースアカウントのロググループを選択します。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でクエリを開始すると、すべての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がデフォルトで選択されます。

```
SOURCE logGroups(accountIdentifiers:['111122223333'])
```

次の例では、名前プレフィックスに基づいてロググループを選択します。

```
SOURCE logGroups(namePrefix: ['namePrefix1', 'namePrefix2'])
```

次の例では、低頻度アクセスログクラスのすべてのロググループを選択します。class 識別子を含めない場合、クエリはデフォルトである標準ログクラスのロググループにのみ適用されます。

```
SOURCE logGroups(class: ['INFREQUENT_ACCESS'])
```

次の の例では、特定の名前プレフィックスで始まり、標準ログクラスにある 111122223333「」アカウントのロググループを選択します。Standard がデフォルトのログクラス値であるため、 クラスは コマンドに記載されません。

```
SOURCE logGroups(accountIdentifiers:['111122223333'], namePrefix: ['namePrefix1',
 'namePrefix2']
```

最後の例は、 SOURCE コマンドで start-query AWS CLI コマンドを使用する方法を示しています。

```
aws logs start-query
--region us-east-1
--start-time 1729728200
--end-time 1729728215
--query-string "SOURCE logGroups(namePrefix: ['Query']) | fields @message | limit 5"
```

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発見することや既知のエラーをモニタリングすることに加えて、頻繁に発生するログラインやコストの高いログラインを特定することができます。CloudWatch Logs Insights は、ログイベント内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、ログを検索して分析するための出発点として使用できます。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。

CloudWatch Logs が動的トークンの表すデータのタイプを推測できない場合は、<トークン - ##> として表示され、##は他の動的トークンと比較して、このトークンがパターン内のどこに表示されるかを示します。

動的トークンの一般的な例には、エラーコード、IP アドレス、タイムスタンプ、リクエスト ID があります。

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、パターン内の異なるフィールドはトークンによって表現されます。CloudWatch Logs が動的トークンの表すデータのタイプを判断できる場合、トークンは <*string*-*number*> として表示されます。###は、トークンが表すデータのタイプの説明です。##は、他の動的トークンと比較して、このトークンがパターン内のどこに表示されるかを示します。

  CloudWatch Logs は、名前を含むログイベントのコンテンツの分析に基づいて、その名前の文字列部分を割り当てます。

  CloudWatch Logs が動的トークンの表すデータのタイプを推測できない場合は、<トークン - ##> として表示され、##は他の動的トークンと比較して、このトークンがパターン内のどこに表示されるかを示します。

  例えば、[INFO] Request time: <Time-1> 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: 選択した期間のロギイベントと、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
```

diff

リクエストされた期間に検出されたログイベントと、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、傾向を検索し、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、「(diff) 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。正規表現の構文の詳細については、「サポートされている正規表現 (regex) 構文」を参照してください。

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 NetworkInterface, @message
```

> **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 表現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。

- 数値以外のすべての値は、すべての数値より前に来ます。数値は数のみを含む値であり、数と他の文字の組み合わせは含まれません。
- 数以外の値の場合、アルゴリズムは、連続する数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て比較します。数以外の部分は Unicode 値の順に並び、数の部分は最初に長さ順に並んでから、数値の順に並びます。

Unicode の順序の詳細については、「List of Unicode character」を参照してください。

例として、昇順のソートの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特定できます。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、統計集計関数と統計非集計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。

関数	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。
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。

関数	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。
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。

制限

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。limit を省略すると、クエリは結果に最大 10,000 個のログイベントを返します。

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 クエリのその他の例については、「一般的なクエリ」を参照してください。

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。

ネスト解除

unnest を使用して、入力として取得したリストをフラット化し、リスト内の各要素に対して 1 つのレコードを持つ複数のレコードを生成します。フィールドに含まれる項目の数に基づいて、このコマンドは現在のレコードを破棄し、新しいレコードを生成します。各レコードにはunnested_field、項目を表す が含まれます。他のすべてのフィールドは、元のレコードから取得されます。

の入力unnestは でLIST、 jsonParse関数から取得されます。詳細については、「構造タイプ」を参照してください。MAP、、 などの他のタイプはnumbers、 に 1 String つの項目を含むリストとして扱われますunnest。

コマンド構造

次の例では、このコマンドの形式について説明します。

```
unnest field into unnested_field
```

クエリの例

次の の例では、JSON オブジェクト文字列を解析し、フィールドイベントのリストを展開します。

```
fields jsonParse(@message) as json_message
| unnest json_message.events into event
| display event.name
```

このクエリ例のログイベントは、次のように JSON 文字列になります。

```
{
   "events": [
        {
            "name": "exception"
        },
        {
            "name": "user action"
        }
   ]
}
```

この場合、サンプルクエリはクエリ結果に 2 つのレコードを生成します。1 つは ユーザーアクションevent.nameとして 、exceptionもう 1 つはユーザーアクションevent.nameとして です。

クエリの例

次の例では、リストをフラット化し、項目を除外します。

```
fields jsonParse(@message) as js
| unnest js.accounts into account
| filter account.type = "internal"
```

クエリの例

次の の例では、集計のリストをフラット化します。

```
fields jsonParse(trimmedData) as accounts
| unnest accounts into account
| stats sum(account.droppedSpans) as n by account.accountId
| sort n desc
```

```
| limit 10
```

ブール、比較、数値、日時、その他の関数

CloudWatch Logs Insights は、以下のセクションで説明するように、クエリ内の他の多くの演算や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
- 構造タイプ
- 日時関数
- 一般関数
- JSON 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。

Operation	説明
`a + b`	加算
`a - b`	減算
`a * b`	乗算
`a / b`	除算
`a ^ b`	指数 (`2 ^ 3` は 8 を返します)
`a % b`	残余または剰余 (`10 % 3` は 1 を返します)

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 (より小さい)
>	Greater than (より大きい)
<=	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。

Operation	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。

Operation	結果タイプ	説明
`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

構造タイプ

マップまたはリストは、CloudWatch Logs Insights の構造タイプであり、クエリの属性にアクセスして使用できます。

例: マップまたはリストを取得するには

jsonParse を使用して、JSON 文字列であるフィールドをマップまたはリストに解析します。

```
fields jsonParse(@message) as json_message
```

例: 属性にアクセスするには

ドットアクセス演算子 (map.attribute) を使用して、マップ内の項目にアクセスします。
マップ内の属性に特殊文字が含まれている場合は、バックティックを使用して属性名 (map.attributes.`special.char`) を囲みます。

```
fields jsonParse(@message) as json_message
| stats count() by json_message.status_code
```

ブラケットアクセス演算子 (list[index]) を使用して、リスト内の特定の位置にある項目を取得します。

```
fields jsonParse(@message) as json_message
| filter json_message.users[1].action = "PutData"
```

キー名に特殊文字が含まれている場合は、特殊文字をバックティック (`) でラップします。

```
fields jsonParse(@message) as json_message
| filter json_message.`user.id` = "123"
```

例: 空の結果

マップとリストは、文字列、数値、および日時関数では null として扱われます。

```
fields jsonParse(@message) as json_message
| display toupper(json_message)
```

マップとリストを他のフィールドと比較すると、 になりますfalse。

> **Note**
>
> dedup、、pattern、および でのマップsortとリストstatsの使用はサポートされていません。

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 はミリ秒
- s は秒
- m は分
- h は時間

たとえば、10m は 10 分、1h は 1 時間です。

> **Note**
>
> 日時関数に最適な時間単位を使用します。CloudWatch Logs は、選択した時間単位に従ってリクエストを制限します。例えば、s 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。したがって、bin(300s) を指定すると、CloudWatch Logs はこれを 60 秒として実際に実装します。60 は 1 分間の秒数であるため、CloudWatch Logs は s で 60 を超える数値を使用しません。5 分間のバケットを作成するには、代わりに bin(5m) を使用します。
> ms の上限は 1000、s と m の上限は 60、h の上限は 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。

関数	結果タイプ	説明

bin(period: Period)	Timestamp	@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/`
		`  | stats count(*) as exceptionCount`
		`by bin(1h)`
		`  | sort exceptionCount desc`

関数	結果タイプ	説明
		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)	Timestamp	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。

関数	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。

now()	数値	クエリ処理が開始された時刻をエポック秒単位で返します。この関数は引数を取りません。
		
		これを使用して、現在の時刻に従ってクエリ結果をフィルタリングできます。
		
		たとえば、次のクエリは、過去 2 時間の 4xx エラーをすべて返します。
		
		`parse @message "Status Code: *;" as statusCode\n`
		`| filter statusCode >= 400 and statusCode <= 499 \n`
		`| filter toMillis(@timestamp) >= (now() * 1000 - 7200000)`
		
		次の の例では、 または のいずれかの単語を含む過去 5 時間のすべてのログエントリを返します。 error failure
		
		`fields @timestamp, @message`
		`| filter @message like /(?i)(error|failure)/`
		`| filter toMillis(@timestamp) >= (now() * 1000 - 18000000)`

> **Note**
>
> 現在、CloudWatch Logs Insights では、可読性のあるタイムスタンプが記録されている口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。

関数	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

JSON 関数

JSON 関数

JSON 関数を コマンド`fields`と `filter` コマンドで使用し、他の関数の引数として使用します。

関数	結果タイプ	説明
`jsonParse(fieldName: string)`	マップ | リスト | 空	入力が JSON オブジェクトまたは JSON 配列の文字列表現である場合、マップまたはリストを返します。入力が表現のいずれでもない場合、空の値を返します。

関数	結果タイプ	説明
`jsonStringify(fieldName: Map | List)`	String	マップまたはリストデータから JSON 文字列を返します。

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。

関数	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)`	boolean	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。

関数	結果タイプ	説明
`isIpv6InSubnet(fieldName: string, subnet: string)`	boolean	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。

関数	結果タイプ	説明
`isempty(fieldName: string)`	数値	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)`	数値	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

関数	結果タイプ	説明
		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。

関数	結果タイプ	説明
tolower(str: string)	文字列	文字列を小文字に変換します。
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、クエリ内でのコメントをサポートしています。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
```

Logs Insights QL の開始方法: クエリチュートリアル

以下のセクションでは、Logs Insights QL 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について説明します。

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
- チュートリアル: 集計関数を使用してクエリを実行する
- 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
- 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

チュートリアル: サンプルクエリを実行および変更する

以下のチュートリアル通りに、CloudWatch Logs Insights の使用を開始できます。Logs Insights QL でサンプルクエリを実行し、そのクエリを変更して再実行する方法を確認します。

クエリを実行するには、CloudWatch Logs に保存済みのログが必要です。CloudWatch Logs をすでに使用していて、ロググループとログストリームが設定済みである場合は、開始する準備が整っています。また AWS CloudTrail、Amazon Route 53 や Amazon VPC などのサービスを使用していて、CloudWatch Logs に移動するようにそれらのサービスのログを設定している場合は、ログ

が既に存在する可能性があります。CloudWatch Logs にログを送信する方法の詳細については、「CloudWatch Logs の開始方法」を参照してください。

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

サンプルクエリを実行する

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

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
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 ページで、クエリエディタには、最新の 20 個のログイベントを返す Logs Insights QL のデフォルトクエリが含まれています。
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。

   これが CloudWatch の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、モニタリングアカウントだけでなくソースアカウントのロググループも選択できます。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。

   標準ログクラスのロググループを選択すると、CloudWatch Logs Insights 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、「ログクラス」を参照してください。
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。

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

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 に用意されている一部のサンプルクエリでは、`limit` の代わりに `head` または `tail` コマンドを使用します。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 の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 クエリの保存と再実行」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 により検出された最も一般的なフィールドと、各フィールドが出現す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 [Fields (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。

Logs Insights QL 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 言語クエリ構文」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 CloudWatch 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。

集計関数を使用したクエリの実行方法

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
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 Insights QL タブが選択されていることを確認します。
4. [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。

   これが CloudWatch の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、モニタリングアカウントだけでなくソースアカウントのロググループも選択できます。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、標準クラスロググループの場合 CloudWatch Logs Insights はログ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。
5. 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
   ```
6. *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。

   [Fields] (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そこに表示されます。
7. [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. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
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。

   これが CloudWatch の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、モニタリングアカウントだけでなくソースアカウントのロググループも選択できます。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. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
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 Insights QL タブが選択されていることを確認します。
4. [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。

   これが CloudWatch の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、モニタリングアカウントだけでなくソースアカウントのロググループも選択できます。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。
5. 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)
   ```

   結果として、CloudWatch Logs が 30 秒間隔で受信したロググループ内のログイベントの数が表示されます。
6. [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で便利なクエリコマンドの一覧を紹介します。クエリコマンドの実行方法については、「Amazon CloudWatch Logs ユーザーガイド」の「チュートリアル: サンプルクエリを実行および変更する」を参照してください。

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

トピック

- 一般的なクエリ

- Lambda ログのクエリ
- Amazon VPC フローログのクエリ
- Route 53 ログのクエリ
- CloudTrail ログのクエリ
- のクエリ Amazon API Gateway
- NAT ゲートウェイに対するクエリ
- Apache サーバーのログに対するクエリ
- Amazon EventBridge のクエリ
- 解析コマンドの例

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"
```

新しく作成した IAM ユーザーの AWS リージョン、ユーザー名、および 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内で最も長く実行されている 10 個の Amazon API Gateway リクエストを特定する

```
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、CloudWatch Logs Insights を使用して上位の寄与要因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 ゲートウェイ経由のトラフィックの上位の寄与要因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 では Apache サーバーのログにクエリできます。以下のクエリの詳細については、 AWS クラウドオペレーションと移行ブログの「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 EventBridge のクエリ

イベント詳細タイプ別にグループ化された EventBridge イベントの数を取得します。

```
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
```

(diff) を以前の時間範囲と比較する

CloudWatch Logs Insights と Logs Insights QL 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。最近の時間範囲に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。これにより、ログ内のエラーが最近発生したものか、または既に発生していたかを調べることができ、他の傾向を見つけやすくなります。

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。返されるパターンを基に、ログイベントの傾向と経時的な変化をすばやく確認できるようになります。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 raw ログイベントのサンプルを確認できます。ログパターンの詳細については、「パターン分析」を参照してください。

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。比較期間は、常に元のクエリ期間と同じ長さです。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**
>
> 比較を使用したクエリでは、組み合わせ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 料金表」をご覧ください。

比較クエリを実行するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. ナビゲーションペインで、[ログ]、[Logs Insights] 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. Logs Insights QL タブが選択されていることを確認します。
4. 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。
5. [ロググループを選択] 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。
6. (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 時間のログデータです。
7. 時間範囲セレクタで、[比較] を選択します。次に、元のログと比較する前の期間を選択し、[適用] を選択します。
8. [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できるようにするため、`diff` コマンドがクエリに追加されます。
9. [パターン] タブを選択して結果を表示します。

   テーブルに表示される情報は、次のとおりです。

   - 各 [パターン] では、パターンの可変部分が動的トークン記号 <*string-number*> に置き換えられます。*###*は、トークンが表すデータのタイプの説明です。*##*は、他の動的トークンと比較して、このトークンがパターン内のどこに表示されるかを示します。詳細については、「パターン分析」を参照してください。
   - [イベント数] は、元のより最新の期間にそのパターンを持つログイベントの数です。
   - [イベント数の差異] 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。正の違いは、現在の期間にこのようなイベントが多いことを意味します。

- [差異の説明] は、現在の期間と比較期間の間のパターンの変化を簡単に要約します。
- [重要度タイプ] は、FATAL、ERROR、WARN などのログイベントで検出された単語に基づいて、このパターンを持つログイベントの推定重要度です。

10. リスト内のパターンの 1 つをさらに検査するには、[検査] 列の アイコンを選択してパターンの 1 つを選択します。

    [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、パターンの発生の急増など、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。
- [トークンの値] タブには、選択している場合は、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した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、追加の詳細を含む INFO とマークされた別のログイベントが伴っていた場合、そのパターンがここに表示されます。

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、積層型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。

自然言語を使用した CloudWatch Logs Insights クエリの生成と更新

CloudWatch Logs は自然言語クエリ機能をサポートしており、CloudWatch Logs Insights と CloudWatch Metrics Insights のクエリを生成し、更新する際に便利です。

この機能を使用すると、どのような CloudWatch Logs データを探しているのかを平易な英語で質問したり、具体的に説明したりすることができます。自然言語機能が働いて、入力したプロンプトに

基づいてクエリが生成され、クエリの仕組みを説明した文が 1 行ずつ表示されます。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?」 や「Lambda 関数の最も遅いリクエストを 10 個見つけてください。」などのプロンプトを入力できます。

> **Note**
>
> 自然言語クエリ機能は、一般に 10 のリージョンで使用できます。一部のリージョンでは、この機能は、クエリ プロンプトを処理するために、米国のリージョンに対してクロスリージョン呼び出しを行います。次の表は、サポートされているリージョンを一覧表示し、各リージョンがプロンプトを処理する場所を示しています。

サポートされているリージョン	プロンプトが処理されるリージョン
米国東部 (バージニア北部)	米国東部 (バージニア北部)
米国東部(オハイオ)	米国東部 (バージニア北部)
米国西部 (オレゴン)	米国西部 (オレゴン)
アジアパシフィック (香港)	米国西部 (オレゴン)
アジアパシフィック (シンガポール)	米国西部 (オレゴン)
アジアパシフィック (シドニー)	米国西部 (オレゴン)
アジアパシフィック (東京)	アジアパシフィック (東京)
欧州 (フランクフルト)	欧州 (フランクフルト)
欧州 (アイルランド)	米国東部 (バージニア北部)

サポートされているリージョン	プロンプトが処理されるリージョン
欧州 (ストックホルム)	米国東部 (バージニア北部)

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、[Generate query] 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 IAM ポリシーでサインインするか、`cloudwatch:GenerateQuery` のアクセス許可が必要です。

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、「CloudWatch Logs Insights query syntax」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。

Prompt

以下は、最も遅い 10 個の Lambda 関数の呼び出しを検索する機能を指示するプロンプトの例です。

```
Find the 10 slowest requests
```

クエリ

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。最も遅い 10 個の Lambda 関数の呼び出しを検索するプロンプトの代わりに、このプロンプトは 最も遅い 20 個の Lambda 関数の呼び出しを検索し、追加のログイベントのために別の列を含めています。

```
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、CloudWatch Logs Insights の品質を高めるために使用される可能性があります。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。クエリ生成機能などのすべての CloudWatch Logs AI 機能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。

OpenSearch PPL 言語

このセクションでは、OpenSearch PPL を使用して CloudWatch Logs をクエリするための基本的な概要を示します。PPL を使用すると、パイプ結合コマンドを使用してデータを取得、クエリ、分析できるため、複雑なクエリの理解と作成が容易になります。構文は Unix パイプに基づいており、コマンドを連鎖させてデータを変換および処理できます。PPL を使用すると、データをフィルタリングして集計し、豊富な数学、文字列、日付、条件付き、その他の関数のセットを分析に使用できます。

OpenSearch PPL は、標準ログクラスのロググループのクエリにのみ使用できます。

CloudWatch Logs でサポートされているすべての OpenSearch PPL クエリコマンドと構文と制限の詳細については、OpenSearch Service デベロッパーガイドの「サポートされている PPL コマンド」を参照してください。

コマンドまたは関数	クエリの例	説明
fields	`fields field1, field2`	射影が必要な一連のフィールドを表示します。
この場合、次のようになります。	`where field1="success" | where field2 != "i-023fe0a90929d8822" | fields field3, field4, field5,field6 | head 1000`	指定した条件に基づいてデータをフィルタリングします。
stats	`stats count(), count(field1), min(field1), max(field1), avg(field1) by field2 | head 1000`	集計と計算を実行します。
parse	`parse field1 ".*/(?<field2>[^/]+$)" | where field2 = "requestId" | fields field1, field2 | head 1000`	文字列から正規表現 (regex) パターンを抽出し、抽出されたパターンを表示します。抽出されたパターンは、さらに新しいフィールドの作成やデータのフィルタリングに使用できます。
sort	`stats count(), count(field1), min(field1) as field1Alias, max(`field1`), avg(`field1`) by field2 | sort -field1Alias | head 1000`	表示された結果をフィールド名でソートします。sort -FieldName を使用して降順でソートします。

コマンドまたは関数	クエリの例	説明
評価	`eval field2 = field1 * 2 | fields field1, field2 | head 20`	フィールドの値を変更または処理し、別のフィールドに保存します。これは、列の数学的変更、列への文字列関数の適用、または列への日付関数の適用に役立ちます。
rename	`rename field2 as field1 | fields field1;`	検索結果の 1 つ以上のフィールドの名前を変更します。
head	`fields `@message` | head 20`	表示されるクエリ結果を最初の N 行に制限します。
top	`top 2 field1 by field2`	フィールドの最も一般的な値を検索します。
重複排除	`dedup field1 | fields field1, field2, field3`	指定したフィールドに基づいて重複するエントリを削除します。

コマンドまたは関数	クエリの例	説明
まれ	`rare field1 by field2`	フィールドリスト内のすべてのフィールドの最も頻度の低い値を検索します。
トレンドライン	`trendline sma(2, field1) as field1Alias`	フィールドの移動平均を計算します。
eventStats	`eventstats sum(field1) by field2`	計算されたサマリー統計でイベントデータを強化します。イベント内の指定されたフィールドを分析し、さまざまな統計測定値を計算し、これらの結果を新しいフィールドとして各元のイベントに追加します。
フィールドの概要	`where field1 != 200 | fieldsummary includefields= field1 nulls=true`	各フィールドの基本統計 (カウント、個別カウント、最小、最大、平均、stddev、平均) を計算します。

コマンドまたは関数	クエリの例	説明
Grok	`grok email '.+@%{HOSTNAME:host}' | fields email, host`	grok パターンでテキストフィールドを解析し、検索結果に結果を追加します。
文字列関数	`eval field1Len = LENGTH(field1) | fields field1Len`	PPL クエリ内の文字列およびテキストデータを操作および変換できる PPL の組み込み関数。たとえば、大文字と小文字の変換、文字列の結合、パーツの抽出、テキストのクリーニングなどです。
数学関数	`eval field2 = ACOS(field1) | fields field1`	PPL クエリで数学的計算と変換を実行するための組み込み関数。例えば、abs (絶対値）、round (丸め数値）、sqrt (平方根）、pow (電力計算）、ceil (最も近い整数に切り上げ) などです。

コマンドまたは関数	クエリの例	説明
日付関数	`eval newDate = ADDDATE(DATE('2020-08-26'), 1) | fields newDate`	PPL クエリで日付とタイムスタンプデータを処理および変換するための組み込み関数。たとえば、date_add、date_format、dateiff、current_date などです。
条件関数	`eval field2 = isnull(field1) | fields field2, field1, field3`	特定のフィールド条件をチェックし、式を条件付きで評価する組み込み関数。たとえば、field1 が null の場合、field2 を返します。

コマンドまたは関数	クエリの例	説明
数学関数	`eval field2 = ACOS(field1) | fields field1`	PPL クエリで数学的計算と変換を実行するための組み込み関数。例えば、abs (絶対値）、round (丸め数値）、sqrt (平方根）、pow (電力計算）、ceil (最も近い整数に切り上げ) などです。
CryptoGraphic 関数	`eval crypto = MD5(field)| head 1000`	指定されたフィールドのハッシュを計算するには

OpenSearch SQL 言語

このセクションでは、OpenSearch SQL を使用して CloudWatch Logs をクエリするための基本的な概要を示します。リレーショナルデータベースの使用に慣れている場合は、使い慣れたオプションを提供します。OpenSearch SQL は SQL 機能のサブセットを提供するため、アドホッククエリやデータ分析タスクの実行に適しています。OpenSearch SQL では、SELECT、FROM、WHERE、GROUP BY、HAVING、その他のさまざまな SQL コマンドや関数などのコマンドを使用できます。ロググループ間で JOINs を実行し、サブクエリを使用してロググループ間でデータを関連付け、豊富な JSON、数学、文字列、条件付き、その他の SQL 関数のセットを使用して、ログとセキュリティデータの強力な分析を実行できます。

OpenSearch SQL は、標準ログクラスのロググループのクエリにのみ使用できます。

次の表に、CloudWatch Logs でサポートされている SQL コマンドと関数を示します。構文を含むすべての OpenSearch SQL コマンドの詳細については、OpenSearch Service デベロッパーガイドの「サポートされている SQL コマンド」を参照してください。

サポートされている SQL コマンド

コマンドまたは関数	クエリの例	説明
SELECT	`SELECT `@message`, Operation FROM `LogGroupA``	射影された値を表示します。
FROM	`SELECT `@message`, Operation FROM `LogGroupA``	データを取得するソーステーブル (複数可) またはビュー (複数可) を指定する組み込み句。さまざまなタイプの結合とサブクエリをサポートします。
WHERE	`SELECT * FROM `LogGroupA` WHERE Operation = 'x'`	指定されたフィールド基準に基づいてログイベントをフィルタリングします。
グループ化の条件	`SELECT `@logStream`, COUNT(*) as log_count FROM `LogGroupA` GROUP BY `@logStream``	グループはカテゴリに基づいてイベントを記録し、統計に基づいて平均を見つけます。

コマンドまたは関数	クエリの例	説明
HAVING	`SELECT `@logStream`, COUNT(*) as log_count FROM `LogGroupA` GROUP BY `@logStream` HAVING log_count > 100`	グループ化条件に基づいて結果をフィルタリングします。
ORDER BY	`SELECT * FROM `LogGroupA` ORDER BY `@timestamp` DESC`	ORDER BY 句のフィールドに基づいて結果を順序付けします。降順または昇順でソートできます。
JOIN	`SELECT A.`@message`, B.`@timestamp`FROM `LogGroupA` as A INNER JOIN `LogGroupB` as B ON A.`requestId` = B.`requestId``	共通フィールドに基づいて 2 つのテーブルの結果を結合します。内部 JOIN または左外部結合を指定する必要があります
LIMIT	`Select * from `LogGroupA` limit 10`	表示されるクエリ結果を最初の N 行に制限します。

コマンドまたは関数	クエリの例	説明
文字列関数	`SELECT upper(Operation) , lower(Operation), Operation FROM `LogGroupA``	SQL クエリ内の文字列およびテキストデータを操作および変換できる SQL の組み込み関数。たとえば、大文字と小文字の変換、文字列の結合、パーツの抽出、テキストのクリーニングなどです。
日付関数	`SELECT current_date() as today, date_add(current_date(), 30) as thirty_days_later, last_day(current_date()) as month_end FROM `LogGroupA``	SQL クエリで日付とタイムスタンプデータを処理および変換するための組み込み関数。たとえば、date_add、date_format、dateiff、current_date などです。

コマンドまたは関数	クエリの例	説明
条件関数	`SELECT Operation, IF(Error > 0, 'High', 'Low') as error_category FROM `LogGroupA`;`	指定された条件に基づいてアクションを実行するか、式を条件付きで評価する組み込み関数。例えば、CASE と IF などです。
関数を移行する	`SELECT AVG(bytes) as bytesWritten FROM `LogGroupA``	複数の行に対して計算を実行して 1 つの要約値を生成する組み込み関数。例えば、SUM、COUNT、AVG、MAX、MIN などです。
JSON 関数	`SELECT get_json_object(json_column, '$.name') as name FROM `LogGroupA``	SQL クエリ (from_json、to_json、get_json_object、json_tuple など) 内の JSON 形式のデータを解析、抽出、変更、クエリするための組み込み関数。データセット内の JSON 構造を操作できます。

コマンドまたは関数	クエリの例	説明
配列関数	`SELECT scores, size(scores) as length, array_contains(scores, 90) as has_90 FROM `LogGroupA`;`	SQL クエリで配列タイプの列を操作するための組み込み関数。配列データ (サイズ、爆発、配列を含むなど) へのアクセス、変更、分析などの操作を可能にします。
Window 関数	`SELECT field1, field2, RANK() OVER (ORDER BY field2 DESC) as field2Rank FROM `LogGroupA`;`	現在の行 (ウィンドウ) に関連する指定された行セットで計算を実行する組み込み関数。ランキング、実行合計、移動平均などのオペレーションを有効にします。例えば、ROW_NUMBER、RANK、LAG、LEAD などです。

コマンドまたは関数	クエリの例	説明
変換関数	`SELECT CAST('123' AS INT) as converted_number, CAST(123 AS STRING) as converted_string FROM `LogGroupA``	SQL クエリ内でデータをあるタイプから別のタイプに変換するための組み込み関数。データ型変換と形式変換を有効にします。たとえば、CAST、TO_DATE、TO_TIMESTAMP、BINARY などです。
述語関数	`SELECT scores, size(scores) as length, array_contains(scores, 90) as has_90 FROM `LogGroupA`;`	条件を評価し、指定された条件またはパターンに基づいてブール値 (true/false) を返す組み込み関数。例えば、IN、LIKE、BETWEEN、IS NULL、EXISTS などです。
複数のロググループを選択する	`SELECT lg1.field1, lg1.field2 from `logGroups(logGroupIdentifier: ['LogGroup1', 'LogGroup2'])` as lg1 where lg1.field3= "Success"`	SELECT ステートメントで複数のロググループを指定できます

multi-log-groupクエリでサポートされている SQL

SQL で複数のロググループをクエリするユースケースをサポートするには、 logGroups コマンドを使用できます。この構文を使用すると、FROM コマンドで指定することで、複数のロググループをクエリできます。

構文:

```
`logGroups(
    logGroupIdentifier: ['LogGroup1','LogGroup2', ...'LogGroupn']
)
```

この構文では、 logGroupIdentifierパラメータに最大 50 個のロググループを指定できます。モニタリングアカウントのロググループを参照するには、LogGroup名前の代わりに ARNs を使用します。

クエリの例:

```
SELECT LG1.Column1, LG1.Column2 from `logGroups(
    logGroupIdentifier: ['LogGroup1', 'LogGroup2']
)` as LG1 WHERE LG1.Column1 = 'ABC'
```

CloudWatch Logs のクエリでは、 FROMステートメントの後に複数のロググループを含む次の構文はサポートされていません。

```
SELECT Column1, Column2 FROM 'LogGroup1', 'LogGroup2', ...'LogGroupn'
WHERE Column1 = 'ABC'
```

制限事項

OpenSearch SQL を使用して CloudWatch Logs Insights でクエリを実行する場合、次の制限が適用されます。

- SELECT ステートメントに含めることができる JOIN は 1 つだけです。
- ネストされたサブクエリは 1 つのレベルのみサポートされます。
- セミコロン (;) で区切られた複数のステートメントクエリはサポートされていません。
- フィールド名を含むクエリは同じですが、大文字と小文字 (field1 や FIELD1 など) でのみ異なります。

たとえば、次のクエリはサポートされていません。

```
Select AWSAccountId, AwsAccountId from LogGroup
```

ただし、フィールド名 (@logStream) は両方のロググループで同じであるため、次のクエリがサポートされています。

```
Select a.`@logStream`, b.`@logStream` from Table A INNER Join Table B on a.id = b.id
```

- 関数と式はフィールド名で動作し、FROM 句で指定されたロググループを持つ SELECT ステートメントの一部である必要があります。

  たとえば、このクエリはサポートされていません。

  ```
  SELECT cos(10) FROM LogGroup
  ```

  このクエリはサポートされています。

  ```
  SELECT cos(field1) FROM LogGroup
  ```

- SQL コマンドまたは PPL コマンドを使用する場合は、特定のフィールドをバックティックで囲み、正常にクエリを実行します。バックティックは、特殊文字 (非アルファベットおよび非数値) を含むフィールドで必要です。たとえば、@message、Operation.Export、 をバックティックTest::Fieldで囲みます。列を単にアルファベット名でバックティックで囲む必要はありません。

  シンプルなフィールドを使用したクエリの例：

  ```
  SELECT SessionToken, Operation, StartTime  FROM `LogGroup-A`
  LIMIT 1000;
  ```

  バックティックが追加された同様のクエリ：

  ```
  SELECT `@SessionToken`, `@Operation`, `@StartTime`  FROM `LogGroup-A` LIMIT 1000;
  ```

サポートされるログと検出されるフィールド

CloudWatch Logs Insights は、さまざまなタイプのログをサポートします。Amazon CloudWatch Logs の標準クラスロググループに送信されるログごとに、CloudWatch Logs Insights は 5 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、InputLogevent の `message` フィールドに相当します。
- `@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、InputLogevent の `timestamp` フィールドに相当します。
- `@ingestionTime` は、ログイベントが CloudWatch Logs によって受信された時間を示します。
- `@logStream` は、ログイベントの追加先のログストリームの名前を示します。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。
- `@entity` には、Explore 関連のテレメトリ機能のエンティティに関連するフラット化された JSON が含まれています。

  たとえば、この JSON はエンティティを表すことができます。

```
{
  "Entity": {
    "KeyAttributes": {
      "Type": "Service",
      "Name": "PetClinic"
    },
    "Attributes": {
      "PlatformType": "AWS::EC2",
      "EC2.InstanceId": "i-1234567890123"
    }
  }
}
```

  このエンティティの場合、抽出されたシステムフィールドは次のとおりです。

```
@entity.KeyAttributes.Type = Service
@entity.KeyAttributes.Name = PetClinic
```

```
@entity.Attributes.PlatformType = AWS::EC2
@entity.Attributes.EC2.InstanceId = i-1234567890123
```

> **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、「ログクラス」を参照してください。

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

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

CloudWatch Logs Insights が自動的に検出しないフィールドを持つ他のログタイプについては、`parse` コマンドを使用して抽出フィールドを抽出および作成してクエリで使用できます。詳細については、「CloudWatch Logs Insights 言語クエリ構文」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 @ 文字で始まる場合は、この名前の先頭に @ が追加されて CloudWatch Logs Insights に表示されます。たとえば、ログフィールド名が `@example.com` である場合、このフィールド名は `@@example.com` と表示されます。

> **Note**
>
> `@message`、、`@timestamp`または を除き`@log`、検出されたフィールドのフィールドインデックスを作成できます。フィールドインデックスの詳細については、「」を参照してくださいフィールドインデックスを作成してクエリのパフォーマンスを向上させ、スキャンボリュームを削減する。

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`

ログタイプ	検出されるログフィールド
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。ただし、各ログイベントに埋め込まれている最初の JSON フラグメントのみが検出されます。Lambda ログイベントに複数の JSON フラグメントが含まれている場合は、**parse** コマンドを使用してログフィールドを解析して抽出できます。詳細については、「JSON ログのフィールド」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、1 つの 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 CloudWatch ユーザーガイド」の「クエリ構文」を参照してください。

フィールドインデックスを作成してクエリのパフォーマンスを向上させ、スキャンボリュームを削減する

ログイベントにフィールドのフィールドインデックスを作成して、等価性ベースの検索を効率的に行うことができます。次に、CloudWatch Logs Insights クエリでフィールドインデックスを使用すると、クエリはインデックス付きフィールドを含まないことがわかっているログイベントの処理をスキップしようとします。これにより、フィールドインデックスを使用するクエリのスキャン量が減り、結果をより迅速に返すことができます。これにより、数千のロググループにわたる合計ログのペタバイトをすばやく検索し、関連するログをより迅速に見つけることができます。インデックスを作成するのに適したフィールドは、クエリを実行する必要があることが多いフィールドです。値のカー

ディナリティが高いフィールドは、フィールドインデックスの候補としても適しています。これらのフィールドインデックスを使用するクエリは、一致するログイベントをターゲット値に制限するため、より速く完了するためです。

たとえば、 のフィールドインデックスを作成したとしますrequestId。次に、そのインデックス付きフィールドとクエリされた値を含むことがわかっており、CloudWatch Logs が過去にそのフィールドの値を検出したログイベントのみを処理する、requestId = *value*または処理requestId IN [*value*, *value*, ...]を試みる、そのロググループの CloudWatch Logs Insights クエリ。

フィールドインデックスを活用して、多数のロググループの効率的なクエリを作成することもできます。filterIndex コマンドの代わりに クエリで filter コマンドを使用すると、フィールドインデックスを持つログイベントで選択したロググループに対してクエリが実行されます。これらのクエリは、最大 5 つのロググループ名のプレフィックスを指定することで、選択した最大 10,000 個のロググループをスキャンできます。これが CloudWatch 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である場合は、すべてのソースアカウントを選択するか、個々のソースアカウントを指定してロググループを選択できます。

インデックス付きフィールドでは、大文字と小文字が区別されます。たとえば、 のフィールドインデックスRequestIdは、 を含むログイベントと一致しませんrequestId。

フィールドインデックスは、JSON ログとサービスログの構造化ログ形式でのみサポートされます。

CloudWatch Logs は、インデックスポリシーの作成後に取り込まれたログイベントのみをインデックス化します。ポリシーの作成前に取り込まれたログイベントにはインデックスを作成しません。フィールドインデックスを作成すると、一致する各ログイベントは、ログイベントの取り込み時間から 30 日間インデックスが付けられたままになります。

> **Note**
>
> モニタリングアカウントにフィールドインデックスポリシーを作成する場合、そのポリシーはリンクされたソースアカウントのロググループには使用されません。フィールドインデックスポリシーは、それが作成されたアカウントにのみ適用されます。

このセクションの残りのトピックでは、フィールドインデックスを作成する方法について説明します。クエリでフィールドインデックスを参照する方法については、filterIndex「」および「」を参照してくださいフィルター。

トピック

フィールドインデックスの構文とクォータ

フィールドインデックスを作成するには、フィールドインデックスポリシーを作成します。アカウント全体に適用するアカウントレベルのインデックスポリシーを作成したり、単一のロググループのみに適用するポリシーを作成したりできます。アカウント全体のインデックスポリシーでは、アカウント内のすべてのロググループに適用されるインデックスポリシーを設定できます。また、ロググループ名のプレフィックスによって選択された、アカウントのロググループのサブセットに適用されるアカウントレベルのインデックスポリシーを作成することもできます。同じアカウントに複数のアカウントレベルのポリシーがある場合、これらのポリシーのロググループ名のプレフィックスは重複できません。

ロググループレベルのフィールドインデックスポリシーは、アカウントレベルのフィールドインデックスポリシーを上書きします。ロググループレベルのインデックスポリシーを作成すると、そのロググループはそのポリシーのみを使用し、アカウントレベルのポリシーを無視します。

ログイベントとフィールドインデックスの名前の一致では、大文字と小文字が区別されます。たとえば、 のフィールドインデックスRequestIdは、 を含むログイベントと一致しませんrequestId。

最大 20 個のアカウントレベルのインデックスポリシーを持つことができます。ロググループ名のプレフィックスにフィルタリングされた複数のアカウントレベルのインデックスポリシーがある場合、その 2 つで同じまたは重複するロググループ名のプレフィックスを使用することはできません。たとえば、 で始まるロググループにフィルタリングされたポリシーがある場合my-log、別のフィールドインデックスポリシーを my-logpprodまたは にフィルタリングすることはできませんmy-logging。

名前のプレフィックスがなく、すべてのロググループに適用されるアカウントレベルのインデックスポリシーがある場合、他のアカウントレベルのインデックスポリシーを作成することはできません。

各インデックスポリシーには、次のクォータと制限があります。

- 最大 20 個のフィールドをポリシーに含めることができます。
- 各フィールド名には最大 100 文字を含めることができます。

- で始まるロググループにカスタムフィールドのインデックスを作成するには@、フィールド名の先頭@に を追加してフィールドを指定する必要があります。たとえば、ログイベントに という名前のフィールドが含まれている場合は@userId、 を指定@@userIdしてこのフィールドのインデックスを作成する必要があります。

生成されたフィールドと予約されたフィールド

CloudWatch Logs Insights は、各ログイベントにシステムフィールドを自動的に生成します。これらの生成されたフィールドにはプレフィックスが付きます@。生成されたフィールドの詳細については、「」を参照してくださいサポートされるログと検出されるフィールド。

これらの生成されたフィールドのうち、フィールドインデックスとして使用するために以下がサポートされています。

- `@logStream`
- `@ingestionTime`
- `@requestId`
- `@type`
- `@initDuration`
- `@duration`
- `@billedDuration`
- `@memorySize`
- `@maxMemoryUsed`
- `@xrayTraceId`
- `@xraySetmentId`

これらの生成されたフィールドのインデックスを作成するには、 で始まるカスタムフィールドに対して行う必要があるため、それらを指定する@ときに を追加する必要はありません@。たとえば、 のフィールドインデックスを作成するには@logStream、 をフィールドインデックス@logStreamとして指定します。

JSON ログの子フィールドと配列フィールド

JSON ログのネストされた子フィールドまたは配列フィールドであるフィールドのインデックスを作成できます。

たとえば、このログ内の フィールド内にaccessKeyId子userIdentityフィールドのインデックスを作成できます。

```
{
    "eventVersion": "1.0",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EXAMPLE_PRINCIPAL_ID",
        "arn": "arn: aws: iam: : 123456789012: user/Alice",
        "accessKeyId": "11112222",
        "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",
                "currentState": {
                    "code": 0,
                    "name": "pending"
                },
                "previousState": {
                    "code": 80,
                    "name": "stopped"
                }
            }]
        }
    }
}
```

このフィールドを作成するには、フィールドインデックスの作成時とクエリで指定時の両方で、ドット表記 (userIdentity.accessKeyId) を使用して参照します。クエリは次のようになります。

```
fields @timestamp, @message
| filterIndex userIdentity.accessKeyId = "11112222"
```

前の例のイベントでは、 instanceIdフィールドは 内の配列にありますrequestParameters.instancesSet.items。フィールドインデックスの作成時とクエリ時の両方でこのフィールドを表すには、0 requestParameters.instancesSet.items.0.instanceId は配列内のそのフィールドの場所を参照するため、それを参照してください。

次に、このフィールドのクエリは次のようになります。

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

アカウントレベルのフィールドインデックスポリシーを作成する

このセクションのステップを使用して、アカウント内のすべてのロググループ、または同じ文字列で始まるロググループ名を持つ複数のロググループに適用されるフィールドインデックスポリシーを作成します。

アカウントレベルのフィールドインデックスポリシーを作成するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. 左側のナビゲーションペインで、設定を選択し、ログタブを選択します。
3. アカウントレベルのインデックスポリシーセクションで、管理を選択します。
4. インデックスポリシーの作成 を選択します。
5. ポリシー名に、新しいポリシーの名前を入力します。
6. Select ロググループでは、次のいずれかを実行します。
   - すべての標準ロググループを選択して、インデックスポリシーをアカウントのすべての標準クラスロググループに適用します。
   - プレフィックス一致でロググループを選択 (複数可) を選択して、同じ文字列で始まる名前を持つロググループのサブセットにポリシーを適用します。次に、これらのロググループのプレフィックスを「プレフィックス名を入力する」に入力します。

     プレフィックスを入力したら、プレビュープレフィックス一致ロググループを選択して、プレフィックスが予期したロググループと一致することを確認できます。
7. カスタムインデックスフィールド設定で、フィールドパスを追加 を選択して、インデックスを作成する最初のフィールドを入力します。

次に、フィールド名の値として使用する文字列を入力します。これは、ログイベントに表示されるものと完全に大文字と小文字が一致する必要があります。たとえば、ログイベントに が含まれている場合はrequestId、requestIdここに と入力する必要があります。RequestId、requestID、 request Idは一致しません。

@ 文字で始まるカスタムログフィールドにインデックスを作成する場合は、インデックス文字列を入力するときに追加の@文字を含める必要があります。たとえば、カスタムログフィールド がある場合は@emailname、フィールドパス@@emailnameの追加ボックスに と入力します。

CloudWatch Logs が自動的に生成する @ingestionTimeおよび @logStreamフィールドのインデックスを作成することもできます。その場合は、指定@時に を追加する必要はありません。

8. 前のステップを繰り返して、最大 20 個のフィールドインデックスを追加します。
9. 終了したら、[Create] (作成) を選択します。

ロググループレベルのフィールドインデックスポリシーを作成する

このセクションのステップを使用して、単一のロググループに適用されるフィールドインデックスポリシーを作成します。

ロググループレベルのフィールドインデックスポリシーを作成するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. 左側のナビゲーションペインで、[Logs] (ログ)、[Log groups] (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. フィールドインデックスタブを選択します。
5. このロググループのフィールドインデックスの管理を選択する
6. ロググループレベルのフィールドインデックスを管理する で、フィールドパスを追加 を選択して、インデックスを作成する最初のフィールドを入力します。

   次に、フィールド名の値として使用する文字列を入力します。これは、ログイベントに表示されるものと完全に大文字と小文字が一致する必要があります。たとえば、ログイベントに が含まれている場合はrequestId、requestIdここに と入力する必要があります。RequestId、requestID、 request Idは一致しません。

@ 文字で始まるカスタムログフィールドにインデックスを作成する場合は、インデックス文字列を入力するときに追加の@文字を含める必要があります。たとえば、カスタムログフィールド があ る場合は@emailname、フィールドパス@@emailnameの追加ボックスに と入力します。

CloudWatch Logs が自動的に生成する @ingestionTimeおよび @logStreamフィールドのインデックスを作成することもできます。その場合は、指定する@ときに を追加する必要はありません。

7. 前のステップを繰り返して、最大 20 個のフィールドインデックスを追加します。
8. 完了したら、[Save] を選択します。

クエリ作成時のロググループの選択オプション

このセクションでは、クエリに含めるロググループを選択するさまざまな方法について説明します。

コンソールでクエリのロググループを選択するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. ナビゲーションペインで、[ログ]、[Logs Insights] を選択します。
3. クエリのロググループを選択する方法は 3 つあります。
   - ロググループ名ボックスを使用します。これはデフォルトの選択方法です。この方法では、最大 50 個のロググループ名を入力できます。これが CloudWatch の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、モニタリングアカウントだけでなくソースアカウントのロググループも選択できます。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。
   - ロググループの条件セクションを使用します。このセクションでは、ロググループ名のプレフィックスに基づいてロググループを選択できます。1 つのクエリに最大 5 つのプレフィックスを含めることができます。これらのプレフィックスが名前に含まれているロググループが選択されます。または、すべてのロググループオプションで、アカウントからすべてのロググループを選択します。
   - これが CloudWatch 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である場合は、アカウントドロップダウンメニューですべてのアカウントを選択して、リンクされたすべてのアカウントからロググループを選択できます。または、このクエリに含めるアカウントを個別に選択することもできます。

選択肢が 10,000 を超えるロググループと一致する場合、選択肢を絞り込むように求めるエラーが表示されます。

4. クエリのデフォルトのログクラスは Standard です。Log クラスを使用して、アクセス頻度を低頻度に変更することができます。

の使用 AWS CLI

コマンドラインからクエリを開始するときにこれらのタイプの選択を行うには、クエリで `source` コマンドを使用できます。詳細な説明と例についてはSOURCEを参照してください。

フィールドインデックスポリシーを削除する効果

しばらく有効になっているフィールドインデックスポリシーを削除すると、次のことが発生します。

- ポリシーが削除されてから最大 30 日間、クエリはインデックス付きログイベントから恩恵を受けることができます。
- ロググループレベルのインデックスポリシーを削除し、そのロググループに適用されるアカウントレベルのポリシーがすでに設定されている場合、アカウントレベルのポリシーは最終的にそのロググループに適用されます。

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することができるため、パターンは大きなログセットを分析する際に役立ちます。

次の 3 つのログイベントの例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 つのパターンに従っています。

```
<Time-1> [INFO] Calling DynamoDB to store for resource id <ID-2>
```

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 で表されます<*string-number*>。*###*は、トークンが表すデータのタイプの説明です。*##*は、他の動的トークンと比較して、このトークンがパターン内のどこに表示されるかを示します。

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 `501` になります。

パターン検出は、CloudWatch Logs 異常ディテクターおよび比較機能でも使用されます。詳細については、「ログ異常検出」および「(diff) 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、CloudWatch Logs Insights 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。

`pattern` コマンドをクエリに含めると、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` コマンドの使用時に未加工のログイベントは返されません。クエリに `pattern` が含まれていない場合、パターン結果は、返された最初の 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いて返されます。`pattern` をクエリに含めると、[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. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. ナビゲーションペインで、[ログ]、[Logs Insights] 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、クエリを次のようにします。

   ```
   fields @timestamp, @message, @logStream, @log
   | sort @timestamp desc
   ```

4. [ロググループを選択] 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。
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 分および 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。
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たログイベントのテーブルが表示されます。表の上にあるのは、クエリに一致するレコードの数に関するメッセージです。これは、一致する 71,101 レコードのうち 10,000 レコードを表示するというメッセージと似ています。
7. [パターン] タブを選択します。
8. テーブルにクエリで検出されたパターンが表示されるようになりました。クエリには `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 10,000 個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] では、各動的トークンが <*string-number*> として表示されます。*###*は、トークンが表すデータのタイプの説明です。*##*は、他の動的トークンと比較して、このトークンがパターン内のどこに表示されるかを示します。
   - [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] は、次のいずれかになります。
     - パターンに Error という単語が含まれている場合は、ERROR です。
     - パターンに Warn という単語が含まれているが、Error が含まれていない場合は WARN です。
     - パターンに Warn または Error が含まれていない場合は INFO です。

   [Severity info] 列の見出しを選択して、重要度別にパターンをソートします。
9. 次に、クエリを変更します。クエリの `| sort @timestamp desc` 行を `| pattern @message` に置き換えて、次のような完全なクエリにします。

```
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] タブに結果は表示されません。ただし、[パターン] タブには、クエリされたログイベントの合計数に応じて、リストされたパターンの数が多く表示される可能性があります。

11. クエリに pattern を含める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することができます。これを行うには、いずれかのパターンについて [検査] 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、パターンの発生の急増など、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。
    - [トークンの値] タブには、選択している場合は、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した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、追加の詳細を含む INFO とマークされた別のログイベントが伴っていた場合、そのパターンがここに表示されます。

パターンコマンドの詳細

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

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

  pattern 構文の詳細については、「pattern」を参照してください。

- クエリで pattern を使用する場合、@message は pattern コマンドで選択されたフィールドの 1 つである必要があります。
- pattern コマンドの前に filter 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をパターン分析の入力として使用できます。
- 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、pattern @fieldname を使用します。
- 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。

CloudWatch Logs Insights クエリの保存と再実行

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。保存したクエリは、フォルダ構造が保持されるため、整理された状態を保つことができます。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。

クエリは、ユーザー固有のレベルではなく、リージョン固有のレベルに保存されます。クエリを作成して保存すると、同じリージョンで CloudWatch Logs にアクセスできる他のユーザーは、リージョンで保存されたすべてのクエリとそのフォルダ構造を表示できます。

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. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
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。

   [Save (保存)] ボタンが表示されない場合、CloudWatch Logs コンソールの新しいデザイン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. [保存] 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>*。このリソースの更新の詳細については、「PutQueryDefinition」を参照してください。

保存されたクエリを実行するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
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. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
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. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
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

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

ダッシュボードに追加したクエリは、ダッシュボードをロードおよび更新するたびに再実行されます。これらのクエリは、30 件の同時実行数といった 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. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
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. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
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。アカウントあたり 30 個の同時実行 CloudWatch Logs Insights クエリ (ダッシュボードに追加したクエリも含む) に制限されます。OpenSearch Service PPL または OpenSearch Service SQL を使用できるクエリは、これら 30 件のうち 15 OpenSearch 件のみです。

最近のクエリ履歴を表示するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
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. CloudWatch Logs コンソールに新しいデザインを使用する場合は、[History (履歴)] 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。

   CloudWatch Logs では、[Status (ステータス)] の下に、現在実行中のクエリが [In progress (進行中)] と表示されます。

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 CloudWatch Logs Insights クエリの保存済み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。暗号化結果に AWS KMS キーを関連付けると、CloudWatch Logs は、そのキーを使用して、アカウントに保存されているすべてのクエリの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、その後のクエリに対してデフォルトの暗号化方法を適用します。しかし、キーが関連付けられていたときに実行されたクエリは、そのキーで暗号化されたままになります。CloudWatch Logs 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 Logs はそれらの結果を返すことができます。ただし、キーを後で無効にすると、CloudWatch Logs 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なくなります。

> **Important**
>
> CloudWatch Logs 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 のアクセスを取り消した場合、または関連付けられた KMS キーを削除した場合、CloudWatch Logs 内の暗号化されたデータを取得できなくなります。
- CloudWatch コンソールを使用して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、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を付与します。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、get-key-policy コマンドを使用して、KMS キーのデフォルトポリシーを `policy.json` として保存します。

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

テキストエディタで `policy.json` ファイルを開き、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::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:*"
```

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、ログデータ内の異常を検出します。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、そのグループの過去 2 週間分のログイベントを使用してトレーニングが行われます。トレーニングの所要時間は最大 15 分間です。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ます。CloudWatch Logs コンソールに異常が表示されたら、調査を行います。

CloudWatch Logs パターン認識では、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によって、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することができるため、パターンは大きなログセットを分析する際に役立ちます。

例えば、次の 3 つのログイベントの例を参照してください。

```
2023-01-01 19:00:01 [INFO] Calling DynamoDB to store for ResourceID: 12342342k124-12345
2023-01-01 19:00:02 [INFO] Calling DynamoDB to store for ResourceID: 324892398123-1234R
2023-01-01 19:00:03 [INFO] Calling DynamoDB to store for ResourceID: 3ff231242342-12345
```

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

```
<Date-1> <Time-2> [INFO] Calling DynamoDB to store for resource id <ResourceID-3>
```

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、パターン内の異なるフィールドは動的トークンと呼ばれます。動的トークンのそれぞれ異なる値は、トークン値と呼ばれます。

CloudWatch Logs が動的トークンの表すデータのタイプを推測できる場合、トークンは <*string-number*> として表示されます。*###*は、トークンが表すデータのタイプの説明です。*##*は、他の動的トークンと比較して、このトークンがパターン内のどこに表示されるかを示します。

CloudWatch Logs は、名前を含むログイベントのコンテンツの分析に基づいて、その名前の文字列部分を割り当てます。

CloudWatch Logs が動的トークンの表すデータのタイプを推測できない場合は、トークンは <トークン - *##*> として表示され、*##*は他の動的トークンと比較して、このトークンがパターン内のどこに表示されるかを示します。

動的トークンの一般的な例には、エラーコード、IP アドレス、タイムスタンプ、リクエスト ID があります。

異常検出ログは、これらのパターンを使用して異常を検出します。異常ディテクターのモデルトレーニング期間が終了したら、ログは既知の傾向に照らして評価されます。異常ディテクターは、大幅な変動を異常とみなし、フラグを立てます。

この章では、異常検出の有効化の方法、異常の表示方法、ログ異常ディテクターのアラームおよび異常ディテクターが発行するメトリクスの作成方法について説明します。また、異常ディテクターとその結果を暗号化する方法についても説明します AWS Key Management Service。

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る各異常には、優先度が割り当てられます。検出された各パターンには重要度が割り当てられます。

- 優先度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% 増加すると、その重要度が NONE であっても、その異常が HIGH 優先度として指定される可能性があります。
- 重要度は、FATAL、ERROR、WARN 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。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 NONE としてマークされます。

異常可視性期間

異常ディテクターを作成するときは、その異常の最大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、ListAnomalies API 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も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、異常検出モデルは異常としてフラグ付けするのを停止します。

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

異常の抑制

異常が検出されたら、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、指定した期間中、異常ディテクターは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、異常が検出され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表示できます。また、抑制を停止することもできます。

よくある質問

は、自分のデータ AWS を使用して機械学習アルゴリズムをトレーニングし、 AWS 使用したり、他の顧客向けに使用したりしますか?

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くものであるため、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、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その他のタイプのログに適しています。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持つロググループは、ログ異常検出に特に適しています。

ログ異常検出は、CloudTrail 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。パターン分析では、ログラインの最初の 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、異常検出で成功しません。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さない可能性もあります。

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 Logs パターン分析を使用してグループのログイベントの中からパターン数を見つけます。パターン数が約 300 以下の場合は、異常検出が適切に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、「パターン分析」を参照してください。

何が異常としてフラグ付けされるのですか?

次の状況が発生すると、ログイベントに異常としてフラグが立てら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かったパターンを持つログイベント。
- 既知のパターンに対する大きな変化。
- 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発生回数の大きな変化。

上記の項目はすべて異常としてフラグ付けされ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、通常よりも多くの 200 成功値が異常としてフラグ付けされる場合もあります。このような場合は、問題を示していないとして、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、異常検出のためにスキャンされることはありません。機密データのマスキングの詳細については、「Help protect sensitive log data with masking」を参照してください。

ロググループの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。

プログラムで異常検知器を作成することもできます。詳細については、「CreateLogAnomalyDetector」を参照してください。

ログ異常ディテクターを作成するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. [ログ]、[ログ異常] を選択します。
3. [異常ディテクターを作成] 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. [異常検出器の名前] にディテクターの名前を入力します。
6. (オプション) [評価頻度] 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は、評価頻度を 15 分に設定するのが適切です。
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、[パターンをフィルタリング] 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、「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。

   (オプション) フィルターパターンをテストするには、[イベントメッセージを log 記録] にいくつかのログメッセージを入力し、[パターンをテスト] 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するか、 AWS KMS キーを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、[異常の最大表示期間 (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、このディテクターによって検出された異常情報は、保管時にキーで暗号化されます。検出された異常に関する情報を取得するには、このキーと異常ディテクターへのアクセス許可が必要です。

      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、「を使用して異常ディテクターとその結果を暗号化する 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む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、異常の検出を開始して表面化させます。

検出された異常を表示する

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、検出された異常を表示できます。

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

すべてのログ異常ディテクターによって検出された異常を表示するには

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

   [ログの異常] 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は、テーブルにリストされているログ異常の数を表します。テーブルの各行に表示される情報は、次のとおりです。

   - [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 CloudWatch Logs によって生成されます。
   - 異常の優先度。優先度は、ログイベントの変化量、ログイベントで発生する `Exception` などのキーワードに基づくなどして自動的に計算されます。

- 異常の根拠となるログパターン。パターンの詳細については、「ログ異常検出」を参照してください。
- [異常ログのトレンド] 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。
- [初回検出時刻] 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] には、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が表示されます。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。

   - この異常の根拠となるパターン。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発生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。
   - [トークンの値] タブには、選択している場合は、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、以下を実行します。

   a. [アクション]、[異常を抑制] を選択します。

   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、[抑制パターン] を選択します。

   d. [異常を抑制] を選択します。

単一のロググループで検出された異常を表示するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. [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は、テーブルにリストされているログ異常の数を表します。テーブルの各行に表示される情報は、次のとおりです。

   - [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 CloudWatch Logs によって生成されます。
   - 異常の優先度。優先度は、ログイベントの変化量、ログイベントで発生する `Exception` などのキーワードに基づくなどして自動的に計算されます。
   - 異常の根拠となるログパターン。パターンの詳細については、「ログ異常検出」を参照してください。
   - [異常ログのトレンド] 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。
   - [初回検出時刻] 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。

   - この異常の根拠となるパターン。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発生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。
   - [トークンの値] タブには、選択している場合は、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、以下を実行します。

   a. [アクション]、[異常を抑制] を選択します。

   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、[抑制パターン] 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された数の異常が検出されたときにアラームの状態が ALARM になるように指定できます。フィルターを使用して、指定された優先順位の異常のみがアラームに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. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[ログ]、[ログ異常] の順に選択します。

   ログ異常ディテクターのテーブルが表示されます。
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、[アラームを作成] 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 AnomalyCount が表示されます。
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。
   - アラームカウントを優先度の高い異常のみにするには、[LogAnomalyPriority] に **HIGH** と入力します。
   - アラームカウントを優先度の高い異常と中程度の異常のみにするには、[LogAnomalyPriority] に **MEDIUM** と入力します。

   優先度の詳細については、「異常とパターンの重要度と優先度」を参照してください。
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。静的しきい値とは、アラームしきい値が選択した静的な定数であることを意味します。異常検出しきい値とは、CloudWatch が通常の値の範囲を決定し、実際の数がこのバンド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で [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、「Using CloudWatch anomaly detection」を参照してください。

6. [*your-metric-name* が次の時...] で、[より大きい]、[以上]、[以下]、または [より低い] から選択します。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、「Evaluating an alarm」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、「Configuring how CloudWatch alarms treat missing data」を参照してください。
9. [Next (次へ)] を選択します。
10. [通知] で、[通知の追加] を選択し、その後アラームが ALARM、OK、INSUFFICIENT_DATA のいずれかの状態に遷移するときに通知するための Amazon SNS トピックを指定します。

    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 のアクションの詳細については、「Configuring CloudWatch

to create OpsItems」(OpsItems を作成するように CloudWatch を設定する) および「Incident creation」(インシデントの作成) を参照してください。

12. [Next (次へ)] を選択します。
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。マークダウン形式は、CloudWatch コンソールのアラームの [詳細] タブにのみ表示されます。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 Logs は AnomalyCount メトリクスを CloudWatch メトリクスに発行します。このメトリクスは、`AWS/Logs` 名前空間に公開されます。

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

を使用して異常ディテクターとその結果を暗号化する AWS KMS

CloudWatch Logs では、異常検出データは常に暗号化されます。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 Logs 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。
- キーと異常ディテクターを関連付けるか、あるいは関連付けを解除すると、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 のアクセスを取り消したり、関連付けられた KMS キーを削除した場合、CloudWatch Logs 内の暗号化されたデータを取得できなくなります。

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、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を付与します。このサービスプリンシパルは、KMS キーが保存されているリージョンと同じ 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、get-key-policy コマンドを使用して、KMS キーのデフォルトポリシーを policy.json として保存します。

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

テキストエディタで policy.json ファイルを開き、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 キーの使用を指定したアカウントのみに制限していますが、これを使用できる異常ディテクター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: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 キーを異常ディテクターに関連付けるには、コンソールで作成するか、 AWS CLI または APIs を使用します。

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。

CloudWatch Logs Live Tail を使用したトラブルシューティング

CloudWatch Logs Live Tail を使用すると、新しいログイベントが取り込まれたときにストリーミングリストを表示して、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。

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] のタブを参照してください。

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ています。ログクラスの詳細については、「ログクラス」を参照してください。

以下のセクションでは、コンソールおよび AWS CLI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、「StartLiveTail」を参照してください。SDK の例については、AWS 「 SDK を使用してライブテールセッションを開始する」を参照してください。

で Live Tail を使用することもできます AWS Toolkit for Visual Studio Code。VS Code コマンドパレットからライブテールセッションを開始するには、 AWS Toolkit for Visual Studio Code ユーザーガイドのAmazon CloudWatch Logs Live Tail」セクションを参照してください。

Live Tail 機能は、すべての commercial AWS Regions で使用できます。中国リージョンまたは AWS GovCloud (米国) リージョンでは利用できません。

> **Note**
>
> `StartLiveTail` API は、SDK ホストプレフィックスインジェクション`streaming-logs.`*`Region`*`.amazonaws.com`を使用してリクエストを にルーティングします。VPC エンドポイントのサポートは、この API では使用できません。

を使用して Live Tail セッションを開始する AWS CLI

`start-live-tail` AWS CLI コマンドは、ターミナル内の 1 つ以上のロググループの Live Tail ストリーミングセッションを開始します。Live Tail セッションは最大 3 時間持続します。1 秒あたり

500 件を超えるログイベントがフィルターと一致する場合、ログイベントがログイベント全体のサンプルとして表示され、リアルタイムのテーリングエクスペリエンスを提供します。`start-live-tail` コマンドの詳細については、「start-live-tail」を参照してください。

`start-live-tail` は、次の 2 つのモードで使用できます。

- print-only – これはデフォルトモードです
- interactive

print-only

`print-only` モードでは、ログイベントはターミナルでストリーミングされます。新しいイベントが 1 秒ごとに下部に追加され、Linux での `tail -f` と同様のほぼリアルタイムのテーリングエクスペリエンスが作成されます。

print-only モードで Live Tail セッションを開始するには、次のコマンドを入力します。

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-
east-1:111111222222:log-group:my-logs
```

print-only モードを使用する場合、他の Linux コマンドを使用してパイプ処理を行い、分析機能を向上させることもできます。次の例では、`error` キーワードを使用してログイベントをフィルタリングし、これらのイベントの 2 列目と 4 列目を印刷して、特定の情報を抽出しやすくします。

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-
east-1:111111222222:log-group:my-logs --mode print-only | grep "error" | awk '{print
 $2, $4}'
```

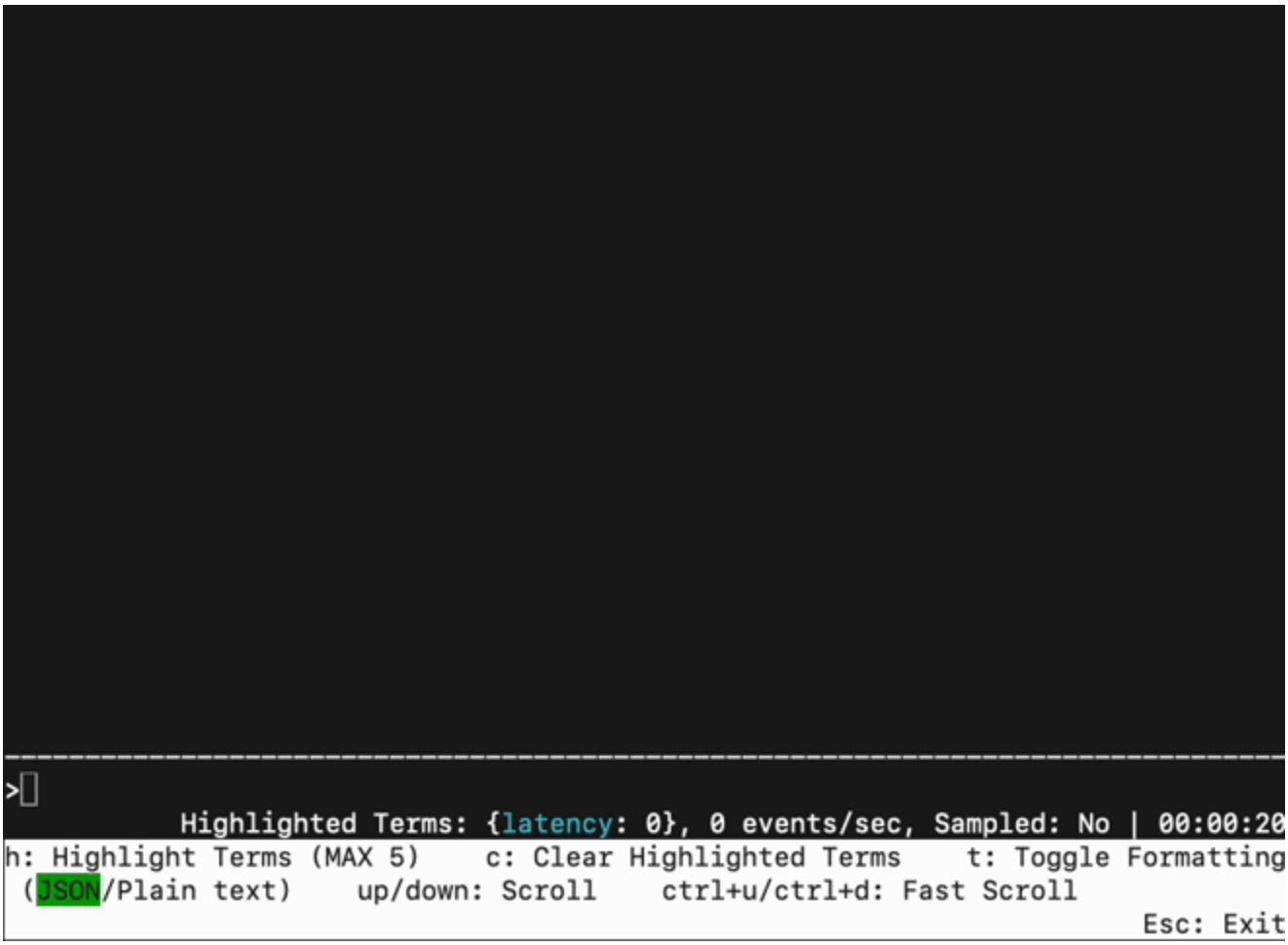
interactive

`interactive` モードでは、用語を強調表示し、出力ログイベントの形式を JSON とプレーンテキストの間で切り替えることができます。Interactive モードでは、セッションの所要時間、セッションのサンプリングの有無、現在強調表示されている用語、およびそれらの発生回数など、Live Tail セッションに関する情報も表示されます。

Interactive モードで Live Tail セッションを開始するには、次のコマンドを入力します。

```
aws logs start-live-tail --log-group-identifiers arn:aws:logs:us-
east-1:111111222222:log-group:my-logs --mode interactive
```

Live Tail セッションが開始されます。次のビデオは、セッションの例の一部を示しています。



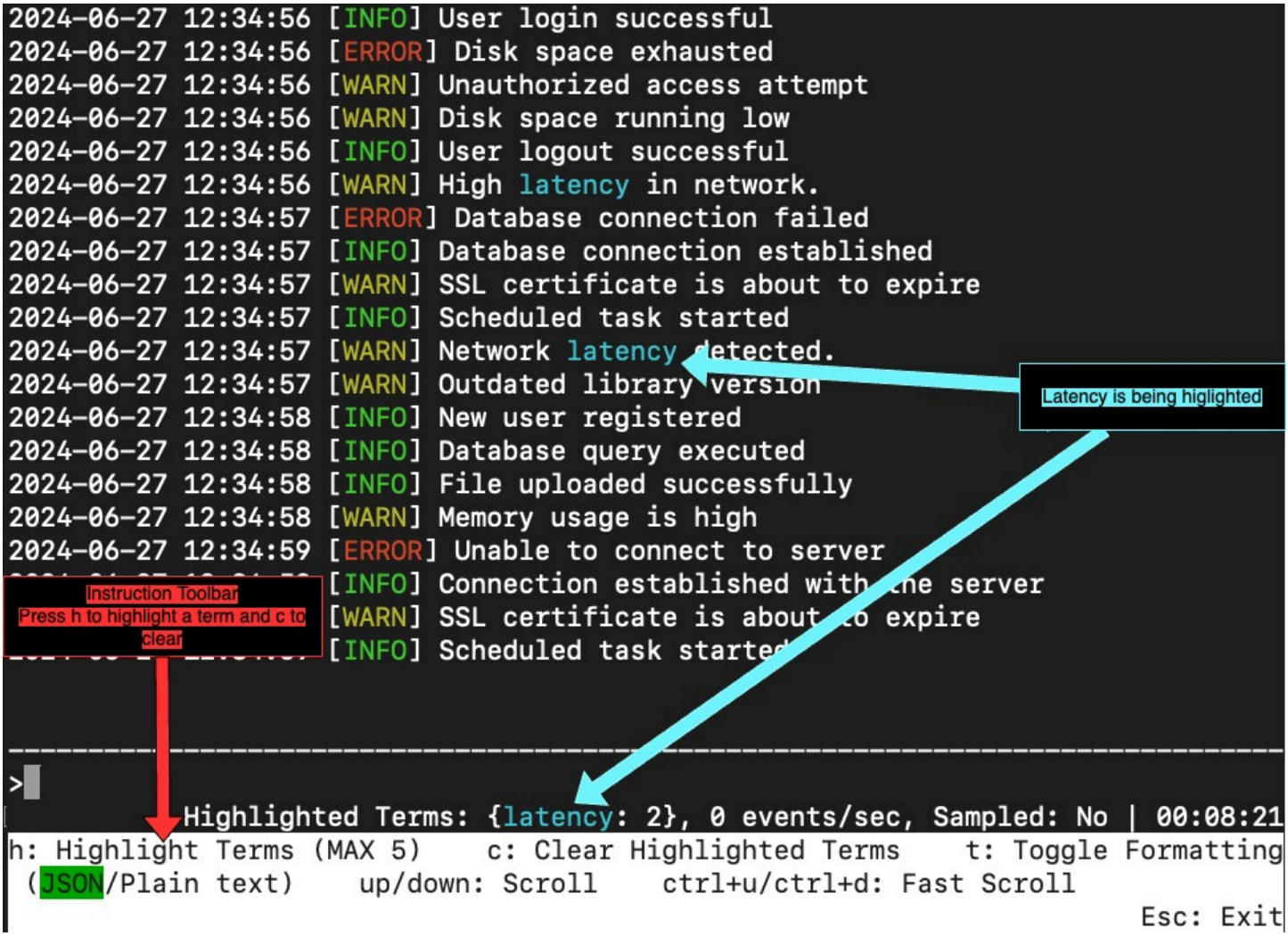
ストリーミングログで用語を強調表示するには、h を押してから用語を入力します。以下は、用語 `latency` が強調表示された後の画面を示しています。

強調表示された用語をクリアするには、c を押し、強調表示を停止する用語を表す数値を入力します。

t を押して、JSON とプレーンテキストの間で受信イベントの表示形式を切り替えることができます。このトグル機能はベストエフォートであり、ログイベント形式が互換性がある場合にのみ発生します。

スクロールするには上矢印キーと下矢印キーを使用し、より速くスクロールするには `CTRL+u` と `CTRL+d` を使用します。

次の図は、Live Tail セッション中の `latency` 用語の強調表示を示しています。



コンソールで 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。Live Tail セッションは、ロググループのページ、または CloudWatch Logs Insights のページから開始することもできます。

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。

> **Important**
>
> ネットワークセキュリティチームがウェブソケットの使用を許可しない場合は、現在 CloudWatch コンソールの Live Tail の部分にアクセスすることはできません。Live Tail は、AWS CLI または APIsで使用できます。詳細については、を使用して Live Tail セッションを開始する AWS CLI「」および「StartLiveTail」を参照してください。

Live Tail セッションを開始するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
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に一致したために、セッションのスクロールが速くなりすぎることを防ぐため、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。

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 でのログの各ソースで各ログストリームが構成されます。

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できます。

CloudWatch Logs にロググループを作成します。

Amazon CloudWatch Logs ユーザーガイドで前のセクションに記載されている手順を使用して、CloudWatch Logs エージェントを Amazon EC2 インスタンスにインストールすると、そのプロセスの一環としてロググループが作成されます。CloudWatch コンソールで、直接ロググループを作成することもできます。

ロググループを作成するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[Log groups] (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 Logs 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、CloudWatch Logs に他のログイベントを送信することもできます。

- CloudWatch エージェント — 統合された CloudWatch エージェントは、メトリックとログの両方を CloudWatch Logs に送信できます。CloudWatch エージェントのインストールと使用については、「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ックとログを収集する」を参照してください。
- AWS CLI - put-log-events は、ログイベントのバッチを CloudWatch Logs にアップロードします。
- プログラムによる – PutLogEvents API を使用して、ログイベントのバッチをプログラムにより CloudWatch Logs にアップロードできます。

CloudWatch Logs に送信されたログデータを表示する

CloudWatch Logs エージェントによって CloudWatch Logs に送信されたログデータは、ストリームごとに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. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[Log groups] (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。

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か、 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 の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、このモニタリングアカウントにリンクされているソースアカウントの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 の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. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[Log groups] (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. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[Log groups] (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. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、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. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで [Metrics (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るが、関連性がわかっているログストリームを 3 つだけ表示する場合は、 AWS CLI を使用して、ロググループ内の 3 つのログストリームのみに検索を制限できます。
- 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。

Change log data retention in CloudWatch Logs

デフォルトでは、ログデータは CloudWatch Logs に無期限に保存されます。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、ログイベントが保持設定に達したときにすぐには削除しません。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. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[ログ]、[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、現在の保持設定 (例: [失効しない]) を選択します。
5. [保持期間の設定] の [次の期間経過後にイベントを失効] で、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**
>
> ロググループや送信先などの CloudWatch Logs のリソースへのアクセスを制御するには、タグを使用します。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、以下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 で使用するために予約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 CloudWatch Logs リソースに 1 つまたは複数のタグ (キーと値のペア) を割り当てます。

list-tags-for-resource

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

CreateLogGroup

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

TagResource

指定された CloudWatch Logs リソースに 1 つまたは複数のタグ (キーと値のペア) を割り当てます。

ListTagsForResource

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 つまたは複数のタグを削除します。

を使用して CloudWatch Logs 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 CloudWatch Logs で暗号化されます。デフォルトでは、CloudWatch Logs は 256 ビットの Advanced Encryption Standard Galois/Counter Mode (AES-GCM) によるサーバー側の暗号化を使用して、保管中のログデータを暗号化します。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 Logs は、`kms:EncryptionContext:aws:logs:arn` をキーとして使用し、ロググループの ARN をそのキーの値として使用して、暗号化コンテキストをサポートするようになりました。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。

> **Important**
>
> CloudWatch Logs で がサポートされ`kms:ViaService`るようになりました。これにより、ログがユーザーに代わって AWS KMS 呼び出しを行うことができます。これは、キーポリシーまたは IAM で CloudWatch Logs を呼び出すロールに追加する必要があります。詳細については、「kms:ViaService」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形式で保存されます。CloudWatch Logs は、リクエストがあればいつでもこのデータを復号化します。暗号化されたデータがリクエストされた場合、CloudWatch Logs に KMS キーのアクセス許可が必要です。

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 CloudWatch Logs デフォルトの暗号化方法を使用して、新たに取り込まれたデータを暗号化します。以前に取り込まれたデータのうち KMS キーで暗号化されたものは、すべて KMS キーで暗号化されたままになります。CloudWatch Logs 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 Logs はそのデータを返すことができます。ただし、キーを後で無効にすると、CloudWatch Logs はそのキーで暗号化されたログを読み取ることができなくなります。

> **Important**
>
> CloudWatch Logs 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 のアクセスを取り消したり、関連付けられた KMS キーを削除した場合、CloudWatch Logs 内の暗号化されたデータを取得できなくなります。
- CloudWatch コンソールを使用して KMS キーを既存の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、CloudWatch Logs サービスプリンシパルと呼び出し元ロールに キーを使用するアクセス許可を付与します。このサービスプリンシパルは、KMS キーが保存されているリージョンと同じ 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 アカウントまたはロググループのみに制限することをお勧めします。

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

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

テキストエディタで `policy.json` ファイルを開き、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:*"
                }
            }
        }
    ]
}
```

次に、CloudWatch Logs を呼び出すロールにアクセス許可を追加します。これを行うには、 AWS KMS キーポリシーにステートメントを追加するか、ロール自体の IAM を使用します。CloudWatch Logs は `kms:ViaService` を使用して、お客様に代わって AWS KMS を呼び出します。詳細については、「kms:ViaService」を参照してください。

AWS KMS キーポリシーでアクセス許可を追加するには、キーポリシーに次のステートメントを追加します。この方法を使用する場合は、ベストプラクティスとして、 AWS KMS 暗号化されたロググループとやり取りするロールのみにポリシーの範囲を設定します。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::account_id:role/role_name"
  },
  "Action": [
    "kms:Encrypt",
    "kms:ReEncrypt*",
```

```
    "kms:Decrypt",
    "kms:GenerateDataKey*"
    "kms:Describe*"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": [
        "logs.region.amazonaws.com"
      ]
    }
  }
}
```

または、IAM でロールのアクセス許可を管理する場合は、次のポリシーを使用して同等のアクセス許可を追加できます。これは、既存のロールポリシーに追加することも、別のポリシーとしてロールにアタッチすることもできます。この方法を使用する場合は、ベストプラクティスとして、ログの暗号化に使用される AWS KMS キーのみにポリシーの範囲を設定します。詳細については、「IAM ポリシーの編集」を参照してください。

```
{
   "Version": "2012-10-17",
   "Statement" : [
     {
        "Effect": "Allow",
        "Action": [
            "kms:Encrypt",
            "kms:ReEncrypt*",
            "kms:Decrypt",
            "kms:GenerateDataKey*
            ",
            "kms:Describe*"
        ],
        "Condition":{
            "StringEquals":{
            "kms:ViaService": [
                "logs.region.amazonaws.com"
                ]
             }
        },
        "Resource": "arn:aws:kms:region:account_id:key/key_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 KMS キーがすでに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ロググループ ARNs を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 Logs で、すべての新しいデータが新しいキーを使用して暗号化され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することで、CloudWatch Logs に取り込まれた機密データを保護できます。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、デフォルトでは、ユーザーの選択したデータ識別子に一致する機密データがすべての出力ポイント (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) 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対してデータ保護ポリシーを作成する場合、ポリシーは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、「ログクラス」を参照してください。

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 は多くのマネージドデータ識別子をサポートしており、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タイプを提供しています。CloudWatch Logs のデータ保護では、パターンマッチング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。マネージドデータ識別子の種類によっては、検出は、機密データに密接に関連する特定のキーワードの検出結果にも依存します。また、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。

選択されたデータ識別子に一致する機密データが検出されると、CloudWatch にメトリクスが発行されます。これは LogEventsWithFindings メトリクスで、AWS/Logs 名前空間で発行されます。このメトリクスは CloudWatch アラームを作成するために使用でき、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 CloudWatch に送信するメトリクスの詳細については、「CloudWatch メトリクスによるモニタリング」を参照してください。

各マネージドデータ識別子は、特定の国またはリージョン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 Logs データ保護では、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 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。目的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。そうすることで、CloudWatch Logs データ保護が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するようになります。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

CloudWatch 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 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 ログがスキャンする必要がある機密データ。名前、住所、電話番号などです。
- Operation – 後続アクション (Audit または De-identify のいずれか)。CloudWatch Logs は、機密データが検出され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 CloudWatch 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。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 CloudWatch に送信するメトリクスの詳細については、「CloudWatch メトリクスによるモニタリング」を参照してください。
- 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。

Operation	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	*
監査先として CloudWatch Logs 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	*
	logs:PutResourcePolicy	*
	logs:DescribeResourcePolicies	*
	logs:DescribeLogGroups	*
監査先として Firehose 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	*
	firehose:TagDeliveryStream	arn:aws:logs:::deliverystream/ *YOUR_DELIVERY_STREAM*

Operation	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。

Operation	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* `:*`

監査先として CloudWatch Logs を使用して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* `:*`
		`*`
		`*`
		`*`
		`*`

監査先として Firehose を使用して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*

Operation	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. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで [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、使用しているすべてのロググループで監査およびマスクするデータの種類を選択します。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、データ型の名前と、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。

   単一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、それらは既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、「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」を参照してください。

```
{
    "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. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[ログ]、[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種類を選択します。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、データ型の名前と、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。

   単一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、それらは既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、「CloudWatch Logs Insights 言語クエリ構文」を参照してください。
- unmask パラメータを含む GetLogEvents オペレーションまたは FilterLogEvents オペレーションを使用します。

CloudWatchLogsFullAccess ポリシーには `logs:Unmask` アクセス許可が含まれています。CloudWatchLogsFullAccess を持たないユーザーにカスタム IAM ポリシーをアタッチして、そ

のユーザーに logs:Unmask を許可できます。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 のデータ保護監査ポリシーを設定した場合、これらの検出結果レポートは以下の例と類似します。CloudWatch Logs は、機密データが含まれるログイベントごとに 1 つの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ログ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、ロ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、機密データがロ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 での使用に必要なキーポリシーについては、「Amazon CloudWatch Logs ユーザーガイド」の「Amazon S3」を参照してください。

保護できるデータの種類

このセクションには、CloudWatch Logs データ保護ポリシーで保護できるデータの種類に関する情報が記載されています。CloudWatch Logs マネージドデータ識別子には、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、事前設定されたデータタイプが用意されています。また、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

機密データの種類についての CloudWatch Logs マネージドデータ識別子

このセクションには、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、およびそれぞれの種類のデータに関連する国と地域に関する情報が記載されています。

機密データの種類によっては、CloudWatch Logs のデータ保護機能でデータと密接に関連するキーワードがないかスキャンされ、そのキーワードが見つかった場合にのみ該当するとみなされます。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 のデータ保護では、スペースを含まないキーワードのバリエーションや、スペースではなくアンダースコア (_) またはハイフン (-) を含むキーワードのバリエーションが自動的に照合されます。場合によっては、CloudWatch Logs ではキーワードの一般的なバリエーションに対処するためにキーワードを省略形にしたり、省略形のときは元に戻したりします。

次のテーブルでは、CloudWatch Logs で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、および保護対象保健情報 (PHI) の種類の一覧を示しています。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 と指定します。

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、US
cepCode	個人	BR
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	US
ElectoralRollNumber	個人	GB
HealthInsuranceCardNumber	健康	EU
HealthInsuranceClaimNumber	健康	US
HealthInsuranceNumber	健康	FR
HealthcareProcedureCode	健康	US
IndividualTaxIdentificationNumber	個人	US
inSeeCode	個人	FR
MedicareBeneficiaryNumber	健康	US
NationalDrugCode	健康	US

識別子	カテゴリ	国と言語
NationalIdentificationNumber	個人	DE、ES、IT
NationalInsuranceNumber	個人	GB
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
電話番号	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 Logs のデータ保護では、次の種類の認証情報を検出できます。

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 Logs のデータ保護では、次の種類のデバイス識別子を検出できます。

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 Logs のデータ保護では、次の種類の財務情報を検索できます。

データ保護ポリシーを設定すると、CloudWatch Logs では、ロググループがどの地理的位置にあるかに関係なく、指定したデータ識別子がないかスキャンします。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、米国	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 チェック式に従った 13 ~ 19 桁のシーケンスである必要があり、American Express、Dankort、Diner's Club、Discover、Electron、Japanese Card Bureau (JCB)、Mastercard、UnionPay、Visa のいずれかの種

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
				類のクレジットカードに対する標準のカード番号プレフィックスが使用されます。
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

銀行口座番号には、次のキーワードを使用します。これには、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) が含まれます。

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 Logs では、クレジットカード発行会社がパブリックテストのために予約している、次のシーケンスの出現はレポートされません。

```
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 Logs のデータ保護では、次の種類の保護対象保健情報 (PHI) が検出できます。

データ保護ポリシーを設定すると、CloudWatch Logs では、ロググループがどの地理的位置にあるかに関係なく、指定したデータ識別子がないかスキャンします。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 Logs のデータ保護では、次の種類の個人を特定できる情報 (PII) を検出できます。

データ保護ポリシーを設定すると、CloudWatch Logs では、ロググループがどの地理的位置にあるかに関係なく、指定したデータ識別子がないかスキャンします。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	いずれか	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, electoralrollno	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	いずれか	

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	いずれか	CloudWatch Logs では、緯度と経度の座標がペアとして保存され、それらが十進角 (DD) 形式の場合 (例: 41.948614,-87.655311)、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°56.9168'N

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
				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。
フルネーム	`Name`	なし	いずれか	CloudWatch Logs で検出できるのはフルネームのみです。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	いずれか	CloudWatch Logs では、17 文字のシーケンスで構成され、ISO 3779 および 3780 規格に従った VIN 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 では番号に密接に関連するキーワードが必要です。次のテーブルに、CloudWatch Logs で特定の国およびリージョンが認識されるキーワードのリストを示します。

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, 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 では番号に密接に関連する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。

次のテーブルに、CloudWatch Logs で特定の国およびリージョンが認識されるキーワードのリストを示します。

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#, nationalidno#, 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ために、CloudWatch Logs では番号に密接に関連するキーワードが必要です。次のテーブルに、CloudWatch Logs で特定の国およびリージョンが認識されるキーワードのリストを示します。

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ために、CloudWatch Logs では番号に密接に関連するキーワードが必要です。次のテーブルに、CloudWatch Logs で特定の国およびリージョンが認識されるキーワードのリストを示します。

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 Logs カスタムデータ識別子には、以下の制限があります。

- 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の組み合わせにより機能します。

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します。例えば、名前には **Employee_ID** と入力し、正規表現として **EmployeeID-\d{9}** を入力します。この正規表現は、EmployeeID- 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。例えば、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 ID を含むログイベントを検出およびマスクします。

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** と指定します。
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}** と指定します。この正規表現は、EmployeeID- の後に 9 桁の EmployeeID- を含むログイベントと一致します。例えば、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。

取り込み中のログの変換

ログ変換とエンリッチメントを使用すると、CloudWatch Logs への取り込み時に、すべてのログを一貫性のあるコンテキスト豊富な形式で正規化できます。 AWS WAF や Amazon Route 53 などの一般的な AWS サービス用に事前設定されたテンプレートを使用してログに構造を追加したり、Grok などのネイティブパーサーを使用してカスタムトランスフォーマーを構築したりできます。既存の属性の名前を変更し、アカウント ID、リージョンなどのメタデータをログに追加することもできます。

ログ変換は、アプリケーション全体のログクエリを簡素化および短縮し、ログへのアラートの作成を簡素化するのに役立ちます。この機能は、VPC フローログ、Route 53、 などの主要な AWS ログソース用のout-of-the-box変換テンプレートを使用して、一般的なログタイプの変換を提供しますAmazon RDS for PostgreSQL。事前設定された変換テンプレートを使用するか、ニーズに合わせてカスタムトランスフォーマーを作成できます。

ログ変換は、形式や属性名が大きく異なるさまざまなソースから出力されるログを管理するのに役立ちます。

トランスフォーマーを作成すると、取り込まれたログイベントが変換され、標準形式で保存されます。これらの変換されたログを活用して、次の機能を使用して分析エクスペリエンスを加速できます。

- フィールドインデックス
- CloudWatch Logs Insights 検出フィールド
- メトリクスフィルターを使用したアラームの柔軟性
- サブスクリプションフィルターによる転送

変換はログの取り込み中にのみ行われます。既に取り込まれたログイベントを変換することはできません。変換は元に戻せません。元のログと変換されたログはどちらも、同じ保持ポリシーで CloudWatch Logs に保存されます。ログ変換とエンリッチメント機能は、既存の標準ログクラスの取り込み料金に含まれています。ログストレージコストは、変換後のログサイズに基づいており、元のログボリュームを超える可能性があります。

> **Important**
>
> ログイベントが変換されたら、CloudWatch Logs Insights クエリを使用して、変換されたバージョンのログを表示する必要があります。GetLogEvents および FilterLogEvents アクションは、変換される前のログイベントの元のバージョンのみを返します。

さまざまな形式に変換することに加えて、アカウント ID、リージョン、キーワードなどの追加のコンテキストでログを強化することもできます。これらは、ロググループ名と静的キーワードから抽出されます。

ログ変換は、形式や属性名が大きく異なるさまざまなソースから出力されるログに役立ちます。

ログ変換とエンリッチメントは、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。

個々のロググループのトランスフォーマーを作成できます。また、アカウント内のすべてのロググループまたは多数のロググループに適用されるアカウントレベルのトランスフォーマーを作成することもできます。ロググループにロググループレベルのトランスフォーマーがある場合、そのトランスフォーマーはそのロググループに適用されるアカウントレベルのトランスフォーマーを上書きします。

トピック

- ログトランスフォーマーの作成と管理
- 使用できるプロセッサ
- 変換メトリクスとエラー

ログトランスフォーマーの作成と管理

ログトランスフォーマーには、論理パイプラインにある 1 つ以上のプロセッサが含まれています。各プロセッサは、トランスフォーマー設定にリストされている順序でログイベントに適用されます。

一部のプロセッサはパーサータイプです。各トランスフォーマーには少なくとも 1 つのパーサーが必要です。トランスフォーマーの最初のプロセッサはパーサーである必要があります。

一部のパーサーは、特定のタイプの AWS 販売ログ用に設定された組み込みパーサーです。

その他のプロセッサタイプは、文字列ミューテーター、JSON ミューテーター、およびデータプロセッサです。

個々のロググループのトランスフォーマーを作成できます。また、アカウント内のすべてのロググループまたは多数のロググループに適用されるアカウントレベルのトランスフォーマーを作成することもできます。ロググループにロググループレベルのトランスフォーマーがある場合、そのトランスフォーマーはそのロググループに適用されるアカウントレベルのトランスフォーマーを上書きします。アカウント内のリージョンには、最大 20 個のアカウントレベルのトランスフォーマーを設定できます。

トランスフォーマーを作成するときは、次のガイドラインに従う必要があります。

- AWS 販売ログのタイプに事前設定済みのパーサーを含める場合は、トランスフォーマーにリストされている最初のプロセッサである必要があります。トランスフォーマーに含めることができるプロセッサは 1 つだけです。
- トランスフォーマーに含めることができるgrokプロセッサは 1 つだけです。
- トランスフォーマーには少なくとも 1 つのパーサータイプのプロセッサが必要です。最大 5 つのパーサータイプのプロセッサを含めることができます。この 5 つの制限には、組み込みパーサーと設定可能なパーサーの両方が含まれます。
- トランスフォーマーには最大 20 個のプロセッサを含めることができます。
- トランスフォーマーに含めることができる addKeys プロセッサは 1 つだけです。
- トランスフォーマーに含めることができる copyValue プロセッサは 1 つだけです。
- 各トランスフォーマーは、ログイベントから最大 200 個のフィールドを抽出できます。

サポートされているすべてのプロセッサとその構文の詳細については、「」を参照してください使用できるプロセッサ。

トピック

- アカウントレベルのトランスフォーマーポリシーを作成する
- アカウントレベルのトランスフォーマーポリシーの編集または削除
- log-group-levelログトランスフォーマーを最初から作成する
- 既存のトランスフォーマーをコピーしてlog-group-levelトランスフォーマーを作成する
- log-group-levelトランスフォーマーを編集する
- log-group-levelトランスフォーマーを削除する

アカウントレベルのトランスフォーマーポリシーを作成する

このセクションのステップを使用して、アカウント内のすべてのロググループ、または同じ文字列 (プレフィックス) で始まるロググループ名を持つ複数のロググループに適用されるトランスフォーマーポリシーを作成します。1 つのリージョンに最大 20 個のアカウントレベルのトランスフォーマーポリシーを持つことができます。

同じプレフィックスを使用する、または別のリージョンに含まれる 1 つのプレフィックスを持つ同じリージョンに 2 つのトランスフォーマーポリシーを作成することはできません。たとえば、文字列プレフィックス に対して 1 つのトランスフォーマーポリシーを作成する場合/aws/lambda、プレフィックス を使用して別のトランスフォーマーポリシーを作成することはできません/aws。ただし、 用のトランスフォーマー/aws/lambdaと 用のトランスフォーマーがあります。 /aws/waf

アカウントレベルのトランスフォーマーポリシーを作成するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. 左側のナビゲーションペインで、設定を選択し、ログタブを選択します。
3. アカウントのトランスフォーマーポリシーセクションで、トランスフォーマーポリシーの作成を選択します。
4. Transformer ポリシー名に、新しいポリシーの名前を入力します。
5. Select ロググループでは、次のいずれかを実行します。
   - すべての標準ロググループを選択して、トランスフォーマーポリシーをアカウントのすべての標準クラスロググループに適用します。
   - プレフィックス一致でロググループを選択し、同じ文字列で始まる名前を持つロググループのサブセットにポリシーを適用します。次に、これらのロググループのプレフィックスを選択基準に入力します。
6. パーサーの選択 エリアで、パーサーを使用してトランスフォーマーに含めるパーサーを選択します。

   あるタイプの AWS 販売済みログ用に事前設定されたパーサーである場合は、その設定を指定する必要はありません。

   別のパーサーの場合は、その設定を指定する必要があります。詳細については、 でそのプロセッサの情報を参照してください設定可能なパーサータイプのプロセッサ。
7. 別のプロセッサを追加するには、プロセッサの選択を選択します。次に、プロセッサボックスで必要なプロセッサを選択し、設定パラメータを入力します。設定パラメータの詳細については、のそのプロセッサの セクションを参照してください使用できるプロセッサ。

プロセッサは、トランスフォーマーに追加する順序でログイベントで動作することに注意してください。

8. (オプション) プロセッサを追加するには、+ Processor を選択し、前のステップを繰り返します。
9. (オプション) これまでサンプルログイベントで構築したトランスフォーマーは、いつでもテストできます。これを行うには、Transformer プレビューセクションで次のいずれかを実行します。
   - ロググループの選択で最大 5 つのロググループを選択し、最新のログイベントのロードを選択します。次に、トランスフォーマーのテストを選択します。
   - ログイベントをサンプルログイベントに直接コピーし、テストトランスフォーマーを選択します。

   変換されたバージョンのログが表示されます。
10. プロセッサの追加が完了し、サンプルログのテストに満足したら、保存を選択します。
11. 終了したら、[Create] (作成) を選択します。

アカウントレベルのトランスフォーマーポリシーの編集または削除

このセクションのステップを使用して、アカウントレベルのトランスフォーマーポリシーを編集または削除します。

アカウントレベルのトランスフォーマーポリシーを編集または削除するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. 左側のナビゲーションペインで、設定を選択し、ログタブを選択します。
3. Transformer アカウントポリシーセクションで、管理を選択します。
4. 管理するトランスフォーマーポリシーでボタンを選択し、編集または削除を選択します。

   ポリシーを編集する場合は、「」のステップ 5 ~ 11 設定可能なパーサータイプのプロセッサ を参照してオプションを確認してください。

log-group-levelログトランスフォーマーを最初から作成する

以下のステップを使用して、log-group-levelトランスフォーマーをゼロから作成します。

コンソールを使用してロググループのログトランスフォーマーを作成するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
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. トランスフォーマーを作成するロググループを選択します。
4. Transformer タブを選択します。タブリストを表示するには、タブリストを右にスクロールする必要がある場合があります。
5. トランスフォーマーの作成 を選択します。
6. パーサーの選択ボックスで、トランスフォーマーに含めるパーサーを選択します。

   あるタイプの AWS 販売済みログ用に事前設定されたパーサーである場合は、その設定を指定する必要はありません。

   別のパーサーの場合は、その設定を指定する必要があります。詳細については、 でそのプロセッサの情報を参照してください設定可能なパーサータイプのプロセッサ。
7. 別のプロセッサを追加するには、 + プロセッサの追加を選択します。次に、プロセッサの選択ボックスで必要なプロセッサを選択し、設定パラメータを入力します。設定パラメータの詳細については、 のそのプロセッサの セクションを参照してください使用できるプロセッサ。

   プロセッサは、トランスフォーマーに追加する順序でログイベントで動作することに注意してください。
8. (オプション) これまでサンプルログイベントで構築したトランスフォーマーは、いつでもテストできます。このためには、以下を実行します。

   - 変換プレビューセクションで、サンプルログをロードを選択して、このトランスフォーマーが属するロググループからサンプルログイベントをロードするか、ログイベントをテキストボックスに貼り付けます。

     トランスフォーマーのテストを選択します。変換されたログのバージョンが表示されます。
9. プロセッサの追加が完了し、サンプルログのテストに合格したら、保存を選択します。

を使用してログトランスフォーマーを最初から AWS CLI 作成するには

- `aws logs put-transformer` コマンドを使用します。以下は、 `parseJSON`および `addKeys`プロセッサを含むトランスフォーマーを作成する例です。

```
aws logs put-transformer \
```

```
--transformer-config '[{"parseJSON":{}},{"addKeys":{"entries":
[{"key":"metadata.transformed_in","value":"CloudWatchLogs"},
{"key":"feature","value":"Transformation"}]}},{"trimString":{"withKeys":
["status"]}}]' \
--log-group-identifier my-log-group-name
```

既存のトランスフォーマーをコピーしてlog-group-levelトランスフォーマーを作成する

コンソールを使用して、既存のトランスフォーマーの JSON 設定をコピーできます。その後、そのコードを使用して を使用して同一のトランスフォーマーを作成するか AWS CLI、最初に設定を変更できます。

既存のログトランスフォーマーをコピーしてログトランスフォーマーを作成するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
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. コピーするトランスフォーマーがあるロググループを選択します。
4. 変換タブを選択します。タブリストを表示するには、タブリストを右にスクロールする必要がある場合があります。
5. トランスフォーマーの管理 を選択します。
6. Copy transformer を選択します。これにより、トランスフォーマー JSON がクリップボードにコピーされます。
7. ファイルを作成し、トランスフォーマー設定に貼り付けます。この例では、 ファイルを呼び出します。 `CopiedTransformer.json`
8. AWS CLI を使用して、その設定で新しいトランスフォーマーを作成します。

```
aws logs put-transformer --log-group-identifier my-log-group-name \
--transformer-config file://CopiedTransformer.json
```

log-group-levelトランスフォーマーを編集する

既存のログトランスフォーマーを編集するには、次の手順に従います。

ログトランスフォーマーを編集するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
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. 編集するトランスフォーマーがあるロググループを選択します。
4. 変換タブを選択します。タブリストを表示するには、タブリストを右にスクロールする必要がある場合があります。
5. トランスフォーマーの管理 を選択します。
6. Parsers and Processors セクションで、変更を加えます。
7. 別のプロセッサを追加するには、+プロセッサの追加を選択します。次に、プロセッサボックスで必要なプロセッサを選択し、設定パラメータを入力します。設定パラメータの詳細については、 のそのプロセッサの セクションを参照してください使用できるプロセッサ。

   プロセッサは、トランスフォーマーに追加する順序でログイベントで動作することに注意してください。
8. (オプション) これまでサンプルログイベントで構築したトランスフォーマーは、いつでもテストできます。このためには、以下を実行します。

   - 変換プレビューセクションで、サンプルログのロードを選択して、このトランスフォーマーが属するロググループからサンプルログイベントをロードするか、ログイベントをテキストボックスに貼り付けます。

     変換のテストを選択します。変換されたログのバージョンが表示されます。
9. プロセッサの追加が完了し、サンプルログのテストに満足したら、保存を選択します。

log-group-levelトランスフォーマーを削除する

ログトランスフォーマーを削除するには、次の手順に従います。

ログトランスフォーマーを削除するには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
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. 編集するトランスフォーマーがあるロググループを選択します。
4. 変換タブを選択します。タブリストを表示するには、タブリストを右にスクロールする必要がある場合があります。

5. [削除] を選択します。
6. 確認ボックスで、ポリシーの削除を選択します。

使用できるプロセッサ

このセクションでは、ログイベントトランスフォーマーで使用できる各プロセッサについて説明します。プロセッサは、パーサー、文字列ミューテーター、JSON ミューテーター、および日付プロセッサに分類できます。

目次

- lowerCaseString
- upperCaseString
- splitString
- substituteString
- trimString

- JSON ミューテーションプロセッサ
  - addKeys
  - deleteKeys
  - moveKeys
  - renameKeys
  - copyValue
  - listToMap
- データ型コンバータプロセッサ
  - typeConverter
  - datetimeConverter

設定可能なパーサータイプのプロセッサ

parseJSON

parseJSON プロセッサは JSON ログイベントを解析し、抽出された JSON キーと値のペアを送信先に挿入します。送信先を指定しない場合、プロセッサはキーと値のペアをルートノードに配置します。

元の`@message`コンテンツは変更されず、新しいキーがメッセージに追加されます。

フィールド	説明	必須?	デフォルト値	制限
ソース	解析されるログイベントの フィールドへのパス。ドット表記を使用して子フィールドにアクセスします。例: `store.book`	いいえ	`@message`	最大長: 128 ネストされたキーの最大深度: 3

フィールド	説明	必須?	デフォルト値	制限
destination	解析された JSON の送信先フィールド	いいえ	`Parent JSON node`	最大長: 128 ネストされたキーの最大深度: 3

例

取り込まれたログイベントは次のようになります。

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

次に、この parseJSON プロセッサがある場合 :

```
[
   "parseJSON": {
        "destination": "new_key"
   }
]
```

変換されたログイベントは次のとおりです。

```
{
    "new_key": {
        "outer_key": {
            "inner_key": "inner_value"
        }
    }
}
```

Grok

grok プロセッサを使用して、パターンマッチングを使用して非構造化データを解析および構造化します。このプロセッサは、ログメッセージからフィールドを抽出することもできます。

フィールド	説明	必須?	デフォルト値	制限
ソース	grok マッチングを適用するログイベントの フィールドへのパス	いいえ	`@message`	最大長: 128 ネストされたキーの最大深度: 3
match	ログイベントと照合する grok パターン。サポートされている grok パターンは、このセクションの最後に一覧表示されます。	はい		最大長: 128 最大 5 つの grok パターン。 grok パターンはタイプ変換をサポートしません。 一般的なログ形式のパターン (APACHE_ACCESS_LOG、NGINX_ACCESS_LOG、SYSLOG5424) では、一般的なログパターンの後に GREEDYDATA または DATA パターンのみを含めることができます。

Grok の例

例 1: grok を使用して非構造化ログからフィールドを抽出する

サンプルログ：

```
293750 server-01.internal-network.local OK "[Thread-000] token generated"
```

使用するトランスフォーマー：

```
[
    "grok": {
            "match": "%{NUMBER:version} %{HOSTNAME:hostname} %{NOTSPACE:status}
 %{QUOTEDSTRING:logMsg}"
        }
    }
]
```

出力:

```
{
  "version": "293750",
  "hostname": "server-01.internal-network.local",
  "status": "OK",
  "logMsg": "[Thread-000] token generated"
}
```

例 2

サンプルログ：

```
23/Nov/2024:10:25:15 -0900 172.16.0.1 200
```

使用するトランスフォーマー：

```
[
    "grok": {
            "match": "%{HTTPDATE:timestamp} %{IPORHOST:clientip}
 %{NUMBER:response_status}"
        }
    }
]
```

出力:

```
{
  "timestamp": "23/Nov/2024:10:25:15 -0900",
  "clientip": "172.16.0.1",
  "response_status": "200"
}
```

例 3: grok を parseJSON と組み合わせて使用して JSON ログイベントからフィールドを抽出する

サンプルログ：

```
{
    "timestamp": "2024-11-23T16:03:12Z",
    "level": "ERROR",
    "logMsg": "GET /page.html HTTP/1.1"
}
```

使用するトランスフォーマー：

```
[
   "parseJSON": {},
    "grok":
        {
           "source": "logMsg",
            "match": "%{WORD:http_method} %{NOTSPACE:request} HTTP/
%{NUMBER:http_version}"
        }
    }
]
```

出力:

```
{
  "timestamp": "2024-11-23T16:03:12Z",
  "level": "ERROR",
  "logMsg": "GET /page.html HTTP/1.1",
  "http_method": "GET",
  "request": "/page.html",
  "http_version": "1.1"
}
```

サポートされている grok パターン

次の表に、grokプロセッサでサポートされているパターンを示します。

一般的な grok パターン

パターン	例	説明
USERNAME または USER	入力: `user123.name-TEST` パターン: `%{USERNAME:name}` 出力: `{"name": "user123.name-TEST"}`	小文字 (a ~ z)、大文字 (A ~ Z)、数字 (0 ~ 9)、ドット (.)、アンダースコア (_)、またはハイフン (-) を使用できる 1 つ以上の文字に一致します
INT	入力: `-456` パターン: `%{INT:num}` 出力: `{"num": "-456"}`	オプションのプラス記号またはマイナス記号の後に 1 つ以上の数字が続きます。
BASE10NUM	入力: `-0.67` パターン: `%{BASE10NUM:num}` 出力: `{"num": "-0.67"}`	整数または浮動小数点数をオプションの符号と小数点で一致させます。
BASE16NUM	入力: `+0xA1B2` パターン: `%{BASE16NUM:num}` 出力: `{"num": "+0xA1B2"}`	10 進数と 16 進数をオプションの符号 (+ または -) とオプションの 0x プレフィックスで一致させます。

パターン	例	説明
POSINT	入力: `123` パターン: `%{POSINT:num}` 出力: `{"num": "123"}`	1 つ以上の桁 (1 ~ 9 の後に 0 ~ 9) で構成される、先頭にゼロを付けずに正の整数全体を一致させます。
NONNEGINT	入力: `008` パターン: `%{NONNEGINT:num}` 出力: `{"num": "008"}`	ゼロを含む整数 (1 つ以上の数字 0 ~ 9 で構成) と先頭に 0 が付いた数字を一致させます。
WORD	入力: `user_123` パターン: `%{WORD:user}` 出力: `{"user": "user_123"}`	文字、数字、アンダースコアを含む 1 つ以上の単語文字 (\w) で構成される単語全体に一致します。
NOTSPACE	入力: `hello_world123` パターン: `%{NOTSPACE:msg}` 出力: `{"msg": "hello_world123"}`	1 つ以上の空白以外の文字に一致します。

パターン	例	説明
SPACE	入力: `"  "` パターン: `%{SPACE:extra}` 出力: `{"extra": "  "}`	0 文字以上の空白文字に一致します。
DATA	入力: `abc def` パターン: `%{DATA:data}` 出力: `{"data": "abc def"}`	貪欲ではない任意の文字 (改行を除く) を 0 回以上一致させます。
GREEDYDATA	入力: `abc def` パターン: `%{GREEDYDATA:data}` 出力: `{"data": "abc def"}`	0 回以上、greedy の任意の文字 (改行を除く) に一致します。
QUOTEDSTRING	入力: `"Hello, world!"` パターン: `%{QUOTEDSTRING:msg}` 出力: `{"msg": "Hello, world!"}`	引用符で囲まれた文字列 (一重引用符または二重引用符) をエスケープ文字と一致させます。

パターン	例	説明
UUID	入力: `550e8400-e29b-41d4-a716-446655440000` パターン: `%{UUID:id}` 出力: `{"id": "550e8400-e29b-41d4-a716-446655440000"}`	標準の UUID 形式に一致します。8 つの 16 進数文字に続いて 3 つの 16 進数文字のグループが続き、12 の 16 進数文字で終わり、すべてハイフンで区切られます。
URN	入力: `urn:isbn:0451450523` パターン: `%{URN:urn}` 出力: `{"urn": "urn:isbn:0451450523"}`	Uniform Resource Name (URN) 構文に一致します。

AWS Grok パターン

パターン	例	説明
ARN	入力: `arn:aws:iam:us-east-1:123456789012:user/johndoe`	AWS Amazon リソースネーム (ARNs) に一致し、パーティション (aws、aws-cn、または aws-us-gov) 、サービス、リージョン、アカウント ID、スラッシュで区切られた最大 5 つの階層リソース識別子をキャプチャします。コロン間で情報が欠落している ARNs には一致しません。

パターン	例	説明
	パターン: `%{ARN:arn}` 出力: `{"arn": "arn:aws:iam:us-east-1:123456789012:user/johndoe"}`	

ネットワーク grok パターン

パターン	例	説明
CISCOMAC	入力: `0123.4567.89AB` パターン: `%{CISCOMAC:MacAddress}` 出力: `{"MacAddress": "0123.4567.89AB"}`	4-4-4 の 16 進形式の MAC アドレスに一致します。
WINDOWSMAC	入力: `01-23-45-67-89-AB` パターン: `%{WINDOWSMAC:MacAddress}`	16 進形式の MAC アドレスをハイフンと一致させます。

パターン	例	説明
	出力: `{"MacAddress": "01-23-45-67-89-AB"}`	
共通	入力: `01:23:45:67:89:AB` パターン: `%{COMMONMAC:MacAddress}` 出力: `{"MacAddress": "01:23:45:67:89:AB"}`	16 進形式の MAC アドレスをコロンと一致させます。
MAC	入力: `01:23:45:67:89:AB` パターン: `%{MAC:m1}` 出力: `{"m1": "01:23:45:67:89:AB"}`	CISCOMAC、WINDOWSMAC、または COMMONMAC パターンのいずれかに一致します。

パターン	例	説明
IPV6	入力: `2001:db8:3333:4444:5555:6666:7777:8888` パターン: `%{IPV6:ip}` 出力: `{"ip": "2001:db8:3333:4444:5555:6666:7777:8888"}`	圧縮フォームや IPv4 マップ IPv6 アドレスを含む IPv6 アドレスに一致します。 IPv4-mapped
IPV4	入力: `192.168.0.1` パターン: `%{IPV4:ip}` 出力: `{"ip": "192.168.0.1"}`	IPv4 アドレスに一致します。
IP	入力: `192.168.0.1` パターン: `%{IP:ip}` 出力: `{"ip": "192.168.0.1"}`	IPV6 パターンでサポートされている IPv6 アドレスまたは IPV6IPv4 パターンでサポートされている IPV4 アドレスのいずれかに一致します。

パターン	例	説明
ホスト名またはホスト	入力: `server-01.internal-network.local` パターン: `%{HOST:host}` 出力: `{"host": "server-01.internal-network.local"}`	サブドメインを含むドメイン名と一致します。
IPORHOST	入力: `2001:db8:3333:4444:5555:6666:7777:8888` パターン: `%{IPORHOST:ip}` 出力: `{"ip": "2001:db8:3333:4444:5555:6666:7777:8888"}`	HOSTNAME パターンでサポートされているホスト名、または IP パターンでサポートされている IP アドレスのいずれかに一致します。

パターン	例	説明
ホストポート	入力: `192.168.0.1:8080` パターン: `%{HOSTPORT:ip}` 出力: `{"ip":"192.168.0.1:8080","PORT":"8080"}`	IPORHOST パターンでサポートされている IP アドレスまたはホスト名にコロンとポート番号が続き、出力でポートを「PORT」としてキャプチャします。

URIHOST	入力: `example.com:443`
	`10.0.0.1`
	
	パターン: `%{URIHOST:host}`
	`%{URIHOST:ip}`
	
	出力: `{"host":"example.com:443","port":"443","ip":"10.0.0.1"}`
		IPORHOST パターンでサポートされている IP アドレスまたはホスト名に一致し、オプションでコロンとポート番号が続き、ポートが存在する場合はポートを「ポート」としてキャプチャします。

パス grok パターン

パターン	例	説明
UNIXPATH	入力: `/search?q=regex`	クエリパラメータを含む可能性のある URL パスに一致します。

パターン	例	説明
	パターン: `%{UNIXPATH:path}` 出力: `{"path":"/search?q=regex"}`	
ウィンパス	入力: `C:\Users\John\Documents\file.txt` パターン: `%{WINPATH:path}` 出力: `{"path":"C:\\Users\\John\\Documents\\file.txt"}`	Windows ファイルパスと一致します。
PATH	入力: `/search?q=regex` パターン: `%{PATH:path}` 出力: `{"path":"/search?q=regex"}`	URL または Windows ファイルパスのいずれかに一致します。

パターン	例	説明
TTY	入力: `/dev/tty1` パターン: `%{TTY:path}` 出力: `{"path":"/dev/tty1"}`	ターミナルと擬似ターミナルの Unix デバイスパスに一致します。
URIPROTO	入力: `web+transformer` パターン: `%{URIPROTO:protocol}` 出力: `{"protocol":"web+transformer"}`	文字に一致し、オプションでプラス (+) 文字と追加の文字、またはプラス (+) 文字が続きます。
URIPATH	入力: `/category/sub-category/product_name` パターン: `%{URIPATH:path}` 出力: `{"path":"/category/sub-category/product_name"}`	URI のパスコンポーネントに一致します。

パターン	例	説明
URIPARAM	入力: `?param1=value1&param2=value2` パターン: `%{URIPARAM:url}` 出力: `{"url":"?param1=value1&param2=value2"}`	URL クエリパラメータに一致します。
URIPATHPARAM	入力: `/category/sub-category/product?id=12345&color=red` パターン: `%{URIPATHPARAM:path}` 出力: `{"path":"/category/sub-category/product?id=12345&color=red"}`	オプションでクエリパラメータが続く URI パスに一致します。

パターン	例	説明
[URI]	入力: `https://user:password@example.com/path/to/resource?param1=value1&param2=value2` パターン: `%{URI:uri}` 出力: `{"path":"https://user:password@example.com/path/to/resource?param1=value1&param2=value2"}`	完全な URI に一致します。

日付と時刻の Grok パターン

パターン	例	説明
MONTH	入力: `Jan` パターン: `%{MONTH:month}` 出力: `{"month":"Jan"}`	完全な月名または省略された月名を単語全体で一致させます。

パターン	例	説明
	入力: `January` パターン: `%{MONTH:month}` 出力: `{"month": "January"}`	
月	入力: `5` パターン: `%{MONTHNUM:month}` 出力: `{"month": "5"}` 入力: `05` パターン: `%{MONTHNUM:month}` 出力: `{"month": "05"}`	1 から 12 までの月番号に一致し、1 桁の月ではオプションで先頭に 0 が付きます。
MONTHNUM2	入力: `05` パターン: `%{MONTHNUM2:month}` 出力: `{"month": "05"}`	01 から 12 までの 2 桁の月番号に一致します。

パターン	例	説明
月曜日	入力: 31 パターン: %{MONTHDAY:monthDay} 出力: {"monthDay":"31"}	1 から 31 までの日付に一致し、オプションで先頭に 0 が付きます。
YEAR	入力: 2024 パターン: %{YEAR:year} 出力: {"year":"2024"} 入力: 24 パターン: %{YEAR:year} 出力: {"year":"24"}	2 桁または 4 桁の形式で年を一致させます。
DAY	入力: Tuesday パターン: %{DAY:day} 出力: {"day":"Tuesday"}	完全な日付名または省略された日付名と一致します。

パターン	例	説明
HOUR	入力: 22 パターン: `%{HOUR:hour}` 出力: `{"hour":"22"}`	24 時間形式の時間をオプションの先頭にゼロ (0)0 ~ 23 と一致させます。
MINUTE	入力: 59 パターン: `%{MINUTE:min}` 出力: `{"min":"59"}`	分 (00 ~ 59) に一致します。

パターン	例	説明
SECOND	入力: `3` パターン: `%{SECOND:second}` 出力: `{"second":"3"}` 入力: `30.5` パターン: `%{SECOND:fractionalSeconds}` 出力: `{"minSec":"30.5"}` 入力: `30:5` パターン: `%{SECOND:fractionalSeconds}` 出力: `{"minSec":"30:5"}`	秒 (0)0 ~ 60 を表す数値に一致し、オプションで小数点またはコロンと小数秒の 1 桁以上が続きます。
TIME	入力: `09:45:32` パターン: `%{TIME:time}` 出力: `{"time":"09:45:32"}`	HOUR パターンが時間に一致し、MINUTE パターンが分に一致し、SECOND パターンが 2 番目に一致する時間形式を時間、分、秒で一致させます。通常はの形式です`(H)H:mm:(s)s` 。秒にはうるう秒 (0)0 ~ 60 が含まれます。

パターン	例	説明
DATE_US	入力: `11/23/2024` パターン: `%{DATE_US:date}` 出力: `{"date":"11/23/2024"}` 入力: `1-01-24` パターン: `%{DATE_US:date}` 出力: `{"date":"1-01-24"}`	`(M)M/(d)d/(yy)yy` または の形式の日付と一致し`(M)M-(d)d-(yy)yy` ます。MONTHNUM パターンは月と一致し、MONTHDAY パターンは日と一致し、YEAR パターンは年と一致します。

パターン	例	説明
DATE_EU	入力: 23/11/2024 パターン: %{DATE_EU:date} 出力: {"date":"23/11/2024"} 入力: 1-01-24 パターン: %{DATE_EU:date} 出力: {"date":"1.01.24"}	、d)d/(M)M/(yy)yy 、(d)d-(M)M-(yy)yy または の形式の日付と一致し(d)d.(M)M.(yy)yyます。MONTHNUM パターンは月と一致し、MONTHDAY パターンは日と一致し、YEAR パターンは年と一致します。
DATE	入力: 11/29/2024 パターン: %{DATE:date} 出力: {"date":"11/29/2024"} 入力: 29.11.2024 パターン: %{DATE:date} 出力: {"date":"29.11.2024"}	DATE_US および DATE_EU パターンのように、米国形式または EU 形式の日付に一致します。

パターン	例	説明
DATESTAMP	入力: `29-11-2024 14:30:00` パターン: `%{DATESTAMP:dateTime}` 出力: `{"dateTime":"29-11-2024 14:30:00"}`	DATE パターンの後にスペースまたはハイフンで区切られた TIME パターンが続きます。
TZ	入力: `PDT` パターン: `%{TZ:tz}` 出力: `{"tz":"PDT"}`	一般的なタイムゾーンの略語 (PST、PDT、MST、MDT、CST CDT、EST、EDT、UTC) と一致します。

パターン	例	説明
ISO8601_TIMEZONE	入力: +05:30 パターン: %{ISO8601_TIMEZONE:tz} 出力: {"tz":"+05:30"} 入力: -530 パターン: %{ISO8601_TIMEZONE:tz} 出力: {"tz":"-530"} 入力: Z パターン: %{ISO8601_TIMEZONE:tz} 出力: {"tz":"Z"}	UTC オフセットの「Z」またはタイムゾーンオフセットをこの形式のオプションのコロンと一致させます。[+-](H)H(:)mm ここで、HOUR パターンは時間と一致し、MINUTE パターンは分と一致します。

パターン	例	説明
ISO8601_SECOND	入力: `60` パターン: `%{ISO8601_SECOND:second}` 出力: `{"second":"60"}`	秒 (0)0 ~ 60 を表す数値に一致し、オプションで小数点またはコロンと小数秒の 1 桁以上が続きます。

パターン	例	説明
TIMESTAMP_ISO8601	入力: 2023-05-15T14:30:00+05:30 パターン: %{TIMESTAMP_ISO8601:timestamp} 出力: {"timestamp":"2023-05-15T14:30:00+05:30"} 入力: 23-5-1T1:25+5:30 パターン: %{TIMESTAMP_ISO8601:timestamp} 出力: {"timestamp":"23-5-1T1:25+5:30"} 入力: 23-5-1T1:25Z パターン: %{TIMESTAMP_ISO8601:timestamp}	ISO8601 日時形式をオプションの秒とタイムゾーン(yy)yy-(M)M-(d)dT(H)H:mm:((s)s)(Z|[+-](H)H:mm) と一致させます。

パターン	例	説明
	出力: {"timestamp":"23-5-1T1:25Z"}	
DATESTAMP_RFC2822	入力: Mon, 15 May 2023 14:30:00 +0530 パターン: %{DATESTAMP_RFC2822:dateTime} 出力: {"dateTime":"Mon, 15 May 2023 14:30:00 +0530"} 入力: Monday, 15 Jan 23 14:30:00 Z パターン: %{DATESTAMP_RFC2822:timestamp} 出力: {"dateTime":"Monday, 15 Jan 23 14:30:00 Z"}	RFC2822 日時形式と一致します。 Day, (d)d MonthName (yy)yy (H)H:mm:(s)s Z|[+-](H)H:mm Day パターンは、「Mon」や「Monday」など、完全な日または省略された日を一致させるために使用されます。 MonthName パターンは、「Jan」や「January」など、完全なまたは省略された英語の月名を一致させるために使用されます。 Timezoneパターンは、UTC オフセット (Z) またはタイムゾーンオフセットをオプションのコロンで一致させるために使用されます。

パターン	例	説明
DATESTAMP_OTHER	入力: `Mon May 15 14:30:00 PST 2023` パターン: `%{DATESTAMP_OTHER:dateTime}` 出力: `{"dateTime":"Mon May 15 14:30:00 PST 2023"}`	日付と時刻を次の形式で一致させます。 `Day MonthName (d)d (H)H:mm:(s)s Timezone (yy)yy` Day パターンは、「Mon」や「Monday」など、完全な日または省略された日を一致させるために使用されます。MonthName パターンは、「Jan」や「January」などの完全または省略された英語の月名を一致させるために使用されます。Day パターンは、「Mon」や「Monday」などの完全または省略された日を一致させるために使用されます。 MonthName パターンは、「Jan」や「January」などの完全または省略された英語の月名を一致させるために使用されます。 Timezoneパターンは、TZ grok パターンでサポートされている任意のタイムゾーンに一致します。
DATESTAMP_EVENTLOG	入力: `20230515143000` パターン: `%{DATESTAMP_EVENTLOG:dateTime}` 出力: `{"dateTime":"20230515143000"}`	区切り文字のないコンパクトな日時形式と一致します。 `(yy)yyMM(d)d(H)Hmm(s)s`

ログ grok パターン

パターン	例	説明
ログレベル	入力: `INFO`	、Alert/ALERT 、、、、、、、、、、 など、さまざまな大文字と小文字の標準ログレベルに一致

パターン	例	説明
	パターン: %{LOGLEVEL:logLevel} 出力: {"logLevel":"INFO"}	しますTrace/TRACE Debug/DEBUG Notice/NOTICE Info/INFO Warn/Warning/WARN/WARNING Err/Error/ERR/ERROR Crit/Critical/CRIT/CRITICAL Fatal/FATAL Severe/SEVERE Emerg/Emergency/EMERG/EMERGENCY 。
HTTPDATE	入力: 23/Nov/2024:14:30:00 +0640 パターン: %{HTTPDATE:date} 出力: {"date":"23/Nov/2024:14:30:00 +0640"}	ログファイルでよく使用される日付と時刻の形式と一致します。形式: (d)d/MonthName/(yy)yy:(H)H:mm:(s)s Timezone MonthName は、「Jan」や「January」など、英語の月名全体または略名と一致し、INT grok パターンTimezoneと一致します。
SYSLOGTIMESTAMP	入力: Nov 29 14:30:00 パターン: %{SYSLOGTIMESTAMP:dateTime} 出力: {"dateTime":"Nov 29 14:30:00"}	日付形式を と一致させますMonthName (d)d (H)H:mm:(s)s 。 MonthName は、「Jan」や「January」など、英語の月名または省略形と一致する

パターン	例	説明
プログラム	入力: `user.profile/settings-page` パターン: `%{PROG:program}` 出力: `{"program":"user.profile/settings-page"}`	文字、数字、ドット、アンダースコア、スラッシュ、パーセント記号、ハイフンの文字列で構成されるプログラム名と一致します。

パターン	例	説明
SYSLOGPROG	入力: `user.profile/settings-page[1234]` パターン: `%{SYSLOGPROG:programWithId}` 出力: `{"programWithId":"user.profile/settings-page[1234]","program":"user.profile/settings-page","pid":"1234"}`	PROG grok パターンに一致し、オプションでプロセス ID が角括弧で続きます。

パターン	例	説明
SYSLOGHOST	入力: `2001:db8:3333:4444:5555:6666:7777:8888` パターン: `%{SYSLOGHOST:ip}` 出力: `{"ip":"2001:db8:3333:4444:5555:6666:7777:8888"}`	ホストパターンまたは IP パターンのいずれかに一致します。
SYSLOGFACILITY	入力: `<13.6>` パターン: `%{SYSLOGFACILITY:syslog}` 出力: `{"syslog":"<13.6>","facility":"13","priority":"6"}`	10 進形式の syslog 優先度に一致します。値は角括弧 (<>) で囲む必要があります。

一般的なログ Grok パターン

事前定義されたカスタム grok パターンを使用して、Apache、NGINX、Syslog Protocol (RFC 5424) のログ形式を一致させることができます。これらの特定のパターンを使用する場合、それらは一致する設定の最初のパターンである必要があり、他のパターンの前に配置することはできません。また、GREEDYDATA パターンまたは DATA パターンのいずれかでのみ追跡できます。

パターン	説明	**match** フィールド内の使用制限
APACHE_ACCESS_LOG	Apache アクセスログに一致	1
NGINX_ACCESS_LOG	NGINX アクセスログに一致	1
SYSLOG5424	Syslog プロトコル (RFC 5424) ログと一致	1

以下に、これらの一般的なログ形式パターンを使用するための有効および無効な例を示します。

```
"%{NGINX_ACCESS_LOG} %{DATA}" // Valid
"%{SYSLOG5424}%{DATA:logMsg}" // Valid
"%{APACHE_ACCESS_LOG} %{GREEDYDATA:logMsg}" // Valid
"%{APACHE_ACCESS_LOG} %{SYSLOG5424}" // Invalid (multiple common log patterns used)
"%{NGINX_ACCESS_LOG} %{NUMBER:num}" // Invalid (Only GREEDYDATA and DATA patterns are
 supported with common log patterns)
"%{GREEDYDATA:logMsg} %{SYSLOG5424}" // Invalid (GREEDYDATA and DATA patterns are
 supported only after common log patterns)
```

一般的なログ形式の例

Apache ログの例

サンプルログ：

```
127.0.0.1 - - [03/Aug/2023:12:34:56 +0000] "GET /page.html HTTP/1.1" 200 1234
```

トランスフォーマー：

```
[
    "grok": {
            "match": "%{APACHE_ACCESS_LOG}"
        }
    }
```

```
]
```

出力:

```
{
  "remote_host": "127.0.0.1",
  "ident": "-",
  "auth_user": "-",
  "timestamp": "2023-08-03T12:34:56Z",
  "http_method": "GET",
  "request": "/page.html",
  "http_version": 1.1,
  "status_code": 200,
  "response_size": 1234
}
```

NGINX ログの例

サンプルログ：

```
192.168.1.100 - Foo [03/Aug/2023:12:34:56 +0000] "GET /account/login.html HTTP/1.1"
 200 42 "https://www.amazon.com/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
```

トランスフォーマー：

```
[
    "grok": {
            "match": "%{NGINX_ACCESS_LOG}"
        }
    }
]
```

出力:

```
{
  "remote_host": "192.168.1.100",
  "ident": "-",
  "auth_user": "Foo",
  "timestamp": "2023-08-03T12:34:56Z",
  "http_method": "GET",
  "request": "/account/login.html",
```

```
  "http_version": 1.1,
  "status_code": 200,
  "response_size": 42,
  "referrer": "https://www.amazon.com/",
  "agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/92.0.4515.131 Safari/537.36"
}
```

Syslog Protocol (RFC 5424) ログの例

サンプルログ：

```
<165>1 2003-10-11T22:14:15.003Z mymachine.example.com evntslog - ID47
 [exampleSDID@32473 iut="3" eventSource= "Application" eventID="1011"]
[examplePriority@32473 class="high"]
```

トランスフォーマー：

```
[
    "grok": {
            "match": "%{SYSLOG5424}"
        }
    }
]
```

出力:

```
{
  "pri": 165,
  "version": 1,
  "timestamp": "2003-10-11T22:14:15.003Z",
  "hostname": "mymachine.example.com",
  "app": "evntslog",
  "msg_id": "ID47",
  "structured_data": "exampleSDID@32473 iut=\"3\" eventSource= \"Application\" eventID=
\"1011\"",
  "message": "[examplePriority@32473 class=\"high\"]"
}
```

csv

csv プロセッサは、ログイベントからカンマ区切り値 (CSV) を列に解析します。

フィールド	説明	必須?	デフォルト値	制限
ソース	解析されるログイベントの フィールドへのパス	いいえ	`@message`	最大長: 128 ネストされたキーの最大深さ: 3
delimiter	元のカンマ区切り値ログイベントの各列を区切るために使用される文字	いいえ	`,`	最大長: 1
quoteCharacter	データの単一の列のテキスト修飾子として使用される文字	いいえ	`"`	最大長: 1
columns	変換されたログイベントの列に使用する名前のリスト。	いいえ	`[column_1, column_2`	最大 CSV 列: 100 最大長: 128 ネストされたキーの最大深さ: 3

例

取り込まれたログイベントの一部は次のようになります。

```
'Akua Mansa',28,'New York, USA'
```

csv プロセッサのみを使用するとします。

```
[
   "csv": {
      "delimiter": ",",
      "quoteCharacter": ":""
   }
]
```

変換されたログイベントは次のとおりです。

```
{
```

```
  "column_1": "Akua Mansa",
  "column_2": "28",
  "column_3": "New York, USA"
}
```

parseKeyValue

parseKeyValue プロセッサを使用して、指定されたフィールドをキーと値のペアに解析します。以下のオプションを使用して、フィールド情報を解析するようにプロセッサをカスタマイズできます。

フィールド	説明	必須?	デフォルト値	制限
ソース	解析されるログイベントの フィールドへのパス	いいえ	`@message`	最大長: 128 ネストされたキーの最大深さ: 3
destination	抽出されたキーと値のペアを配置する送信先フィールド	いいえ		最大長: 128
fieldDelimiter	元のログイベントのキーと値のペア間で使用されるフィールド区切り文字文字列	いいえ	&	最大長: 128
keyValueDelimiter	変換されたログイベントの各ペアのキーと値の間に使用する区切り文字文字列	いいえ	=	最大長: 128
nonMatchValue	キーと値のペアが正常に分割されない場合に、結果の値フィールドに挿入する値。	いいえ		最大長: 128
keyPrefix	変換されたすべてのキーにプレフィックスを追加する場合は、ここで指定します。	いいえ		最大長: 128
overwriteIfExists	送信先キーが既に存在する場合に値を上書きするかどうか	いいえ	`false`	

例

次のログイベントの例を取ります。

```
key1:value1!key2:value2!key3:value3!key4
```

次のプロセッサ設定を使用しているとします。

```
[
   "parseKeyValue": {
      "destination": "new_key",
      "fieldDelimiter": "!",
      "keyValueDelimiter": ":",
      "nonMatchValue": "defaultValue",
      "keyPrefix": "parsed_"
   }
]
```

変換されたログイベントは次のとおりです。

```
{
  "new_key": {
    "parsed_key1": "value1",
    "parsed_key2": "value2",
    "parsed_key3": "value3",
    "parsed_key4": "defaultValue"
  }
}
```

AWS 販売ログ用の組み込みプロセッサ

parseWAF

このプロセッサを使用して AWS WAF 、提供されたログを解析`httpRequest.headers`し、 の内容を取得し、対応する値を使用して各ヘッダー名から JSON キーを作成します。また、 でも同じことを行います`labels`。これらの変換により、 AWS WAF ログのクエリが大幅に簡単になります。AWS WAF ログ形式の詳細については、「ウェブ ACL トラフィックのログ例」を参照してください。

このプロセッサは、 を入力`@message`としてのみ受け入れます。

> **Important**
>
> このプロセッサを使用する場合は、トランスフォーマーの最初のプロセッサである必要があります。

例

次のログイベントの例を取ります。

```
{
  "timestamp": 1576280412771,
  "formatVersion": 1,
  "webaclId": "arn:aws:wafv2:ap-southeast-2:111122223333:regional/webacl/
STMTest/1EXAMPLE-2ARN-3ARN-4ARN-123456EXAMPLE",
  "terminatingRuleId": "STMTest_SQLi_XSS",
  "terminatingRuleType": "REGULAR",
  "action": "BLOCK",
  "terminatingRuleMatchDetails": [
    {
      "conditionType": "SQL_INJECTION",
      "sensitivityLevel": "HIGH",
      "location": "HEADER",
      "matchedData": ["10", "AND", "1"]
    }
  ],
  "httpSourceName": "-",
  "httpSourceId": "-",
  "ruleGroupList": [],
  "rateBasedRuleList": [],
  "nonTerminatingMatchingRules": [],
  "httpRequest": {
    "clientIp": "1.1.1.1",
    "country": "AU",
    "headers": [
      { "name": "Host", "value": "localhost:1989" },
      { "name": "User-Agent", "value": "curl/7.61.1" },
      { "name": "Accept", "value": "*/*" },
      { "name": "x-stm-test", "value": "10 AND 1=1" }
    ],
    "uri": "/myUri",
    "args": "",
    "httpVersion": "HTTP/1.1",
```

```
    "httpMethod": "GET",
    "requestId": "rid"
  },
  "labels": [{ "name": "value" }]
}
```

プロセッサ設定は次のとおりです。

```
[
   "parseWAF": {}
]
```

変換されたログイベントは次のとおりです。

```
{
  "httpRequest": {
    "headers": {
      "Host": "localhost:1989",
      "User-Agent": "curl/7.61.1",
      "Accept": "*/*",
      "x-stm-test": "10 AND 1=1"
    },
    "clientIp": "1.1.1.1",
    "country": "AU",
    "uri": "/myUri",
    "args": "",
    "httpVersion": "HTTP/1.1",
    "httpMethod": "GET",
    "requestId": "rid"
  },
  "labels": { "name": "value" },
  "timestamp": 1576280412771,
  "formatVersion": 1,
  "webaclId": "arn:aws:wafv2:ap-southeast-2:111122223333:regional/webacl/
STMTest/1EXAMPLE-2ARN-3ARN-4ARN-123456EXAMPLE",
  "terminatingRuleId": "STMTest_SQLi_XSS",
  "terminatingRuleType": "REGULAR",
  "action": "BLOCK",
  "terminatingRuleMatchDetails": [
    {
      "conditionType": "SQL_INJECTION",
      "sensitivityLevel": "HIGH",
      "location": "HEADER",
```

```
      "matchedData": ["10", "AND", "1"]
    }
  ],
  "httpSourceName": "-",
  "httpSourceId": "-",
  "ruleGroupList": [],
  "rateBasedRuleList": [],
  "nonTerminatingMatchingRules": []
}
```

parsePostgres

このプロセッサを使用して、 Amazon RDS for PostgreSQL 提供されたログを解析し、フィールドを抽出して、JSON 形式に変換します。RDS for PostgreSQL ログ形式の詳細については、「RDS for PostgreSQL データベースログファイル」を参照してください。

このプロセッサは、 を入力`@message`としてのみ受け入れます。

> **Important**
>
> このプロセッサを使用する場合は、トランスフォーマーの最初のプロセッサである必要があります。

例

次のログイベントの例を取ります。

```
2019-03-10 03:54:59 UTC:10.0.0.123(52834):postgres@logtestdb:[20175]:ERROR: column
 "wrong_column_name" does not exist at character 8
```

プロセッサ設定は次のとおりです。

```
[
   "parsePostgres": {}
]
```

変換されたログイベントは次のとおりです。

```
{
```

```
  "logTime": "2019-03-10 03:54:59 UTC",
  "srcIp": "10.0.0.123(52834)",
  "userName": "postgres",
  "dbName": "logtestdb",
  "processId": "20175",
  "logLevel": "ERROR"
}
```

parseCloudfront

このプロセッサを使用して、 Amazon CloudFront 提供されたログを解析し、フィールドを抽出して、JSON 形式に変換します。エンコードされたフィールド値はデコードされます。整数と倍精度の値は、そのように扱われます。 Amazon CloudFront ログ形式の詳細については、「標準ログの設定と使用 (アクセスログ) 」を参照してください。

このプロセッサは、 を入力@messageとしてのみ受け入れます。

> **Important**
>
> このプロセッサを使用する場合は、トランスフォーマーの最初のプロセッサである必要があります。

例

次のログイベントの例を取ります。

```
2019-12-04  21:02:31   LAX1   392    192.0.2.24    GET
 d111111abcdef8.cloudfront.net  /index.html    200    -  Mozilla/5.0%20(Windows
%20NT%2010.0;%20Win64;%20x64)%20AppleWebKit/537.36%20(KHTML,
%20like%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36  -  -  Hit
    SOX4xwn4XV6Q4rgb7XiVGOHms_BGlTAC4KyHmureZmBNrjGdRLiNIQ==
 d111111abcdef8.cloudfront.net  https  23 0.001  -  TLSv1.2    ECDHE-RSA-AES128-GCM-
SHA256    Hit    HTTP/2.0   -  -  11040  0.001  Hit    text/html  78 -  -
```

プロセッサ設定は次のとおりです。

```
[
   "parseCloudfront": {}
]
```

変換されたログイベントは次のとおりです。


```
{
  "date": "2019-12-04",
  "time": "21:02:31",
  "x-edge-location": "LAX1",
  "sc-bytes": 392,
  "c-ip": "192.0.2.24",
  "cs-method": "GET",
  "cs(Host)": "d111111abcdef8.cloudfront.net",
  "cs-uri-stem": "/index.html",
  "sc-status": 200,
  "cs(Referer)": "-",
  "cs(User-Agent)": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36",
  "cs-uri-query": "-",
  "cs(Cookie)": "-",
  "x-edge-result-type": "Hit",
  "x-edge-request-id": "SOX4xwn4XV6Q4rgb7XiVGOHms_BGlTAC4KyHmureZmBNrjGdRLiNIQ==",
  "x-host-header": "d111111abcdef8.cloudfront.net",
  "cs-protocol": "https",
  "cs-bytes": 23,
  "time-taken": 0.001,
  "x-forwarded-for": "-",
  "ssl-protocol": "TLSv1.2",
  "ssl-cipher": "ECDHE-RSA-AES128-GCM-SHA256",
  "x-edge-response-result-type": "Hit",
  "cs-protocol-version": "HTTP/2.0",
  "fle-status": "-",
  "fle-encrypted-fields": "-",
  "c-port": 11040,
  "time-to-first-byte": 0.001,
  "x-edge-detailed-result-type": "Hit",
  "sc-content-type": "text/html",
  "sc-content-len": 78,
  "sc-range-start": "-",
  "sc-range-end": "-"
}
```


parseRoute53

このプロセッサを使用して、 Amazon Route 53 Public Data Plane 提供されたログを解析し、フィールドを抽出して、JSON 形式に変換します。エンコードされたフィールド値はデコードされます。

このプロセッサは、 を入力@messageとしてのみ受け入れます。

> **Important**
>
> このプロセッサを使用する場合は、トランスフォーマーの最初のプロセッサである必要があります。

例

次のログイベントの例を取ります。

```
1.0 2017-12-13T08:15:50.235Z Z123412341234 example.com AAAA NOERROR TCP IAD12 192.0.2.0
 198.51.100.0/24
```

プロセッサ設定は次のとおりです。

```
[
   "parseRoute53": {}
]
```

変換されたログイベントは次のとおりです。

```
{
  "version": 1.0,
  "queryTimestamp": "2017-12-13T08:15:50.235Z",
  "hostZoneId": "Z123412341234",
  "queryName": "example.com",
  "queryType": "AAAA",
  "responseCode": "NOERROR",
  "protocol": "TCP",
  "edgeLocation": "IAD12",
  "resolverIp": "192.0.2.0",
  "ednsClientSubnet": "198.51.100.0/24"
}
```

parseVPC

このプロセッサを使用して、Amazon VPC 提供のログを解析し、フィールドを抽出して、JSON 形式に変換します。エンコードされたフィールド値はデコードされます。

> **Important**
>
> parseVPC プロセッサは、デフォルトの Amazon VPC フローログ形式のログでのみ機能します。カスタム VPC フローログでは機能しません。

このプロセッサは、 を入力@messageとしてのみ受け入れます。

> **Important**
>
> このプロセッサを使用する場合は、トランスフォーマーの最初のプロセッサである必要があります。

例

次のログイベントの例を取ります。

```
2 123456789010 eni-abc123de 192.0.2.0 192.0.2.24 20641 22 6 20 4249 1418530010
 1418530070 ACCEPT OK
```

プロセッサ設定は次のとおりです。

```
[
   "parseVPC": {}
]
```

変換されたログイベントは次のとおりです。

```
{
  "version": 2,
  "accountId": "123456789010",
  "interfaceId": "eni-abc123de",
  "srcAddr": "192.0.2.0",
  "dstAddr": "192.0.2.24",
  "srcPort": 20641,
  "dstPort": 22,
  "protocol": 6,
  "packets": 20,
  "bytes": 4249,
```

```
  "start": 1418530010,
  "end": 1418530070,
  "action": "ACCEPT",
  "logStatus": "OK"
}
```

文字列ミューテーションプロセッサ

lowerCaseString

lowerCaseString プロセッサは文字列を小文字のバージョンに変換します。

フィールド	説明	必須?	デフォルト値	制限
withKeys	小文字に変換するキーのリスト	はい		最大エントリ: 10

例

次のログイベントの例を取ります。

```
{
    "outer_key": {
        "inner_key": "INNER_VALUE"
    }
}
```

トランスフォーマーの設定は次のとおりです。 lowerCaseStringで を使用しますparseJSON。

```
[
   "parseJSON": {},
    "lowerCaseString": {
        "withKeys":["outer_key.inner_key"]
    }
]
```

変換されたログイベントは次のとおりです。

```
{
```

```
  "outer_key": {
    "inner_key": "inner_value"
  }
}
```

upperCaseString

upperCaseString プロセッサは文字列を大文字バージョンに変換します。

フィールド	説明	必須?	デフォルト値	制限
withKeys	大文字に変換するキーのリスト	はい		最大エントリ: 10

例

次のログイベントの例を取ります。

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

トランスフォーマーの設定は次のようになります。 upperCaseStringで を使用しますparseJSON。

```
[
   "parseJSON": {},
    "upperCaseString": {
        "withKeys":["outer_key.inner_key"]
    }
]
```

変換されたログイベントは次のとおりです。

```
{
  "outer_key": {
    "inner_key": "INNER_VALUE"
  }
```

```
}
```

splitString

splitString プロセッサは、区切り文字を使用してフィールドを配列に分割します。

フィールド	説明	必須?	デフォルト値	制限
エントリ	エントリの配列。配列内の各項目には、 source および delimiter フィールドが含まれている必要があります。	はい		最大エントリ: 100
ソース	分割するキー	はい		最大長: 128
delimiter	分割を担当する区切り文字	はい		最大長: 1

例

次のログイベントの例を取ります。

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

トランスフォーマーの設定は次のようになります。 splitStringで を使用しますparseJSON。

```
[
   "parseJSON": {},
    "splitString": {
        "entries": [
            {
                "source": "outer_key.inner_key",
                "delimiter": "_"
            }
        ]
    }
```

```
]
```

変換されたログイベントは次のとおりです。

```
{
  "outer_key": {
    "inner_key": [
      "inner",
      "value"
    ]
  }
}
```

substituteString

substituteString プロセッサは、キーの値を正規表現と照合し、すべての一致を置き換え文字列に置き換えます。

フィールド	説明	必須?	デフォルト値	制限
エントリ	エントリの配列。配列の各項目にはsource、、from、およびtoフィールドが含まれている必要があります。	はい		最大エントリ: 10
ソース	変更するキー	はい		最大長: 128 ネストされたキーの最大深さ: 3
送信元	置き換える正規表現文字列。[や] などの特別な正規表現文字は、二重引用符を使用する場合は \\、一重引用符を使用する場合は \ を使用してエスケープする必要があります。詳細については、Oracle ウェブサイトの「Class Pattern」を参照してください。	はい		最大長: 128

フィールド	説明	必須?	デフォルト値	制限
次のように変更します。	の一致ごとに置き換えられる文字列 from	はい		最大長: 128

例

次のログイベントの例を取ります。

```
{
    "outer_key": {
        "inner_key1": "[]",
        "inner_key2": "123-345-567"
    }
}
```

トランスフォーマーの設定は次のようになります。 substituteStringで を使用しますparseJSON。

```
[
   "parseJSON": {},
    "substituteString": {
        "entries": [
            {
                "source": "outer_key.inner_key1",
                "from": "\\[\\]",
                "to": "value1"
            },
            {
                "source": "outer_key.inner_key2",
                "from": "[0-9]{3}-[0-9]{3}-[0-9]{3}",
                "to": "xxx-xxx-xxx"
            }
        ]
    }
]
```

変換されたログイベントは次のとおりです。

```
{
  "outer_key": {
    "inner_key1": "value1",
    "inner_key2": "xxx-xxx-xxx"
  }
}
```

trimString

trimString プロセッサは、キーの先頭と末尾から空白を削除します。

フィールド	説明	必須?	デフォルト値	制限
withKeys	トリミングするキーのリスト	はい		最大エントリ: 10

例

次のログイベントの例を取ります。

```
{
    "outer_key": {
        "inner_key": "   inner_value  "
    }
}
```

トランスフォーマーの設定は次のようになります。 trimStringで を使用しますparseJSON。

```
[
   "parseJSON": {},
    "trimString": {
        "withKeys":["outer_key.inner_key"]
    }
]
```

変換されたログイベントは次のとおりです。

```
{
  "outer_key": {
```

```
    "inner_key": "inner_value"
  }
}
```

JSON ミューテーションプロセッサ

addKeys

addKeys プロセッサを使用して、ログイベントに新しいキーと値のペアを追加します。

フィールド	説明	必須?	デフォルト値	制限
エントリ	エントリの配列。配列内の各項目にはkey、、value、および overwriteIfExists フィールドを含めることができます。	はい		最大エントリ: 5
キー	追加する新しいエントリのキー	はい		最大長: 128 ネストされたキーの最大深さ: 3
値	追加する新しいエントリの値	はい		最大長: 256
overwriteIfExists	これを に設定するとtrue、イベントにkey既に存在する場合、既存の値が上書きされます。デフォルト値は false です。	いいえ	false	

例

次のログイベントの例を取ります。

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

トランスフォーマーの設定は次のようになります。 addKeysで を使用しますparseJSON。

```
[
   "parseJSON": {},
   "addKeys": {
        "entries": [
            {
                "source": "outer_key.new_key",
                "value": "new_value"
            }
        ]
    }
]
```

変換されたログイベントは次のとおりです。

```
{
  "outer_key": {
    "inner_key": "inner_value",
    "new_key": "new_value"
  }
}
```

deleteKeys

deleteKeys プロセッサを使用して、ログイベントからフィールドを削除します。これらのフィールドには、キーと値のペアを含めることができます。

フィールド	説明	必須?	デフォルト値	制限
withKeys	削除するキーのリスト。	はい		最大エントリ: 5

例

次のログイベントの例を取ります。

```
{
    "outer_key": {
        "inner_key": "inner_value"
```

```
    }
}
```

トランスフォーマーの設定は次のようになります。 deleteKeysで を使用しますparseJSON。

```
[
   "parseJSON": {},
    "deleteKeys": {
        "withKeys":["outer_key.inner_key"]
    }
]
```

変換されたログイベントは次のとおりです。

```
{
  "outer_key": {}
}
```

moveKeys

moveKeys プロセッサを使用して、あるフィールドから別のフィールドにキーを移動します。

フィールド	説明	必須?	デフォルト値	制限
エントリ	エントリの配列。配列内の各項目にはsource、、target、およびoverwriteIfExists フィールドを含めることができます。	はい		最大エントリ: 5
ソース	移動するキー	はい		最大長: 128 ネストされたキーの最大深さ: 3
target	移動先のキー	はい		最大長: 128 ネストされたキーの最大深さ: 3

フィールド	説明	必須?	デフォルト値	制限
overwriteIfExists	これを に設定するとtrue、イベントにkey既に存在する場合、既存の値が上書きされます。デフォルト値は false です。	いいえ	false	

例

次のログイベントの例を取ります。

```
{
    "outer_key1": {
        "inner_key1": "inner_value1"
    },
    "outer_key2": {
        "inner_key2": "inner_value2"
    }
}
```

トランスフォーマーの設定は次のようになります。 moveKeysで を使用しますparseJSON。

```
[
   "parseJSON": {},
   "moveKeys": {
        "entries": [
            {
                "source": "outer_key1.inner_key1",
                "target": "outer_key2"
            }
        ]
    }
]
```

変換されたログイベントは次のとおりです。

```
{
  "outer_key1": {},
  "outer_key2": {
```

```
    "inner_key2": "inner_value2",
    "inner_key1": "inner_value1"
  }
}
```

renameKeys

renameKeys プロセッサを使用して、ログイベントのキーの名前を変更します。

フィールド	説明	必須?	デフォルト値	制限
エントリ	エントリの配列。配列内の各項目にはkey、、target、およびoverwriteIfExists フィールドを含めることができます。	はい		最大エントリ: 5
キー	名前を変更するキー	はい		最大長: 128
target	新しいキー名	はい		最大長: 128 ネストされたキーの最大深さ: 3
overwriteIfExists	これを に設定するとtrue、イベントにkey既に存在する場合、既存の値が上書きされます。デフォルト値は false です。	いいえ	false	

例

次のログイベントの例を取ります。

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

トランスフォーマーの設定は次のようになります。 renameKeysで を使用しますparseJSON。

```
[
   "parseJSON": {},
   "renameKeys": {
        "entries": [
            {
                "key": "outer_key",
                "target": "new_key"
            }
        ]
    }
]
```

変換されたログイベントは次のとおりです。

```
{
  "new_key": {
    "inner_key": "inner_value"
  }
}
```

copyValue

copyValue プロセッサを使用して、ログイベント内の値をコピーします。このプロセッサを使用して、次のメタデータキーの値をログイベントにコピーすることで、ログイベントにメタデータを追加することもできます: @logGroupName、@logGroupStream、@accountId、@regionName。これは次の例に示されています。

フィールド	説明	必須?	デフォルト値	制限
エントリ	エントリの配列。配列内の各項目にはsource、、target、およびoverwriteIfExists フィールドを含めることができます。	はい		最大エントリ: 5
ソース	コピーするキー	はい		最大長: 128 ネストされたキーの最大深さ: 3

フィールド	説明	必須?	デフォルト値	制限
target	値をコピーする キー	はい		最大長: 128 ネストされたキーの最大深さ: 3
overwriteIfExists	これを に設定するとtrue、イベントにkey既に存在する場合、既存の値が上書きされます。デフォルト値は false です。	いいえ	false	

例

次のログイベントの例を取ります。

```
{
    "outer_key": {
        "inner_key": "inner_value"
    }
}
```

トランスフォーマーの設定は次のようになります。 copyValueで を使用しますparseJSON。

```
[
   "parseJSON": {},
   "copyValue": {
        "entries": [
            {
                "source": "outer_key.new_key",
                "target": "new_key"
            },
            {
                "source": "@logGroupName",
                "target": "log_group_name"
            },
            {
                "source": "@logGroupStream",
                "target": "log_group_stream"
```

```
            },
            {
                "source": "@accountId",
                "target": "account_id"
            },
            {
                "source": "@regionName",
                "target": "region_name"
            }
        ]
    }
]
```

変換されたログイベントは次のとおりです。

```
{
  "outer_key": {
    "inner_key": "inner_value"
  },
  "new_key": "inner_value",
  "log_group_name": "myLogGroupName",
  "log_group_stream": "myLogStreamName",
  "account_id": "012345678912",
  "region_name": "us-east-1"
}
```

listToMap

`listToMap` プロセッサは、キーフィールドを含むオブジェクトのリストを取得し、ターゲットキーのマップに変換します。

フィールド	説明	必須?	デフォルト値	制限
ソース	マップに変換されるオブジェクトのリストを含む ProcessingEvent のキー	はい		最大長: 128 ネストされたキーの最大深度: 3
キー	生成されたマップでキーとして抽出されるフィールドのキー	はい		最大長: 128

フィールド	説明	必須?	デフォルト値	制限
valueKey	これを指定すると、このパラメータで指定した値がsourceオブジェクトから抽出され、生成されたマップの値に挿入されます。それ以外の場合、ソースリスト内の元のオブジェクトは、生成されたマップの値に配置されます。	いいえ		最大長: 128
target	生成されたマップを保持するフィールドのキー	いいえ	ルートノード	最大長: 128 ネストされたキーの最大深度: 3
flatten	リストを単一の項目にフラット化するか、生成されたマップの値がリストになるかを示すブール値。 デフォルトでは、一致するキーの値は配列で表されます。flattenに設定するtrueと、配列は の値に基づいて 1 つの値に変換されますflattenedElement 。	いいえ	false	
flattenedElement	flatten を に設定する場合はtrue、 flattenedElement を使用して、保持lastする 要素firstまたは を指定します。	flattenが に設定されている場合に必要です true		値は firstまたは のみです last

例

次の口グイベントの例を取ります。

```
{
```

```
    "outer_key": [
        {
            "inner_key": "a",
            "inner_value": "val-a"
        },
        {
            "inner_key": "b",
            "inner_value": "val-b1"
        },
        {
            "inner_key": "b",
            "inner_value": "val-b2"
        },
        {
            "inner_key": "c",
            "inner_value": "val-c"
        }
    ]
}
```

ユースケース 1 のトランスフォーマー： は flattenです false

```
[
   "parseJSON": {},
   "listToMap": {
        "source": "outer_key"
        "key": "inner_key",
        "valueKey": "inner_value",
        "flatten": false
    }
]
```

変換されたログイベントは次のとおりです。

```
{
    "outer_key": [
        {
            "inner_key": "a",
            "inner_value": "val-a"
        },
        {
            "inner_key": "b",
            "inner_value": "val-b1"
```

```
        },
        {
            "inner_key": "b",
            "inner_value": "val-b2"
        },
        {
            "inner_key": "c",
            "inner_value": "val-c"
        }
    ],
    "a": [
        "val-a"
    ],
    "b": [
        "val-b1",
        "val-b2"
    ],
    "c": [
        "val-c"
    ]
}
```

ユースケース 2 のトランスフォーマー： `flatten` は `true`で、 `flattenedElement`は です `first`

```
[
   "parseJSON": {},
   "listToMap": {
        "source": "outer_key"
        "key": "inner_key",
        "valueKey": "inner_value",
        "flatten": true,
        "flattenedElement": "first"
    }
]
```

変換されたログイベントは次のとおりです。

```
{
    "outer_key": [
        {
            "inner_key": "a",
            "inner_value": "val-a"
        },
```

```
        {
            "inner_key": "b",
            "inner_value": "val-b1"
        },
        {
            "inner_key": "b",
            "inner_value": "val-b2"
        },
        {
            "inner_key": "c",
            "inner_value": "val-c"
        }
    ],
    "a": "val-a",
    "b": "val-b1",
    "c": "val-c"
}
```

ユースケース 3 のトランスフォーマー： `flatten` は `true`で、 `flattenedElement`は です `last`

```
[
   "parseJSON": {},
   "listToMap": {
        "source": "outer_key"
        "key": "inner_key",
        "valueKey": "inner_value",
        "flatten": true,
        "flattenedElement": "last"
    }
]
```

変換されたログイベントは次のとおりです。

```
{
    "outer_key": [
        {
            "inner_key": "a",
            "inner_value": "val-a"
        },
        {
            "inner_key": "b",
            "inner_value": "val-b1"
        },
```

```
        {
            "inner_key": "b",
            "inner_value": "val-b2"
        },
        {
            "inner_key": "c",
            "inner_value": "val-c"
        }
    ],
    "a": "val-a",
    "b": "val-b2",
    "c": "val-c"
}
```

データ型コンバータプロセッサ

typeConverter

typeConverter プロセッサを使用して、指定されたキーに関連付けられた値タイプを指定されたタイプに変換します。これは、指定されたフィールドのタイプを変更するキャストプロセッサです。値は、、、integer、 double string のいずれかのデータ型に変換できますboolean。

フィールド	説明	必須?	デフォルト値	制限
エントリ	エントリの配列。配列の各項目には、 key および typeフィールドが含まれている必要があります。	はい		最大エントリ: 10
キー	別のタイプに変換される値を持つキー	はい		最大長: 128 ネストされたキーの最大深度: 3
type	変換するタイプ。有効な値は、integer、double、string、です。	はい		

例

次のログイベントの例を取ります。

```
{
    "name": "value",
    "status": "200"
}
```

トランスフォーマーの設定は次のようになります。 typeConverterで を使用しますparseJSON。

```
[
   "parseJSON": {},
   "typeConverter": {
        "entries": [
            {
                "key": "status",
                "type": "integer"
            }
        ]
    }
]
```

変換されたログイベントは次のとおりです。

```
{
    "name": "value",
    "status": 200
}
```

datetimeConverter

datetimeConverter プロセッサを使用して、日時文字列を指定した形式に変換します。

フィールド	説明	必須?	デフォルト値	制限
ソース	日付変換を適用するキー。	はい		最大エントリ: 10
matchPatterns	source フィールドに一致するパターンのリスト	はい		最大エントリ: 5

フィールド	説明	必須?	デフォルト値	制限
target	結果を保存する JSON フィールド。	はい		最大長: 128 ネストされたキーの最大深度: 3
targetFormat	ターゲットフィールドで変換されたデータに使用する日時形式。	いいえ	`yyyy-MM-dd'T'HH:mm:ss.SSS'Z`	最大長: 64
sourceTimezone	ソースフィールドのタイムゾーン。 可能な値のリストについては、「Java でサポートされているゾーン ID とオフセット」を参照してください。	いいえ	UTC	最小長: 1
targetTimezone	ターゲットフィールドのタイムゾーン。 可能な値のリストについては、「Java でサポートされているゾーン ID とオフセット」を参照してください。	いいえ	UTC	最小長: 1
サイト	ソースフィールドのロケール。 可能な値のリストについては、「例を含む Java のロケール getAvailableLocales() メソッド」を参照してください。	はい		最小長: 1

例

次のログイベントの例を取ります。

```
{"german_datetime": "Samstag 05. Dezember 1998 11:00:00"}
```

トランスフォーマーの設定は次のようになります。 dateTimeConverterで を使用しますparseJSON。

```
[
   "parseJSON": {},
   "dateTimeConverter": {
        "source": "german_datetime",
        "target": "target_1",
        "locale": "de",
        "matchPatterns": ["EEEE dd. MMMM yyyy HH:mm:ss"],
        "sourceTimezone": "Europe/Berlin",
        "targetTimezone": "America/New_York",
        "targetFormat": "yyyy-MM-dd'T'HH:mm:ss z"
    }
]
```

変換されたログイベントは次のとおりです。

```
{
    "german_datetime": "Samstag 05. Dezember 1998 11:00:00",
    "target_1": "1998-12-05T17:00:00 MEZ"
}
```

変換メトリクスとエラー

CloudWatch Logs は、変換メトリクスを CloudWatch に発行します。これらのメトリクスには、TransformedLogEvents、TransformedBytes、および が含まれますTransformationErrors。詳細については、「ログトランスフォーマーのメトリクスとディメンション」を参照してください。

CloudWatch Logs がログイベントを変換しようとして変換に失敗するたびに、そのログイベントに@transformationErrorシステムフィールドが追加されます。CloudWatch Logs Insights クエリを実行すると、変換に失敗したすべてのログイベントにこのフィールドが表示されます。このフィールドを などのクエリでクエリfilter ispresent(@transformationError)して、失敗した変換イベントをすべて見つけることができます。

Amazon OpenSearch Service で分析する

CloudWatch Logs は と統合 Amazon OpenSearch Service されており、OpenSearch Service が AWS サービスから提供されたログから派生する主要なメトリクスを表示する自動キュレーションダッシュボードを作成できます。次のダッシュボードを使用できます。

- Amazon VPC フローログダッシュボードは、Amazon VPC のネットワークフローデータをキャプチャします。ネットワークトラフィックの分析、異常なパターンの検出、リソースの使用状況のモニタリングに役立ちます。表示される主要なメトリクスは次のとおりです。
  - フローの合計と、これらのフローの承諾と拒否
  - 時間の経過に伴うトラフィックパターン
  - 送信元 IP と送信先 IPs (トップトーカー) 間のデータフローを示すサンキー図
  - 転送されたバイト数とパケット数の上位 IPs

  > **Note**
  >
  > 現在、VPC バージョン 2 のフィールド形式のみがサポートされています。

- AWS WAF ログダッシュボードは、モニタリング対象のウェブトラフィックに関するインサイトを提供します AWS WAF。このダッシュボードは、特定のリージョンまたは IPs からのトラフィックパターン、ブロックされたリクエスト、および潜在的な脅威を特定するのに役立ちます。表示される主要なメトリクスは次のとおりです。
  - 「許可」および「ブロック」カウントによるリクエストの合計。
  - 時間の経過に伴うリクエスト履歴。許可されたリクエストとブロックされたリクエストが表示されます。
  - ウェブ ACL 名によるリクエストの内訳、終了ルールによるブロックされたリクエスト、およびソース IPs。
  - リクエストオリジンの地理的分散。
  - リクエスト数別の上位クライアント IPs と終了ルール。
- CloudTrail ログダッシュボードには、CloudTrail ログを使用して AWS 環境内の API アクティビティの概要が表示されます。API アクティビティのモニタリング、アクションの監査、潜在的なセキュリティまたはコンプライアンスの問題の特定に役立ちます。表示される主要なメトリクスは次のとおりです。
  - 経時的なイベント数とイベント履歴の合計

- アカウント IDs、カテゴリ、リージョン別のイベントの内訳。
- イベントの生成に関連する上位APIs、サービス、ソース IPs。
- イベントを生成している上位ユーザーのテーブル。ユーザーアカウント情報とイベント数の詳細を示します。

これらの厳選されたダッシュボードに表示されるメトリクスは、 Amazon OpenSearch Service 分析から派生します。

これらのダッシュボードを表示する前に、IAM ロールを作成し、CloudWatch Logs と 1 回限りの統合を実行する必要があります Amazon OpenSearch Service。この 1 回限りの統合により、ダッシュボードの作成とレンダリングに必要な Amazon OpenSearch Service リソースが設定されます。使用した OpenSearch サービスには料金が発生します。詳細については、「Amazon CloudWatch 料金表」をご覧ください。

これらのキュレートされたダッシュボードは、標準ログクラスのロググループに対してのみ作成できます。

> **Important**
>
> 提供されたログダッシュボードを作成するロググループには、ログトランスフォーマーを使用しないでください。ログイベントを変換すると、ダッシュボードのデータが空になります。

トピック

ステップ 1: OpenSearch Service との統合を作成する

最初のステップは、OpenSearch Service との統合を作成することです。これは 1 回だけ行う必要があります。統合を作成すると、アカウントに次のリソースが作成されます。

- 高可用性のないOpenSearch Service 時系列コレクション。

  コレクションは、ワークロードをサポートするために連携する一連の OpenSearch Service インデックスです。
- コレクションの 2 つのセキュリティポリシー。1 つは、カスタマーマネージド AWS KMS キーまたはサービス所有キーのいずれかを使用して暗号化タイプを定義します。もう 1 つのポリシーはネットワークアクセスを定義し、OpenSearch Service アプリケーションがコレクションにアクセスできるようにします。詳細については、「Amazon OpenSearch Service の保管中のデータの暗号化」を参照してください。
- コレクション内のデータにアクセスできるユーザーを定義する OpenSearch Service データアクセスポリシー。
- ソースとして CloudWatch Logs が定義された OpenSearch Service ダイレクトクエリデータソース。
- という名前の OpenSearch Service アプリケーションaws-analytics。アプリケーションは、ワークスペースの作成を許可するように設定されます。という名前のアプリケーションがaws-analytics既に存在する場合は、このコレクションをデータソースとして追加するように更新されます。
- ダッシュボードをホストし、ワークスペースから読み取るためのアクセス許可が付与されたすべてのユーザーを許可する OpenSearch Service ワークスペース。

トピック

必要なアクセス許可

統合を作成するには、CloudWatchOpenSearchDashboardsFullAccess マネージド IAM ポリシーまたは同等のアクセス許可を持つアカウントにサインインする必要があります。統合の削除、ダッシュボードの作成、編集、削除、およびダッシュボードの手動更新を行うには、これらのアクセス許可も必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Sid": "CloudWatchOpenSearchDashboardsIntegration",
            "Effect": "Allow",
```

```
            "Action": [
                "logs:ListIntegrations",
                "logs:GetIntegration",
                "logs:DeleteIntegration",
                "logs:PutIntegration",
                "logs:DescribeLogGroups",
                "opensearch:ApplicationAccessAll",
                "iam:ListRoles",
                "iam:ListUsers"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchLogsOpensearchReadAPIs",
            "Effect": "Allow",
            "Action": [
                "aoss:BatchGetCollection",
                "aoss:BatchGetLifecyclePolicy",
                "es:ListApplications"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsOpensearchCreateServiceLinkedAccess",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
opensearchservice.amazonaws.com/AWSServiceRoleForAmazonOpenSearchService",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "opensearchservice.amazonaws.com",
                    "aws:CalledViaFirst": "logs.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsObservabilityCreateServiceLinkedAccess",
```

```
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
observability.aoss.amazonaws.com/AWSServiceRoleForAmazonOpenSearchServerless",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "observability.aoss.amazonaws.com",
                    "aws:CalledViaFirst": "logs.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsCollectionRequestAccess",
            "Effect": "Allow",
            "Action": [
                "aoss:CreateCollection"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
                    "aws:RequestTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ]
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "CloudWatchOpenSearchIntegration"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsApplicationRequestAccess",
            "Effect": "Allow",
            "Action": [
                "es:CreateApplication"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
                    "aws:RequestTag/OpenSearchIntegration": [
                        "Dashboards"
```

```
                    ]
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "OpenSearchIntegration"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsCollectionResourceAccess",
            "Effect": "Allow",
            "Action": [
                "aoss:DeleteCollection"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
                    "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ]
                }
            }
        },
        {
            "Sid": "CloudWatchLogsApplicationResourceAccess",
            "Effect": "Allow",
            "Action": [
                "es:UpdateApplication",
                "es:GetApplication"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
                    "aws:ResourceTag/OpenSearchIntegration": [
                        "Dashboards"
                    ]
                }
            }
        },
        {
            "Sid": "CloudWatchLogsCollectionPolicyAccess",
            "Effect": "Allow",
            "Action": [
```

```
                "aoss:CreateSecurityPolicy",
                "aoss:CreateAccessPolicy",
                "aoss:DeleteAccessPolicy",
                "aoss:DeleteSecurityPolicy",
                "aoss:GetAccessPolicy",
                "aoss:GetSecurityPolicy"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aoss:collection": "cloudwatch-logs-*",
                    "aws:CalledViaFirst": "logs.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsAPIAccessAll",
            "Effect": "Allow",
            "Action": [
                "aoss:APIAccessAll"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aoss:collection": "cloudwatch-logs-*"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsIndexPolicyAccess",
            "Effect": "Allow",
            "Action": [
                "aoss:CreateAccessPolicy",
                "aoss:DeleteAccessPolicy",
                "aoss:GetAccessPolicy",
                "aoss:CreateLifecyclePolicy",
                "aoss:DeleteLifecyclePolicy"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aoss:index": "cloudwatch-logs-*",
                    "aws:CalledViaFirst": "logs.amazonaws.com"
                }
```

```
            }
        },
        {
            "Sid": "CloudWatchLogsDQSRequestQueryAccess",
            "Effect": "Allow",
            "Action": [
                "es:AddDirectQueryDataSource"
            ],
            "Resource": "arn:aws:opensearch:*:*:datasource/cloudwatch_logs_*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
                    "aws:RequestTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ]
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "CloudWatchOpenSearchIntegration"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsStartDirectQueryAccess",
            "Effect": "Allow",
            "Action": [
                "opensearch:StartDirectQuery",
                "opensearch:GetDirectQuery"
            ],
            "Resource": "arn:aws:opensearch:*:*:datasource/cloudwatch_logs_*"
        },
        {
            "Sid": "CloudWatchLogsDQSResourceQueryAccess",
            "Effect": "Allow",
            "Action": [
                "es:GetDirectQueryDataSource",
                "es:DeleteDirectQueryDataSource"
            ],
            "Resource": "arn:aws:opensearch:*:*:datasource/cloudwatch_logs_*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
                    "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ]
```

```
                }
            }
        },
        {
            "Sid": "CloudWatchLogsPassRoleAccess",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService":
"directquery.opensearchservice.amazonaws.com",
                    "aws:CalledViaFirst": "logs.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsAossTagsAccess",
            "Effect": "Allow",
            "Action": [
                "aoss:TagResource"
            ],
            "Resource": "arn:aws:aoss:*:*:collection/*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
                    "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ]
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "CloudWatchOpenSearchIntegration"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsEsApplicationTagsAccess",
            "Effect": "Allow",
            "Action": [
                "es:AddTags"
            ],
            "Resource": "arn:aws:opensearch:*:*:application/*",
```

```
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/OpenSearchIntegration": [
                        "Dashboards"
                    ],
                    "aws:CalledViaFirst": "logs.amazonaws.com"
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "OpenSearchIntegration"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsEsDataSourceTagsAccess",
            "Effect": "Allow",
            "Action": [
                "es:AddTags"
            ],
            "Resource": "arn:aws:opensearch:*:*:datasource/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ],
                    "aws:CalledViaFirst": "logs.amazonaws.com"
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "CloudWatchOpenSearchIntegration"
                }
            }
        }
    ]
}
```

統合を作成する

以下のステップを使用して統合を作成します。

CloudWatch Logs を と統合するには Amazon OpenSearch Service

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. 左側のナビゲーションペインで、Logs Insights を選択し、OpenSearch で分析タブを選択します。

3. [統合を作成する] を選択します。
4. [統合名] に、統合の名前を入力します。
5. （オプション) OpenSearch Service Serverless に書き込まれるデータを暗号化するには、KMS AWS KMS キー ARN で使用するキーの ARN を入力します。 詳細については、「Amazon OpenSearch Service デベロッパーガイド」の「保管時の暗号化」を参照してください。
6. データ保持には、OpenSearch Service データインデックスを保持する時間を入力します。これにより、ダッシュボードでデータを表示できる最大期間も定義されます。データ保持期間を長くすると、追加の検索とインデックス作成のコストが発生します。詳細については、OpenSearch Service Serverless の料金」を参照してください。

   最大保持期間は 30 日間です。

   データ保持期間の長さは、OpenSearch Service コレクションライフサイクルポリシーの作成にも使用されます。
7. OpenSearch コレクションに書き込む IAM ロールの場合は、新しい IAM ロールを作成するか、OpenSearch Service コレクションへの書き込みに使用する既存の IAM ロールを選択します。

   新しいロールの作成は最も簡単な方法です。ロールは必要なアクセス許可で作成されます。

   > **Note**
   >
   > ロールを作成すると、アカウント内のすべてのロググループから読み取るアクセス許可が付与されます。

   既存のロールを選択する場合は、 に記載されているアクセス許可が必要です統合に必要なアクセス許可。または、既存のロールを使用する を選択し、選択したロールのアクセス許可の検証セクションでロールの作成 を選択します。これにより、 にリストされているアクセス許可をテンプレート統合に必要なアクセス許可として使用して変更できます。例えば、ロググループのよりきめ細かな制御を指定する場合などです。
8. ダッシュボードを表示できる IAM ロールとユーザーの場合、提供されたログダッシュボードアクセスのために IAM ロールと IAM ユーザーへのアクセスを許可する方法を選択します。
   - ダッシュボードへのアクセスを一部のユーザーのみに制限するには、ダッシュボードを表示できる IAM ロールとユーザーの選択を選択し、テキストボックスでアクセスを許可する IAM ロールと IAM ユーザーを検索して選択します。

- ダッシュボードへのアクセス権をすべてのユーザーに付与するには、このアカウントのすべてのロールとユーザーにダッシュボードの表示を許可するを選択します。

> **Important**
>
> ロールまたはユーザーを選択するか、すべてのユーザーを選択すると、 はダッシュボードデータを保存する OpenSearch Service コレクションへのアクセスに必要なデータアクセスポリシーにのみ追加します。 OpenSearch 販売者が販売されたログダッシュボードを表示できるようにするには、それらのロールとユーザーにCloudWatchOpenSearchDashboardAccessマネージド IAM ポリシーも付与する必要があります。

9. 統合の作成を選択する

   統合の作成には数分かかります。

ステップ 2: 発行されたログダッシュボードを作成する

統合を作成したら、ダッシュボードを作成できます。ダッシュボードは、Amazon VPC フローログ、CloudTrail ログ、および AWS WAF ログで使用できます。

OpenSearch Service によって導出されたメトリクスを使用して提供されるログダッシュボードを作成するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. 左側のナビゲーションペインで、Logs Insights を選択し、OpenSearch で分析タブを選択します。
3. [ダッシュボードの作成] を選択します。
4. ダッシュボードを作成するログのタイプ、Amazon VPC AWS WAFフローログ、または CloudTrail を選択します。
5. ダッシュボードの名前を入力し、オプションで説明を入力します。
6. データ同期頻度には、OpenSearch Service で作成されたメトリクスとインデックスを新しいデータに同期して更新できるように、OpenSearch Service で CloudWatch をクエリする頻度を入力します。OpenSearch Service は、ダッシュボードをレンダリングするためのメトリクスとインデックスをログに作成します。

   より短い時間を選択すると、データが最新の状態になり、コストも高くなります。

7. このダッシュボードのデータを収集するロググループを選択します。作成するダッシュボードのタイプに一致するロググループを必ず選択してください。

   ロググループの参照 ボタンと、選択したロググループからログサンプルを表示 オプションを使用して、必要なロググループを取得できます。

8. [ダッシュボードの作成] を選択します。

   最初は、ダッシュボードはデータなしで表示されます。数分後、データがダッシュボードに表示されます。データが最初に表示されると、最新の 15 分間のログエントリになります。

発行されたログダッシュボードの表示、編集、または削除

CloudWatch Logs または OpenSearch Service で提供されるログダッシュボードを表示する

ダッシュボードを表示するには、CloudWatchOpenSearchDashboardAccess IAM ポリシーを持つ IAM プリンシパルにサインインする必要があります。

発行されたログダッシュボードを表示するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. 左側のナビゲーションペインで、Logs Insights を選択し、OpenSearch で分析タブを選択します。
3. OpenSearch ダッシュボードボックスでダッシュボードを選択します。
4. (オプション) 右上で、OpenSearch で表示を選択します。

   OpenSearch Service コンソールが開き、そこに同じダッシュボードが表示されます。OpenSearch Service コンソールでは、ダッシュボードとそのウィジェットに変更を加えることができます。これらの変更は、CloudWatch Logs でダッシュボードを表示するときにも表示されます。

追加の IAM ロールまたは IAM ユーザーへのアクセス権をダッシュボードに付与する

統合の作成後に追加の IAM プリンシパルへのアクセスを許可するには、次の手順を実行します。

追加の IAM ロールまたはユーザーへのアクセス権を販売ログダッシュボードに付与するには

1. コレクションのデータアクセスポリシーを編集して、これらのロールまたはユーザーを追加します。詳細については、OpenSearch Service デベロッパーガイド」の「Amazon OpenSearch Service Serverless のデータアクセスコントロール」を参照してください。 OpenSearch
2. これらのユーザーに CloudWatchOpenSearchDashboardAccess を付与します。このポリシーの内容の詳細については、「CloudWatchOpenSearchDashboardAccess」を参照してください。

ダッシュボード設定の編集

既存の販売済みログダッシュボードの名前、説明、同期頻度を編集できます。

発行されたログダッシュボードを編集するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. 左側のナビゲーションペインで、Logs Insights を選択し、OpenSearch で分析タブを選択します。
3. OpenSearch ダッシュボードボックスでダッシュボードを選択します。
4. アクション、ダッシュボードの詳細の変更を選択します。
5. 変更を行い、変更の確認を選択します。

提供されたログダッシュボードを削除する

発行されたログダッシュボードを削除できます。これを行うと、OpenSearch Service コレクションで作成されたダッシュボード、メトリクス、インデックスがすべて削除されます。

> **Note**
>
> 発行されたログダッシュボードを削除した後、少なくとも 6 時間待ってから、同じダッシュボードを再作成してください。待機しない場合、再作成されたダッシュボードは正しく機能しません。

発行されたログダッシュボードを削除するには

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

2. 左側のナビゲーションペインで、Logs Insights を選択し、OpenSearch で分析タブを選択します。
3. OpenSearch ダッシュボードボックスでダッシュボードを選択します。
4. [アクション]、[削除] の順に選択します。
5. を入力して決定を確認し**delete**、削除を選択します。

OpenSearch Service とのすべての提供されるログダッシュボード統合を削除する

OpenSearch 統合全体を削除できます。これを行うと、発行されたすべてのログダッシュボードとダッシュボードに表示されたデータが削除されます。

> **Important**
>
> 継続的なコストを避けるため、統合を削除する前に、次のリソースを手動で削除することを強くお勧めします。統合を削除しても、これらのリソースは自動的に削除されません。統合を削除した後は、これらのリソースにアクセスして削除することはできません。削除するリソースの名前を確認するには、次の手順を参照してください。
>
> - データソース
> - コレクション
> - データアクセスポリシー
> - 暗号化ポリシー
> - ネットワークポリシー
> - ライフサイクルポリシー

OpenSearch Service と提供されるログダッシュボードの統合全体を削除するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. 左のナビゲーションペインの [設定] を選択します。
3. [ログ] タブを選択します。
4. OpenSearch 統合セクションで、統合の削除を選択します。

次の画面には、統合を削除する前に削除する必要がある OpenSearch Service リソースの名前が表示されます。

5. を入力して決定を確認し**delete**、統合の削除を選択します。

ユーザーの IAM ポリシー

CloudWatch Logs は、CloudWatchOpenSearchDashboardsFullAccess と CloudWatchOpenSearchDashboardAccess の 2 つの IAM ポリシーを作成しました。次の表に、これらのポリシーで有効にするアクションを示します。

アクション	IAM ポリシー	必要な追加のアクセス許可
統合を作成する	CloudWatchOpenSearchDashboardsFullAccess	
統合を削除する	CloudWatchOpenSearchDashboardsFullAccess	
ダッシュボードの作成	CloudWatchOpenSearchDashboardsFullAccess	
ダッシュボードの編集	CloudWatchOpenSearchDashboardsFullAccess	
ダッシュボードを削除する	CloudWatchOpenSearchDashboardsFullAccess	
今すぐ同期を使用してダッシュボードを更新する	CloudWatchOpenSearchDashboardsFullAccess	
設定で統合を表示する	CloudWatchOpenSearchDashboardAccess または CloudWatchOpenSearchDashboardsFullAccess	
ダッシュボードの表示	CloudWatchOpenSearchDashboardAccess また	統合を作成するときにロールまたはユーザーを指定するか、コレクションのデー

アクション	IAM ポリシー	必要な追加のアクセス許可
	は CloudWatchOpenSearchDashboardsFullAccess	タアクセスポリシーを編集してこれらのロールまたはユーザーを追加します。詳細については、OpenSearch Service デベロッパーガイド」の「Amazon OpenSearch Service Serverless のデータアクセスコントロール」を参照してください。 OpenSearch
OpenSearch Service コンソールでダッシュボードを表示する	CloudWatchOpenSearchDashboardAccess または CloudWatchOpenSearchDashboardsFullAccess	統合を作成するときにロールまたはユーザーを指定するか、コレクションのデータアクセスポリシーを編集してこれらのロールまたはユーザーを追加します。詳細については、OpenSearch Service デベロッパーガイド」の「Amazon OpenSearch Service Serverless のデータアクセスコントロール」を参照してください。 OpenSearch

統合に必要なアクセス許可

統合で使用する IAM ロールを作成する場合、CloudWatch Logs にロールの作成を許可する代わりに、次のアクセス許可と信頼ポリシーを含める必要があります。IAM ロールの作成方法の詳細については、「 AWS サービスにアクセス許可を委任するロールの作成」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudWatchLogsAccess",
      "Effect": "Allow",
      "Action": [
```

```
        "logs:StartQuery",
        "logs:GetLogGroupFields",
        "logs:GetQueryResults"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "CloudWatchLogsDescribeLogGroupsAccess",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": "*"
    },
    {
        "Sid": "AmazonOpenSearchCollectionAccess",
        "Effect": "Allow",
        "Action": [
            "aoss:APIAccessAll"
        ],
        "Resource": "*",
        "Condition": {
            "StringLike": {
                "aoss:collection": "cloudwatch-logs-*"
            }
        }
    }
  ]
}

//Trust Policy
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TrustPolicyForAmazonOpenSearchDirectQueryService",
            "Effect": "Allow",
            "Principal": {
                "Service": "directquery.opensearchservice.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
```

```
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:opensearch:us-
east-1:123456789012:datasource/cloudwatch_logs_*"
                }
            }
        }
    ]
}
```

Note

前のロールは、アカウント内のすべてのロググループから読み取るためのアクセスを許可し、クロスアカウントロググループを含む任意のログアカウントのダッシュボードを作成できるようにします。特定のロググループへのアクセスを制限し、それらのロググループのみのダッシュボードを作成する場合は、そのポリシーの最初のステートメントを次のように更新できます。

```
{
      "Sid": "CloudWatchLogsAccess",
      "Effect": "Allow",
      "Action": [
        "logs:StartQuery",
        "logs:GetLogGroupFields",
        "logs:GetQueryResults"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:123456789012:log-group:myLogGroup:*",
        "arn:aws:logs:us-east-1:123456789012:log-group:myLogGroup"
      ]
}
```

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 つまたは複数のメトリクスフィルターを作成することで、CloudWatch Logs で受信す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は CloudWatch Logs に送信されたログデータを検索するための語句とパターンを定義します。CloudWatch Logs は、これらのメトリクスフィルタを使用して、ログデータを数値の CloudWatch メトリクスに変換し、グラフを作成したり、アラームを設定したりできます。

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。

サブスクリプションを持つロググループがログ変換を使用する場合、フィルターパターンは変換されたバージョンのログイベントに適用されます。詳細については、「取り込み中のログの変換」を参照してください。

> **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、「ログクラス」を参照してください。

これらのメトリクスを表示する、またはアラームを設定するときは、パーセンタイル統計など、任意のタイプの CloudWatch 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

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

各ログイベントのデータを CloudWatch Logs がどのように解釈するかについての記号による説明です。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

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

メトリクス名前空間

新しい CloudWatch メトリクスの送信先名前空間です。

メトリクス値

一致するログが見つかるたびにメトリクスに発行する数値。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
> 一致するログが見つかるたびに、指定された数値がメトリックスフィルターに追加されるという点において、メトリックスフィルターは CloudWatch Logs Insights クエリと異なります。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 の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、CloudWatch Logs Insights 言語クエリ構文 を参照してください。
> [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

メトリックスフィルターを使用すると、CloudWatch ログに入力されるログデータの検索とフィルタリング、フィルタリングされたログデータからメトリックス観測データの抽出、データポイントを CloudWatch ログメトリックスに変換できます。ユーザーは、CloudWatch ログに送信されるログデータを検索するための語句とパターンを定義します。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。CloudWatch がサポートする単位の詳細なリストについては、「Amazon CloudWatch API リファレンス」の「MetricDatum」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と、メトリクスフィルターで一致するものが見つからない場合、CloudWatch 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、CloudWatch がデータをより正確にレポートし、CloudWatch がむらがあるメトリクスを集計するのを防ぐことができます。CloudWatch は、1 分ごとにメトリクス値を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、CloudWatch はメトリクスのデフォルト値をレポートします。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 CLI AWS 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「Dimensions」
- 「Amazon CloudWatch Logs ユーザーガイドhttps://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/ExtractBytesExample.html」の「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」

> **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、「予想AWS 請求額をモニタリングするための請求アラームの作成」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. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。

2. ナビゲーションペインで、[ログ]、[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手順については、「Amazon CloudWatch Logs ユーザーガイド」の「CloudWatch Logs にロググループを作成する」を参照してください。
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、CloudWatch がデータをより正確にレポートし、CloudWatch がむらがあるメトリクスを集計するのを防ぐことができます。CloudWatch は、1 分ごとにメトリクス値を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を作成するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[ログ]、[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. [Metric details (メトリクスの詳細)] の [Metric namespace (メトリクス名前空間)] で、メトリクスを発行する 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、MyAppAccessCount というメトリクスフィルタをロググループ MyApp/access.log に適用して、CloudWatch の名前空間 MyNamespace にメトリクス EventCount を生成します。フィルタは、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を作成するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[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 MyAppAccessEventCount に発行されたデータポイントを参照する必要があります。

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

メトリクスフィルターを作成した後、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を作成するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[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、「MyNameSpace」と入力します。
9. [メトリクス名] に「ErrorCount」と入力します。
10. [メトリクス値] 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 を使用して、Apache サーバーが HTTP 404 レスポンス (ページが見つからない場合のレスポンスコード) を返した数をモニタリングできます。サイトの訪問者が目的のリソースを見つけられなかった頻度を理解するためにモニタリングする場合があります。ログレコードが各ログイベント (サイト訪問) に関する次の情報を含むように設定されている前提です。

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

例は次のようになります。

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

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

CloudWatch コンソールを使用してメトリクスフィルタを作成するには

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

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

例: HTTP 4xx コードをカウントする

前の例と同じように、ウェブサービスアクセスログをモニタリングしたり HTTP 応答コードレベルをモニタリングする場合があります。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を作成するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[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を作成するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[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を一覧表示するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[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を削除するには

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[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 Logs からログイベントのリアルタイムフィードにアクセスし、カスタム処理、分析、他のシステムへのロードを行うために、Amazon Kinesis ストリーム、Amazon Data Firehose ストリーム、 AWS Lambda などの他のサービスに配信することができます。ログイベント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。ログイベントは、取り込み後すぐに、通常は 3 分未満で受信リソースに送信されます。

> **Note**
>
> サブスクリプションを持つロググループがログ変換を使用する場合、フィルターパターンは変換されたバージョンのログイベントと比較されます。詳細については、「取り込み中のログの変換」を参照してください。

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。各アカウントは、リージョンごとに 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を持つことができます。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**
>
> 送信先のサービスが、スロットリングの例外や再試行可能なサービス例外 (HTTP 5xx など) の再試行可能なエラーを返した場合、CloudWatch Logs は最大 24 時間配信を再試行し続けます。AccessDeniedException や ResourceNotFoundException などの再試行不可能なエラーの場合、CloudWatch Logs は再配信を試みません。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 Logs 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。

また、CloudWatch Logs は、サブスクリプションへのログイベントの転送に関する CloudWatch メトリクスを作成します。詳細については、「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、Amazon OpenSearch Service クラスターにほぼリアルタイムでログデータを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 Logs 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、「ログクラス」を参照してください。

> **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、送信先への呼び出しの量を減らすことができます。バッチ処理は必ず行われるわけではありませんが、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

CloudWatch Logs が各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、送信先 AWS リソースに配信される内容を制限するフィルタリング式。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、Firehose ストリーム、または Lambda 関数の Amazon リソースネーム (ARN)。

role arn

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

ディストリビューション

送信先にログデータを配信するのに使用する方法。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基準。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、無限のログループを防ぐために使用します。無限ロググループ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。

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

Amazon Kinesis Data Streams、、Amazon Data Firehose AWS Lambda、または Amazon OpenSearch Service でサブスクリプションフィルターを使用できます。サブスクリプションフィ

ルターを介して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。このセクションでは、Firehose、Lambda、Kinesis Data Streams にログデータを送信する CloudWatch Logs サブスクリプションフィルターを作成するための例を示します。

> **Note**
>
> ログデータを検索する場合は、「Filter and pattern syntax」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 4: Amazon OpenSearch Service 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 AWS CloudTrail イベントを含むロググループに関連付けます。サブスクリプションフィルターは、ログが記録された「Root」 AWS 認証情報のすべてのアクティビティを、「RootAccess」という Kinesis データストリームに配信します。CloudWatch Logs 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CloudTrail イベントを CloudWatch Logs に送信する」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 put-subscription-filter を使用してサブスクリプションフィルターを作成するときは、`distribution` に `random` 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たストリームのモニタリング 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。たとえば、DeliveryThrottling メトリクスを使用してデータをサブスクリプション送信先に転送するときに CloudWatch Logs がスロットルされたログイベントの数を追跡できます。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。
- CloudWatch サブスクリプションのフィルターパターンを Kinesis Data Streams のストリームの容量に合わせて制限します。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 データストリーム describe-stream コマンドを使用して、StreamDescription.StreamStatus プロパティをチェックできます。さらに、後のステップで必要になるため StreamDescription.StreamARN 値を書き留めます。

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

出力例を次に示します。

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

```
                    "EndingHashKey": "340282366920938463463374607431768211455",
                    "StartingHashKey": "0"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber":
                    "49551135218688818456679503831981458784591352702181572610"
                }
            }
        ]
    }
}
```

3. 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-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. 権限ポリシーを作成し、CloudWatch Logs がアカウントで実行できるアクションを定義します。まず、ファイル (~/PermissionsForCWL-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 Logs サブスクリプションフィルタを作成できます。サブスクリプションフィルタにより、選択されたロググループから 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を設定したら、CloudWatch Logs によりフィルタパターンに一致するすべての受信ログイベントがストリームに転送されます。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)

元のログデータの AWS アカウント ID。

logGroup

発行元ログデータのロググループ名。

logStream

発行元ログデータのログストリーム名。

subscriptionFilters

発行元ログデータと一致したサブスクリプションフィルタ名のリスト。

messageType

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

logEvents

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

例 2: 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 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. CloudWatch Logs に、関数を実行するためのアクセス権限を付与します。次のコマンドを使用します。プレースホルダーは自身のアカウントに置き換え、プレースホルダーロググループは処理するロググループに置き換えます。

```
aws lambda add-permission \
    --function-name "helloworld" \
    --statement-id "helloworld" \
    --principal "logs.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)

元のログデータの AWS アカウント ID。

logGroup

発行元ログデータのロググループ名。

logStream

発行元ログデータのログストリーム名。

subscriptionFilters

発行元ログデータと一致したサブスクリプションフィルタ名のリスト。

messageType

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

logEvents

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

例 3: Amazon 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、CloudWatch Logs サブスクリプションを作成して、定義されたフィルターに一致する受信ログイベントを Amazon Data Firehose 配信ストリームに送信します。CloudWatch Logs から Amazon Data Firehose に送信されたデータは、既に 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 Logs を Firehose に送信する」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 Firehose ストリームを作成するように注意してください。ストリームがこのボリュームを処理できない場合、ログストリームはスロットリングされます。Firehose ストリームボリュームの制限事項の詳細については、「Amazon Data Firehose Data Limits」をご参照ください。

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 専用に作成したバケットを使用することをお勧めします。ただし、既存のバケットを使用する場合は、ステップ 2 に進みます。

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

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket2 --create-bucket-configuration
 LocationConstraint=region
```

出力例を次に示します。

```
{
    "Location": "/amzn-s3-demo-bucket2"
}
```

2. Amazon S3 バケットにデータを置くためのアクセス許可を 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 開発者ガイド」の「Controlling Access with Amazon Data Firehose」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、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:::amzn-s3-demo-bucket2",
          "arn:aws:s3:::amzn-s3-demo-bucket2/*" ]
    }
  ]
}
```

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

```
aws firehose create-delivery-stream \
   --delivery-stream-name 'my-delivery-stream' \
```

```
   --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::123456789012:role/FirehosetoS3Role", "BucketARN":
 "arn:aws:s3:::amzn-s3-demo-bucket2"}'
```

Firehose は、配信された Amazon S3 オブジェクトに対して、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 プロパティをチェックできます。さらに、後のステップで必要になるため、DeliveryStreamDescription.DeliveryStreamARN 値を書き留めます。

```
aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
{
    "DeliveryStreamDescription": {
        "HasMoreDestinations": false,
        "VersionId": "1",
        "CreateTimestamp": 1446075815.822,
        "DeliveryStreamARN": "arn:aws:firehose:us-
east-1:123456789012:deliverystream/my-delivery-stream",
        "DeliveryStreamStatus": "ACTIVE",
        "DeliveryStreamName": "my-delivery-stream",
        "Destinations": [
            {
                "DestinationId": "destinationId-000000000001",
                "S3DestinationDescription": {
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "RoleARN": "delivery-stream-role",
                    "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket2",
                    "BufferingHints": {
                        "IntervalInSeconds": 300,
                        "SizeInMBs": 5
                    }
                }
            }
        ]
```

```
    }
}
```

8. Firehose 配信ストリームにデータを置くためのアクセス許可を 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. 権限ポリシーを作成し、CloudWatch Logs がアカウントで実行できるアクションを定義します。まず、テキストエディタを使用して権限ポリシーファイル (例: ~/PermissionsForCWL.json) を作成します。

```
{
    "Statement":[
      {
        "Effect":"Allow",
        "Action":["firehose:PutRecord"],
        "Resource":[
            "arn:aws:firehose:region:account-id:deliverystream/delivery-stream-name"]
      }
    ]
}
```

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

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

12. Amazon 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 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により、選択され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がすぐに開始されます。

```
aws logs put-subscription-filter \
    --log-group-name "CloudTrail" \
    --filter-name "Destination" \
```

```
    --filter-pattern "{$.userIdentity.type = Root}" \
    --destination-arn "arn:aws:firehose:region:123456789012:deliverystream/my-
delivery-stream" \
    --role-arn "arn:aws:iam::123456789012:role/CWLtoKinesisFirehoseRole"
```

13. サブスクリプションフィルターを設定したら、CloudWatch Logs からフィルタパターンに一致するすべての受信ログイベントが Amazon Data Firehose 配信ストリームに転送されます。Amazon Data Firehose 配信ストリームに設定された時間バッファ間隔に基づいて、Amazon S3 にデータが表示されるようになります。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 'amzn-s3-demo-bucket2' --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 'amzn-s3-demo-bucket2' --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
```

例 4: Amazon OpenSearch Service を使用したサブスクリプションフィルター

この例では、定義されたフィルターに一致する受信ログイベントを OpenSearch Service ドメインに送信する CloudWatch Logs サブスクリプションを作成します。

OpenSearch Service のサブスクリプションフィルターを作成するには

1. OpenSearch Service ドメインを作成する。詳細については、OpenSearch Service ドメインの作成」を参照してください。
2. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
3. 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。
4. ロググループの名前を選択します。
5. [Actions] (アクション)、[Subscription filters] (サブスクリプションフィルター)、[Create Amazon OpenSearch Service subscription filter] (Amazon OpenSearch Service サブスクリプションフィルターを作成) の順に選択します。
6. 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、ステップ 1 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、そのドメインの ARN とエンドポイントを入力します。

7. 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。
8. Lambda IAM 実行ロールで、OpenSearch Service 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。選択した IAM ロールは、次の要件を満たしている必要があります。

   - 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 Service ドメインが VPC アカウントを使用している場合、IAM ロールには AWSLambdaVPCAccessExecutionRole ポリシーもアタッチされている必要があります。Amazon が管理するこのポリシーにより、お客様の VPC へのアクセスが Lambda に許可され、Lambda は VPC の OpenSearch エンドポイントに書き込むことができます。
9. ログ形式を選択します。
10. サブスクリプションフィルターパターンには、ログイベントで検索する用語または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 Service クラスターに送信できます。詳細については、「メトリックスフィルターのフィルターパターン構文」を参照してください。
11. (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。
12. [Start streaming] (ストリーミングの開始) を選択します。

アカウントレベルのサブスクリプションフィルター

> **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、対処しなければ取り込みにかかる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。この問題および除外するロググループを判断する方法については、「ログの再帰防止」を参照してください。

アカウント内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、Amazon Kinesis Data Streams AWS Lambda、または Amazon Data Firehose と連携できます。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。このセクションでは、Kinesis Data Streams、Lambda、Firehose のアカウントレベルのサブスクリプションを作成するための例を示します。

> **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、`--policy-type` パラメータに `SUBSCRIPTION_FILTER_POLICY` の値を持つ `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制限に関する詳細は、Kinesis Data Streams ドキュメントの「Quotas and Limits」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宛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、必要に応じて構成を調整することができます。例えば、`DeliveryThrottling` メトリクスはデータをサブスクリプション送信先に転送するときに CloudWatch Logs がスロットリングされたログイベントの数を追跡できます。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、「On-demand mode」をご参照ください。
> - CloudWatch Logs サブスクリプションのフィルターパターンを Kinesis Data Streams のストリームの容量に合わせて制限します。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 `Test` と照合し、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。ストリームがアクティブ化どうか確認するには、describe-stream コマンドを使用して、StreamDescription.StreamStatus プロパティをチェックできます。

```
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": "EXAMPLE846346337460743176821145 5",
                    "StartingHashKey": "0"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber":
                    "EXAMPLE688818456679503831981458784591352702181572610"
                }
            }
        ]
    }
}
```

3. 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-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. 権限ポリシーを作成し、CloudWatch Logs がアカウントで実行できるアクションを定義します。まず、ファイル (~/PermissionsForCWL-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/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 Logs 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 ERROR を含むすべてのログイベントがストリーミングされます。ただし、LogGroupToExclude1 および LogGroupToExclude2 という名前のロググループ内のイベントは除外されます。

```
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したら、CloudWatch Logs によりフィルタパターンと選択基準に一致するすべての受信ログイベントがストリームに転送されます。

   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のログ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の重要なフィールドです。この問題および除外するロググループを判断する方法については、「ログの再帰防止」を参照してください。現在、NOT IN は selection-criteria でサポートされている唯一の演算子です。

   Kinesis Data Streams シャードイテレータと Kinesis Data Streams 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、ログイベントのフローを確認できます。

```
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"
```

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、このコマンドを数回使用する必要があります。

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、主に送信先が到達可能であるかどうかをチェックするために、"CONTROL_MESSAGE" 型の Kinesis データストリームレコードを発行することがあります。

owner (オーナー)

元のログデータの 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」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 `policyLevel` 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。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。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. CloudWatch Logs に、関数を実行するためのアクセス権限を付与します。次のコマンドを使用して、プレースホルダーアカウントを独自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独自のアカウントに置き換えます。この例では、文字列 ERROR を含むすべてのログイベントがストリーミングされます。ただし、LogGroupToExclude1 および LogGroupToExclude2 という名前のロググループ内のイベントは除外されます。

```
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したら、CloudWatch Logs によりフィルタパターンと選択基準に一致するすべての受信ログイベントがストリームに転送されます。

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のログ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の重要なフィールドです。この問題および除外するロググループを判断する方法については、「ログの再帰防止」を参照してください。現在、NOT IN は selection-criteria でサポートされている唯一の演算子です。

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、取り込み料金の増加につながる可能性のある無限のログ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、主に送信先が到達可能であるかどうかをチェックするために、"CONTROL_MESSAGE" 型の Kinesis データストリームレコードを発行することがあります。

owner (オーナー)

元のログデータの 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」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 `policyLevel` 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、CloudWatch Logs 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して、定義されたフィルタに一致する受信ログイベントを Amazon Kinesis Data Firehose 配信ストリームに送信します。CloudWatch Logs から Amazon Data Firehose に送信されたデータは、既に 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、「Writing to Kinesis Data Firehose Using CloudWatch Logs」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 Firehose ストリームを作成するように注意してください。ストリームがこのボリュームを処理できない場合、ログストリームはスロットリングされます。Firehose ストリームボリュームの制限事項の詳細については、「Amazon Data Firehose Data Limits」をご参照ください。

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 専用に作成したバケットを使用することをお勧めします。ただし、既存のバケットを使用する場合は、ステップ 2 に進みます。

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

   ```
   aws s3api create-bucket --bucket amzn-s3-demo-bucket2 --create-bucket-configuration
    LocationConstraint=region
   ```

   出力例を次に示します。

   ```
   {
       "Location": "/amzn-s3-demo-bucket2"
   }
   ```

2. Amazon S3 バケットにデータを置くためのアクセス許可を 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 開発者ガイド」の「Controlling Access with Amazon Data Firehose」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": "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、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:::amzn-s3-demo-bucket2",
            "arn:aws:s3:::amzn-s3-demo-bucket2/*" ]
      }
    ]
}
```

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

```
aws firehose create-delivery-stream \
   --delivery-stream-name 'my-delivery-stream' \
   --s3-destination-configuration \
  '{"RoleARN": "arn:aws:iam::123456789012:role/FirehosetoS3Role", "BucketARN":
 "arn:aws:s3:::amzn-s3-demo-bucket2"}'
```

NFirehose は、配信された Amazon S3 オブジェクトに対して、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 プロパティをチェックできます。さらに、後のステップで必要になるため、DeliveryStreamDescription.DeliveryStreamARN 値を書き留めます。

```
aws firehose describe-delivery-stream --delivery-stream-name "my-delivery-stream"
{
    "DeliveryStreamDescription": {
        "HasMoreDestinations": false,
        "VersionId": "1",
        "CreateTimestamp": 1446075815.822,
        "DeliveryStreamARN": "arn:aws:firehose:us-
east-1:123456789012:deliverystream/my-delivery-stream",
```

```
        "DeliveryStreamStatus": "ACTIVE",
        "DeliveryStreamName": "my-delivery-stream",
        "Destinations": [
            {
                "DestinationId": "destinationId-000000000001",
                "S3DestinationDescription": {
                    "CompressionFormat": "UNCOMPRESSED",
                    "EncryptionConfiguration": {
                        "NoEncryptionConfig": "NoEncryption"
                    },
                    "RoleARN": "delivery-stream-role",
                    "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket2",
                    "BufferingHints": {
                        "IntervalInSeconds": 300,
                        "SizeInMBs": 5
                    }
                }
            }
        ]
    }
}
```

8. Firehose 配信ストリームにデータを置くためのアクセス許可を 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. 権限ポリシーを作成し、CloudWatch Logs がアカウントで実行できるアクションを定義します。まず、テキストエディタを使用して権限ポリシーファイル (例: ~/PermissionsForCWL.json) を作成します。

```
{
    "Statement":[
      {
        "Effect":"Allow",
        "Action":["firehose:PutRecord"],
        "Resource":[
            "arn:aws:firehose:region:account-id:deliverystream/delivery-stream-
name"]
```

```
      }
    ]
}
```

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

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

12. Amazon 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 のアカウントレベルのサブスクリプションフィルタポリシーを作成できます。ポリシーにより、選択され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がすぐに開始されます。

```
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したら、CloudWatch Logs から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が Amazon Data Firehose 配信ストリームに転送されます。

    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のログ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の重要なフィールドです。この問題および除外するロググループを判断する方法については、「ログの再帰防止」を参照してください。現在、NOT IN は selection-criteria でサポートされている唯一の演算子です。

    Amazon Data Firehose 配信ストリームに設定された時間バッファ間隔に基づいて、Amazon S3 にデータが表示されるようになります。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 'amzn-s3-demo-bucket2' --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 'amzn-s3-demo-bucket2' --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、クロスリージョンのサブスクリプション

別の AWS アカウントの所有者とコラボレーションし、Amazon Kinesis や Amazon Data Firehose ストリーム (これはクロスアカウントデータ共有と呼ばれます) などの AWS リソースでログイベントを受信できます。例えば、このログイベントデータを集中管理型の Kinesis Data Streams または Firehose ストリームから読み取り、カスタム処理や分析を実行することができます。カスタム処理は、多数のアカウントが協力しデータを分析する場合に特に便利です。

たとえば、ある会社の情報セキュリティグループがリアルタイムで侵入または異常な挙動を検出するためにデータを分析するとします。この場合、会社の全部署のアカウントのフェデレーションされた本稼働ログを中央処理のために収集することによって、これらのアカウントの監査を行うことができます。これらすべてのアカウントのイベントデータのリアルタイムストリームは、アセンブルした後に、Kinesis 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。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

- Kinesis Data Streams を使用したクロスアカウント、クロスリージョンのログデータ共有
- Firehose を使用したクロスアカウント、クロスリージョンのログデータ共有
- 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
- 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

Kinesis Data Streams を使用したクロスアカウント、クロスリージョンのログデータ共有

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。

- [Log data sender] (ログデータの送信者) — 受信者から送信先情報を取得し、そのログイベントを特定の送信先に送信する準備が完了していることを CloudWatch Logs に通知します。このセクションの残りの手順では、ログデータ送信者は架空の AWS アカウント番号 111111111111「」と表示されます。

  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。
- [ログデータの受信者] — Kinesis データストリームストリーミングをカプセル化する送信先を設定し、受取人がログデータの受け取りを希望していることを CloudWatch Logs に通知します。この後、受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「」と表示されます。

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

送信先名

作成する送信先の名前。

ターゲット ARN

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

ロールの ARN

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 Logs に付与する AWS Identity and Access Management (IAM) ロール。

アクセスポリシー

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

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

送信先が作成されると、CloudWatch Logs は受信者アカウントに代わってテストメッセージを宛先に送信します。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宛先に送信します。

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 プロパティをチェックできます。さらに、StreamDescription.StreamARN の値は、後で CloudWatch Logs に渡すので書き留めておいてください。

```
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置くアクセス権限を CloudWatch Logs に付与する IAM ロールを作成します。まず、ファイル ~/TrustPolicyForCWL.json で信頼ポリシーを作成する必要があります。このポリシーの作成にはテキストエディタを使用します。IAM コンソールは使用しないでください。

   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 値を書き留めておきます。これも後で CloudWatch Logs に渡されるからです。

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

{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "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. CloudWatch Logs がアカウントで実行できるアクションを定義するアクセス許可ポリシーを作成します。まず、テキストエディタを使用してファイル ~/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 ロールが作成されたら、CloudWatch Logs の送信先を作成できます。

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

```
aws logs put-destination \
    --destination-name "testDestination" \
    --target-arn "arn:aws:kinesis:region:999999999999:stream/RecipientStream" \
    --role-arn "arn:aws:iam::999999999999:role/CWLtoKinesisRole"

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

   b. ステップ 7a が完了したら、ログデータの受取人アカウントで、アクセスポリシーを送信先に関連付けます。このポリシーでは、送信者アカウントが送信先にアクセスするためのアクセス許可が付与され、logs:PutSubscriptionFilter アクションが指定されている必要があります。

      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 が 111111111111 AWS アカウントのユーザーは、ARN arn:aws:logs:*region*:999999999999:destination:testDestination を使用して送信先に対して PutSubscriptionFilter を呼び出すことができます。他のユーザーがこの送信先に対して PutSubscriptionFilter を呼び出そうとしても、それは却下されます。

アクセスポリシーに照らし合わせてユーザーの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、クロスアカウントのアクセス許可 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により、CloudWatch が想定する IAM ロールを作成し、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権限を持っているかどうかを検証できます。

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、CloudWatch Logs がアカウントで実行できるアクションを定義します。

   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 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 ID ベースのポリシーを、送信アカウントに保持する必要があります。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 *UserName* で表される値を書き留めておいてください。

2. 送信アカウント 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 AWS アカウント、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、サブスクリプションフィルターを送信アカウントで作成します。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、以前に作成した送信先に配信されます。この送信先は、「RecipientStream」という ストリームをカプセル化します。

以降のセクションの残りのステップは、「AWS CloudTrail ユーザーガイド」の「CloudWatch Logs への CloudTrail 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が作成済みであ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を作成したら、CloudWatch Logs が、フィルタパターンと一致するすべての受信ログイベントを、送信先ストリーム内でカプセル化されている「RecipientStream」という名前のストリームに転送します。送信先の所有者は、aws kinesis get-shard-iterator コマンドを使用して Kinesis データストリームシャードを取得し、aws kinesis get-records コマンドを使用していくつかの Kinesis データストリームレコードをフェッチすることにより、これが実際に行われていることを確認できます。

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

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

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

> **Note**
>
> 場合によっては、Kinesis 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)

元のログデータの AWS アカウント ID。

logGroup

発行元ログデータのロググループ名。

logStream

発行元ログデータのログストリーム名。

subscriptionFilters

発行元ログデータと一致したサブスクリプションフィルタ名のリスト。

messageType

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

logEvents

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

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 testDestination という送信先に関連付けられているポリシーをフェッチし、AccessPolicy を書き留めておきます。

```
aws logs describe-destinations \
    --destination-name-prefix "testDestination"

{
 "Destinations": [
   {
```

```
    "DestinationName": "testDestination",
    "RoleArn": "arn:aws:iam::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 ファイルに配置します。

```
{
  "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ているポリシーを送信先に関連付けます。

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

これにより、最終的には、アカウント ID 111111111111 からのログイベントが無効になります。アカウント ID 222222222222 の所有者が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。この更新により IAM ロールが追加されます。IAM ロールは CloudWatch が引き受けることができ、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権限を持っていることを検証できます。

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、CloudWatch Logs がアカウントで実行できるアクションを定義します。

   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**
>
> にリストされている AWS 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、クロスリージョンのログデータ共有

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。

- [Log data sender (ログデータの送信者)] — 受取人から送信先情報を取得し、そのログイベントを特定の送信先に送信する準備が完了していることを CloudWatch Logs に通知します。このセクションの残りの手順では、ログデータ送信者は架空の AWS アカウント番号 111111111111「」と表示されます。
- [ログデータの受信者] — Kinesis データストリームストリーミングをカプセル化する送信先を設定し、受取人がログデータの受け取りを希望していることを CloudWatch Logs に通知します。この後、受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「」と表示されます。

このセクションの例では、Amazon S3 ストレージで Firehose 配信ストリームを使用しています。異なる設定で 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、「Creating a Firehose Delivery Stream」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**
>
> [same account] と [cross-Region] 配信ストリームに対して 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

- ログイベントの送信の検証
- 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**
>
> 以下の手順を実行する前に、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 開発者ガイド」の「Controlling Access」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 amzn-s3-demo-bucket --create-bucket-configuration
 LocationConstraint=us-east-1
```

2. バケットにデータを配置するためのアクセス許可を Firehose に付与する 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、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:::amzn-s3-demo-bucket",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ]
    }]
}
```

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* および *amzn-s3-demo-bucket2-arn* をデプロイに適した値に置き換えます。

```
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:::amzn-s3-demo-bucket"}'
```

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テストメッセージを宛先に送信します。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宛先に送信します。

送信先を作成するには

1. ステップ 1: Firehose 配信ストリームを作成する で作成した Firehose ストリームがアクティブ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、後の手順で使用する必要があるため、DeliveryStreamDescription.DeliveryStreamARN 値を書き留めます。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:::amzn-s3-demo-bucket",
                    "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:::amzn-s3-demo-bucket",
                    "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ためのアクセス許可を 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 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。CloudWatch Logs エンドポイントの詳細については、Amazon CloudWatch Logs エンドポイントおよび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. CloudWatch Logs がアカウントで実行できるアクションを定義するアクセス許可ポリシーを作成します。まず、テキストエディタを使用してファイル ~/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 ロールが作成されたら、CloudWatch Logs の送信先を作成できます。

   a. このステップでは、アクセスポリシーと送信先は関連付けられません。送信先の作成を完了するには 2 つのステップを行う必要がありますが、このステップはその最初のステップです。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",
      "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 アクションが指定されている必要があります。クロスアカウントのユーザーは、PutSubscriptionFilter アクションを使用して送信先にログイベントを送信します。

```
aws logs put-destination-policy \
    --destination-name "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 ID ベースのポリシーを、送信アカウントに保持する必要があります。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 *UserName* で表される値を書き留めておいてください。

2. 送信アカウント 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 AWS アカウント、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、以前に作成した宛先に配信されます。CloudWatch Logs 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CloudTrail イベントを CloudWatch Logs に送信する」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。データは、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、Amazon S3 バケットに順次表示されます。十分な時間が経過すると、Amazon S3 バケットをチェックしてデータを確認できます。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 'amzn-s3-demo-bucket'
```

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 'amzn-s3-demo-bucket' --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が関連付けられている送信先に対して PutDestinationPolicy アクションを使用できます。次の例では、先ほど追加したアカウント 111111111111 がログデータの送信を停止し、アカウント 333333333333 が有効になります。

1. 現在 testDestination という送信先に関連付けられているポリシーをフェッチし、AccessPolicy を書き留めておきます。

```
aws logs describe-destinations \
    --destination-name-prefix "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 ファイルに配置します。

```
{
  "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に関連付けます。

```
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。

- [Log data sender] (ログデータの送信者) — 受信者から送信先情報を取得し、そのログイベントを特定の送信先に送信する準備が完了していることを CloudWatch Logs に通知します。このセクションの残りの手順では、ログデータ送信者は架空の AWS アカウント番号 111111111111「」と表示されます。

  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。
- [ログデータの受信者] — Kinesis データストリームストリーミングをカプセル化する送信先を設定し、受取人がログデータの受け取りを希望していることを CloudWatch Logs に通知します。この後、受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「」と表示されます。

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

送信先名

作成する送信先の名前。

ターゲット ARN

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

ロールの ARN

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 Logs に付与する AWS Identity and Access Management (IAM) ロール。

アクセスポリシー

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

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

送信先が作成されると、CloudWatch Logs は受信者アカウントに代わってテストメッセージを宛先に送信します。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宛先に送信します。

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 プロパティをチェックできます。さらに、StreamDescription.StreamARN の値は、後で CloudWatch Logs に渡すので書き留めておいてください。

```
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置くアクセス権限を CloudWatch Logs に付与する IAM ロールを作成します。まず、ファイル ~/TrustPolicyForCWL.json で信頼ポリシーを作成する必要があります。このポリシーの作成にはテキストエディタを使用します。IAM コンソールは使用しないでください。

   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 値を書き留めておきます。これも後で CloudWatch Logs に渡されるからです。

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

{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Statement": {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "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. CloudWatch Logs がアカウントで実行できるアクションを定義するアクセス許可ポリシーを作成します。まず、テキストエディタを使用してファイル ~/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 ロールが作成されたら、CloudWatch Logs の送信先を作成できます。

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

```
aws logs put-destination \
    --destination-name "testDestination" \
    --target-arn "arn:aws:kinesis:region:999999999999:stream/RecipientStream" \
    --role-arn "arn:aws:iam::999999999999:role/CWLtoKinesisRole"

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

   b. ステップ 7a が完了したら、ログデータの受取人アカウントで、アクセスポリシーを送信先に関連付けます。このポリシーでは、送信者アカウントが送信先にアクセスするためのアクセス許可が付与され、logs:PutSubscriptionFilter アクションが指定されている必要があります。

      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 が 111111111111 AWS アカウントのユーザーは、ARN arn:aws:logs:*region*:999999999999:destination:testDestination を使用して送信先に対して PutSubscriptionFilter を呼び出すことができます。他のユーザーがこの送信先に対して PutSubscriptionFilter を呼び出そうとしても、それは却下されます。

アクセスポリシーに照らし合わせてユーザーの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により、CloudWatch が想定する IAM ロールを作成し、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権限を持っているかどうかを検証できます。

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、CloudWatch Logs がアカウントで実行できるアクションを定義します。

   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が送信アカウントで作成されます。フィルターは送信者アカウント 111111111111 に関連付けられ、フィルターと選択基準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。この送信先は、「RecipientStream」という ストリームをカプセル化します。

`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のログ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の重要なフィールドです。この問題および除外するロググループを判断する方法については、「ログの再帰防止」を参照してください。現在、NOT IN は `selection-criteria` でサポートされている唯一の演算子です。

```
aws logs put-account-policy \
```

```
    --policy-name "CrossAccountStreamsExamplePolicy" \
    --policy-type "SUBSCRIPTION_FILTER_POLICY" \
    --policy-document
 '{"DestinationArn":"arn:aws:logs:region:999999999999:destination:testDestination",
 "FilterPattern": "", "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したら、CloudWatch Logs が、フィルタパターンおよび選択基準と一致するすべての受信ログイベントを、送信先ストリーム内でカプセル化されている「RecipientStream」という名前のストリームに転送します。送信先の所有者は、aws kinesis get-shard-iterator コマンドを使用して Kinesis データストリームシャードを取得し、aws kinesis get-records コマンドを使用していくつかの Kinesis データストリームレコードをフェッチすることにより、これが実際に行われていることを確認できます。

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

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

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

> **Note**
>
> 場合によっては、Kinesis 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
\"}"
```

```
        }
    ]
}
```

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、主に送信先が到達可能であるかどうかをチェックするために、"CONTROL_MESSAGE" 型の Kinesis データストリームレコードを発行することがあります。

owner (オーナー)

元のログデータの 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」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 `policyLevel` 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 testDestination という送信先に関連付けられているポリシーをフェッチし、AccessPolicy を書き留めておきます。

```
aws logs describe-destinations \
    --destination-name-prefix "testDestination"

{
 "Destinations": [
   {
     "DestinationName": "testDestination",
     "RoleArn": "arn:aws:iam::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 ファイルに配置します。

```
{
  "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ているポリシーを送信先に関連付けます。

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

これにより、最終的には、アカウント ID 111111111111 からのログイベントが無効になります。アカウント ID 222222222222 の所有者が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。この更新により IAM ロールが追加されます。IAM ロールは CloudWatch が引き受けることができ、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権限を持っていることを検証できます。

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、CloudWatch Logs がアカウントで実行できるアクションを定義します。

   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 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。

- [Log data sender (ログデータの送信者)] — 受取人から送信先情報を取得し、そのログイベントを特定の送信先に送信する準備が完了していることを CloudWatch Logs に通知します。このセクションの残りの手順では、ログデータ送信者は架空の AWS アカウント番号 111111111111「」と表示されます。
- [ログデータの受信者] — Kinesis データストリームストリーミングをカプセル化する送信先を設定し、受取人がログデータの受け取りを希望していることを CloudWatch Logs に通知します。この後、受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「」と表示されます。

このセクションの例では、Amazon S3 ストレージで Firehose 配信ストリームを使用しています。異なる設定で 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、「Creating a Firehose Delivery Stream」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**
>
> [same account] と [cross-Region] 配信ストリームに対して 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**
>
> 以下の手順を実行する前に、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 開発者ガイド」の「Controlling Access」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 amzn-s3-demo-bucket --create-bucket-configuration
 LocationConstraint=us-east-1
```

2. バケットにデータを配置するためのアクセス許可を Firehose に付与する 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、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:::amzn-s3-demo-bucket",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ]
    }]
}
```

   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* および *amzn-s3-demo-bucket2-arn* をデプロイに適した値に置き換えます。

```
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:::amzn-s3-demo-bucket"}'
```

   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テストメッセージを宛先に送信します。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宛先に送信します。

送信先を作成するには

1. ステップ 1: Firehose 配信ストリームを作成する で作成した Firehose ストリームがアクティブ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、後の手順で使用する必要があるため、DeliveryStreamDescription.DeliveryStreamARN 値を書き留めます。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:::amzn-s3-demo-bucket",
                    "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:::amzn-s3-demo-bucket",
                    "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ためのアクセス許可を 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 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。CloudWatch Logs エンドポイントの詳細については、Amazon CloudWatch Logs エンドポイントおよび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. CloudWatch Logs がアカウントで実行できるアクションを定義するアクセス許可ポリシーを作成します。まず、テキストエディタを使用してファイル ~/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 ロールが作成されたら、CloudWatch Logs の送信先を作成できます。

   a. このステップでは、アクセスポリシーと送信先は関連付けられません。送信先の作成を完了するには 2 つのステップを行う必要がありますが、このステップはその最初のステップです。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。クロスアカウントのユーザーは、PutSubscriptionFilter および PutAccountPolicy アクションを使用して送信先にログイベントを送信します。

```
aws logs put-destination-policy \
    --destination-name "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、送信側のアカウントに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 つを除くすべてのロググループに含まれる文字列 ERROR 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 から、フィルタパターンと選択基準に一致するすべての受信ログイベントが Kinesis Data Firehose 配信ストリームに転送されます。データは、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、Amazon S3 バケットに順次表示されます。十分な時間が経過すると、Amazon S3 バケットをチェックしてデータを確認できます。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 'amzn-s3-demo-bucket'
```

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 'amzn-s3-demo-bucket' --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が関連付けられている送信先に対して 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 testDestination という送信先に関連付けられているポリシーをフェッチし、AccessPolicy を書き留めておきます。

   ```
   aws logs describe-destinations \
       --destination-name-prefix "testFirehoseDestination"
   ```

   返されたデータは、次のように表示される場合があります。

   ```
   {
       "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 ファイルに配置します。

```
{
  "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に関連付けます。

```
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許可を持たないエンティティが、より特権のあるエンティティにアクションを実行するように強制できるセキュリティの問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。これを防ぐために、 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、aws:SourceOrgID、aws:SourceOrgPaths のグローバル条件コンテキストキーを使用して、別のサービスに付与する許可をその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 「エンティティパスを理解する」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*:*。

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。

ステップ 1: 送信先を作成する と ステップ 2: 送信先を作成する に記載されている、Kinesis データストリームと Firehose にデータを書き込むためのアクセス許可を CloudWatch Logs に付与するポリシーでは、 aws:SourceArn グローバル条件コンテキストキーを使用して、混乱した代理の問題を防止する方法を示しています。

ログの再帰防止

サブスクリプションフィルターの使用によって無限のログ再帰が引き起こされるリスクがあります。これを防止しなければ CloudWatch Logs 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宛先に配信され、ロググループがより多くのログを取り込むようになり、その後再び宛先に転送され、再帰ループが作成されます。

例えば、Amazon S3 にログイベントを配信する送信先を Firehose とするサブスクリプションフィルターがあるとします。さらに、Amazon S3 に配信された新しいイベントを処理し、一部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は送信先で Firehose と Amazon S3 に転送され、その後再び関数が呼び出されます。これにより、より多くのログが生成されて Firehose と Amazon S3 に転送され、関数が次々と呼び出されます。その結果、無限ループが発生し、ログの取り込み、Firehose、Amazon S3 で予期しない利用料金の増加につながります。

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では、`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、ログを生成し、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る以下の AWS サービスを検討してください。

- Amazon EC2 と Fargate
- Lambda
- AWS Step Functions
- CloudWatch Logs で有効になっている Amazon VPC 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 関数には転送されません。この場合、アカウントサブスクリプションポリシーでは、`selectionCriteria` を使用する送信先 Lambda 関数のロググループを除外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 の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、CloudWatch Logs Insights 言語クエリ構文 を参照してください。

CloudWatch Logs では、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、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、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、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、ログイベント、Live Tail 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。
- ?: 前の語句のインスタンスが 0 以上一致。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」。例 :
  - %gra|ey% は「グレー」または「グレー」と一致できます
  - %^starting|^initializing|^shutting down% は「開始中...」、「初期化中...」、または「シャットダウン中」と一致できますが、「スキップ初期化中...」とは一致しません。
  - %abcc|ab[^c]$ は「abcc ...」と「aba ...」と一致できますが、「aac ...」と一致しません
- \: 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。

- *: 前の語句のインスタンスが 0 以上一致。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。
- +: 前の語句のインスタンスが 1 以上一致。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、次のような口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

次のコードス二ペットは、メッセージに ERROR と ARGUMENTS という単語が含まれるすべての口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、次のような口グイベントメッセージを返します。

- [ERROR 419] MISSING ARGUMENTS
- [ERROR 420] INVALID ARGUMENTS

次の口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口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。次のコードス二ペットは、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口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、包含語や除外語などの他のフィルターパターンと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、すべての疑問符用語は無視されます。
> たとえば、次のフィルターパターンは という単語を含むすべてのイベントに一致しますがREQUEST、疑問符 ("?") フィルター用語は無視され、効果はありません。
>
> ```
> ?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 CloudWatch Logs にのみログを発行しますが、一部の AWS サービスは Amazon Simple Storage Service または Amazon Data Firehose に直接ログを発行できます。ログの主な要件がストレージまたはこのいずれかのサービスでの処理である場合、追加の設定を行わずに、サービスで簡単にログを作成し、直接 Amazon S3 または Firehose に配信することができます。

ログが Amazon S3 または Firehose に直接公開される場合でも、料金が適用されます。詳細については、[Logs (ログ)] タブの Amazon CloudWatch 料金表で提供されたログを参照してください。

一部の AWS サービスは、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。

ログソース	ログタイプ	CloudWatch Logs	Amazon S3	Firehose
Amazon API Gateway アクセスログ	提供されるログ	サポートあり [V1 アクセス許可]		
AWS AppSync ログ	カスタムログ	サポート		
Amazon Aurora MySQL ログ	カスタムログ	サポート		
Amazon Bedrock ナレッジベースのログ記録	提供されるログ	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]

ログソース	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	提供されるログ	サポートあり [V1 アクセス許可]		
CloudFront: アクセスログ	提供されるログ	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]
AWS CloudHSM 監査ログ	カスタムログ	サポート		
CloudWatch Evidently 評価イベントログ	提供されるログ	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 インターネット モニタリング ログ	提供されるログ		サポートあり [V1 アクセス許可]	
CloudTrail のログ	カスタムログ	サポート		
AWS CodeBuild ログ	カスタムログ	サポート		
Amazon CodeWhisperer イベントログ	提供されるログ	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	提供されるログ	サポートあり [V1 アクセス許可]		
Amazon Connect のログ	カスタムログ	サポート		

ログソース	ログタイプ	CloudWatch Logs	Amazon S3	Firehose
AWS DataSync ログ	カスタムログ	サポート		
Amazon ElastiCache (Redis OSS) ログ	提供されるログ	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	カスタムログ	サポート		
Amazon Elastic Container Service のログ	カスタムログ	サポート		
Amazon Elastic Kubernetes Service コントロールプレーンのログ	提供されるログ	サポート		
AWS Elemental MediaPackage アクセスログ	提供されるログ	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]
AWS Elemental MediaTailor ログ	提供されるログ	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 EventBridge パイプのログ記録	提供されるログ	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 Fargate ログ	カスタムログ	サポート		
AWS Fault Injection Service 実験ログ	提供されるログ		サポートあり [V1 アクセス許可]	

ログソース	ログタイプ	CloudWatch Logs	Amazon S3	Firehose
Amazon FinSpace	提供されるログ	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 Global Accelerator フローログ	提供されるログ		サポートあり [V1 アクセス許可]	
AWS Glue ジョブログ	カスタムログ	サポート		
IAM アイデンティティセンターエラーログ	提供されるログ	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 Interactive Video Service チャット ログ	提供されるログ	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	カスタムログ	サポート		
AWS IoT FleetWise ログ	提供されるログ	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	提供されるログ	サポート	サポート対象	サポート
Amazon Macie のログ	カスタムログ	サポート		
Amazon SES メールマネージャーのログ記録	提供されるログ	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]

ログソース	ログタイプ	CloudWatch Logs	Amazon S3	Firehose
AWS Mainframe Modernization	提供されるログ	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 Managed Service for Prometheus のログ	提供されるログ	サポートあり [V1 アクセス許可]		
Amazon MSK ブローカーログ	提供されるログ	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	提供されるログ	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	カスタムログ	サポート		
AWS Network Firewall ログ	提供されるログ	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]
Network Load Balancer アクセスログ	提供されるログ		サポートあり [V1 アクセス許可]	
OpenSearch のログ	カスタムログ	サポート		
Amazon OpenSearch Service インジェスト ログ	提供されるログ	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	カスタムログ	サポート		

ログソース	ログタイプ	CloudWatch Logs	Amazon S3	Firehose
Amazon Relational Database ServicePostgreSQL のログ	カスタムログ	サポート		
Amazon Q Business の会話ログ	提供されるログ	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]
AWS RoboMaker ログ	カスタムログ	サポート		
Amazon Route 53 パブリック DNS クエリログ	提供されるログ	サポート		
Amazon Route 53 Resolver クエリログ	提供されるログ	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 AI イベント	提供されるログ	サポートあり [V1 アクセス許可]		
Amazon SageMaker AI ワーカーイベント	提供されるログ	サポートあり [V1 アクセス許可]		
AWS Site-to_Site VPN ログ	提供されるログ	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 Email Service ログ	提供されるログ	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 Simple Notification Service のログ	カスタムログ	サポート		

ログソース	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	カスタムログ	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	提供されるログ		サポートあり [V1 アクセス許可]	
AWS Step Functions Express ワークフローログと標準ワークフローログ	提供されるログ	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	提供されるログ	サポートあり [V1 アクセス許可]		
AWS Transfer Family ログ	提供されるログ	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	提供されるログ	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	提供されるログ	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	提供されるログ	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	提供されるログ	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

ログソース	ログタイプ	CloudWatch Logs	Amazon S3	Firehose
Amazon WorkMail 監査ログ	提供されるログ	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 サービスは、共通のインフラストラクチャを使用して CloudWatch Logs、Amazon S3、または Firehose にログを送信します。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 することも、ログを設定する前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では、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。

自分または組織内のユーザー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、ログの送信を設定するユーザーには、このセクションで後述する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。

CloudWatch Logs に送信されたログ

> **Important**
>
> 以下のリストにあるログタイプを CloudWatch Logs に設定するようにセットアップすると、AWS がその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を必要に応じて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掲載されているタイプのログが 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の CloudWatch Logs 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`
- `logs:PutResourcePolicy`
- `logs:DescribeResourcePolicies`
- `logs:DescribeLogGroups`

> **Note**
>
> `logs:DescribeLogGroups`、`logs:DescribeResourcePolicies`、または `logs:PutResourcePolicy` アクセス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 で * ワイルドカードを使用するように設定してください。例: `"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、これらの中の別のログを同じロググループ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。

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にロググループの `logs:PutResourcePolicy`、`logs:DescribeResourcePolicies`、および アクセス`logs:DescribeLogGroups`許可がある場合、CloudWatch Logs へのログの送信を開始すると、によって次のポリシー AWS が自動的に作成されます。

```
{
  "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、ログを送信している各ロググループをリソースポリシーにリストアップする必要がありますが、CloudWatch Logs リソースポリシーは 5,120 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、この上限に到達する可能性があります。

この問題を緩和するため、CloudWatch Logs はログを送信するサービスによって使用されるリソースポリシーのサイズを監視し、ポリシーが 5,120 文字のサイズ上限に近づいていることを CloudWatch Logs が検知すると、そのサービスのリソースポリシーで `/aws/vendedlogs/*` を自動的に有効化します。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、S3 に vended log を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、CloudFront 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にもアクセス許可が付与されます。

> **Important**
>
> Amazon S3 バケットにログを送信していて、バケットポリシーに `NotAction` または `NotPrincipal` 要素が含まれている場合、バケットにログ配信アクセス許可を自動的に追加し、ログサブスクリプションを作成すると失敗します。ログサブスクリプションを正常に作成するには、バケットポリシーにログ配信アクセス許可を手動で追加してから、ログサブスクリプションを作成する必要があります。詳細については、このセクションの指示を参照してください。
> バケットにカスタマーマネージド AWS KMS キーを使用したサーバー側の暗号化がある場合は、カスタマーマネージドキーのキーポリシーも追加する必要があります。詳細については、「Amazon S3」を参照してください。
> レプリケート先バケットで SSE-KMS が有効になっていてバケットキーが有効になっている場合、アタッチされたカスタマー管理 KMS キーポリシーはすべてのリクエストで想定どおりに動作しなくなりました。詳細については、「Amazon S3 バケットキーを使用した SSE-KMS のコスト削減」を参照してください。

カスタマーマネージド AWS KMS キーで供給ログと S3 暗号化を使用している場合は、バケットを設定するときに AWS KMS キー ID の代わりに完全修飾キー ARN を使用する必要があります。詳細については、「put-bucket-encryption」を参照してください。

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、Amazon S3 へのログの送信を開始すると、 によって次のポリシー AWS が自動的に作成されます。

```
{
    "Version": "2012-10-17",
    "Id": "AWSLogDeliveryWrite20150319",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryAclCheck",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
                },
            "Action": "s3:GetBucketAcl",
            "Resource": "arn:aws:s3:::amzn-s3-demo-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:::amzn-s3-demo-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。CloudTrail ログにこのようなエラーが表示されないようにするには、s3:ListBucket が delivery.logs.amazonaws.com にアクセスア

クセス許可を付与し、前述のバケットポリシーで設定された s3:GetBucketAcl アクセス許可で示されている 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-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) 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、このシナリオでは マネージドキーの使用はサポートされていないため、カスタマー AWS マネージドキーを使用する必要があります。 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は、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、このシナリオでは マネージドキーの使用はサポートされていないため、カスタマー AWS マネージドキーを使用する必要があります。カスタマーマネージド AWS KMS キーを使用する場合は、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 に送信されるログ

このセクションは、前のセクションの表に掲載されているタイプのログが 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の Firehose への送信を初めて設定するには、以下のアクセス許可を使用して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 に既に送信されている場合、これらのタイプの別のログを Firehose に送信するように設定するには、logs:CreateLogDelivery および firehose:TagDeliveryStream 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、 はこれらのログを 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。 は、 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 に設定されているすべての Firehose 配信ストリームにアクセス許可を付与しますtrue。 は、ログ記録を設定するときに、このタグを送信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 CloudWatch Logs、Amazon S3、または Firehose のうち 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。

動作しているログ配信は、次の 3 つの要素で構成されます。

- 実際にログを送信するリソースを表す論理オブジェクトである `DeliverySource`。
- 実際の配信先を表す論理オブジェクトである `DeliveryDestination`。
- 配信ソースを配信先に接続する `Delivery`。

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。

- PutDeliverySource を使用して配信ソースを作成します。
- PutDeliveryDestination を使用して配信先を作成します。
- クロスアカウントでログを配信する場合は、送信先アカウントで PutDeliveryDestinationPolicy 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 の配信ソースからアカウント 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。
- CreateDelivery を使用して、1 つの配信元と 1 つの配信先だけ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ことはお客様の責任です。そのためには、以下の手順を実行します。
>
> 1. DeleteDelivery オペレーションを使用して Delivery を削除します。
> 2. DeleteDeliverySource オペレーションを使用して DeliverySource を削除します。
> 3. 削除した DeliverySource に関連付けられた DeliveryDestination がこの特定の DeliverySource にのみ使用されている場合は、DeleteDeliveryDestinations オペレーションを使用して削除できます。

目次

CloudWatch Logs に送信されたログ

ユーザーアクセス許可

CloudWatch Logs 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",
                "logs:UpdateDeliveryConfiguration"
            ],
            "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",
                "logs:DescribeConfigurationTemplates"
            ],
            "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にロググループの `logs:PutResourcePolicy`、`logs:DescribeResourcePolicies`、および アクセス`logs:DescribeLogGroups`許可がある場合、CloudWatch Logs へのログの送信を開始すると、によって次のポリシー AWS が自動的に作成されます。

```
{
  "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、ログを送信している各ロググループをリソースポリシーにリストアップする必要がありますが、CloudWatch Logs リソースポリシーは 5,120 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。

この問題を緩和するため、CloudWatch Logs はログを送信するサービスによって使用されるリソースポリシーのサイズを監視し、ポリシーが 5,120 文字のサイズ上限に近づいていることを CloudWatch Logs が検知すると、そのサービスのリソースポリシーで `/aws/vendedlogs/*` を自動的に有効化します。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",
                "logs:UpdateDeliveryConfiguration"
            ],
            "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",
                "logs:DescribeConfigurationTemplates"
            ],
            "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、Amazon S3 へのログの送信を開始すると、 によって次のポリシー AWS が自動的に作成されます。

```
{
    "Version": "2012-10-17",
    "Id": "AWSLogDeliveryWrite20150319",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-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。CloudTrail ログにこのようなエラーが表示されないようにするには、s3:ListBucket が delivery.logs.amazonaws.com にアクセスアクセス許可を付与し、前述のバケットポリシーで設定された s3:GetBucketAcl アクセス許可で示されている 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-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) 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、このシナリオでは マネージドキーの使用はサポートされていないため、カスタマー AWS マネージドキーを使用する必要があります。 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は、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、このシナリオでは マネージドキーの使用はサポートされていないため、カスタマー AWS マネージドキーを使用する必要があります。カスタマーマネージド AWS KMS キーを使用する場合は、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 に送信されるログ

ユーザーアクセス許可

Firehose へのログ送信を有効にするには、次のアクセス許可を使用して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",
                "logs:UpdateDeliveryConfiguration"
            ],
            "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",
                "logs:DescribeConfigurationTemplates"
            ],
            "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、 はこれらのログを 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。 は、 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 に設定されているすべての Firehose 配信ストリームにアクセス許可を付与しますtrue。 は、ログ記録を設定するときに、このタグを送信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のレイヤーとして、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承認が必要です。これにより、そのサービス内でログを記録したリソースの AllowVendedLogDeliveryForResource アクションが承認されます。これらのサービスでは、

次のポリシーを使用し、*####*と*#######*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 vended log に関する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

API 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、前のセクションに記載されているアクセス許可に加えて、以下の追加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": [
                "*"
            ]
        }
    ]
}
```

アカウント間の配信の例

この例では、2 つのアカウントが関係しています。ログ生成リソースを持つアカウントはアカウント A、ID: *AAAAAAAAAAAA*、ログ消費リソースを持つアカウントはアカウント B、ID: *BBBBBBBBBBBB* です。

アカウント A は、ARN arn:aws:bedrock:*region*：*AAAAAAAAAAAA*:knowledge-base/*XXXXXXXXXX* を使用して、アカウントの Amazon Bedrock ナレッジベースからログを配信したいと考えています。

この例では、アカウント A には次のアクセス許可が必要です。

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowVendedLogDeliveryForKnowledgeBase",
            "Effect": "Allow",
            "Action": [
                "bedrock:AllowVendedLogDeliveryForResource"
            ],
            "Resource": "arn:aws:bedrock:region:AAAAAAAAAAAA:knowledge-base/XXXXXXXXXX"
```

```
        },
        {
            "Sid": "CreateLogDeliveryPermissions",
            "Effect": "Allow",
            "Action": [
                "logs:PutDeliverySource",
                "logs:CreateDelivery"
            ],
            "Resource": [
                "arn:aws:logs:region:AAAAAAAAAAAA:delivery-source:*",
                "arn:aws:logs:region:AAAAAAAAAAAA:delivery:*",
                "arn:aws:logs:region:BBBBBBBBBBBB:delivery-destination:*"
            ]
        }
    ]
}
```

配信ソースを作成する

まず、アカウント A は、Bedrock ナレッジベースを使用して配信ソースを作成します。

```
aws logs put-delivery-source --name my-delivery-source --log-type APPLICATION_LOGS --
resource-arn arn:aws:bedrock:region:AAAAAAAAAAAA:knowledge-base/XXXXXXXXXX
```

次に、アカウント B は、次のいずれかのフローを使用して配信先を作成する必要があります。

- Amazon S3 バケットへの配信を設定する
- Firehose ストリームへの配信を設定する

Amazon S3 バケットへの配信を設定する

アカウント B は ARN arn:aws:s3:::amzn-s3-demo-bucket を使用して S3 バケットでログを受信したいと考えています。この例の場合、アカウント B には次のアクセス許可が必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PutLogDestinationPermissions",
            "Effect": "Allow",
            "Action": [
```

```
                "logs:PutDeliveryDestination",
                "logs:PutDeliveryDestinationPolicy"
            ],
            "Resource": "arn:aws:logs:region:BBBBBBBBBBBB:delivery-destination:*"
        }
    ]
}
```

バケットポリシーには、次のアクセス許可が必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AWSLogsDeliveryWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/AWSLogs/AAAAAAAAAAAA/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": ["AAAAAAAAAAAA"]
                },
                "ArnLike": {
                    "aws:SourceArn": ["arn:aws:logs:region:AAAAAAAAAAAA:delivery-
source:my-delivery-source"]
                }
            }
        }
    ]
}
```

バケットが SSE-KMS で暗号化されている場合は、 AWS KMS キーポリシーに適切なアクセス許可があることを確認してください。例えば、KMS キーが arn:aws:kms:*region*:*BBBBBBBBBBBB*:key/*X* の場合、以下を使用します。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowLogsGenerateDataKey",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            }
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:region:BBBBBBBBBBBB:key/X",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": ["AAAAAAAAAAAA"]
                },
                "ArnLike": {
                    "aws:SourceArn": ["arn:aws:logs:region:AAAAAAAAAAAA:delivery-
source:my-delivery-source"]
                }
            }
        }
    ]
}
```

アカウント B は、S3 バケットを宛先リソースとして配信先を作成できます。

```
aws logs put-delivery-destination --name my-s3-delivery-destination --delivery-
destination-configuration "destinationResourceArn=arn:aws:s3:::amzn-s3-demo-bucket"
```

次に、アカウント B は新しく作成された配信先に配信先ポリシーを作成し、アカウント A が口グ配信を作成するアクセス許可を付与します。新しく作成された配信先に追加されるポリシーは次のとおりです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateDelivery",
            "Effect": "Allow",
            "Principal": {
                "AWS": "AAAAAAAAAAAA"
```

```
            },
            "Action": [
                "logs:CreateDelivery"
            ],
            "Resource": "arn:aws:logs:region:BBBBBBBBBBBB:delivery-destination:my-s3-
delivery-destination"
        }
    ]
}
```

このポリシーは、アカウント B のコンピュータに destination-policy-s3.json として保存されます。このリソースをアタッチするために、アカウント B は次のコマンドを実行します。

```
aws logs put-delivery-destination-policy --delivery-destination-name my-s3-delivery-
destination --delivery-destination-policy file://destination-policy-s3.json
```

最後に、アカウント A は配信を作成し、アカウント A の配信ソースをアカウント B の配信先にリンクします。

```
aws logs create-delivery --delivery-source-name my-delivery-source --delivery-
destination-arn arn:aws:logs:region:BBBBBBBBBBBB:delivery-destination:my-s3-delivery-
destination
```

Firehose ストリームへの配信を設定する

この例では、アカウント B は Firehose ストリームでログを受信したいと考えています。Firehose ストリームには次の ARN があり、DirectPut 配信ストリームタイプを使用するように設定されています。

arn:aws:firehose:*region*:*BBBBBBBBBBBB*:deliverystream/*X*

この例では、アカウント B には次のアクセス許可が必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowFirehoseCreateSLR",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
```

```
            "Resource": "arn:aws:iam::BBBBBBBBBBBB:role/aws-service-role/
delivery.logs.amazonaws.com/AWSServiceRoleForLogDelivery",
        },
        {
            "Sid": "AllowFirehoseTagging",
            "Effect": "Allow",
            "Action": [
                "firehose:TagDeliveryStream"
            ],
            "Resource": "arn:aws:firehose:region:BBBBBBBBBBBB:deliverystream/X"
        },
        {
            "Sid": "AllowFirehoseDeliveryDestination",
            "Effect": "Allow",
            "Action": [
                "logs:PutDeliveryDestination",
                "logs:PutDeliveryDestinationPolicy"
            ],
            "Resource": "arn:aws:logs:region:BBBBBBBBBBBB:delivery-destination:*"
        }
    ]
}
```

Firehose ストリームのタグ LogDeliveryEnabled は true に設定する必要があります。

その後、アカウント B は、Firehose ストリームを宛先リソースとして配信先を作成します。

```
aws logs put-delivery-destination --name my-fh-delivery-destination --delivery-
destination-configuration
 "destinationResourceArn=arn:aws:firehose:region:BBBBBBBBBBBB:deliverystream/X"
```

次に、アカウント B は新しく作成された配信先に配信先ポリシーを作成し、アカウント A がログ配信を作成するアクセス許可を付与します。新しく作成された配信先に追加されるポリシーは次のとおりです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateDelivery",
            "Effect": "Allow",
            "Principal": {
                "AWS": "AAAAAAAAAAAA"
```

```
            },
            "Action": [
                "logs:CreateDelivery"
            ],
            "Resource": "arn:aws:logs:region:BBBBBBBBBBBB:delivery-destination:my-fh-
delivery-destination"
        }
    ]
}
```

このポリシーは、アカウント B のコンピュータに destination-policy-fh.json として保存されます。このリソースをアタッチするために、アカウント B は次のコマンドを実行します。

```
aws logs put-delivery-destination-policy --delivery-destination-name my-fh-delivery-
destination --delivery-destination-policy file://destination-policy-fh.json
```

最後に、アカウント A は配信を作成し、アカウント A の配信ソースをアカウント B の配信先にリンクします。

```
aws logs create-delivery --delivery-source-name my-delivery-source --delivery-
destination-arn arn:aws:logs:region:BBBBBBBBBBBB:delivery-destination:my-fh-delivery-
destination
```

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。

リソースポリシーで aws:SourceArn、aws:SourceAccount、aws:SourceOrgID、aws:SourceOrgPaths のグローバル条件コンテキストキーを使用して、CloudWatch Logs が別のサービスに付与する許可をその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 「エンティティパスを理解する」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*:*。

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。

AWS マネージドポリシーに対する CloudWatch Logs の更新

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 Logs 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入手するには、CloudWatch Logs ドキュメントの履歴ページから、RSS 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	CloudWatch Logs で AWSServiceRoleForL	2021 年 7 月 15 日

変更	説明	日付
されたロールポリシー – 既存のポリシーに更新	ogDelivery サービスにリンクされたロールに関連付けられた IAM ポリシーの許可が変更されました。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。	
CloudWatch Logs が変更の追跡を開始しました	CloudWatch Logs は AWS 、管理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

この章では、ロググループのログデータを Amazon S3 バケットにエクスポートし、このデータをカスタム処理や分析で使用したり、別のシステムの読み込んだりする方法を説明します。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 Object Lock 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、「ログクラス」を参照してください。

ログを継続的にアーカイブする方法として、Amazon S3 に定期的にエクスポートしないことをお勧めします。そのユースケースでは、代わりにサブスクリプションを使用することをお勧めします。サブスクリプションの詳細については、「」を参照してください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。

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。この時間以降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。

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

以下の例では、Amazon CloudWatch コンソールを使用して、`my-log-group` という名前の Amazon CloudWatch Logs ロググループからすべてのデータを `my-exported-logs` という名前の Amazon S3 バケットにエクスポートします。

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

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 専用に作成したバケットを使用することをお勧めします。ただし、既存のバケットを使用する場合は、ステップ 2 に進むことができます。

> **Note**
>
> S3 バケットは、エクスポートするログデータと同じリージョンに存在している必要があります。CloudWatch Logs 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 Logs があるリージョンを選択します。
3. バケットの作成 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. [Region (リージョン)] で、CloudWatch Logs データが存在するリージョンを選択します。

6. [Create] (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。

- 以下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) 用のロールを作成する」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 ユーザーのロールの作成」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します。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 ID のリストにあるのは、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です。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 Logs は S3 バケットにログデータをエクスポートできるようになります。バケット所有者には、エクスポートされたすべてのオブジェクトに対する完全なアクセス権限があります。

> **Warning**
>
> バケットにアタッチされているポリシーがすでに 1 つ以上ある場合は、そのポリシーに CloudWatch Logs アクセスのステートメントを追加します。バケットにアクセスするユーザーに適したアクセス許可であることを確認するために、アクセス許可の結果セットを評価することをお勧めします。

(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. [Save changes] (変更の保存) をクリックします。
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. [Save changes] (変更の保存) をクリックします。

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. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
3. 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。
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 専用に作成したバケットを使用することをお勧めします。ただし、既存のバケットを使用する場合は、ステップ 2 に進むことができます。

> **Note**
>
> S3 バケットは、エクスポートするログデータと同じリージョンに存在している必要があります。CloudWatch Logs 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 Logs があるリージョンを選択します。
3. バケットの作成 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. [Region (リージョン)] で、CloudWatch Logs データが存在するリージョンを選択します。

6. [Create] (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、CloudWatch Logs がエクスポート先アカウントの宛先 Amazon S3 バケットへの `s3:PutObject` アクセス許可を有効にする必要があります。

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。

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. [Create policy] 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. [Next (次へ)] を選択します。
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。

- 以下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) 用のロールを作成する」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 ユーザーのロールの作成」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します。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 ID のリストにあるのは、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です。`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 Logs は S3 バケットにログデータをエクスポートできるようになります。バケット所有者には、エクスポートされたすべてのオブジェクトに対する完全なアクセス権限があります。

> **Warning**
>
> バケットにアタッチされているポリシーがすでに 1 つ以上ある場合は、そのポリシーに CloudWatch Logs アクセスのステートメントを追加します。バケットにアクセスするユーザーに適したアクセス許可であることを確認するために、アクセス許可の結果セットを評価することをお勧めします。

(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. [Save changes] (変更の保存) をクリックします。
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. [Save changes] (変更の保存) をクリックします。

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. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
3. 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。
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、すべてのデータを `my-log-group` という名前の CloudWatch Logs ロググループから `my-exported-logs` という名前の Amazon S3 バケットにエクスポートします。この例では、「`my-log-group`」というロググループを作成済みであることを前提としています。

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 専用に作成したバケットを使用することをお勧めします。ただし、既存のバケットを使用する場合は、ステップ 2 に進むことができます。

> **Note**
>
> S3 バケットは、エクスポートするログデータと同じリージョンに存在している必要があります。CloudWatch Logs 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。

- 以下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) 用のロールを作成する」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 ユーザーのロールの作成」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します。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 ID のリストにあるのは、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です。`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、バケットでアクセスポリシーとして先ほど追加したポリシーを設定します。このポリシーにより、CloudWatch Logs は S3 バケットにログデータをエクスポートできるようになります。バケット所有者には、エクスポートされたすべてのオブジェクトに対する完全なアクセス権限があります。

```
aws s3api put-bucket-policy --bucket my-exported-logs --policy file://policy.json
```

> **Warning**
>
> バケットにアタッチされているポリシーがすでに 1 つ以上ある場合は、そのポリシーに CloudWatch Logs アクセスのステートメントを追加します。バケットにアクセスするユーザーに適したアクセス許可であることを確認するために、アクセス許可の結果セットを評価することをお勧めします。

(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 専用に作成したバケットを使用することをお勧めします。ただし、既存のバケットを使用する場合は、ステップ 2 に進むことができます。

> **Note**
>
> S3 バケットは、エクスポートするログデータと同じリージョンに存在している必要があります。CloudWatch Logs 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 Logs が宛先 Amazon S3 s3:PutObject バケットに対するアクセス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。

- 以下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) 用のロールを作成する」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 ユーザーのロールの作成」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します。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 ID のリストにあるのは、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です。`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、バケットでアクセスポリシーとして先ほど追加したポリシーを設定します。このポリシーにより、CloudWatch Logs は S3 バケットにログデータをエクスポートできるようになります。バケット所有者には、エクスポートされたすべてのオブジェクトに対する完全なアクセス権限があります。

```
aws s3api put-bucket-policy --bucket my-exported-logs --policy file://policy.json
```

> **Warning**
>
> バケットにアタッチされているポリシーがすでに 1 つ以上ある場合は、そのポリシーに CloudWatch Logs アクセスのステートメントを追加します。バケットにアクセスするユーザーに適したアクセス許可であることを確認するために、アクセス許可の結果セットを評価することをお勧めします。

(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る点に注意してください。

CloudWatch Logs データの Amazon OpenSearch Service へのストリーミング

Amazon CloudWatch Logs でロググループを設定して、ほぼリアルタイムで Amazon OpenSearch Service クラスターにデータをストリーミングできます。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 Service 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ています。ログクラスの詳細については、「ログクラス」を参照してください。

ストリーミングされるログデータの量に応じて、関数レベルの同時実行制限を設定することを検討してください。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 Service にストリーミングすると、使用料が高くなる可能性があるため、 AWS Billing and Cost Management コンソールで予算を作成することをお勧めします。詳細については、AWS 「 Budgets によるコストの管理」を参照してください。

このセクションでは、ロググループを OpenSearch Service にサブスクライブする前に完了する必要がある前提条件について説明します。また、ロググループを OpenSearch Service にサブスクライブする方法も説明します。

前提条件

開始する前に、OpenSearch Service 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 Service ドメイン設定を確認し、クラスターで処理されるデータの量に基づいてクラスター設定を変更することができます。ドメインの作成手順については、「OpenSearch Service のドメインの作成」を参照してください。

OpenSearch Service 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 Service 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. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[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] (アクション)、[Subscription filters] (サブスクリプションフィルター)、[Create Amazon OpenSearch Service subscription filter] (Amazon OpenSearch Service サブスクリプションフィルターを作成) の順に選択します。
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 Execution Role] (Lambda IAM 実行ロール) では、OpenSearch 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。

   選択した IAM ロールは、これらの要件を満たす必要があります。

   - 信頼関係に `lambda.amazonaws.com` が含まれている必要があります。
   - 以下のポリシーが含まれている必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "es:*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:es:region:account-id:domain/target-domain-name/
*"
```

```
        }
    ]
}
```

- ターゲットの OpenSearch Service ドメインで VPC アクセスを使用する場合、ロールには AWSLambdaVPCAccessExecutionRole ポリシーがアタッチされている必要があります。Amazon が管理するこのポリシーにより、お客様の VPC へのアクセスが Lambda に許可され、Lambda は VPC の OpenSearch エンドポイントに書き込むことができます。

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、関心のあるデータだけが OpenSearch クラスターに送信されるようになります。詳細については、「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。

SDK を使用した CloudWatch Logs のコード例 AWS SDKs

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、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

AWS SDK 開発者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 Logs 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

- SDK を使用した CloudWatch Logs の基本的な例 AWS SDKs
  - SDK を使用した CloudWatch Logs のアクション AWS SDKs
    - AWS SDK AssociateKmsKeyで を使用する
    - AWS SDK CancelExportTaskで を使用する
    - AWS SDK 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 DeleteSubscriptionFilterで を使用する
    - AWS SDK DescribeExportTasksで を使用する
    - AWS SDK または CLI DescribeLogGroupsで を使用する
    - AWS SDK DescribeSubscriptionFiltersで を使用する
    - AWS SDK GetQueryResultsで を使用する
    - AWS SDK PutSubscriptionFilterで を使用する
    - AWS SDK StartLiveTailで を使用する
    - AWS SDK StartQueryで を使用する
- SDK を使用した CloudWatch Logs のシナリオ AWS SDKs

- CloudWatch Logs を使用して大規模なクエリを実行する
- 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

SDK を使用した CloudWatch Logs の基本的な例 AWS SDKs

次のコード例は、 AWS SDK で Amazon CloudWatch Logs の基本的な機能を使用する方法を示しています。

例

- SDK を使用した CloudWatch Logs のアクション AWS SDKs
  - AWS SDK AssociateKmsKeyで を使用する
  - AWS SDK CancelExportTaskで を使用する
  - AWS SDK 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 DeleteSubscriptionFilterで を使用する
  - AWS SDK DescribeExportTasksで を使用する
  - AWS SDK または CLI DescribeLogGroupsで を使用する
  - AWS SDK DescribeSubscriptionFiltersで を使用する
  - AWS SDK GetQueryResultsで を使用する
  - AWS SDK PutSubscriptionFilterで を使用する
  - AWS SDK StartLiveTailで を使用する
  - AWS SDK StartQueryで を使用する

SDK を使用した CloudWatch Logs のアクション AWS SDKs

次のコード例は、 AWS SDKs を使用して個々の CloudWatch Logs アクションを実行する方法を示しています。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

これらは CloudWatch Logs API を呼び出すもので、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抜粋です。アクションは SDK を使用した CloudWatch Logs のシナリオ AWS SDKs のコンテキスト内で確認できます。

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。詳細な一覧については、「Amazon CloudWatch Logs API Reference」(Amazon CloudWatch Logs API リファレンス) を参照してください。

例

- AWS SDK AssociateKmsKeyで を使用する
- AWS SDK CancelExportTaskで を使用する
- AWS SDK 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 DeleteSubscriptionFilterで を使用する
- AWS SDK DescribeExportTasksで を使用する
- AWS SDK または CLI DescribeLogGroupsで を使用する
- AWS SDK DescribeSubscriptionFiltersで を使用する
- AWS SDK GetQueryResultsで を使用する
- AWS SDK PutSubscriptionFilterで を使用する
- AWS SDK StartLiveTailで を使用する
- AWS SDK StartQueryで を使用する

AWS SDK **AssociateKmsKey**で を使用する

次のコード例は、`AssociateKmsKey` を使用する方法を示しています。

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
    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、「AWS SDK for .NET API リファレンス」にある「AssociateKmsKey」を参照してください。

AWS SDK 開発者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 Logs 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 **CancelExportTask**で を使用する

次のコード例は、CancelExportTask を使用する方法を示しています。

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
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、「AWS SDK for .NET API リファレンス」にある「CancelExportTask」を参照してください。

AWS SDK 開発者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 Logs 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 **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

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
    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 = "amzn-s3-demo-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、「AWS SDK for .NET API リファレンス」の「CreateExportTask」を参照してください。

AWS SDK 開発者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 Logs 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

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
    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、「AWS SDK for .NET API リファレンス」の「CreateLogGroup」を参照してください。

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、AWS CLI コマンドリファレンスの「CreateLogGroup」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
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、「AWS SDK for JavaScript API リファレンス」の「CreateLogGroup」を参照してください。

AWS SDK 開発者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 Logs 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

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
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、「AWS SDK for .NET API リファレンス」の「CreateLogStream」を参照してください。

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、AWS CLI コマンドリファレンスの「CreateLogStream」を参照してください。

AWS SDK 開発者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 Logs 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

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
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、「AWS SDK for .NET API リファレンス」の「DeleteLogGroup」を参照してください。

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、AWS CLI コマンドリファレンスの「DeleteLogGroup」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
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、「AWS SDK for JavaScript API リファレンス」の「DeleteLogGroup」を参照してください。

AWS SDK 開発者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 Logs 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 **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**
>
> GitHub には、その他のリソースもあります。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、「AWS SDK for C++ SDK for Kotlin API リファレンス」の「DeleteAlarms」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
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、「AWS SDK for Java 2.x SDK for Kotlin API リファレンス」の「DeleteAlarms」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
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、「AWS SDK for JavaScript SDK for Kotlin API リファレンス」の「DeleteAlarms」を参照してください。

SDK for JavaScript (v2)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
// 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、「AWS SDK for JavaScript SDK for Kotlin API リファレンス」の「DeleteAlarms」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
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」の「DeleteSubscriptionFilter」を参照してください。

AWS SDK 開発者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 Logs 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 **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

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
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、「AWS SDK for .NET API リファレンス」の「DescribeExportTasks」を参照してください。

AWS SDK 開発者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 Logs 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

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
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、「AWS SDK for .NET API リファレンス」の「DescribeLogGroups」を参照してください。

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、AWS CLI コマンドリファレンスの「DescribeLogGroups」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
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?.every((lg) => !!lg)) {
      logGroups.push(...page.logGroups);
    }
  }

  console.log(logGroups);
  return logGroups;
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「DescribeLogGroups」を参照してください。

AWS SDK 開発者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 Logs 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 **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**
>
> GitHub には、その他のリソースもあります。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、「AWS SDK for C++ API リファレンス」の「DescribeSubscriptionFilters」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
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、「AWS SDK for Java 2.x API リファレンス」の「DescribeSubscriptionFilters」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
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、「AWS SDK for JavaScript API リファレンス」の「DescribeSubscriptionFilters」を参照してください。

SDK for JavaScript (v2)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
// 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、「AWS SDK for JavaScript API リファレンス」の「DescribeSubscriptionFilters」を参照してください。

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
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」の「DescribeSubscriptionFilters」を参照してください。

AWS SDK 開発者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 Logs 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 **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**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
/**
 * 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、「AWS SDK for JavaScript API リファレンス」の「GetQueryResults」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
    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、「AWS SDK for Python (Boto3) API Reference」の「GetQueryResults」を参照してください。

AWS SDK 開発者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 Logs 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 **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**
>
> GitHub には、その他のリソースもあります。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、「AWS SDK for C++ API リファレンス」の「PutSubscriptionFilter」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
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、「AWS SDK for Java 2.x API リファレンス」の「PutSubscriptionFilter」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
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、「AWS SDK for JavaScript API リファレンス」の「PutSubscriptionFilter」を参照してください。

SDK for JavaScript (v2)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
// 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、「AWS SDK for JavaScript API リファレンス」の「PutSubscriptionFilter」を参照してください。

AWS SDK 開発者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 Logs 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 **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

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、「AWS SDK for .NET API リファレンス」の「StartLiveTail」を参照してください。

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、「AWS SDK for Go API リファレンス」の「StartLiveTail」を参照してください。

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、「AWS SDK for Java 2.x API リファレンス」の「StartLiveTail」を参照してください。

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、「AWS SDK for JavaScript API リファレンス」の「StartLiveTail」を参照してください。

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」の「StartLiveTail」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、「AWS SDK for Python (Boto3) API リファレンス」の「StartLiveTail」を参照してください。

AWS SDK 開発者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 Logs 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 **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**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
/**
 * 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、「AWS SDK for JavaScript API リファレンス」の「StartQuery」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

```
    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_group,
                    startTime=start_time,
                    endTime=end_time,
                    queryString=self.query_string,
                    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_group,
                startTime=start_time,
                endTime=end_time,
                queryString=self.query_string,
                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、「AWS SDK for Python (Boto3) API Referenc」の「StartQuery」を参照してください。

AWS SDK 開発者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 Logs 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。

SDK を使用した CloudWatch Logs のシナリオ AWS SDKs

次のコード例は、 AWS SDKs を使用して CloudWatch Logs に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 Logs 内または他の AWS のサービスと組み合わせた複数の関数を呼び出し、特定のタスクを実行する方法を示します。各シナリオには、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

シナリオは、サービスアクションをコンテキストで理解するのに役立つ中級レベルの経験を対象としています。

例

- CloudWatch Logs を使用して大規模なクエリを実行する
- 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

CloudWatch Logs を使用して大規模なクエリを実行する

次のコード例は、CloudWatch Logs 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**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

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(Number.parseInt(process.env.QUERY_START_DATE)),
    new Date(Number.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]);
}

/**
 * Simple wrapper for the GetQueryResultsCommand.
 * @param {string} queryId
 */
_getQueryResults(queryId) {
  return this.client.send(new GetQueryResultsCommand({ queryId }));
```

```
}

/**
 * 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 [];
    }
    throw err;
  }
}

/**
 * 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;
    }
  }

  /**
   * 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**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、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。

このファイルは、10,000 を超える結果の CloudWatch 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。

```
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",
)

DEFAULT_QUERY_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"


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.
        Fetches the environment variable for log group, returning the default
value if it
        does not exist.

        :return: Tuple of query start date and end date as integers and the log
group.
        :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)

        try:
            log_group = os.environ["QUERY_LOG_GROUP"]
        except KeyError:
            logging.warning("No QUERY_LOG_GROUP environment variable, using
default value")
            log_group = DEFAULT_QUERY_LOG_GROUP

        return query_start_date, query_end_date, log_group

    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",
        query="fields @timestamp, @message | sort @timestamp asc"
    ):
        """
```

```
        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
        :param query: Query string to pass to the CloudWatchQuery instance
        :type query: str
        """
        cloudwatch_query = CloudWatchQuery(
            log_group=log_group,
            query_string=query
        )
        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, log_group =
 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,
 log_group=log_group)
```

```
if __name__ == "__main__":
    main()
```

このモジュールは、10,000 を超える結果の CloudWatch クエリを処理します。

```
import logging
import time
from datetime import datetime
import threading
import boto3

from date_utilities import DateUtilities

DEFAULT_QUERY = "fields @timestamp, @message | sort @timestamp asc"
DEFAULT_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"

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.

    :vartype date_range: tuple
    :ivar limit: Maximum number of log entries to return.
    :vartype limit: int
    :log_group str: Name of the log group to query
    :query_string str: query
    """

    def __init__(self, log_group: str = DEFAULT_LOG_GROUP, query_string:
 str=DEFAULT_QUERY) -> None:
        self.lock = threading.Lock()
        self.log_group = log_group
        self.query_string = query_string
        self.query_results = []
        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}"
            f"\n       LOG GROUP: {self.log_group}"
        )
        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

    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_group,
                    startTime=start_time,
                    endTime=end_time,
                    queryString=self.query_string,
                    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_group,
                startTime=start_time,
                endTime=end_time,
                queryString=self.query_string,
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")


    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、『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 開発者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 Logs 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。

スケジュールされたイベントを使用した Lambda 関数の呼び出し

次のコード例は、Amazon EventBridge のスケジュールされたイベントによって呼び出される AWS Lambda 関数を作成する方法を示しています。

Java

SDK for Java 2.x

AWS Lambda 関数を呼び出す Amazon EventBridge のスケジュールされたイベントを作成する方法を示します。cron 式を使用して Lambda 関数が呼び出されるタイミングをスケジュールするように EventBridge を設定します。この例では、Lambda Java ランタイム API を使用して Lambda 関数を作成します。この例では、さまざまな AWS サービスを呼び出して、特定のユースケースを実行します。この例では、年間の記念日に従業員を祝福するモバイルテキストメッセージを従業員に送信するアプリを作成する方法を示します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK for JavaScript (v3)

AWS Lambda 関数を呼び出す Amazon EventBridge のスケジュールされたイベントを作成する方法を示します。cron 式を使用して Lambda 関数が呼び出されるタイミングをスケジュールするように EventBridge を設定します。この例では、Lambda JavaScript ランタイム API を使用して Lambda 関数を作成します。この例では、さまざまな AWS サービスを呼び出して、特定のユースケースを実行します。この例では、年間の記念日に従業員を祝福するモバイルテキストメッセージを従業員に送信するアプリを作成する方法を示します。

完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例は、AWS SDK for JavaScript v3 デベロッパーガイドでも使用できます。

この例で使用されているサービス

- CloudWatch Logs

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK for Python (Boto3)

この例では、スケジュールされた Amazon EventBridge イベントのターゲットとして AWS Lambda 関数を登録する方法を示します。Lambda ハンドラーは、後で取得するために Amazon CloudWatch Logs にわかりやすいメッセージと完全なイベントデータを書き込みます。

- Lambda 関数をデプロイします。
- EventBridge スケジュールイベントを作成し、Lambda 関数をターゲットにします。
- EventBridge に Lambda 関数を呼び出す許可を付与します
- CloudWatch Logs 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ています。
- 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 GitHub で最もよく確認できます。完全なソースコードとセットアップおよび実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

AWS SDK 開発者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 Logs 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 カスタマーは、最もセキュリティの影響を受けやすい組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、 AWS 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 AWS 責任があります。 AWS また、 では、安全に使用できるサービスも提供しています。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。WorkSpaces に適用されるコンプライアンスプログラムの詳細については、「コンプライアンスプログラム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。
- クラウド内のセキュリティ – 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、セキュリティとコンプライアンスの目標を満たすように Amazon CloudWatch Logs を設定する方法について説明します。また、CloudWatch Logs リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

内容

Amazon CloudWatch Logs におけるデータ保護

> **Note**
>
> CloudWatch Logs では AWS、 の一般的なデータ保護に関する以下の情報に加えて、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。

Amazon CloudWatch Logs でのデータ保護には、 AWS 責任共有モデルが適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーに関するよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された AWS 責任共有モデルおよび GDPR のブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM 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 が必須で、TLS 1.3 をお勧めします。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、その中のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、または SDK を使用して CloudWatch Logs AWS CLIまたは他の AWS のサービス を使用する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

保管中の暗号化

CloudWatch Logs は、暗号化を使用して保管時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に対応し、256 ビットの Advanced Encryption Standard Galois/Counter Mode (AES-GCM) によるサーバー側の暗号化を使用して、保管中のログデータを暗号化します。

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 CloudWatch Logs 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 Logs は、転送中のデータのエンドツーエンドの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 Logs 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。次のセクションでは、AWS Identity and Access Management (IAM) と CloudWatch Logs を使用して、リソースにアクセスできるユーザーを制御することで、リソースをセキュリティで保護する方法について詳しく説明します。

- 認証
- 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) 用のロールを作成する」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 ユーザーのロールの作成」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します。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

有効な認証情報があればリクエストを認証できますが、アクセス許可がなければ CloudWatch Logs リソースの作成やアクセスはできません。たとえば、ログストリーム、ロググループなどを作成する許可が必要となります。

以下のセクションでは、CloudWatch Logs のアクセス許可を管理する方法について説明します。最初に概要のセクションを読むことをお勧めします。

- CloudWatch Logs リソースへの許可の管理の概要
- CloudWatch Logs 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 の許可リファレンス

CloudWatch Logs リソースへの許可の管理の概要

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) 用のロールを作成する」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 ユーザーのロールの作成」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します。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 Logs がリソースとオペレーションをログに記録する

CloudWatch Logs では、プライマリリソースはロググループ、ログストリーム、送信先です。CloudWatch Logs はサブリソース (プライマリリソースと使用する他のリソース) をサポートしていません。

これらのリソースとサブリソースには、次の表に示すとおり、一意の Amazon リソースネーム (ARN) が関連付けられています。

リソースタイプ	ARN 形式
ロググループ	次の 2 つの形式が使用されます。末尾に `:*` が付いている 2 つ目の方は、`describe-log-groups` CLI コマンドと [DescribeLogGroups] API によって返される形式です。 arn:aws:logs:*region*:*account-id* :log-group:*log_group_name*

リソースタイプ	ARN 形式
	arn:aws:logs:*region*:*account-id* :log-group:*log_group_name* :* 次の状況では、末尾に :* がない最初のバージョンを使用します。 • CloudWatch Logs APIslogGroupIdentifier の入力フィールド。 • タグ付け API の resourceArn フィールド • IAM ポリシーで、TagResource、UntagResource、および ListTagsForResource のアクセス許可を指定する場合。 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、末尾に :* がある 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*
送信先	arn:aws:logs:*region*:*account-id* :destination:*destination_name*

ARN の詳細については、IAM ユーザーガイドの「ARN」を参照してください。CloudWatch Logs ARN の詳細については、「Amazon Web Services 全般のリファレンス」の「Amazon リソースネーム (ARN)」を参照してください。CloudWatch Logs を対象とするポリシーの例については、CloudWatch Logs で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 を参照してください。

CloudWatch Logs には、CloudWatch Logs リソースを操作するための一連のオペレーションが用意されています。使用可能なオペレーションのリストについては、「CloudWatch Logs の許可リファレンス」を参照してください。

リソース所有権についての理解

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 CloudWatch Logs 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 CloudWatch Logs リソースを作成できます。ただし、ユーザーが属する AWS アカウントは CloudWatch Logs リソースを所有しているとします。
- CloudWatch Logs リソースを作成するアクセス許可を持つ AWS IAM ロールをアカウントに作成する場合、ロールを引き受けることのできるすべてのユーザーが CloudWatch Logs リソースを作成できます。ロールが属する AWS アカウントは CloudWatch Logs リソースを所有しているとします。

リソースへのアクセスの管理

アクセス権限ポリシー 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、アクセス権限のポリシーを作成するために使用可能なオプションについて説明します。

> **Note**
>
> このセクションでは、CloudWatch Logs のコンテキストでの IAM の使用について説明します。これは、IAM サービスに関する詳細情報を取得できません。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 ポリシーのリファレンス」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、およびクロスアカウントのサブスクリプションを有効にするために使用する、送信先のリソースベースのポリシーをサポートします。詳細については、「クロスアカウント、クロスリージョンの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、コントリビューターデータを表示する時系列を作成できる CloudWatch 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` と `cloudwatch:GetInsightRuleReport` アクセス許可を付与すると、そのユーザーは CloudWatch Logs でロググループを評価し、その結果を参照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 Logs は、クロスアカウントのサブスクリプションを有効にするために使用する、送信先のリソースベースのポリシーをサポートします。詳細については、「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、PutDestinationPolicy API を使用して、送信先にリソースポリシーを追加できます。次の例では、 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 にサブスクライブできるようにします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、このサービスは、一連の API オペレーションを定義します。これらの API オペレーションを実行するためのアクセス許可を付与するために、CloudWatch Logs ではポリシーに一連のアクションを定義できます。一部の API オペレーションは、API オペレーションを実行するために複数のアクションに対するアクセス許可を要求できます。リソースおよび API オペレーションに関する詳細については、「CloudWatch Logs がリソースとオペレーションをログに記録する」および「CloudWatch Logs の許可リファレンス」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 Logs がリソースとオペレーションをログに記録する」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、権限 (リソースベースのポリシーにのみ適用) を受け取りたいユーザー、アカウント、サービス、またはその他のエンティティを指定します。CloudWatch Logs 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 ポリシー構文の詳細と説明については、IAM ユーザーガイドの「AWS IAM ポリシーリファレンス」を参照してください。

すべての CloudWatch Logs API アクションとそれらが適用されるリソースの表については、「CloudWatch Logs の許可リファレンス」を参照してください。

ポリシーでの条件を指定する

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ポリシー全体のキーのリストについては、「AWS サービスのアクション、リソース、および条件キー」とAWS「グローバル条件コンテキストキー」を参照してください。

> **Note**
>
> ロググループや送信先などの CloudWatch Logs のリソースへのアクセスを制御するには、タグを使用します。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 Logs 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**
>
> 初めに、CloudWatch Logs リソースへのアクセスを管理するための基本概念と使用可能なオプションについて説明する概要トピックをお読みにな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 コンソールの使用に必要な許可
- AWS CloudWatch Logs 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 Logs アクションを示すリストについては、「CloudWatch Logs の許可リファレンス」を参照してください。

CloudWatch コンソールの使用に必要な許可

ユーザーが CloudWatch コンソールで CloudWatch Logs を使用するには、そのユーザーにアカウントの他の AWS リソース AWS 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 Logs コンソールで CloudWatch Logs を使用するには、次のサービスからのアクセス許可が必要になります。

- CloudWatch
- CloudWatch Logs
- OpenSearch Service
- IAM
- Kinesis

- Lambda
- Amazon S3

これらの最小限必要なアクセス権限よりも制限された 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。`CloudWatchReadOnlyAccess` で説明されているとおり、ユーザーが CloudWatch コンソールを使用できること、および、AWS CloudWatch Logs の マネージド (事前定義) ポリシー 管理ポリシーがユーザーにアタッチされていることを確認してください。

AWS CLI または CloudWatch Logs API のみを呼び出すユーザーには、最小限のコンソールアクセス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のサブスクリプションを管理していないユーザーが、コンソールを使用するために必要なフルセットのアクセス許可は、次のとおりです。

- cloudwatch:GetMetricData
- cloudwatch:ListMetrics
- logs:CancelExportTask
- logs:CreateExportTask
- logs:CreateLogGroup
- logs:CreateLogStream
- logs:DeleteLogGroup
- logs:DeleteLogStream
- logs:DeleteMetricFilter
- logs:DeleteQueryDefinition
- logs:DeleteRetentionPolicy
- logs:DeleteSubscriptionFilter
- logs:DescribeExportTasks
- logs:DescribeLogGroups
- logs:DescribeLogStreams
- logs:DescribeMetricFilters
- logs:DescribeQueryDefinitions
- logs:DescribeQueries
- logs:DescribeSubscriptionFilters

- logs:FilterLogEvents
- logs:GetLogEvents
- logs:GetLogGroupFields
- logs:GetLogRecord
- logs:GetQueryResults
- logs:PutMetricFilter
- logs:PutQueryDefinition
- logs:PutRetentionPolicy
- logs:StartQuery
- logs:StopQuery
- logs:PutSubscriptionFilter
- logs:TestMetricFilter

コンソールを使用してログのサブスクリプションを管理するユーザーには、以下のアクセス許可も必要です。

- es:DescribeElasticsearchDomain
- es:ListDomainNames
- iam:AttachRolePolicy
- iam:CreateRole
- iam:GetPolicy
- iam:GetPolicyVersion
- iam:GetRole
- iam:ListAttachedRolePolicies
- iam:ListRoles
- kinesis:DescribeStreams
- kinesis:ListStreams
- lambda:AddPermission
- lambda:CreateFunction
- lambda:GetFunctionConfiguration
- lambda:ListAliases

- lambda:ListFunctions
- lambda:ListVersionsByFunction
- lambda:RemovePermission
- s3:ListBuckets

AWS CloudWatch Logs 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以下の AWS 管理ポリシーは、CloudWatch Logs に固有です。

- CloudWatchLogsFullAccess – CloudWatch Logs へのフルアクセスを付与します。
- CloudWatchLogsReadOnlyAccess – CloudWatch Logs への読み取り専用アクセスを付与します。

CloudWatchLogsFullAccess

CloudWatchLogsFullAccess ポリシーは、CloudWatch Logs へのフルアクセスを付与します。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、このポリシーには `cloudwatch:GenerateQuery` アクセス許可が含まれています。これには、一部の機能で CloudWatch Logs と OpenSearch Service の統合を有効にするための Amazon OpenSearch Service および IAM のアクセス許可が含まれています。完全な内容は次のとおりです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchLogsFullAccess",
            "Effect": "Allow",
            "Action": [
                "logs:*",
                "cloudwatch:GenerateQuery",
                "opensearch:ApplicationAccessAll",
                "iam:ListRoles",
                "iam:ListUsers",
```

```
                "aoss:BatchGetCollection",
                "aoss:BatchGetLifecyclePolicy",
                "es:ListApplications"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchLogsOpenSearchCreateServiceLinkedAccess",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
opensearchservice.amazonaws.com/AWSServiceRoleForAmazonOpenSearchService",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "opensearchservice.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsObservabilityCreateServiceLinkedAccess",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/*/
AWSServiceRoleForAmazonOpenSearchServerless",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "observability.aoss.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsCollectionRequestAccess",
            "Effect": "Allow",
            "Action": [
                "aoss:CreateCollection"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CloudWatchOpenSearchIntegration": [
```

```
                    "Dashboards"
                ]
            },
            "ForAllValues:StringEquals": {
                "aws:TagKeys": "CloudWatchOpenSearchIntegration"
            }
        }
    },
    {
        "Sid": "CloudWatchLogsApplicationRequestAccess",
        "Effect": "Allow",
        "Action": [
            "es:CreateApplication"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/OpenSearchIntegration": [
                    "Dashboards"
                ]
            },
            "ForAllValues:StringEquals": {
                "aws:TagKeys": "OpenSearchIntegration"
            }
        }
    },
    {
        "Sid": "CloudWatchLogsCollectionResourceAccess",
        "Effect": "Allow",
        "Action": [
            "aoss:DeleteCollection"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                    "Dashboards"
                ]
            }
        }
    },
    {
        "Sid": "CloudWatchLogsApplicationResourceAccess",
        "Effect": "Allow",
```

```
            "Action": [
                "es:UpdateApplication",
                "es:GetApplication"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/OpenSearchIntegration": [
                        "Dashboards"
                    ]
                }
            }
        },
        {
            "Sid": "CloudWatchLogsCollectionPolicyAccess",
            "Effect": "Allow",
            "Action": [
                "aoss:CreateSecurityPolicy",
                "aoss:CreateAccessPolicy",
                "aoss:DeleteAccessPolicy",
                "aoss:DeleteSecurityPolicy",
                "aoss:GetAccessPolicy",
                "aoss:GetSecurityPolicy",
                "aoss:APIAccessAll"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aoss:collection": "logs-collection-*"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsIndexPolicyAccess",
            "Effect": "Allow",
            "Action": [
                "aoss:CreateAccessPolicy",
                "aoss:DeleteAccessPolicy",
                "aoss:GetAccessPolicy",
                "aoss:CreateLifecyclePolicy",
                "aoss:DeleteLifecyclePolicy"
            ],
            "Resource": "*",
            "Condition": {
```

```
                "StringLike": {
                    "aoss:index": "logs-collection-*"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsStartDirectQueryAccess",
            "Effect": "Allow",
            "Action": [
                "opensearch:StartDirectQuery"
            ],
            "Resource": "arn:aws:opensearch:*:*:datasource/logs_datasource_*"
        },
        {
            "Sid": "CloudWatchLogsDQSRequestQueryAccess",
            "Effect": "Allow",
            "Action": [
                "es:AddDirectQueryDataSource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ]
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "CloudWatchOpenSearchIntegration"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsDQSResourceQueryAccess",
            "Effect": "Allow",
            "Action": [
                "es:GetDirectQueryDataSource",
                "es:DeleteDirectQueryDataSource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ]
```

```
                }
            }
        },
        {
            "Sid": "CloudWatchLogsPassRoleAccess",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService":
"directquery.opensearchservice.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsAossTagsAccess",
            "Effect": "Allow",
            "Action": [
                "aoss:TagResource",
                "es:AddTags"
            ],
            "Resource": "arn:aws:aoss:*:*:collection/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ]
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "CloudWatchOpenSearchIntegration"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsEsApplicationTagsAccess",
            "Effect": "Allow",
            "Action": [
                "es:AddTags"
            ],
            "Resource": "arn:aws:opensearch:*:*:application/*",
            "Condition": {
```

```
                "StringEquals": {
                    "aws:ResourceTag/OpenSearchIntegration": [
                        "Dashboards"
                    ]
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "OpenSearchIntegration"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsEsDataSourceTagsAccess",
            "Effect": "Allow",
            "Action": [
                "es:AddTags"
            ],
            "Resource": "arn:aws:opensearch:*:*:datasource/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ]
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "CloudWatchOpenSearchIntegration"
                }
            }
        }
    ]
}
```

[CloudWatchLogsReadOnlyAccess]

CloudWatchLogsReadOnlyAccess ポリシーは、CloudWatch Logs への読み取り専用のアクセス権限を付与します。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、cloudwatch:GenerateQuery アクセス許可が含まれています。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": "*"
        }
    ]
}
```

CloudWatchOpenSearchDashboardsFullAccess

CloudWatchOpenSearchDashboardsFullAccess ポリシーは、OpenSearch Service との統合を作成、管理、削除し、それらの統合で提供されたログダッシュボードを作成および管理するためのアクセスを許可します。詳細については、「Amazon OpenSearch Service で分析する」を参照してください。

内容は次のとおりです。

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Sid": "CloudWatchOpenSearchDashboardsIntegration",
            "Effect": "Allow",
            "Action": [
                "logs:ListIntegrations",
                "logs:GetIntegration",
                "logs:DeleteIntegration",
                "logs:PutIntegration",
                "logs:DescribeLogGroups",
                "opensearch:ApplicationAccessAll",
                "iam:ListRoles",
                "iam:ListUsers"
            ],
            "Resource": "*"
        },
```

```
        {
            "Sid": "CloudWatchLogsOpensearchReadAPIs",
            "Effect": "Allow",
            "Action": [
                "aoss:BatchGetCollection",
                "aoss:BatchGetLifecyclePolicy",
                "es:ListApplications"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsOpensearchCreateServiceLinkedAccess",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
opensearchservice.amazonaws.com/AWSServiceRoleForAmazonOpenSearchService",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "opensearchservice.amazonaws.com",
                    "aws:CalledViaFirst": "logs.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsObservabilityCreateServiceLinkedAccess",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
observability.aoss.amazonaws.com/AWSServiceRoleForAmazonOpenSearchServerless",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "observability.aoss.amazonaws.com",
                    "aws:CalledViaFirst": "logs.amazonaws.com"
                }
            }
```

```
        },
        {
            "Sid": "CloudWatchLogsCollectionRequestAccess",
            "Effect": "Allow",
            "Action": [
                "aoss:CreateCollection"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
                    "aws:RequestTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ]
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "CloudWatchOpenSearchIntegration"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsApplicationRequestAccess",
            "Effect": "Allow",
            "Action": [
                "es:CreateApplication"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
                    "aws:RequestTag/OpenSearchIntegration": [
                        "Dashboards"
                    ]
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "OpenSearchIntegration"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsCollectionResourceAccess",
            "Effect": "Allow",
            "Action": [
                "aoss:DeleteCollection"
```

```
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
                    "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ]
                }
            }
        },
        {
            "Sid": "CloudWatchLogsApplicationResourceAccess",
            "Effect": "Allow",
            "Action": [
                "es:UpdateApplication",
                "es:GetApplication"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
                    "aws:ResourceTag/OpenSearchIntegration": [
                        "Dashboards"
                    ]
                }
            }
        },
        {
            "Sid": "CloudWatchLogsCollectionPolicyAccess",
            "Effect": "Allow",
            "Action": [
                "aoss:CreateSecurityPolicy",
                "aoss:CreateAccessPolicy",
                "aoss:DeleteAccessPolicy",
                "aoss:DeleteSecurityPolicy",
                "aoss:GetAccessPolicy",
                "aoss:GetSecurityPolicy"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aoss:collection": "cloudwatch-logs-*",
                    "aws:CalledViaFirst": "logs.amazonaws.com"
```

```
                }
            }
        },
        {
            "Sid": "CloudWatchLogsAPIAccessAll",
            "Effect": "Allow",
            "Action": [
                "aoss:APIAccessAll"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aoss:collection": "cloudwatch-logs-*"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsIndexPolicyAccess",
            "Effect": "Allow",
            "Action": [
                "aoss:CreateAccessPolicy",
                "aoss:DeleteAccessPolicy",
                "aoss:GetAccessPolicy",
                "aoss:CreateLifecyclePolicy",
                "aoss:DeleteLifecyclePolicy"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aoss:index": "cloudwatch-logs-*",
                    "aws:CalledViaFirst": "logs.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsDQSRequestQueryAccess",
            "Effect": "Allow",
            "Action": [
                "es:AddDirectQueryDataSource"
            ],
            "Resource": "arn:aws:opensearch:*:*:datasource/cloudwatch_logs_*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
```

```
                    "aws:RequestTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ]
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "CloudWatchOpenSearchIntegration"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsStartDirectQueryAccess",
            "Effect": "Allow",
            "Action": [
                "opensearch:StartDirectQuery",
                "opensearch:GetDirectQuery"
            ],
            "Resource": "arn:aws:opensearch:*:*:datasource/cloudwatch_logs_*"
        },
        {
            "Sid": "CloudWatchLogsDQSResourceQueryAccess",
            "Effect": "Allow",
            "Action": [
                "es:GetDirectQueryDataSource",
                "es:DeleteDirectQueryDataSource"
            ],
            "Resource": "arn:aws:opensearch:*:*:datasource/cloudwatch_logs_*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
                    "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ]
                }
            }
        },
        {
            "Sid": "CloudWatchLogsPassRoleAccess",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
```

```
                    "iam:PassedToService":
"directquery.opensearchservice.amazonaws.com",
                    "aws:CalledViaFirst": "logs.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsAossTagsAccess",
            "Effect": "Allow",
            "Action": [
                "aoss:TagResource",
                "es:AddTags"
            ],
            "Resource": "arn:aws:aoss:*:*:collection/*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
                    "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ]
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "CloudWatchOpenSearchIntegration"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsEsApplicationTagsAccess",
            "Effect": "Allow",
            "Action": [
                "es:AddTags"
            ],
            "Resource": "arn:aws:opensearch:*:*:application/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/OpenSearchIntegration": [
                        "Dashboards"
                    ],
                    "aws:CalledViaFirst": "logs.amazonaws.com"
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "OpenSearchIntegration"
                }
            }
```

```
        },
        {
            "Sid": "CloudWatchLogsEsDataSourceTagsAccess",
            "Effect": "Allow",
            "Action": [
                "es:AddTags"
            ],
            "Resource": "arn:aws:opensearch:*:*:datasource/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ],
                    "aws:CalledViaFirst": "logs.amazonaws.com"
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "CloudWatchOpenSearchIntegration"
                }
            }
        }
    ]
}
```

CloudWatchOpenSearchDashboardAccess

CloudWatchOpenSearchDashboardAccess ポリシーは、 Amazon OpenSearch Service 分析で作成された公開ログダッシュボードを表示するアクセスを許可します。詳細については、「Amazon OpenSearch Service で分析する」を参照してください。

> **Important**
>
> このポリシーの付与に加えて、ロールまたはユーザーが提供されたログダッシュボードを表示できるようにするには、OpenSearch Service との統合を作成するときにそれらも指定する必要があります。詳細については、「ステップ 1: OpenSearch Service との統合を作成する」を参照してください。

CloudWatchOpenSearchDashboardAccess の内容は次のとおりです。

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

```
            "Sid": "CloudWatchOpenSearchDashboardsIntegration",
            "Effect": "Allow",
            "Action": [
                "logs:ListIntegrations",
                "logs:GetIntegration",
                "logs:DescribeLogGroups",
                "opensearch:ApplicationAccessAll",
                "iam:ListRoles",
                "iam:ListUsers"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchLogsOpensearchReadAPIs",
            "Effect": "Allow",
            "Action": [
                "aoss:BatchGetCollection",
                "aoss:BatchGetLifecyclePolicy",
                "es:ListApplications"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsAPIAccessAll",
            "Effect": "Allow",
            "Action": [
                "aoss:APIAccessAll"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aoss:collection": "cloudwatch-logs-*"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsDQSCollectionPolicyAccess",
            "Effect": "Allow",
            "Action": [
```

```
                "aoss:GetAccessPolicy",
                "aoss:GetSecurityPolicy"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
                    "aoss:collection": "cloudwatch-logs-*"
                }
            }
        },
        {
            "Sid": "CloudWatchLogsApplicationResourceAccess",
            "Effect": "Allow",
            "Action": [
                "es:GetApplication"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
                    "aws:ResourceTag/OpenSearchIntegration": [
                        "Dashboards"
                    ]
                }
            }
        },
        {
            "Sid": "CloudWatchLogsDQSResourceQueryAccess",
            "Effect": "Allow",
            "Action": [
                "es:GetDirectQueryDataSource"
            ],
            "Resource": "arn:aws:opensearch:*:*:datasource/cloudwatch_logs_*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "logs.amazonaws.com",
                    "aws:ResourceTag/CloudWatchOpenSearchIntegration": [
                        "Dashboards"
                    ]
                }
            }
        },
        {
```

```
            "Sid": "CloudWatchLogsDirectQueryStatusAccess",
            "Effect": "Allow",
            "Action": [
                "opensearch:GetDirectQuery"
            ],
            "Resource": "arn:aws:opensearch:*:*:datasource/cloudwatch_logs_*"
        }
    ]
}
```

CloudWatchLogsCrossAccountSharingConfiguration

CloudWatchCrossAccountSharingConfiguration 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、および表示するためのアクセス許可を付与します。詳細については、「CloudWatch の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/*"
            ]
        }
    ]
}
```

AWS マネージドポリシーに対する CloudWatch Logs の更新

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 Logs 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入手するには、CloudWatch Logs ドキュメントの履歴ページから、RSS フィードにサブスクライブしてください。

変更	説明	日付
CloudWatchLogsFullAccess — 既存のポリシーへの更新	CloudWatch Logs が、CloudWatchLogsFullAccess に対するアクセス許可を追加しました。 一部の機能で CloudWatch Logs と OpenSearch Service の統合を有効にするために、Amazon OpenSearch Service および IAM のアクセス許可が追加されました。	2024 年 12 月 1 日
CloudWatchOpenSearchDashboardsFullAccess – 新しい IAM ポリシー。	CloudWatch Logs に新しい IAM ポリシー CloudWatchOpenSearchDashboardsFullAccess が追加されました。- このポリシーは、O	2024 年 12 月 1 日

変更	説明	日付
	penSearch Service との統合を作成、管理、削除し、それらの統合で提供されたログダッシュボードを作成、管理、削除するアクセス許可を付与します。詳細については、「Amazon OpenSearch Service で分析する」を参照してください。	
CloudWatchOpenSearchDashboardAccess – 新しい IAM ポリシー。	CloudWatch Logs に、新しい IAM ポリシーである CloudWatchOpenSearchDashboardAccess が追加されました。- このポリシーは、によって提供されるログダッシュボードを表示するアクセス許可を付与します Amazon OpenSearch Service。詳細については、「Amazon OpenSearch Service で分析する」を参照してください。	2024 年 12 月 1 日

変更	説明	日付
CloudWatchLogsFullAccess — 既存のポリシーへの更新	CloudWatch Logs が、CloudWatchLogsFullAccess にアクセス許可を追加しました。 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、`cloudwatch:GenerateQuery` アクセス許可を追加しました。	2023 年 11 月 27 日
CloudWatchLogsReadOnlyAccess — 既存のポリシーへの更新。	CloudWatch が、CloudWatchLogsReadOnlyAccess にアクセス許可を追加しました。 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、`cloudwatch:GenerateQuery` アクセス許可を追加しました。	2023 年 11 月 27 日

変更	説明	日付
CloudWatchLogsReadOnlyAccess — 既存のポリシーへの更新	CloudWatch Logs が、CloudWatchLogsReadOnlyAccess に対するアクセス許可を追加しました。 `logs:StartLiveTail` および `logs:StopLiveTail` のアクセス権限が追加されたため、このポリシーを持つユーザーは、コンソールを使用して CloudWatch Logs の Live Tail セッションを開始および停止できます。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
CloudWatchCrossAccountSharingConfiguration – 新しいポリシー	CloudWatch Logs が、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のリンクを管理できるようにする新しいポリシーを追加しました。 詳細については、「CloudWatch の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 — 既存のポリシーへの更新	CloudWatch Logs が、CloudWatchLogsReadOnlyAccess に対するアクセス許可を追加しました。 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 のクロスアカウントオブザーバビリティでソースアカウントから共有されたデータを表示できるようにするための `oam:ListSinks` および `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ためのアクセス権限を許可することができます。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例を示しています。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 Logs へのフルアクセスを許可する
- 例 2: CloudWatch Logs 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 Logs へのフルアクセスを許可する

以下のポリシーでは、ユーザーにすべての CloudWatch Logs アクションへのアクセスを許可します。

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

例 2: CloudWatch Logs への読み取り専用アクセスを許可する

AWS は CloudWatchLogsReadOnlyAccess ポリシーを提供し、CloudWatch Logs データへの読み取り専用アクセスを有効にします。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特定の CloudWatch Logs Insights クエリなど、リソースとして分類されていないものでのみ動作します。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 の許可リファレンス

アクセスコントロール をセットアップし、IAM アイデンティティ (アイデンティティベースのポリシー) にアタッチできるアクセス権限ポリシーを作成する際、以下の表をリファレンスとして使用できます。この表には、各 CloudWatch Logs API オペレーション、およびその実行のためのアクセス権限を付与できる対応するアクションを示しています。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、IAM ユーザーガイドのAWS 「グローバルおよび 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。たとえば、`logs:CreateLogGroup`、`logs:CreateLogStream`、または `logs:*` (すべての CloudWatch Logs アクションの場合)。

CloudWatch Logs API オペレーションおよびアクションに必要な許可

CloudWatch Logs API のオペレーション	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 Logs API のオペレーション	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のに必要です。
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 Logs API のオペレーション	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 で保存されたクエリ定義のリストを表示するために必要です。

CloudWatch Logs API のオペレーション	必要なアクセス許可 (API アクション)
DescribeQueries	`logs:DescribeQueries` スケジュールされた、実行された、または最近実行された 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 Logs API のオペレーション	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。
ListEntitiesForLogGroup (CloudWatch コンソールのみのアクセス許可)	`logs:ListEntitiesForLogGroup` ロググループに関連付けられているエンティティを検索するために必要です。CloudWatch コンソール内の関連ログを調べるために必要です。
ListLogGroupsForEntity (CloudWatch コンソールのみのアクセス許可)	`logs:ListLogGroupsForEntity` エンティティに関連付けられたロググループを見つけるために必要です。CloudWatch コンソール内の関連ログを調べるために必要です。
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。

CloudWatch Logs API のオペレーション	必要なアクセス許可 (API アクション)
PutMetricFilter	`logs:PutMetricFilter` メトリクスフィルタの作成や更新、またはそれを口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 リソースポリシーを作成するために必要です。
PutRetentionPolicy	`logs:PutRetentionPolicy` ロググループでログイベント (保持) を維持する日数を設定するのに必要です。
PutSubscriptionFilter	`logs:PutSubscriptionFilter` サブスクリプションフィルタの作成や更新、またはそれを口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。

CloudWatch Logs API のオペレーション	必要なアクセス許可 (API アクション)
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 Logs 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 CloudWatch Logs によって事前定義されており、サービスがユーザーに代わって他の AWS 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では、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 Logs の設定が効率的になります。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。特に定義されていない限り、CloudWatch Logs のみがこれらのロールを引き受けることができます。定義されたアクセス権限には、信頼ポリシーとアクセス権限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許可

CloudWatch Logs は AWSServiceRoleForLogDelivery というサービスにリンクされたロールを使用します。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、指定したリソースに対して以下のアクションを完了することを CloudWatch Logs に許可します。

- アクション: 値が `True` の `LogDeliveryEnabled` キーを持つタグが付いたすべての Firehose ストリーム上で `firehose:PutRecordBatch` および `firehose:PutRecord`。このタグは、ログを Firehose に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するように再度ログを設定すると、CloudWatch Logs によってサービスにリンクされたロールが再び作成されます。

CloudWatch Logs のサービスにリンクされたロールの編集

CloudWatch Logs では、ロールを作成した後に、AWSServiceRoleForLogDelivery 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「サービスにリンクされたロールの編集」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を必要とする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、サービスにリンクされた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**
>
> リソースを削除する際に、CloudWatch Logs サービスでそのロールが使用されている場合、削除は失敗することがあります。失敗した場合は数分待ってから操作を再試行してください。

AWSServiceRoleForLogDelivery サービスにリンクされたロールによって使用されている 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。

サービスリンクロールを IAM で手動削除するには

IAM コンソール、 AWS CLI、または AWS API を使用して、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 のサービスにリンクされたロールでサポートされるリージョン

CloudWatch Logs 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ています。詳細については、「CloudWatch Logs 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

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、AWS のサービス「コンプライアンスプログラムによる対象範囲内」を参照して、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「Downloading Reports in AWS Artifact 」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。 は、コンプライアンスに役立つ以下のリソース AWS を提供します。

- セキュリティのコンプライアンスとガバナンス – これらのソリューション実装ガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスの機能をデプロイする手順を示します。
- HIPAA 対応サービスのリファレンス – HIPAA 対応サービスの一覧が提供されています。すべて AWS のサービス HIPAA の対象となるわけではありません。
- 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、お客様の業界や地域に適用される場合があります。
- AWS カスタマーコンプライアンスガイド – コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) など) にわたるセキュリティコントロールを保護し、そのガイダンスに AWS のサービス マッピングするためのベストプラクティスをまとめています。
- 「 デベロッパーガイド」の「ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が内部プラクティス、業界ガイドライン、および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 – これにより AWS のサービス 、 内のセキュリティ状態を包括的に把握できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールの一覧については、Security Hub のコントロールリファレンスを参照してください。
- Amazon GuardDuty – 環境をモニタリングして AWS アカウント、疑わしいアクティビティや悪意のあるアクティビティがないか調べることで、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS のサービス を検出します。GuardDuty を使用すると、特定のコンプライアンスフレームワークで義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件に対応できます。
- AWS Audit Manager – これにより AWS のサービス 、 AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

Amazon CloudWatch Logs の耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョンとアベイラビリティーゾーンを中心に構築されています。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が高く、フォールトトレラントで、スケーラブルです。

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 グローバル AWS 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 フレームワーク」の「インフラストラクチャの保護」を参照してください。

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

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 Service (AWS 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 Logs の使用

Amazon Virtual Private Cloud (Amazon VPC) を使用して AWS リソースをホストする場合は、VPC と CloudWatch Logs の間にプライベート接続を確立できます。この接続を使用して、インターネットを経由して送信せずに CloudWatch Logs にログを送信できます。CloudWatch Logs は、すべてのリージョンで IPv4 VPC エンドポイントをサポートし、アジアパシフィック (マレーシア) 、アジアパシフィック (タイ) 、メキシコ (中部) を除くすべてのリージョンで IPv6 エンドポイントをサポートします。

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 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、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、プライベート IP アドレスを持つ Elastic Network Interface 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 AWS PrivateLink を利用しています。詳細については、「New – AWS PrivateLink for AWS Services」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 Logs 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、CloudWatch Logs のインターフェイス VPC エンドポイントを作成します。選択するサービスは、[com.amazonaws.***Region***.logs] です。CloudWatch Logs のあらゆる設定を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 との間の接続のテスト

エンドポイントの作成が完了したら、接続をテストできます。

VPC と CloudWatch Logs のエンドポイント間の接続をテストするには

1. VPC にある Amazon EC2 インスタンスに接続します。接続の詳細については、Amazon EC2 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、 を使用して AWS CLI 、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 の 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 IAM リソースポリシーです。エンドポイントの作成時にポリシーをアタッチしない場合、サービスへのフルアクセスを許可するデフォルトのポリシーがアタッチされます。エンドポイントポリシーは、IAM 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 Logs のエンドポイントポリシーの例を次に示します。このポリシーでは、VPC を介して CloudWatch Logs に接続するユーザーはログストリームを作成してログを CloudWatch Logs に送信できますが、他の CloudWatch Logs アクションは実行できません。

```
{
  "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 コンソールの https://console.aws.amazon.com/vpc/ を開いてください。
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 Logs のエンドポイントをまだ作成していない場合は、[Create Endpoint (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 Logs では、特定の VPC または特定の VPC エンドポイントへのアクセスを制限できる `aws:SourceVpc` および `aws:SourceVpce` コンテキストキーをサポートしています。これらのキーは、ユーザーが VPC エンドポイントを使用している場合にのみ使用できます。詳細については、「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、ユーザー、ロール、または AWS のサービスが実行したアクションの記録を提供するサービスである AWS CloudTrail と統合されています。CloudTrail は、CloudWatch Logs の API コールをイベントとしてキャプチャします。キャプチャされたコールには、CloudWatch Logs コンソールからの呼び出しと、CloudWatch Logs API オペレーションへのコード呼び出しが含まれます。CloudTrail で収集した情報を使用して、CloudWatch Logs に対するリクエスト、リクエスト元の IP アドレス、リクエストの作成日時、その他の詳細を確認できます。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- ルートユーザーまたはユーザー認証情報のどちらを使用してリクエストが送信されたか。
- リクエストが IAM Identity Center ユーザーに代わって行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS のサービスによって送信されたかどうか。

CloudTrail は、アカウント AWS アカウント を作成すると でアクティブになり、CloudTrail イベント履歴に自動的にアクセスできます。CloudTrail の [イベント履歴] では、 AWS リージョンで過去 90 日間に記録された 管理イベントの表示、検索、およびダウンロードが可能で、変更不可能な記録を確認できます。詳細については、「AWS CloudTrail ユーザーガイド」の「CloudTrail イベント履歴の使用」を参照してください。[イベント履歴] の閲覧には CloudTrail の料金はかかりません。

AWS アカウント 過去 90 日間のイベントの継続的な記録については、証跡または CloudTrail Lake イベントデータストアを作成します。

CloudTrail 証跡

追跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。を使用して作成された証跡はすべてマルチリージョン AWS Management Console です。 AWS CLIを使用する際は、単一リージョンまたは複数リージョンの証跡を作成できます。 AWS リージョン アカウントのすべての でアクティビティをキャプチャするため、マルチリージョン証跡を作成することをお勧めします。単一リージョンの証跡を作成する場合、証跡の AWS リージョンに記録された

イベントのみを表示できます。証跡の詳細については、「AWS CloudTrail ユーザーガイド」の「AWS アカウントの証跡の作成」および「組織の証跡の作成」を参照してください。

証跡を作成すると、進行中の管理イベントのコピーを 1 つ無料で CloudTrail から Amazon S3 バケットに配信できますが、Amazon S3 ストレージには料金がかかります。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。Amazon S3 の料金に関する詳細については、「Amazon S3 の料金」を参照してください。

CloudTrail Lake イベントデータストア

[CloudTrail Lake] を使用すると、イベントに対して SQL ベースのクエリを実行できます。CloudTrail Lake は、行ベースの JSON 形式の既存のイベントを Apache ORC 形式に変換します。ORC は、データを高速に取得するために最適化された単票ストレージ形式です。イベントは、イベントデータストアに集約されます。イベントデータストアは、高度なイベントセレクタを適用することによって選択する条件に基づいた、イベントのイミュータブルなコレクションです。どのイベントが存続し、クエリに使用できるかは、イベントデータストアに適用するセレクタが制御します。CloudTrail Lake の詳細については、「 AWS CloudTrail ユーザーガイド」の「Working with AWS CloudTrail Lake」を参照してください。

CloudTrail Lake のイベントデータストアとクエリにはコストがかかります。イベントデータストアを作成する際に、イベントデータストアに使用する料金オプションを選択します。料金オプションによって、イベントの取り込みと保存にかかる料金、および、そのイベントデータストアのデフォルトと最長の保持期間が決まります。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。

CloudWatch Logs は、CloudTrail ログファイルのイベントとして以下のアクションを記録します。

- CancelExportTask
- CreateExportTask
- CreateLogGroup
- CreateLogStream
- DeleteDestination
- DeleteLogGroup
- DeleteLogStream
- DeleteMetricFilter
- DeleteRetentionPolicy
- DeleteSubscriptionFilter

- PutDestination
- PutDestinationPolicy
- PutMetricFilter
- PutResourcePolicy
- PutRetentionPolicy
- PutSubscriptionFilter
- StartQuery
- StopQuery
- TestMetricFilter

次の CloudWatch Logs API アクションでは、リクエスト要素のみが CloudTrail に記録されます。

- 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 要素を参照してください。

CloudTrail でのクエリ生成情報

クエリジェネレーターコンソールイベントの CloudTrail 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 CloudWatch Metric Insights でサポートされています。これらの CloudTrail イベントでは、`eventSource` は `monitoring.amazonaws.com` です。

次の例は、CloudWatch Logs Insights の GenerateQuery アクションを示す CloudTrail 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の理解

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

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

```
{
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/someuser",
            "accountId": "123456789012",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "someuser"
        },
        "eventTime": "2016-02-08T06:35:14Z",
```

```
        "eventSource": "logs.amazonaws.com",
        "eventName": "CreateExportTask",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
        "requestParameters": {
            "destination": "yourdestination",
            "logGroupName": "yourloggroup",
            "to": 123456789012,
            "from": 0,
            "taskName": "yourtask"
        },
        "responseElements": {
            "taskId": "15e5e534-9548-44ab-a221-64d9d2b27b9b"
        },
        "requestID": "1cd74c1c-ce2e-12e6-99a9-8dbb26bd06c9",
        "eventID": "fd072859-bd7c-4865-9e76-8e364e89307c",
        "eventType": "AwsApiCall",
        "apiVersion": "20140328",
        "recipientAccountId": "123456789012"
}
```

CloudWatch Logs エージェントのリファレンス

> **Important**
>
> このセクションは、廃止された古い CloudWatch Logs エージェントを使用している方のためのリファレンスです。インスタンスメタデータサービスのバージョン 2 (IMDSv2) を使用している場合は、新しい統合 CloudWatch エージェントを使用する必要があります。ただし、IMDSv2 を使用していない場合でも、古い CloudWatch Logs エージェントではなく、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。新しい統合エージェントの詳細については、「Collecting metrics and logs from Amazon EC2 instance and on-premises servers with the CloudWatch agent」を参照してください。廃止された 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、「Create the CloudWatch agent configuration file with the wizard」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 Logs 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。
- データを CloudWatch Logs にプッシュするプロセスを開始するスクリプト (デーモン)。
- デーモンが常に実行中であることを確認する cron ジョブ。

エージェント設定ファイル

CloudWatch Logs エージェント設定ファイルには、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。このファイルは、Python の設定ファイル形式 (https://docs.python.org/2/library/logging.config.html#logging-config-fileformat) です。以下の名前のロガーはカスタマイズできます。

```
cwlogs.push
cwlogs.push.reader
cwlogs.push.publisher
cwlogs.push.event
cwlogs.push.batch
cwlogs.push.stream
cwlogs.push.watcher
```

以下のサンプルは、デフォルトの値が INFO であるリーダーとパブリッシャーのレベルを WARNING に変更します。

```
[loggers]
keys=root,cwlogs,reader,publisher

[handlers]
keys=consoleHandler

[formatters]
keys=simpleFormatter

[logger_root]
level=INFO
handlers=consoleHandler

[logger_cwlogs]
level=INFO
handlers=consoleHandler
qualname=cwlogs.push
propagate=0

[logger_reader]
level=WARNING
handlers=consoleHandler
qualname=cwlogs.push.reader
propagate=0

[logger_publisher]
level=WARNING
handlers=consoleHandler
qualname=cwlogs.push.publisher
propagate=0

[handler_consoleHandler]
class=logging.StreamHandler
level=INFO
formatter=simpleFormatter
args=(sys.stderr,)

[formatter_simpleFormatter]
format=%(asctime)s - %(name)s - %(levelname)s - %(process)d - %(threadName)s -
 %(message)s
```

use_gzip_http_content_encoding

true (デフォルト) に設定すると、CloudWatch Logs への圧縮されたペイロードの送信に対して、gzip による HTTP コンテンツのエンコードが有効になります。これにより、CPU の使用率が減り、NetworkOut が少なくなり、Put のレイテンシーが短くなります。この機能を無効にするには、CloudWatch Logs エージェント設定ファイルの [general] (全般) セクションに [use_gzip_http_content_encoding = false] を追加してから、エージェントを再起動します。

> **Note**
>
> この設定は awscli-cwlogs バージョン 1.3.3 以降でのみ使用できます。

log_group_name

送信先ロググループを指定します。ロググループが存在しない場合には、自動的に作成されます。ロググループの名前は 1～512 文字で指定します。ここで使えるのは、a～z、A～Z、0～9、"_" (アンダーバー)、"-" (ハイフン)、"/" (スラッシュ) および "." (ピリオド) です。

log_stream_name

送信先ログストリームを指定します。リテラル文字列、定義済み変数 ({instance_id}、{hostname}、{ip_address})、またはこれらの組み合わせを使用して、ログストリーム名を定義できます。ログストリームが存在しない場合には、自動的に作成されます。

datetime_format

ログからタイムスタンプを入手する方法を指定します。タイムスタンプはログイベントを取得し、メトリクスを生成するために使用されます。現在の時刻は、[datetime_format] が提供されていない場合に各ログイベントで使用されます。提供された [datetime_format] の値がそのログメッセージに対して無効の場合は、適切に解析されたタイムスタンプを持つ最後のログイベントのタイムスタンプが使用されます。以前のログイベントが存在しない場合は、現在の時刻が使用されます。

一般的な datetime_format コードは次のとおりです。Python がサポートする datetime_format コード (datetime.strptime()) も使用できます。タイムゾーンオフセット (%z) もサポートされています。ただし、Python 3.2 までは、コロン (:) のない [+-] HHMM はサポートされていません。詳細については、「strftime() および strptime() Behavior ()」を参照してください。

[%y]: ゼロ詰め 10 進数での年 (世紀なし) です。00, 01, ..., 99

%Y: 10 進数での年 (世紀あり) です。1970、1988、2001、2013

%b: ロケールの省略名称での月です。Jan、Feb ... Dec (en_US) ;

%B: ロケールの正式名称での月です。January、February...December (en_US) ;

%m: ゼロ詰め 10 進数での月です。01, 02, ..., 12

%d: ゼロ詰め 10 進数での日です。01, 02, ..., 31

%H: ゼロ詰め 10 進数での時 (24 時間形式の時計) です。00, 01, ..., 23

%I: ゼロ詰め 10 進数での時 (12 時間形式の時計) です。01, 02, ..., 12

%p: ロケールで AM または PM に相当するものです。

%M: ゼロ詰め 10 進数での分です。00, 01, ..., 59

%S: ゼロ詰め 10 進数での秒です。00, 01, ..., 59

%f: 左ゼロ詰め 10 進数でのマイクロ秒です。000000, ..., 999999

%z: +HHMM または -HHMM 形式の UTC オフセットです。+0000、-0400、+1030

形式の例:

```
Syslog: '%b %d %H:%M:%S', e.g. Jan 23 20:59:29

Log4j: '%d %b %Y %H:%M:%S', e.g. 24 Jan 2014 05:00:00

ISO8601: '%Y-%m-%dT%H:%M:%S%z', e.g. 2014-02-20T05:20:20+0000
```

time_zone

ログイベントのタイムスタンプのタイムゾーンを指定します。サポートされる 2 つの値は UTC および LOCAL です。デフォルトは LOCAL です。タイムゾーンが [datetime_format] に基づいて推定できない場合に使用されます。

ファイル

CloudWatch Logs にプッシュするログファイルを指定します。ファイルは、特定のファイルまたは複数のファイルを指すことができます (/var/log/system.log* のようにワイルドカードを使用) 。ファイルの変更時間に基づいて、最新のファイルのみが CloudWatch Logs にプッシュされます。access_log.2014-06-01-01 と access_log.2014-06-01-02 など同じ形式の一連のファイルを指定するにはワイルドカードの使用をお勧めします。ただし、access_log_80 と access_log_443 のように複数の種類のファイルには使用しないでください。複数の種類のファイルを指定するに

は、設定ファイルに別のストリームログのエントリを追加して、各種類のログファイルが異なるログストリームに行くようにします。圧縮ファイルはサポートされていません。

file_fingerprint_lines

ファイルを識別するための行範囲を指定します。有効な値は「1」「2-5」のように単一の数字またはハイフンで区切られた 2 つの数字です。デフォルト値は「1」です。最初の行を使用してフィンガープリントを計算します。指定された行がすべて存在しない限り、フィンガープリント行は CloudWatch Logs に送信されません。

multi_line_start_pattern

ログメッセージの開始を識別するパターンを指定します。ログメッセージは、パターンに一致する 1 行と、それに続くパターンに一致しない行で構成されます。有効な値は正規表現または {datetime_format} です。{datetime_format} を使用する場合は、datetime_format オプションを指定する必要があります。デフォルト値は「^[^\s]」です。よって、空白文字以外の文字で始まる行で前のログメッセージを終了し、新しいログメッセージを開始します。

initial_position

データの読み出しをどこから開始するかを指定します (start_of_file または end_of_file)。デフォルトは start_of_file です。そのログストリームに保持されている状態がない場合にのみ使用されます。

encoding

ファイルを正しく読み込むことができるように、ログファイルのエンコードを指定します。デフォルトは utf_8 です。Python の codecs.decode() がサポートするエンコードを使用できます。

> **Warning**
> 正しくないエンコードを指定すると、デコードできない文字がそのほかの文字に置き換えられるため、データ損失が生じる可能性があります。

一般的なエンコードを次に示します。

```
ascii, big5, big5hkscs, cp037, cp424, cp437, cp500, cp720, cp737,
cp775, cp850, cp852, cp855, cp856, cp857, cp858, cp860, cp861, cp862,
cp863, cp864, cp865, cp866, cp869, cp874, cp875, cp932, cp949, cp950,
cp1006, cp1026, cp1140, cp1250, cp1251, cp1252, cp1253, cp1254, cp1255,
cp1256, cp1257, cp1258, euc_jp, euc_jis_2004, euc_jisx0213, euc_kr,
```

```
gb2312, gbk, gb18030, hz, iso2022_jp, iso2022_jp_1, iso2022_jp_2,
iso2022_jp_2004, iso2022_jp_3, iso2022_jp_ext, iso2022_kr, latin_1,
iso8859_2, iso8859_3, iso8859_4, iso8859_5, iso8859_6, iso8859_7,
iso8859_8, iso8859_9, iso8859_10, iso8859_13, iso8859_14, iso8859_15,
iso8859_16, johab, koi8_r, koi8_u, mac_cyrillic, mac_greek, mac_iceland,
mac_latin2, mac_roman, mac_turkish, ptcp154, shift_jis, shift_jis_2004,
shift_jisx0213, utf_32, utf_32_be, utf_32_le, utf_16, utf_16_be,
utf_16_le, utf_7, utf_8, utf_8_sig
```

buffer_duration

ログイベントのバッチ期間を指定します。最小値は 5000ms で、デフォルト値は 5000ms です。

batch_count

バッチのログイベントの最大値を 10000 までの値で指定します。デフォルト値は 10000 です。

batch_size

バッチのログイベントの最大値を 1048576 バイトまでのバイト値で指定します。デフォルト値は 1048576 バイトです。このサイズは、UTF-8 のすべてのイベントメッセージの合計に各ログイベントにつき 26 バイトを加算して計算されます。

HTTP プロキシでの CloudWatch Logs エージェントの使用

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 エージェントを新たにインストールする場合は、以下のコマンドを実行します。

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

```
sudo python awslogs-agent-setup.py --region us-east-1 --http-proxy http://your/
proxy --https-proxy http://your/proxy --no-proxy 169.254.169.254
```

EC2 インスタンスで Amazon EC2 メタデータサービスへのアクセスを維持するには、[--no-proxy 169.254.169.254](推奨) を使用します。詳細については、「Amazon EC2 ユーザーガイド」の「Instance Metadata and User Data」を参照してください。

`http-proxy` および `https-proxy` の値で、URL 全体を指定します。

b. CloudWatch Logs エージェントが既にインストールされている場合は、/var/awslogs/etc/proxy.conf 編集し、プロキシを追加します。

```
HTTP_PROXY=
HTTPS_PROXY=
NO_PROXY=
```

2. エージェントを再起動して、変更を有効にします。

```
sudo service awslogs restart
```

Amazon Linux 2 を使用している場合は、次のコマンドを使用してエージェントを再起動します。

```
sudo service awslogsd restart
```

CloudWatch Logs エージェント設定ファイルのコンパートメント化

awslogs-agent-setup.py バージョン 1.3.8 以降と awscli-cwlogs 1.3.3 以降を使用している場合は、/var/awslogs/etc/config/ に追加の設定ファイルを作成することで、さまざまなコンポーネントのストリーム設定をそれぞれ個別にインポートできます。CloudWatch Logs エージェントが起動すると、これらの追加の設定ファイルにストリーム設定が追加されます。[general] セクションの設定プロパティはメインの設定ファイル (/var/awslogs/etc/awslogs.conf) で定義する必要があり、/var/awslogs/etc/config/ にある追加の設定ファイルで定義しても無視されます。

rpm でエージェントをインストールしたため /var/awslogs/etc/config/ ディレクトリがない場合は、代わりに /etc/awslogs/config/ ディレクトリを使用できます。

エージェントを再起動して、変更を有効にします。

```
sudo service awslogs restart
```

Amazon Linux 2 を使用している場合は、次のコマンドを使用してエージェントを再起動します。

```
sudo service awslogsd restart
```

CloudWatch Logs エージェントに関するよくある質問

どのようなファイルローテーションがサポートされていますか。

次のファイルローテーション機能がサポートされています。

- 既存のファイルを数字サフィックスをつけた名前に変更し、その後、元の名前の空のログファイルを作成し直します。たとえば、/var/log/syslog.log が /var/log/syslog.log.1 という名前に変更されます。/var/log/syslog.log.1 が前回のローテーションにより既に存在する場合は、/var/log/syslog.log.2 という名前に変更されます。
- 元のログファイルを、コピーを作成した後切り捨てます。たとえば、/var/log/syslog.log を /var/log/syslog.log.1 にコピーし、/var/log/syslog.log を切り捨てます。この場合、データを損失する恐れがあるため、このファイルローテーション機能の使用にはご注意ください。
- 古いファイルと共通のパターンを持つ新しいファイルを作成します。たとえば /var/log/syslog.log.2014-01-01 をそのまま残し、/var/log/syslog.log.2014-01-02 を作成します。

ファイルのフィンガープリント (ソース ID) は、ログストリームキーとファイルのコンテンツの 1 行目をハッシュして計算されます。この動作をオーバーライドするには、[file_fingerprint_lines] オプションを使用できます。ファイルのローテーションが発生した場合、新しいファイルには新しいコンテンツがあり古いファイルにはコンテンツの追加がないと思われるため、エージェントは古いファイルの読み込みが完了した後は、新しいファイルをプッシュします。

使用しているエージェントのバージョンを確認する方法

セットアップスクリプトから CloudWatch Logs エージェントをインストールした場合、[/var/awslogs/bin/awslogs-version.sh] で使用しているエージェントのバージョンを確認することができます。エージェントのバージョンと主要な依存関係がプリントアウトされます。yum から CloudWatch Logs エージェントをインストールした場合には、["yum info awslogs"] と ["yum info aws-cli-plugin-cloudwatch-logs"] で CloudWatch Logs エージェントとプラグインのバージョンを確認することができます。

ログのエントリは、どのようにログイベントに変換されるのですか。

ログイベントには 2 つのプロパティが含まれます。イベント発生時のタイムスタンプおよび生のログメッセージです。デフォルトでは、空白文字以外の文字で始まる行は、前のログメッセージがある場合はこれを終了して新しいログメッセージを開始します。この動作をオーバーライドするには、[multi_line_start_pattern] を使用します。パターンに一致する行が新しいログメッセージを開始します。パターンには正規表現または「{datetime_format}」を使用できます。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た場合、CloudWatch Logs エージェントはログに警告を書き込みます。

1. ログイベントのサイズが 256 KB を超過した場合、ログイベントは完全にスキップされます。
2. ログイベントのタイムスタンプが 2 時間以上未来の場合、ログイベントはスキップされます。
3. ログイベントのタイムスタンプが 14 日以上過去の場合、ログイベントはスキップされます。
4. ログイベントがロググループの保持期間よりも古い場合、バッチはすべてスキップされます。
5. 単一の PutLogEvents リクエストでログイベントのバッチが 24 時間実行されている場合、PutLogEvents オペレーションは失敗します。

エージェントを停止させた場合、データ損失や重複が発生しますか。

状態ファイルが使用可能であり、最後に実行されたときからファイルのローテーションが発生していなければ、発生しません。CloudWatch Logs エージェントは停止した場所から再開してログデータのプッシュを続行できます。

同一または異なるホストの異なるログデータを同じログストリームに指定できますか。

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

エージェントはどの API に呼び出しを作成しますか (またはどのアクションを IAM ポリシーに含める必要がありますか)?

CloudWatch Logs エージェントには CreateLogGroup、CreateLogStream、DescribeLogStreams、および PutLogEvents オペレーションが必要です。最新のエージェントを使用している場合には、DescribeLogStreams は必要ありません。以下の IAM ポリシーの例を参照してください。

```
{
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "logs:CreateLogGroup",
      "logs:CreateLogStream",
      "logs:PutLogEvents",
```

```
      "logs:DescribeLogStreams"
    ],
    "Resource": [
      "arn:aws:logs:*:*:*"
    ]
  }
 ]
}
```

CloudWatch Logs エージェントに自動的にロググループまたはログストリームを作成させたくありません。エージェントによるロググループとログストリームの再作成を禁止する方法を教えてください。

IAM ポリシーで、エージェントを次のオペレーション (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

このセクションの表を使用して、Amazon CloudWatch Logs が毎分 Amazon CloudWatch に送信するメトリクスを確認できます。

CloudWatch Logs のメトリック

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 CloudWatch Logs サービスの使用状況メトリクスです。詳細については、「CloudWatch Logs サービスの使用状況メトリクス」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	データをサブスクリプション送信先に転送するときに CloudWatch Logs がエラーを受け取ったログイベントの数。送信先のサービスが、スロットリングの例外や再試行可能なサービス例外 (HTTP 5xx など) の再試行可能なエラーを返した場合、CloudWatch Logs は最大 24 時間配信を再試行し続けます。AccessDeniedException や ResourceNotFoundException などの再試行不可能なエラーの場合、CloudWatch Logs は再配信を試みません。 有効なディメンション: LogGroupName、DestinationType、FilterName、PolicyLevel 有効な統計: Sum 単位: なし

メトリクス	説明
DeliveryThrottling	データをサブスクリプション送信先に転送するときに CloudWatch Logs がスロットルされたログイベントの数。 送信先のサービスが、スロットリングの例外や再試行可能なサービス例外 (HTTP 5xx など) の再試行可能なエラーを返した場合、CloudWatch Logs は最大 24 時間配信を再試行し続けます。AccessDeniedException や ResourceNotFoundException などの再試行不可能なエラーの場合、CloudWatch Logs は再配信を試みません。 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。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。データタイプ MetricDatum の詳細については、「Amazon CloudWatch API リファレンス」の「MetricDatum」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 CloudWatch Logs サービスの使用状況メトリクスです。詳細については、「CloudWatch Logs サービスの使用状況メトリクス」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	CloudWatch Logs にアップロードされたログイベントのボリューム (非圧縮バイト数)。LogGroupName ディメンションと同時に使用すると、ロググループにアップロードされたログイベントのボリューム (非圧縮バイト数) になります。 有効なディメンション: LogGroupName 有効な統計: Sum 単位: バイト
IncomingLogEvents	CloudWatch Logs にアップロードされたログイベントの数。LogGroupName ディメンションと同時に使用すると、ロググループにアップロードされたログイベントの数になります。 有効なディメンション: LogGroupName 有効な統計: Sum 単位: なし

メトリクス	説明
LogEvents WithFindings	CloudWatch Logs データ保護機能を使用して監査しているデータ文字列に一致したログイベントの数。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 CloudWatch Logs サービスの使用状況メトリクスです。詳細については、「CloudWatch Logs サービスの使用状況メトリクス」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 メトリックのディメンション

ほとんどの 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先であり、AWS Lambda、Amazon Kinesis Data Streams、または Amazon Data Firehose を使用できます。
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名は

ディメンション	説明
	CloudWatch で自動的に ASCII に変換され、サポートされていない文字は疑問符 (?) に置き換えられます。

サブスクリプションフィルターメトリクスディメンション

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の有効な値は `AccountPolicy` のみです。
`DestinationType`	CloudWatch Logs データのサブスクリプション先であり、AWS Lambda、Amazon Kinesis Data Streams、または Amazon Data Firehose を使用できます。
`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名は CloudWatch で自動的に ASCII に変換され、サポートされていない文字は疑問符 (?) に置き換えられます。

ログトランスフォーマーのメトリクスとディメンション

CloudWatch Logs は、次のログトランスフォーマーメトリクスを `AWS/Logs`名前空間の CloudWatch に発行します。

メトリクス	説明
`TransformationErrors`	指定されたトランスフォーマーでログイベントの変換中に発生したエラーの数。 単位: なし 有効な統計: Sum

メトリクス	説明
TransformedBytes	変換されたログイベントの出力ボリューム。非圧縮バイト単位です。 単位: バイト 有効な統計: Sum
TransformedLogEvents	変換されたログイベントの数。 単位: なし 有効な統計: Sum

トランスフォーマーメトリクスでは、次のディメンションが使用されます。

ディメンション	説明
LogGroupname	このディメンションは、log-group-levelトランスフォーマーにのみ使用されます。
PolicyLevel	このディメンションは、アカウントレベルのトランスフォーマーにのみ使用されます。現在、このディメンションの有効な値は のみです。 AccountPolicy

CloudWatch Logs サービスの使用状況メトリクス

CloudWatch Logs はCloudWatch Logs API オペレーションの使用を追跡するメトリクスを CloudWatch 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 のサービスクォータの詳細については、「CloudWatch Logs クォータ」を参照してください。

CloudWatch Logs は、AWS/Usage と AWS/Logs の名前空間の両方で 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。

次の表は、CloudWatch Logs によって発行されるサービス使用状況メトリクスを示しています。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 API のメトリクスの合計を取得します。

ディメンション	説明
	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。
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 Logs です。
Class	追跡されているリソースのクラス。CloudWatch Logs API 使用状況メトリクスでは、値が None のこのディメンションを使用します。
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 Logs クォータ

このセクションの表を使用して、Amazon CloudWatch Logs の AWS アカウントのデフォルトのサービスクォータを確認できます。これは制限とも呼ばれます。Service Quotas は、すべてではありませんが、そのほとんどが Service Quotas コンソールの Amazon CloudWatch Logs 名前空間に一覧表示されます。

> **Note**
>
> 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	アカウントごとにリージョンごとに 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。 アカウントごとにリージョンごとに 1 つのアカウントレベルのデータ保護ポリシー。 アカウントあたり 20 のアカウントレベルのフィールドインデックスポリシー。ロググループ名のプレフィックスは重複できません。 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 500 個の異常ディテクター。クォータの引き上げをリクエストできます。
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。

リソース	デフォルトのクォータ
CreateLogGroup	10 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スロットリングされます。クォータの引き上げをリクエストできます。
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。クォータの引き上げをリクエストできます。
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。クォータの引き上げをリクエストできます。 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	10 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スロットリングされます。クォータの引き上げをリクエストできます。
DeleteLogStream	15 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スロットリングされます。クォータの引き上げをリクエストできます。
DescribeLogGroups	10 件のトランザクション/秒 (TPS/アカウント/リージョン)。クォータの引き上げをリクエストできます。
DescribeLogStreams	25 件のトランザクション/秒 (TPS/アカウント/リージョン)。クォータの引き上げをリクエストできます。
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 1,000 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、1 つの 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。
フィールドインデックス	ポリシーあたり最大 20 個のインデックス付きフィールド。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 件のリクエスト: • 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、1 秒あたり 10 件のリクエスト。 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 件のリクエスト。 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。クォータの引き上げをリクエストできます。
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	1 MB (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。クォータの引き上げをリクエストできます。 1 つのロググループに属することができるログストリームの数に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 CloudWatch ユーザーガイドの仕様: 埋め込みメトリクス形式を参照してください。
PutLogEvents	PutLogEvents の最大バッチサイズは 1 MB 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。 リージョンごとにアカウントごとに 1 秒あたり 5,000 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	ロググループを個別に指定すると、1 つの CloudWatch Logs Insights クエリで最大 50 個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。 ロググループ条件を使用して、名前のプレフィックスに基づいてロググループを選択するか、「すべてのロググループ」をクエリするように を選択すると、1 つのクエリで最大 10,000 個のロググループをクエリできます。

リソース	デフォルトのクォータ
クエリの同時実行数	標準クラスロググループの場合、最大 30 件の同時実行 CloudWatch Logs Insights クエリ (ダッシュボードに追加したクエリを含む)。この最大 30 は、使用するクエリ言語に関係なく、同時クエリの合計数に適用されます。OpenSearch Service PPL および/または OpenSearch Service SQL では、これらの同時クエリのうち 15 OpenSearch 個しか使用できません。 低頻度クラスロググループの場合、最大 5 件の同時実行 CloudWatch Logs Insights クエリ (ダッシュボードに追加したクエリを含む)。 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 件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。 PutQueryDefinition を使用して作成されたクエリ定義には有効期限がありません。
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	最大 10,000 行のクエリ結果がコンソールに表示されます。

リソース	デフォルトのクォータ
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。
リソースポリシー	アカウントあたり最大 10 個の 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、リージョンあたり最大 1000 件の 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。
トランスフォーマー	ログトランスフォーマーは、最大 5 つのパーサータイプのプロセッサを持つことができます。全体で最大 20 個のプロセッサを持つことができます。 各ロググループは、ロググループレベルのトランスフォーマーを 1 つだけ持つことができます。 各アカウントには、最大 20 個のアカウントレベルのトランスフォーマーを設定できます。これらのトランスフォーマーのいずれも、同一または重複するロググループのプレフィックスに適用できません。 これら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 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 を使用すれば、CloudWatch Logs サービスクォータの値を簡単に調べることができます。

AWS Management Console

コンソールを使用して CloudWatch Logs の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 のサービスのリストから、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

AWS CLIを使用して CloudWatch Logs のサービスクォータを表示するには

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

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

変更	説明	日付
CloudWatchOpenSearchDashboardAccess と CloudWatchOpenSearchDashboardsFullAccess は新しい IAM ポリシーです	CloudWatch Logs に、CloudWatchOpenSearchDashboardsFullAccess と CloudWatchOpenSearchDashboardAccess という 2 つの新しい IAM ポリシーが追加されました。CloudWatchOpenSearchDashboardsFullAccess は、OpenSearch Service との統合を作成および管理するためのアクセス許可を付与します。CloudWatchOpenSearchDashboardAccess は、これらの統合で作成された公開ログダッシュボードを表示するアクセス許可を付与します。詳細については、「Amazon OpenSearch Service を利用した Vended Log ダッシュボード」を参照してください。	2024 年 12 月 1 日
CloudWatchLogsFullAccess ポリシーを更新	CloudWatch Logs は、CloudWatchLogsFullAccess ポリシーに Amazon OpenSearch Service および IAM のアクセス許可を追加し、一部の機能で CloudWatch	2024 年 12 月 1 日

	Logs と OpenSearch Service の統合を有効にしました。	
CloudWatch Logs Insights は、クエリ sytnax に新しい構造タイプを追加します。	CloudWatch Logs Insights はunnest、 コマンドと 2 つの JSON 関数を追加します。これにより、JSON 文字列をマップおよびリストとして操作できます。詳細については、「構造タイプ」を参照してください。	2024 年 11 月 21 日
CloudWatch Logs が ログ取り込み中のログ変換をサポート	取り込み時にログイベントを変更できるログトランスフォーマーを作成できます。これにより、さまざまな形式やさまざまなソースのログを一貫性のあるコンテキスト豊富な形式に正規化できます。詳細については、「取り込み中のログの変換」を参照してください。	2024 年 11 月 20 日

CloudWatch Logs Insights でフィールドインデックス作成を追加	CloudWatch Logs Insights に、ログのフィールドインデックス作成のサポートが追加されました。次に、CloudWatch Logs Insights クエリでフィールドインデックスを使用すると、クエリはインデックス付きフィールドを含まないことがわかっているログイベントの処理をスキップしようとします。詳細については、「フィールドインデックスの作成」を参照してクエリのパフォーマンスを向上させ、スキャンボリュームを減らします。	2024 年 11 月 20 日
CloudWatch Logs Insights による自然言語クエリの生成のサポートを一般向けに提供	CloudWatch Logs Insights で、自然言語を使用したクエリの生成と更新がサポートされます。詳細については、「Use natural language to generate and update CloudWatch Logs Insights queries」を参照してください。	2024 年 6 月 20 日
CloudWatchLogsReadOnlyAccess ポリシーが更新	CloudWatch Logs が CloudWatchLogsReadOnlyAccess に `cloudwatch:GenerateQuery` アクセス許可を追加し、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しました。	2023 年 11 月 26 日

CloudWatchLogsFullAccess ポリシーを更新	CloudWatch Logs が CloudWatchLogsFullAccess に `cloudwatch:GenerateQuery` アクセス許可を追加し、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しました。	2023 年 11 月 26 日
CloudWatch Logs がログパターン分析を追加	CloudWatch Logs で、ユーザーが CloudWatch Logs Insights クエリを実行するたびにログイベントのパターンがスキャンされるようになりました。詳細については、「Pattern analysis」を参照してください。	2023 年 11 月 26 日
CloudWatch Logs がログ異常検出を追加	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、ログデータ内の異常を検出します。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 Logs が比較機能を追加	CloudWatch Logs Insights を使用して、ログイベントの変化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、「Compare (diff) with previous time ranges」を参照してください。	2023 年 11 月 26 日

CloudWatch Logs が新しいログクラスを追加	CloudWatch Logs で 2 つのクラスのロググループがサポートされます。これにより、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し、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はフル機能オプションを使用することができます。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 が自然言語クエリの生成をサポート	CloudWatch Logs Insights で、自然言語を使用したクエリの生成と更新がサポートされます。詳細については、「Use natural language to generate and update CloudWatch Logs Insights queries」を参照してください。	2023 年 11 月 26 日
CloudWatch Logs にライブテール用の正規表現フィルターパターンの構文サポートを追加	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 CloudWatch 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。	2023 年 11 月 13 日

CloudWatch Logs に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サポートを追加	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 CloudWatch 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。	2023 年 9 月 5 日
CloudWatch Logs Insights がパターンコマンドを追加	CloudWatch Logs Insights クエリでパターンを使用して、自動的にログデータをパターン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 CloudWatch Logs ユーザーガイド」の「pattern」を参照してください。	2023 年 7 月 17 日
CloudWatch Logs Insights が重複排除コマンドを追加	CloudWatch Logs Insights の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き、重複する結果を削除できるようになりました。詳細については、「Amazon CloudWatch Logs ユーザーガイド」の「dedup」を参照してください。	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「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。	2023 年 6 月 8 日
Live Tail 機能が追加	CloudWatch Logs に Live Tail 機能が追加されました。ログを取り込み時にスキャンして、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	CloudWatch Logs が、CloudWatchLogsReadOnlyAccess に対するアクセス許可を追加しました。`logs:StartLiveTail` および `logs:StopLiveTail` のアクセス権限が追加されたため、このポリシーを持つユーザーは、コンソールを使用して CloudWatch Logs の Live Tail セッションを開始および停止できます。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	その後、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 CloudWatch Logs ユーザーガイドの「Analyze Log Data with CloudWatch Logs Insights」を参照してください。	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 Logs との間でプライベート接続を確立できます。詳細については、Amazon CloudWatch Logs 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 CloudWatch Logs の使用」を参照してください。	2018 年 6 月 28 日

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

変更	説明	リリース日
インターフェイス VPC エンドポイント	一部のリージョンでは、インターフェイス VPC エンドポイントを使用して、Amazon VPC と CloudWatch Logs との間のトラフィックが Amazon ネットワークから離れないように維持できます。詳細については、「インターフェイス VPC エンドポイントでの CloudWatch Logs 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 Logs を使用して Route 53 が受け取った DNS クエリについてのログを保存できます。詳細については、「Amazon 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、「CloudWatch Logs に送信された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 CloudWatch	CloudWatch Logs AWS CloudTrail のサポートが追加されました。詳細については、「での CloudWatch Logs API およびコンソールオペレー	2016 年 3 月 10 日

変更	説明	リリース日
Logs メトリクス AWS CloudTrail のサポートを追加	ションの口グ記録 AWS CloudTrail」を参照してください。	
Amazon S3 への CloudWatch Logs のエクスポートに対するサポートを追加しました	CloudWatch Logs データを Amazon 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。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 AWS CloudTrail 口グに記録されたイベントのサポートを追加	CloudWatch にアラームを作成して、CloudTrail がキャプチャした特定 API アクティビティの通知を受け取り、通知をトラブルシューティングの実行に使用できます。	2014 年 11 月 10 日
Amazon CloudWatch Logs のサポートが追加されました	Amazon CloudWatch Logs を使用して、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスや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監視、保存、およびアクセスすることができます。その後、Amazon CloudWatch コンソール、 の Amazon CloudWatch Logs コマンド、または CloudWatch Logs SDK を使用して AWS CLI、CloudWatch Logs から関連するログデータを取得できます。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。