aws

デベロッパーガイド

# Amazon Data Firehose

# Amazon Data Firehose: デベロッパーガイド

# Table of Contents

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。

# Amazon Data Firehose とは何ですか?

Amazon Data Firehose は、宛先 (Amazon Simple Storage Service (Amazon S3)、Amazon Redshift、Amazon OpenSearch Service、Amazon OpenSearch Serverless、Splunk、Apache Iceberg テーブル、カスタム HTTP エンドポイント、または Datadog、Dynatrace、LogicMonitor、MongoDB、New Relic、Coralogix、Elastic などのサポートされているサードパーティーのサービスプロバイダーが所有する HTTP エンドポイントなど) にストリーミングデータをリアルタイムで配信するフルマネージドサービスです。Amazon Data Firehose を使用すると、アプリケーションを記述したり、リソースを管理したりする必要はありません。Amazon Data Firehose にデータを送信するデータプロデューサーを作成すると、それにより、指定した宛先にデータが自動配信されます。データを配信前に変換するように、Amazon Data Firehose を設定することもできます。

AWS ビッグデータソリューションの詳細については、「 でのビッグデータ AWS」を参照してください。 AWS ストリーミングデータソリューションの詳細については、「ストリーミングデータとは?」を参照してください。

## 主要な概念を学ぶ

Amazon Data Firehose の使用を開始すると、次の概念を理解することができます。

Firehose ストリーム

Amazon Data Firehose の基礎となるエンティティ。Firehose ストリームを作成し、それにデータを送信することで Amazon Data Firehose を使用します。詳細については、「チュートリアル: コンソールから Firehose ストリームを作成する」および「Firehose ストリームにデータを送信する」を参照してください。

レコード

データプロデューサーが Firehose ストリームに送信する、関心のあるデータ。最大レコードサイズは 1,000 KB です。

データプロデューサー

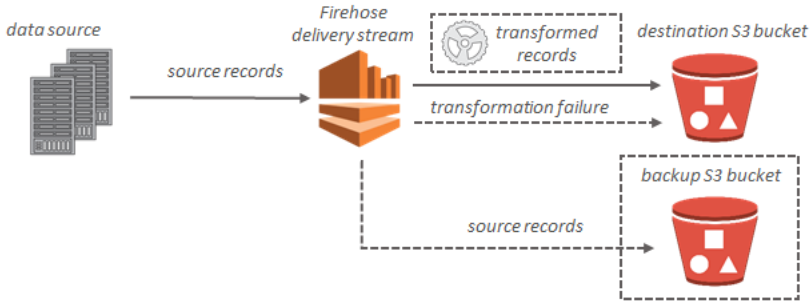
プロデューサーは Firehose ストリームにレコードを送信します。例えば、Firehose ストリームにログデータを送信するウェブサーバーはデータプロデューサーです。Firehose ストリームが自動的に既存のKinesis データストリームからデータを読み取り、宛先にロードするよう設定することもできます。詳細については、「Firehose ストリームにデータを送信する」を参照してください。

バッファリングサイズおよびバッファリングの間隔

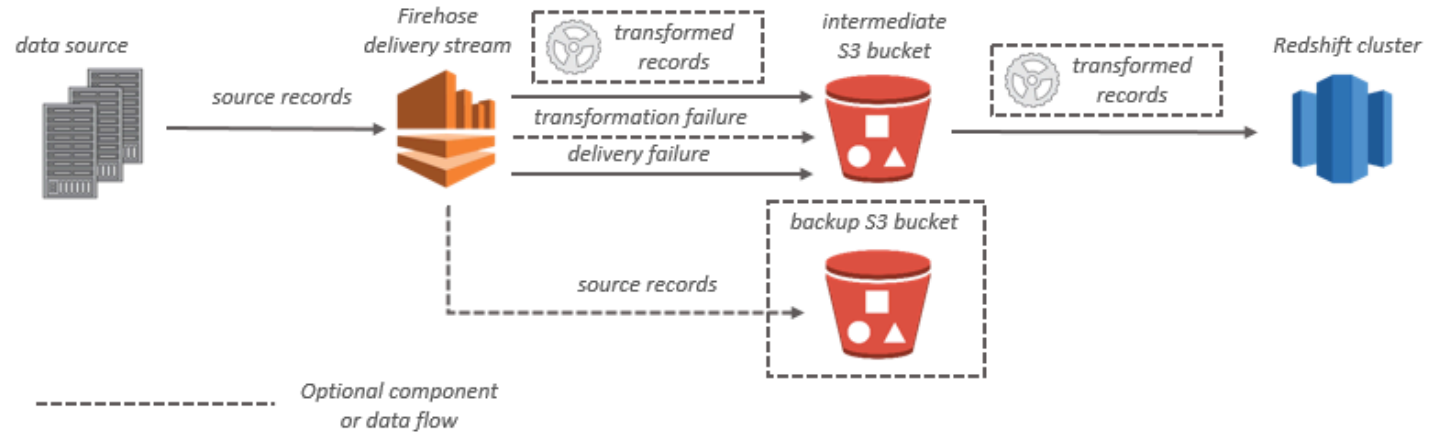
Amazon Data Firehose は特定の期間、着信ストリーミングデータを特定のサイズにバッファリングしてから、宛先に配信します。Buffer Size は MB 単位で、Buffer Interval は秒単位です。

# Amazon Data Firehose のデータフローを理解する

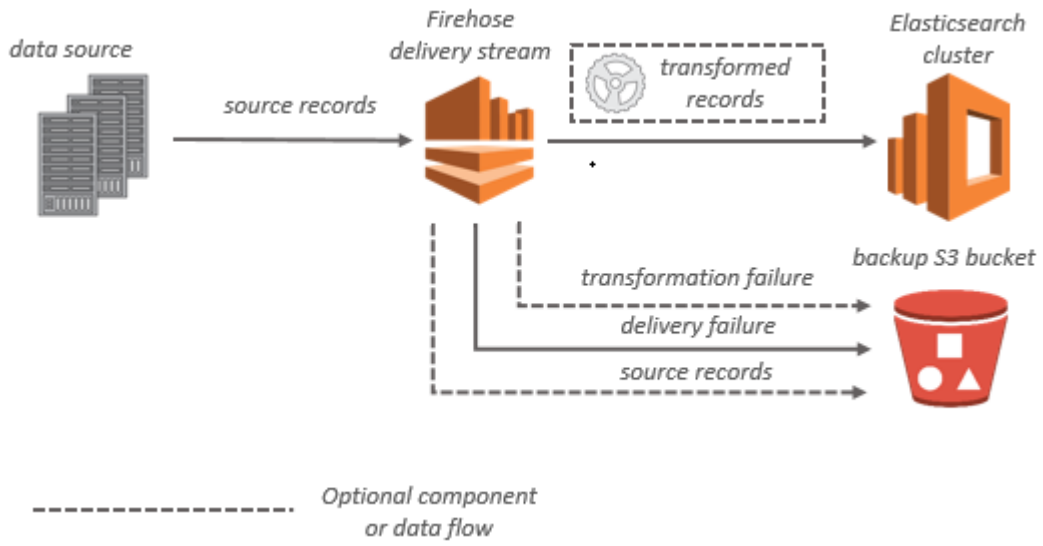
Amazon S3 の送信先の場合、ストリーミングデータは S3 バケットに配信されます。データ変換が有効な場合は、オプションで、送信元データを別の Amazon S3 バケットにバックアップすることもできます。



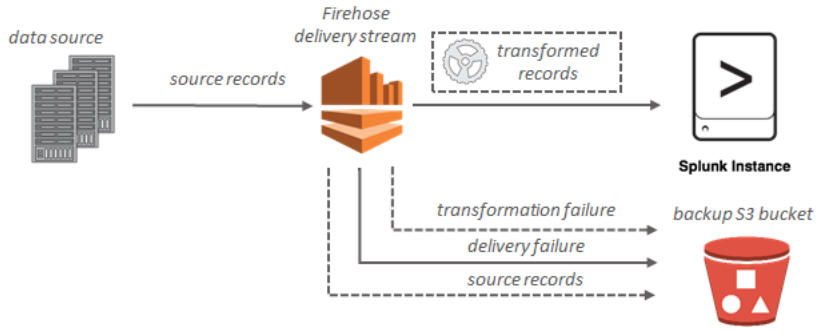

Amazon Redshift の送信先の場合、ストリーミングデータは S3 バケットに配信されます。次に Amazon Data Firehose は、Amazon Redshift COPY コマンドを発行して、S3 バケットから Amazon Redshift クラスターにデータをロードします。データ変換が有効な場合は、オプションで、送信元データを別の Amazon S3 バケットにバックアップすることもできます。




OpenSearch Service の送信先の場合、ストリーミングデータは OpenSearch Service クラスターに配信され、オプションで、配信と同時に S3 バケットにバックアップすることもできます。




Splunk の送信先を使用する場合、ストリーミングデータは Splunk に配信され、オプションで、配信と同時に S3 バケットにバックアップすることもできます。




# AWS SDK での Firehose 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なる 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 |

| SDK ドキュメント | 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 | AWS Tools for PowerShell 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 |

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] リンクから、コードの例をリクエストしてください。

# Amazon Data Firehose を設定するための前提条件を完了する

Amazon Data Firehose を初めて使用する場合は、事前に次のタスクをすべて実行してください。

タスク

- にサインアップする AWS
- (オプション) ライブラリとツールをダウンロードする

## にサインアップする AWS

Amazon Web Services (AWS) にサインアップすると AWS、Amazon Data Firehose を含む のすべてのサービスに AWS アカウントが自動的にサインアップされます。請求されるのは、使用したサービスの料金のみです。

AWS アカウントがすでにある場合は、次のタスクに進んでください。 AWS アカウントをお持ちでない場合は、以下の手順に従ってアカウントを作成してください。

AWS アカウントにサインアップするには

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

   サインアップ手順の一環として、電話またはテキストメッセージを受け取り、電話キーパッドで検証コードを入力します。

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

## (オプション) ライブラリとツールをダウンロードする

次のライブラリとツールは、Amazon Data Firehose のプログラムによる操作、またはコマンドラインからの操作に役立ちます。

- Firehose API オペレーションは、Amazon Data Firehose がサポートする基本的なオペレーションセットです。
- Go、Java、.NET、Node.js、Python、Ruby AWS SDKs には、Amazon Data Firehose のサポートとサンプルが含まれています。

  のバージョンに Amazon Data Firehose のサンプルが含まれ AWS SDK for Java ていない場合は、GitHub から最新の AWS SDK をダウンロードすることもできます。
- AWS Command Line Interface は Amazon Data Firehose をサポートします。 AWS CLI を使用すると、コマンドラインから複数の AWS サービスを制御して、スクリプトを使用して自動化できます。

# チュートリアル: コンソールから Firehose ストリームを作成する

AWS マネジメントコンソール または AWS SDK を使用して、選択した送信先への Firehose ストリームを作成できます。

Firehose ストリームの設定は、Amazon Data Firehose コンソールまたは UpdateDestination を使用して、作成後にいつでも更新できます。設定の更新中は Firehose ストリームは `Active` 状態のままとなり、データの送信を続行できます。通常、更新された設定は数分以内に有効になります。設定を更新する度に、Firehose ストリームのバージョン番号は 1 の値ずつ上がり、配信される Amazon S3 オブジェクト名に反映されます。詳細については、「Amazon S3 オブジェクト名の形式を設定する」を参照してください。

Firehose ストリームを作成するには、次のトピックのステップを実行します。

トピック



## Firehose ストリームのソースと宛先を選択する

1. Firehose コンソール (https://console.aws.amazon.com/firehose/) を開きます。
2. [Firehose ストリームを作成] を選択します。
3. [Firehose ストリームを作成] ページで、次のいずれかのオプションから Firehose ストリームのソースを選択します。

    - Direct PUT – プロデューサーアプリケーションが直接書き込む Firehose ストリームを作成するときは、こちらを選択します。Amazon Data Firehose の Direct PUT と統合する AWS サービス、エージェント、オープンソースサービスのリストを次に示します。このリストは網羅的

なものではなく、Firehose に直接データを送信するために使用できる他のサービスが存在する可能性もあります。

- AWS SDK
- AWS Lambda
- AWS CloudWatch Logs
- AWS CloudWatch Events
- AWS Cloud メトリクスストリーム
- AWS IoT
- AWS Eventbridge
- Amazon Simple Email Service
- Amazon SNS
- AWS WAF ウェブ ACL ログ
- Amazon API Gateway - アクセスログ
- Amazon Pinpoint
- Amazon MSK ブローカーログ
- Amazon Route 53 Resolver クエリログ
- AWS Network Firewall アラートログ
- AWS Network Firewall フローログ
- Amazon Elasticache Redis SLOWLOG
- Kinesis Agent (linux)
- Kinesis Tap (windows)
- Fluentbit
- Fluentd
- Apache Nifi
- Snowflake

- Amazon Kinesis Data Streams – Kinesis データストリームをデータソースとして使用する Firehose ストリームを設定するには、このオプションを選択します。これで、Firehose を使用して、既存の Kinesis データストリームからデータを簡単に読み取り、宛先にロードすることができるようになります。Kinesis Data Streams をデータソースとして使用する方法の詳細については、「Sending data to a Firehose stream with Kinesis Data Streams」を参照してください。

- Amazon MSK – Amazon MSK をデータソースとして使用するように Firehose ストリームを設定するときは、このオプションを選択します。これで、Firehose を使用して既存の Amazon MSK クラスターから簡単にデータを読み込み、指定した S3 バケットにロードすることができます。詳細については、「Sending data to a Firehose stream with Amazon MSK」を参照してください。

4. Firehose がサポートする次のいずれかの宛先から、Firehose ストリームの宛先を選択します。

   - Amazon OpenSearch Service
   - Amazon OpenSearch Serverless
   - Amazon Redshift
   - Amazon S3
   - Apache Iceberg テーブル
   - Coralogix
   - Datadog
   - Dynatrace
   - Elastic
   - HTTP エンドポイント
   - Honeycomb
   - Logic Monitor
   - Logz.io
   - MongoDB Cloud
   - New Relic
   - Splunk
   - Splunk Observability Cloud
   - Sumo Logic
   - Snowflake

5. [Firehose ストリーム名] で、コンソールが生成する名前を使用するか、任意の Firehose ストリームを追加できます。

# ソース設定を構成する

ソース設定は、コンソールから Firehose ストリームに情報を送信するために選択するソースに基づいて設定できます。Amazon MSK および Amazon Kinesis Data Streams のソース設定をソースとして構成できます。Direct PUT のためにソースとして使用できるソース設定はありません。

## Amazon MSK のソース設定を構成する

Amazon MSK を選択して Firehose ストリームに情報を送信する場合は、MSK プロビジョンドクラスターと MSK サーバーレスクラスターから選択できます。これで、Firehose を使用して、特定の Amazon MSK クラスターとトピックから簡単に読み取りを行い、これを指定した S3 宛先にロードすることができるようになります。

このページにある [ソース設定] セクションで、次のフィールドに値を入力します。

Amazon MSK クラスター接続

クラスター設定に基づいて、[プライベートブートストラップブローカー] (推奨) か [パブリックブートストラップブローカー]] のいずれかを選択します。ブートストラップブローカーは、Apache Kafka クライアントがクラスターに接続するときの出発点として使用するものです。パブリックブートストラップブローカーは AWS の外部からのパブリックアクセスが対象で、プライベートブートストラップブローカーは AWS の内部からのアクセスが対象です。詳細については、「Amazon Managed Streaming for Apache Kafka」を参照してください。

プライベートブートストラップブローカーを介して Amazon MSK プロビジョンドクラスターまたはサーバーレスクラスターに接続するには、クラスターが以下の要件をすべて満たしている必要があります。

- クラスターがアクティブである必要があります。
- アクセスコントロール方法の 1 つとしてクラスターが IAM を持っている必要があります。
- IAM アクセスコントロール方法でマルチ VPC プライベート接続が有効になっている必要があります。
- このクラスターに、Amazon MSK `CreateVpcConnection` API オペレーションを呼び出す許可を Firehose サービスプリンシパルに付与するリソースベースのポリシーを追加する必要があります。

パブリックブートストラップブローカーを介して Amazon MSK プロビジョンドクラスターに接続するには、クラスターが以下の要件をすべて満たしている必要があります。

- クラスターがアクティブである必要があります。

- アクセスコントロール方法の 1 つとしてクラスターが IAM を持っている必要があります。
- クラスターはパブリックにアクセス可能でなければなりません。

MSK クラスターアカウント

Amazon MSK クラスターが存在するアカウントを選択できます。これには、次のいずれかを指定できます。

- [現在のアカウント] – 現在の AWS アカウントの MSK クラスターからデータを取り込むことを許可します。そのためには、Firehose ストリームがデータを読み取る Amazon MSK クラスターの ARN を指定する必要があります。
- [クロスアカウント] – 別の AWS アカウントの MSK クラスターからデータを取り込むことを許可します。詳細については、「Amazon MSK からのクロスアカウント配信」を参照してください。

トピック

Firehose ストリームでデータを取り込む Apache Kafka トピックを指定します。Firehose ストリームの作成が完了した後に、このトピックを更新することはできません。

> **Note**
>
> Firehose は Apache Kafka メッセージを自動解凍します。

## Amazon Kinesis Data Streams のソース設定を構成する

Amazon Kinesis Data Streams のソース設定を構成して、次のように Firehose ストリームに情報を送信します。

> **Important**
>
> Kinesis Producer Library (KPL) を使用して Kinesis データストリームにデータを書き込む場合、集約を使用してその Kinesis データストリームに書き込むレコードを結合できます。その後そのデータストリームを Firehose ストリームのソースとして使用する場合、Amazon Data Firehose はレコードの集約を解除してから宛先に配信します。データを変換するように Firehose ストリームを設定する場合、Amazon Data Firehose はレコードの集約を解除してから AWS Lambda に配信します。詳細については、「Kinesis Producer Library を使用した Amazon Kinesis Data Streams プロデューサーの開発」および「集約」を参照してください。

[ソース設定] で、[Kinesis データストリーム] リストで既存のストリームを選択するか、または形式 `arn:aws:kinesis:[Region]:[AccountId]:stream/[StreamName]` でデータストリーム ARN を入力します。

既存のデータストリームがない場合は、[作成] を選択して、Amazon Kinesis コンソールから新しいデータストリームを作成します。Kinesis ストリームに必要な許可を持つ IAM ロールが必要な場合があります。詳細については、「???」を参照してください。新しいストリームを作成した後に、[更新] アイコンを選択して [Kinesis ストリーム] のリストを更新します。多数のストリームがある場合は、[Filter by name] を使用してリストをフィルタリングします。

> **Note**
>
> Firehose ストリームのソースとして Kinesis データストリームを設定すると、Amazon Data Firehose の `PutRecord` および `PutRecordBatch` オペレーションは無効になります。この場合、Firehose ストリームにデータを追加するには、Kinesis Data Streams の `PutRecord` および `PutRecords` オペレーションを使用します。

Amazon Data Firehose は、Kinesis ストリームの `LATEST` の場所からデータの読み取りを開始します。Kinesis Data Streams の場所の詳細については、「GetShardIterator」を参照してください。

Amazon Data Firehose は、Kinesis Data Streams の GetRecords オペレーションを各シャードにつき 1 秒に 1 回呼び出します。ただし、フルバックアップが有効になっている場合、Firehose は、各シャードについて 1 秒に 2 回、Kinesis Data Streams `GetRecords` オペレーションを呼び出します。1 つは主な配信先用、もう 1 つはフルバックアップ用です。

複数の Firehose ストリームが同じ Kinesis ストリームから読み取ることができます。他の Kinesis アプリケーション (コンシューマー) も同じストリームから読み取ることができます。任意の Firehose ストリームまたは他のコンシューマーアプリケーションからの呼び出しはすべて、シャードの全体的なスロットリング制限数に含まれます。スロットリングを回避するため、アプリケーションを注意深く計画してください。Kinesis Data Streams の制限事項の詳細については、「Amazon Kinesis Streams の制限」を参照してください。

レコード変換と形式転換を設定するには、次のステップに進みます。

## (オプション) レコード変換と形式転換を設定する

Amazon Data Firehose を設定して、レコードデータを変換します。

## Firehose ストリームのソースとして Amazon MSK を選択する場合

AWS Lambda でソースレコードを変換セクションで、次のフィールドに値を指定します。

1. データ変換

   着信データを変換しない Firehose ストリームを作成するには、[データ変換を有効にする] チェックボックスのチェックを外します。

   着信データを配信前に変換するために Firehose が呼び出し、使用する Lambda 関数を指定するには、[データ変換を有効にする] を選択します。いずれかの Lambda 設計図を使用して新しい Lambda 関数を設定するか、既存の Lambda 関数を選択することができます。Lambda 関数には、Firehose に必要なステータスモデルが含まれる必要があります。詳細については、「Amazon Data Firehose でソースデータを変換する」を参照してください。

2. [Convert record format (レコード形式を変換)] セクションで、次のフィールドに値を入力します。

   レコード形式の変換

   入力データの形式を変換しない Firehose ストリームを作成するには、[無効] を選択します。

   受信レコードの形式を変換するには、[Enabled (有効)] を選択し、目的の出力形式を指定します。Firehose がレコード形式を変換するために使用するスキーマを保持する AWS Glue テーブルを指定する必要があります。詳細については、「入力データ形式を変換する」を参照してください。

   を使用してレコード形式の変換を設定する方法の例については CloudFormation、AWS「::KinesisFirehose::DeliveryStream」を参照してください。

## Firehose ストリームのソースとして Amazon Kinesis Data Streams または Direct PUT を選択する場合

[ソース設定] セクションで、次のフィールドを入力します。

1. [レコードを変換] で、次のいずれかを選択します。

   a. 宛先が Amazon S3 または Splunk の場合、[Amazon CloudWatch Logs でソースレコードを解凍] セクションで、[解凍をオンにする] を選択します。

b. AWS Lambda を使用したソースレコードの変換セクションで、次のフィールドの値を指定します。

データ変換

着信データを変換しない Firehose ストリームを作成するには、[データ変換を有効にする] チェックボックスのチェックを外します。

着信データを配信前に変換するために Amazon Data Firehose が呼び出し、使用する Lambda 関数を指定するには、[データ変換を有効にする] を選択します。いずれかの Lambda 設計図を使用して新しい Lambda 関数を設定するか、既存の Lambda 関数を選択することができます。Lambda 関数には、Amazon Data Firehose に必要なステータスモデルが含まれる必要があります。詳細については、「Amazon Data Firehose でソースデータを変換する」を参照してください。

2. [Convert record format (レコード形式を変換)] セクションで、次のフィールドに値を入力します。

レコード形式の変換

入力データの形式を変換しない Firehose ストリームを作成するには、[無効] を選択します。

受信レコードの形式を変換するには、[Enabled (有効)] を選択し、目的の出力形式を指定します。Amazon Data Firehose がレコード形式を変換するために使用するスキーマを保持する AWS Glue テーブルを指定する必要があります。詳細については、「入力データ形式を変換する」を参照してください。

でレコード形式変換を設定する方法の例については CloudFormation、AWS「::KinesisFirehose::DeliveryStream」を参照してください。

# 宛先の設定を構成する

このセクションでは、選択した宛先に基づいて Firehose ストリームのために構成する必要がある設定について説明します。

トピック

- Amazon S3 の宛先の設定を構成する
- Apache Iceberg テーブルの宛先の設定を構成する
- Amazon Redshift の宛先の設定を構成する

- OpenSearch Service の宛先の設定を構成する
- OpenSearch Serverless の宛先の設定を構成する
- HTTP エンドポイントの宛先の設定を構成する
- Datadog の宛先の設定を構成する
- Honeycomb の宛先の設定を構成する
- Coralogix の宛先の設定を構成する
- Dynatrace の宛先の設定を構成する
- LogicMonitor の宛先の設定を構成する
- Logz.io の宛先の設定を構成する
- MongoDB Atlas の送信先の設定を構成する
- New Relic の宛先の設定を構成する
- Snowflake の宛先の設定を構成する
- Splunk の宛先の設定を構成する
- Splunk Observability Cloud の宛先の設定を構成する
- Sumo Logic の宛先の設定を構成する
- Elastic の宛先の設定を構成する

## Amazon S3 の宛先の設定を構成する

Firehose ストリームの宛先として Amazon S3 を使用するには、次の設定を指定する必要があります。

- 以下のフィールドに値を入力します。

  S3 バケット

  ストリーミングデータの配信先となる、お客様が所有している S3 バケットを選択します。新しい S3 バケットを作成するか、既存のバケットを選択することができます。

  改行区切り記号

  Amazon S3 に配信されるオブジェクトのレコード間に改行区切り記号を追加するように Firehose ストリームを設定できます。これを行うには、[[Enabled (有効)] をクリックします。Amazon S3 に配信されるオブジェクトのレコード間に改行区切り文字を追加しない場

合は、 [Disabled (無効)] をクリックします。集約されたレコードを持つ S3 オブジェクトを、Athena を使用してクエリする場合は、このオプションを有効にします。

動的パーティショニング

[Enabled (有効)] をクリックして、動的パーティショニングを有効にして設定します。

マルチレコードの集約解除

これは、Firehose ストリーム内のレコードを解析し、有効な JSON または指定された改行区切り文字に基づいてレコードを分離するプロセスです。

複数のイベント、ログ、またはレコードを 1 つの PutRecord および PutRecordBatch API コールに集約した場合でも、動的パーティショニングを有効にして設定できます。集約データでは、動的パーティショニングを有効にすると、Amazon Data Firehose はレコードを解析し、各 API コール内で複数の有効な JSON オブジェクトを検索します。Firehose ストリームがソースとして Kinesis Data Stream で設定されている場合、Kinesis Producer Library (KPL) の組み込み集約を使用することもできます。データパーティション機能は、データが集約解除された後に実行されます。したがって、各 API コールの各レコードを異なる Amazon S3 プレフィックスに配信できます。また、Lambda 関数の統合を活用して、データパーティショニング機能の前に、他の集約解除や他の変換を実行することもできます。

> **Important**
>
> データが集約されている場合、動的パーティショニングは、データの集約解除が実行された後にのみ適用できます。したがって、集約データに対して動的パーティショニングを有効にする場合は、[Enabled (有効)] をクリックして、マルチレコード集約解除を有効にします。

Firehose ストリームは、KPL (protobuf) の集約解除、JSON または区切り文字の集約解除、Lambda 処理、データパーティショニング、データ形式変換、および Amazon S3 配信の順序で処理ステップを実行します。

マルチレコードの集約解除のタイプ

マルチレコードの集約解除を有効にした場合、Firehose でデータを集約解除する方法を指定する必要があります。ドロップダウンメニューから [JSON] または [Delimited (区切り)] をクリックします。

インライン解析

これは、Amazon S3 にバインドされたデータの動的パーティショニングを行うためにサポートされているメカニズムの 1 つです。データの動的パーティショニングにインライン解析を使用するには、パーティショニングキーとして使用するデータレコードパラメータを指定し、指定したパーティショニングキーの値を提供する必要があります。[Enabled (有効)] をクリックして、インライン解析を有効にして設定します。

> **Important**
>
> 上記のステップでソースレコードを変換するために AWS Lambda 関数を指定した場合、この関数を使用して S3 にバインドされたデータを動的にパーティション分割し、インライン解析を使用してパーティショニングキーを作成できます。動的パーティショニングでは、インライン解析または AWS Lambda 関数を使用してパーティショニングキーを作成できます。または、インライン解析と AWS Lambda 関数の両方を同時に使用して、パーティショニングキーを作成できます。

動的パーティショニングキー

[キー] および [値] フィールドを使用して、動的パーティションニングキーとして使用するデータレコードパラメータを指定し、動的パーティショニングキーの値を生成するための jq クエリを指定することができますます。Firehose では jq 1.6 のみをサポートしています。最大 50 個の動的パーティショニングキーを指定できます。Firehose ストリームの動的パーティショニングを正常に設定するには、動的パーティショニングキーの値に有効な jq 式を入力する必要があります。

S3 バケットプレフィックス

動的パーティショニングを有効にして設定する場合は、Amazon Data Firehose がパーティション化されたデータを配信する S3 バケットプレフィックスを指定する必要があります。

動的パーティショニングを正しく設定するには、S3 バケットプレフィクスの数が、指定したパーティショニングキーの数と同じである必要があります。

ソースデータは、インライン解析または指定した AWS Lambda 関数でパーティション化できます。ソースデータのパーティショニングキーを作成するために AWS Lambda 関数を指定した場合、「partitionKeyFromLambda:keyID」の形式を使用して S3 バケットプレフィックス値を手動で入力する必要があります。インライン解析を使用してソースデータのパー

ティショニングキーを指定する場合は、「partitionKeyFromQuery:keyID」の形式を使用して S3 バケットプレビューの値を手動で入力するか、[動的パーティショニングキーの適用] ボタンをクリックし、動的パーティショニングキーと値のペアを使用して S3 バケットプレフィックスを自動生成することができます。インライン解析または AWS Lambda を使用してデータをパーティション化するときに、S3 バケットプレフィックス !{namespace:value} で次の式フォームを使用することもできます。ここで、名前空間は partitionKeyFromQuery または partitionKeyFromLambda のいずれかになります。

S3 バケットと S3 エラー出力プレフィックスのタイムゾーン

Amazon S3 オブジェクトのカスタムプレフィックスで日付と時刻で使用するタイムゾーンを選択します。デフォルトでは、Firehose は時刻プレフィックスを UTC で追加します。異なるタイムゾーンを使用する場合は、S3 プレフィックスで使用されるタイムゾーンを変更できます。

バッファリングのヒント

Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。送信先の推奨バッファサイズは、サービスプロバイダーによって異なります。

S3 圧縮

GZIP、Snappy、Zip、または Hadoop 互換の Snappy データ圧縮、またはデータ圧縮なしを選択します。Snappy、Zip、および Hadoop 互換の Snappy 圧縮は、Amazon Redshift を宛先とする Firehose ストリームには使用できません。

S3 のファイル拡張子の形式 (オプション)

Amazon S3 の宛先バケットに配信されるオブジェクトのファイル拡張子の形式を指定します。この機能を有効にすると、指定されたファイル拡張子は、データ形式変換または .parquet や .gz などの S3 圧縮機能によって付加されたデフォルトのファイル拡張子を上書きします。この機能をデータ形式変換または S3 圧縮で使用する際には、適切なファイル拡張子を設定しているかどうかを確認します。ファイル拡張子はピリオド (.) で始まらなければならず、次の文字を含めることができます: 0-9a-z!-_.*‘()。ファイル拡張子は最大 128 文字です。

S3 暗号化

Firehose は、Amazon S3 で配信されたデータを暗号化するための AWS Key Management Service (SSE-KMS) による Amazon S3 サーバー側の暗号化をサポートしています。送信先 S3 バケットで指定されたデフォルトの暗号化タイプを使用するか、所有するキーのリストから AWS KMS キーで暗号化するかを選択できます。 AWS KMS キーでデータを暗号化する

場合は、デフォルトの AWS マネージドキー (aws/s3) またはカスタマーマネージドキーを使用できます。詳細については、AWS 「KMS マネージドキーによるサーバー側の暗号化を使用したデータの保護 (SSE-KMS)」を参照してください。

## Apache Iceberg テーブルの宛先の設定を構成する

Firehose は、中国リージョン、アジアパシフィック (台北 ) 、アジアパシフィック (マレーシア ) 、アジアパシフィック (ニュージーランド) AWS GovCloud (US) Regions、メキシコ (中部) AWS リージョンを除くすべての で Apache Iceberg Tables を送信先としてサポートしています。

宛先としての Apache Iceberg テーブルの詳細については、「Amazon Data Firehose を使用して Apache Iceberg テーブルにデータを配信する」を参照してください。

## Amazon Redshift の宛先の設定を構成する

このセクションでは、Firehose ストリームの宛先として Amazon Redshift を使用するための設定について説明します。

Amazon Redshift プロビジョンドクラスターまたは Amazon Redshift Serverless ワークグループのどちらを使用しているかに基づき、以下の手順のいずれかを選択します。

- Amazon Redshift プロビジョンドクラスター
- Amazon Redshift Serverless ワークグループの宛先の設定を構成する

> **Note**
>
> Firehose は、拡張 VPC ルーティングを使用する Amazon Redshift クラスターに書き込むことはできません。

### Amazon Redshift プロビジョンドクラスター

このセクションでは、Firehose ストリームの宛先として Amazon Redshift プロビジョンドクラスターを使用するときの設定について説明します。

- 以下のフィールドに値を入力します。

クラスター

S3 バケットデータのコピー先となる Amazon Redshift クラスター。Amazon Redshift クラスターをパブリックアクセス可能に設定し、Amazon Data Firehose の IP アドレスをブロック解除します。詳細については、「Amazon Redshift の宛先へのアクセスを Firehose に付与する」を参照してください。

認証

ユーザー名/パスワードを直接入力するか、 からシークレットを取得 AWS Secrets Manager して Amazon Redshift クラスターにアクセスするかを選択できます。

- [ユーザーネーム]

  Amazon Redshift クラスターへの許可がある Amazon Redshift ユーザーを指定します。このユーザーには、S3 バケットから Amazon Redshift クラスターにデータをコピーする Amazon Redshift `INSERT` アクセス許可が必要です。

- [パスワード]

  クラスターへの許可を持つユーザーのパスワードを指定します。

- シークレット

  Amazon Redshift クラスターの認証情報 AWS Secrets Manager を含むシークレットを から選択します。ドロップダウンリストにシークレットが表示されない場合は、Amazon Redshift の認証情報用に AWS Secrets Manager でシークレットを作成します。詳細については、「Amazon Data Firehose AWS Secrets Manager で を使用して認証する」を参照してください。

データベース

データのコピー先となる Amazon Redshift データベース。

テーブル

データのコピー先となる Amazon Redshift テーブル。

列

(オプション) データのコピー先となるテーブル内の特定の列。Amazon S3 オブジェクトで定義した列数が Amazon Redshift テーブル内の列数より少ない場合に、このオプションを使用します。

中間の S3 送信先

Firehose は最初にデータを S3 バケットに配信してから、Amazon Redshift COPY コマンドを発行してデータを Amazon Redshift クラスターにロードします。ストリーミングデータの配信先となる、お客様が所有している S3 バケットを指定します。新しい S3 バケットを作成するか、お客様が所有する既存のバケットを選択します。

Firehose は、Amazon Redshift クラスターにロードした後で S3 バケットからデータを削除することはしません。ライフサイクル設定を使用して、S3 バケットでデータを管理できます。詳細については、Amazon Simple Storage Service ユーザーガイドの「オブジェクトのライフサイクルの管理」を参照してください。

中間の S3 プレフィックス

(オプション) Amazon S3 オブジェクトに対してデフォルトのプレフィックスを使用するには、このオプションを空白のままにします。Firehose は、Amazon S3 オブジェクトに提供された「YYYY/MM/dd/HH」UTC 時間形式をプレフィックスで自動的に使用する点に注意してください。このプレフィックスの開始に追加できます。詳細については、「Amazon S3 オブジェクト名の形式を設定する」を参照してください。

COPY オプション

Amazon Redshift COPY コマンドで指定できるパラメータです。これらのパラメータは、設定に必要になる場合があります。例えば、Amazon S3 データ圧縮が有効になっている場合はGZIP「」が必要です。S3 バケットが Amazon Redshift クラスターと同じ AWS リージョンにない場合はREGION「」が必要です。詳細については、Amazon Redshift データベース開発者ガイドの「COPY」を参照してください。

COPY コマンド

Amazon Redshift COPY コマンド。詳細については、Amazon Redshift データベース開発者ガイドの「COPY」を参照してください。

再試行の期間

Amazon Redshift クラスターへのデータ COPY が失敗した場合に Firehose が再試行する時間 (0 ~ 7200 秒)です。Firehose は再試行の期間が終了するまで 5 分ごとに再試行します。再試行の期間を 0 (ゼロ) 秒に設定した場合、Firehose は COPY コマンドが失敗しても再試行しません。

バッファリングのヒント

Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。送信先の推奨バッファサイズは、サービスプロバイダーによって異なります。

S3 圧縮

GZIP、Snappy、Zip、または Hadoop 互換の Snappy データ圧縮、またはデータ圧縮なしを選択します。Snappy、Zip、および Hadoop 互換の Snappy 圧縮は、Amazon Redshift を宛先とする Firehose ストリームには使用できません。

S3 のファイル拡張子の形式 (オプション)

S3 のファイル拡張子の形式 (オプション) – Amazon S3 宛先バケットに配信されるオブジェクトのファイル拡張子の形式を指定します。この機能を有効にすると、指定されたファイル拡張子は、データ形式変換または .parquet や .gz などの S3 圧縮機能によって付加されたデフォルトのファイル拡張子を上書きします。この機能をデータ形式変換または S3 圧縮で使用する際には、適切なファイル拡張子を設定しているかどうかを確認します。ファイル拡張子はピリオド (.) で始まらなければならず、次の文字を含めることができます: 0-9a-z!-_.*'()。ファイル拡張子は最大 128 文字です。

S3 暗号化

Firehose は、Amazon S3 で配信されたデータを暗号化するための AWS Key Management Service (SSE-KMS) による Amazon S3 サーバー側の暗号化をサポートしています。送信先 S3 バケットで指定されたデフォルトの暗号化タイプを使用するか、所有するキーのリストから AWS KMS キーで暗号化するかを選択できます。 AWS KMS キーでデータを暗号化する場合は、デフォルトの AWS マネージドキー (aws/s3) またはカスタマーマネージドキーを使用できます。詳細については、AWS 「KMS マネージドキーによるサーバー側の暗号化を使用したデータの保護 (SSE-KMS)」を参照してください。

## Amazon Redshift Serverless ワークグループの宛先の設定を構成する

このセクションでは、Firehose ストリームの宛先として Amazon Redshift Serverless ワークグループを使用するときの設定について説明します。

- 以下のフィールドに値を入力します。

ワークグループ名

S3 バケットデータのコピー先となる Amazon Redshift Serverless ワークグループ。Amazon Redshift Serverless ワークグループをパブリックアクセス可能に設定し、Firehose の IP アドレスをブロック解除します。詳細については、「Amazon Redshift Serverless への接続」の「パブリックにアクセス可能なときの Amazon Redshift Serverless に接続する」と、「Amazon Redshift の宛先へのアクセスを Firehose に付与する」を参照してください。

認証

ユーザー名/パスワードを直接入力するか、 からシークレットを取得 AWS Secrets Manager して Amazon Redshift Serverless ワークグループにアクセスするかを選択できます。

- [ユーザーネーム]

  Amazon Redshift Serverless ワークグループへの許可を持つ Amazon Redshift ユーザーを指定します。このユーザーには、S3 バケットから Amazon Redshift Serverless ワークグループにデータをコピーする Amazon Redshift `INSERT` アクセス許可が必要です。

- [パスワード]

  Amazon Redshift Serverless ワークグループにアクセスするための許可を持っているユーザーのパスワードを指定します。

- シークレット

  Amazon Redshift Serverless ワークグループの認証情報 AWS Secrets Manager を含むシークレットを から選択します。ドロップダウンリストにシークレットが表示されない場合は、Amazon Redshift の認証情報用に AWS Secrets Manager でシークレットを作成します。詳細については、「Amazon Data Firehose AWS Secrets Manager で を使用して認証する」を参照してください。

データベース

データのコピー先となる Amazon Redshift データベース。

テーブル

データのコピー先となる Amazon Redshift テーブル。

列

(オプション) データのコピー先となるテーブル内の特定の列。Amazon S3 オブジェクトで定義した列数が Amazon Redshift テーブル内の列数より少ない場合に、このオプションを使用します。

中間の S3 送信先

Amazon Data Firehose は最初にデータを S3 バケットに配信してから、Amazon Redshift COPY コマンドを発行してデータを Amazon Redshift Serverless ワークグループにロードします。ストリーミングデータの配信先となる、お客様が所有している S3 バケットを指定します。新しい S3 バケットを作成するか、お客様が所有する既存のバケットを選択します。

Firehose は、Amazon Redshift Serverless ワークグループにロードした後は、S3 バケットからデータを削除しません。ライフサイクル設定を使用して、S3 バケットでデータを管理できます。詳細については、Amazon Simple Storage Service ユーザーガイドの「オブジェクトのライフサイクルの管理」を参照してください。

中間の S3 プレフィックス

(オプション) Amazon S3 オブジェクトに対してデフォルトのプレフィックスを使用するには、このオプションを空白のままにします。Firehose は、Amazon S3 オブジェクトに提供された「YYYY/MM/dd/HH」UTC 時間形式をプレフィックスで自動的に使用する点に注意してください。このプレフィックスの開始に追加できます。詳細については、「Amazon S3 オブジェクト名の形式を設定する」を参照してください。

COPY オプション

Amazon Redshift COPY コマンドで指定できるパラメータです。これらのパラメータは、設定に必要になる場合があります。例えば、Amazon S3 データ圧縮が有効になっている場合はGZIP「」が必要です。S3 バケットが Amazon Redshift Serverless ワークグループと同じ AWS リージョンにない場合はREGION「」が必要です。詳細については、Amazon Redshift データベース開発者ガイドの「COPY」を参照してください。

COPY コマンド

Amazon Redshift COPY コマンド。詳細については、Amazon Redshift データベース開発者ガイドの「COPY」を参照してください。

再試行の期間

Amazon Redshift Serverless ワークグループへのデータ COPY が失敗した場合に Firehose が再試行する時間 (0 ~ 7200 秒)。Firehose は再試行の期間が終了するまで 5 分ごとに再試行

します。再試行の期間を 0 (ゼロ) 秒に設定した場合、Firehose は COPY コマンドが失敗しても再試行しません。

バッファリングのヒント

Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。送信先の推奨バッファサイズは、サービスプロバイダーによって異なります。

S3 圧縮

GZIP、Snappy、Zip、または Hadoop 互換の Snappy データ圧縮、またはデータ圧縮なしを選択します。Snappy、Zip、および Hadoop 互換の Snappy 圧縮は、Amazon Redshift を宛先とする Firehose ストリームには使用できません。

S3 のファイル拡張子の形式 (オプション)

S3 のファイル拡張子の形式 (オプション) – Amazon S3 宛先バケットに配信されるオブジェクトのファイル拡張子の形式を指定します。この機能を有効にすると、指定されたファイル拡張子は、データ形式変換または .parquet や .gz などの S3 圧縮機能によって付加されたデフォルトのファイル拡張子を上書きします。この機能をデータ形式変換または S3 圧縮で使用する際には、適切なファイル拡張子を設定しているかどうかを確認します。ファイル拡張子はピリオド (.) で始まらなければならず、次の文字を含めることができます: 0-9a-z!-_.*'()。ファイル拡張子は最大 128 文字です。

S3 暗号化

Firehose は、Amazon S3 で配信されたデータを暗号化するための AWS Key Management Service (SSE-KMS) による Amazon S3 サーバー側の暗号化をサポートしています。送信先 S3 バケットで指定されたデフォルトの暗号化タイプを使用するか、所有するキーのリストから AWS KMS キーで暗号化するかを選択できます。 AWS KMS キーでデータを暗号化する場合は、デフォルトの AWS マネージドキー (aws/s3) またはカスタマーマネージドキーを使用できます。詳細については、AWS 「KMS マネージドキーによるサーバー側の暗号化を使用したデータの保護 (SSE-KMS)」を参照してください。

## OpenSearch Service の宛先の設定を構成する

Firehose は、Elasticsearch バージョン 1.5、2.3、5.1、5.3、5.5、5.6 と、すべての 6.*、7.*、8.* バージョンをサポートします。Firehose は Amazon OpenSearch Service 2.x および 3.x をサポートしています。

このセクションでは、送信先に OpenSearch Service を使用するためのオプションについて説明します。

- 以下のフィールドに値を入力します。

  OpenSearch Service ドメイン

  データの配信先となる OpenSearch Service ドメイン。

  インデックス

  OpenSearch Service クラスターに対してデータのインデックスを作成するときに使用する OpenSearch Service のインデックス名。

  インデックスのローテーション

  OpenSearch Service インデックスをローテーションするかどうかと、その頻度を選択します。インデックスのローテーションが有効になっている場合、Amazon Data Firehose は指定されたインデックス名に対応するタイムスタンプを追加し、ローテーションします。詳細については、「OpenSearch Service のインデックスローテーションを設定する」を参照してください。

  タイプ

  OpenSearch Service クラスターに対してデータのインデックスを作成するときに使用する OpenSearch Service タイプ名。Elasticsearch 7.x と OpenSearch 1.x では、インデックスあたり 1 つのタイプのみが存在できます。既に別のタイプを持つ既存のインデックスに新しいタイプを指定しようとすると、Firehose は実行時にエラーを戻します。

  Elasticsearch 7.x では、このフィールドは空のままにしておきます。

  再試行の期間

  OpenSearch へのインデックスのリクエストが失敗した場合に Firehose が再試行する時間。再試行期間については、0 ~ 7,200 秒の任意の値を設定できます。デフォルトの再試行期間は 300 秒です。Firehose は、再試行期間が終了するまで、エクスポネンシャルバックオフで複数回再試行します。

  再試行期間が終了すると、Firehose は、設定された S3 エラーバケットであるデッドレターキュー (DLQ) にデータを配信します。DLQ に配信されるデータについては、設定された S3 エラーバケットから OpenSearch の宛先にデータをリドライブする必要があります。

OpenSearch クラスターのダウンタイムまたはメンテナンスを理由として、Firehose ストリームによる DLQ へのデータ配信をブロックしたい場合は、再試行期間を秒単位でより大きい値に設定できます。AWS サポートに問い合わせて、再試行期間を 7,200 秒超に増やすことができます。

DocumentID タイプ

ドキュメント ID を設定する方法を示します。サポートされている方法は、Firehose が生成したドキュメント ID と OpenSearch Service が生成したドキュメント ID です。Firehose が生成したドキュメント ID は、ドキュメント ID の値が設定されていない場合のデフォルトオプションです。推奨されている方法は OpenSearch Service が生成したドキュメント ID です。ログ分析やオブザーバビリティなど書き込み負荷が高いオペレーションに対応しており、OpenSearch Service ドメインで消費される CPU リソースが少なく、パフォーマンスが向上するためです。

送信先 VPC 接続

OpenSearch Service ドメインがプライベート VPC 内にある場合は、このセクションを使用してその VPC を指定します。また、OpenSearch Service ドメインにデータを送信するときに Amazon Data Firehose で使用するサブネットとサブグループも指定します。OpenSearch Service ドメインで使用しているものと同じセキュリティグループを使用できます。別のセキュリティグループを指定する場合は、そのセキュリティグループで、OpenSearch Service ドメインのセキュリティグループへのアウトバウンド HTTPS トラフィックを必ず許可します。また、OpenSearch Service ドメインのセキュリティグループで、Firehose ストリームの設定時に指定したセキュリティグループからの HTTPS トラフィックを必ず許可します。Firehose ストリームと OpenSearch Service ドメインの両方に同じセキュリティグループを使用する場合は、セキュリティグループのインバウンドルールで HTTPS トラフィックを必ず許可します。セキュリティグループのルールの詳細については、Amazon VPCドキュメントの「セキュリティグループのルール」を参照してください。

> **Important**
>
> プライベート VPC の宛先にデータを配信するためのサブネットを指定する場合は、選択したサブネットに十分な数の空き IP アドレスがあることを確認してください。指定されたサブネットに使用可能な空き IP アドレスがない場合、Firehose はプライベート VPC でデータ配信用の ENI を作成または追加できず、配信のパフォーマンスが低下するか、または配信が失敗します。

バッファリングのヒント

Amazon Data Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。送信先の推奨バッファサイズは、サービスプロバイダーによって異なります。

## OpenSearch Serverless の宛先の設定を構成する

このセクションでは、送信先に OpenSearch Serverless を使用する方法ついて説明します。

- 以下のフィールドに値を入力します。

  OpenSearch Serverless コレクション

  データの配信先となる OpenSearch Serverless インデックスのグループのエンドポイント。

  [Index] (インデックス)

  OpenSearch Serverless コレクションにデータのインデックスを作成するときに使用する OpenSearch Serverless のインデックス名。

  送信先 VPC 接続

  OpenSearch Serverless コレクションがプライベート VPC 内にある場合は、このセクションを使用してその VPC を指定します。また、OpenSearch Serverless コレクションにデータを送信するときに Amazon Data Firehose で使用するサブネットとサブグループも指定します。

  > **Important**
  > プライベート VPC の宛先にデータを配信するためのサブネットを指定する場合は、選択したサブネットに十分な数の空き IP アドレスがあることを確認してください。指定されたサブネットに使用可能な空き IP アドレスがない場合、Firehose はプライベート VPC でデータ配信用の ENI を作成または追加できず、配信のパフォーマンスが低下するか、または配信が失敗します。

再試行の期間

OpenSearch Serverless へのインデックスのリクエストが失敗した場合に Firehose が再試行する時間。再試行期間については、0～7,200 秒の任意の値を設定できます。デフォルトの再試行期間は 300 秒です。Firehose は、再試行期間が終了するまで、エクスポネンシャルバックオフで複数回再試行します。

再試行期間が終了すると、Firehose は、設定された S3 エラーバケットであるデッドレターキュー (DLQ) にデータを配信します。DLQ に配信されるデータについては、設定された S3 エラーバケットから OpenSearch Serverless の宛先にデータをリドライブする必要があります。

OpenSearch Serverless クラスターのダウンタイムまたはメンテナンスを理由として、Firehose ストリームによる DLQ へのデータ配信をブロックしたい場合は、再試行期間を秒単位でより大きい値に設定できます。AWS サポートに問い合わせて、再試行期間を 7,200 秒超に増やすことができます。

バッファリングのヒント

Amazon Data Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。送信先の推奨バッファサイズは、サービスプロバイダーによって異なります。

## HTTP エンドポイントの宛先の設定を構成する

このセクションでは、送信先に HTTP エンドポイントを使用するためのオプションについて説明します。

> **Important**
>
> HTTP エンドポイントを送信先として選択した場合は、HTTP エンドポイント配信リクエストとレスポンスの仕様を理解する の手順を確認して従ってください。

- 以下のフィールドに値を入力します。

HTTP エンドポイント名 - オプション

HTTP エンドポイントのわかりやすい名前を指定します。例えば、`My HTTP Endpoint Destination`。

HTTP エンドポイント URL

HTTP エンドポイントの URL を次の形式で指定します: `https://xyz.httpendpoint.com`。URL は HTTPS URL であることが必要です。

認証

アクセスキーを直接入力するか、 からシークレットを取得 AWS Secrets Manager して HTTP エンドポイントにアクセスするかを選択できます。

- (オプション) アクセスキー

  Firehose からエンドポイントへのデータ配信を有効にするためにアクセスキーを取得する必要がある場合は、エンドポイントの所有者に連絡します。

- シークレット

  HTTP エンドポイントのアクセスキー AWS Secrets Manager を含むシークレットを から選択します。ドロップダウンリストにシークレットが表示されない場合は、アクセスキー AWS Secrets Manager の にシークレットを作成します。詳細については、「Amazon Data Firehose AWS Secrets Manager で を使用して認証する」を参照してください。

コンテンツのエンコーディング

Amazon Data Firehose は、リクエストを宛先に送信する前に、コンテンツのエンコードを使用してリクエストの本文を圧縮します。[GZIP] または [Disabled (無効)] をクリックして、リクエストのコンテンツエンコーディングを有効/無効にします。

再試行の期間

選択された HTTP エンドポイントへのデータの送信を Amazon Data Firehose が再試行する時間を指定します。

データの送信後、Amazon Data Firehose はまず HTTP エンドポイントからの送達確認を待機します。エラーが発生した場合、または確認タイムアウト期間内に確認が到着しない場合、Amazon Data Firehose で再試行期間カウンターが開始されます。再試行期間が終わるまで再試行が続けられます。その後、Amazon Data Firehose はデータ配信が失敗したとみなしてデータを Amazon S3 バケットにバックアップします。

Amazon Data Firehose がデータを HTTP エンドポイントに送信するたびに (初回か再試行かにかかわらず)、確認応答タイムアウトカウンターが再度開始され、HTTP エンドポイントからの確認応答を待機します。

再試行期間が切れた場合でも、Amazon Data Firehose は確認応答が到着するか確認応答タイムアウト期間に達するまで確認応答を待機し続けます。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターの残り時間があるかどうかを判別します。残り時間がある場合は、確認が到着するか再試行時間が切れたと判断されるまで再試行され口ジックが繰り返されます。

Amazon Data Firehose でデータの送信を再試行しない場合は、この値を 0 に設定します。

Parameters - オプション

Amazon Data Firehose では、各 HTTP コールにこれらのキーと値のペアが含まれます。これらのパラメータを使用すると、送信先の識別や整理に役立ちます。

バッファリングのヒント

Amazon Data Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。送信先の推奨バッファサイズは、サービスプロバイダーによって異なります。

> **Important**
>
> HTTP エンドポイントの宛先で、CloudWatch Logs で宛先エンドポイントからの 413 のレスポンスコードが表示される場合は、Firehose ストリームのバッファリングのヒントサイズを小さくして、もう一度試してください。

## Datadog の宛先の設定を構成する

このセクションでは、送信先に Datadog を使用するためのオプションについて説明します。Datadog の詳細については、「https://docs.datadoghq.com/integrations/amazon_web_services/」を参照してください。

- 次のフィールドに値を入力します。

HTTP エンドポイント URL

ドロップダウンメニューで、次のいずれかのオプションからデータを送信する場所を選択します。

- Datadog ログ - US1
- Datadog ログ - US3
- Datadog ログ - US5
- Datadog ログ - AP1
- Datadog ログ - EU
- Datadog ログ - GOV
- Datadog メトリクス - 米国
- Datadog メトリクス - US5
- Datadog メトリクス - AP1
- Datadog メトリクス - EU
- Datadog の設定 - US1
- Datadog の設定 - US3
- Datadog の設定 - US5
- Datadog の設定 - AP1
- Datadog の設定 - EU
- Datadog の設定 - US GOV

認証

API キーを直接入力するか、 からシークレットを取得 AWS Secrets Manager して Datadog にアクセスするかを選択できます。

- API キー

  Datadog に連絡し、Firehose からこのエンドポイントへのデータ配信を有効にするために必要な API キーを取得します。

- シークレット

  Datadog の API キー AWS Secrets Manager を含むシークレットを から選択します。ドロップダウンリストにシークレットが表示されない場合は、 AWS Secrets Managerで

シークレットを作成します。詳細については、「Amazon Data Firehose AWS Secrets Manager で を使用して認証する」を参照してください。

コンテンツのエンコーディング

Amazon Data Firehose は、リクエストを宛先に送信する前に、コンテンツのエンコードを使用してリクエストの本文を圧縮します。[GZIP] または [Disabled (無効)] をクリックして、リクエストのコンテンツエンコーディングを有効/無効にします。

再試行の期間

選択された HTTP エンドポイントへのデータの送信を Amazon Data Firehose が再試行する時間を指定します。

データの送信後、Amazon Data Firehose はまず HTTP エンドポイントからの送達確認を待機します。エラーが発生した場合、または確認タイムアウト期間内に確認が到着しない場合、Amazon Data Firehose で再試行期間カウンターが開始されます。再試行期間が終わるまで再試行が続けられます。その後、Amazon Data Firehose はデータ配信が失敗したとみなしてデータを Amazon S3 バケットにバックアップします。

Amazon Data Firehose がデータを HTTP エンドポイントに送信するたびに (初回か再試行かにかかわらず)、確認応答タイムアウトカウンターが再度開始され、HTTP エンドポイントからの確認応答を待機します。

再試行期間が切れた場合でも、Amazon Data Firehose は確認応答が到着するか確認応答タイムアウト期間に達するまで確認応答を待機し続けます。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターの残り時間があるかどうかを判別します。残り時間がある場合は、確認が到着するか再試行時間が切れたと判断されるまで再試行されロジックが繰り返されます。

Amazon Data Firehose でデータの送信を再試行しない場合は、この値を 0 に設定します。

Parameters - オプション

Amazon Data Firehose では、各 HTTP コールにこれらのキーと値のペアが含まれます。これらのパラメータを使用すると、送信先の識別や整理に役立ちます。

バッファリングのヒント

Amazon Data Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。送信先の推奨バッファサイズは、サービスプロバイダーによって異なります。

# Honeycomb の宛先の設定を構成する

このセクションでは、送信先に Honeycomb を使用する方法について説明します。Honeycomb の詳細については、https://docs.honeycomb.io/getting-data-in/metrics/aws-cloudwatch-metrics/ を参照してください。

- 以下のフィールドに値を入力します。

  Honeycomb Kinesis エンドポイント

  HTTP エンドポイントの URL を次の形式で指定します。: https://api.honeycomb.io/1/kinesis_events/{{dataset}}

  認証

  API キーを直接入力するか、 からシークレットを取得 AWS Secrets Manager して Honeycomb にアクセスするかを選択できます。

  - API キー

    Honeycomb に連絡し、Firehose からこのエンドポイントへのデータ配信を有効にするために必要な API キーを取得します。

  - シークレット

    Honeycomb の API キー AWS Secrets Manager を含むシークレットを から選択します。ドロップダウンリストにシークレットが表示されない場合は、 AWS Secrets Manager でシークレットを作成します。詳細については、「Amazon Data Firehose AWS Secrets Manager で を使用して認証する」を参照してください。

  コンテンツのエンコーディング

  Amazon Data Firehose は、リクエストを宛先に送信する前に、コンテンツのエンコードを使用してリクエストの本文を圧縮します。[GZIP] を選択して、リクエストのコンテンツエンコーディングを有効にします。こちらは、Honeycomb が送信先である場合に推奨される方法です。

  再試行の期間

  選択された HTTP エンドポイントへのデータの送信を Amazon Data Firehose が再試行する時間を指定します。

データの送信後、Amazon Data Firehose はまず HTTP エンドポイントからの送達確認を待機します。エラーが発生した場合、または確認タイムアウト期間内に確認が到着しない場合、Amazon Data Firehose で再試行期間カウンターが開始されます。再試行期間が終わるまで再試行が続けられます。その後、Amazon Data Firehose はデータ配信が失敗したとみなしてデータを Amazon S3 バケットにバックアップします。

Amazon Data Firehose がデータを HTTP エンドポイントに送信するたびに (初回か再試行かにかかわらず)、確認応答タイムアウトカウンターが再度開始され、HTTP エンドポイントからの確認応答を待機します。

再試行期間が切れた場合でも、Amazon Data Firehose は確認応答が到着するか確認応答タイムアウト期間に達するまで確認応答を待機し続けます。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターの残り時間があるかどうかを判別します。残り時間がある場合は、確認が到着するか再試行時間が切れたと判断されるまで再試行され口ジックが繰り返されます。

Amazon Data Firehose でデータの送信を再試行しない場合は、この値を 0 に設定します。

Parameters - オプション

Amazon Data Firehose では、各 HTTP コールにこれらのキーと値のペアが含まれます。これらのパラメータを使用すると、送信先の識別や整理に役立ちます。

バッファリングのヒント

Amazon Data Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。送信先の推奨バッファサイズは、サービスプロバイダーによって異なります。

## Coralogix の宛先の設定を構成する

このセクションでは、送信先に Coralogix を使用する方法について説明します。Coralogix の詳細については、「Get Started with Coralogix」を参照してください。

- 以下のフィールドに値を入力します。

  HTTP エンドポイント URL

  ドロップダウンメニューの次のオプションから HTTP エンドポイント URL を選択します。

  - Coralogix - 米国

- Coralogix - シンガポール
- Coralogix - アイルランド
- Coralogix - インド
- Coralogix - ストックホルム

認証

プライベートキーを直接入力するか、 からシークレットを取得 AWS Secrets Manager して Coralogix にアクセスできます。

- プライベートキー

  Coralogix に連絡し、Firehose からこのエンドポイントへのデータ配信を有効にするために必要なプライベートキーを取得します。

- シークレット

  Coralogix のプライベートキー AWS Secrets Manager を含むシークレットを から選択します。ドロップダウンリストにシークレットが表示されない場合は、 AWS Secrets Manager でシークレットを作成します。詳細については、「Amazon Data Firehose AWS Secrets Manager で を使用して認証する」を参照してください。

コンテンツのエンコーディング

Amazon Data Firehose は、リクエストを宛先に送信する前に、コンテンツのエンコードを使用してリクエストの本文を圧縮します。[GZIP] を選択して、リクエストのコンテンツエンコーディングを有効にします。こちらは、Coralogix が送信先である場合に推奨される方法です。

再試行の期間

選択された HTTP エンドポイントへのデータの送信を Amazon Data Firehose が再試行する時間を指定します。

データの送信後、Amazon Data Firehose はまず HTTP エンドポイントからの送達確認を待機します。エラーが発生した場合、または確認タイムアウト期間内に確認が到着しない場合、Amazon Data Firehose で再試行期間カウンターが開始されます。再試行期間が終わるまで再試行が続けられます。その後、Amazon Data Firehose はデータ配信が失敗したとみなしてデータを Amazon S3 バケットにバックアップします。

Amazon Data Firehose がデータを HTTP エンドポイントに送信するたびに (初回か再試行かにかかわらず)、確認応答タイムアウトカウンターが再度開始され、HTTP エンドポイントからの確認応答を待機します。

再試行期間が切れた場合でも、Amazon Data Firehose は確認応答が到着するか確認応答タイムアウト期間に達するまで確認応答を待機し続けます。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターの残り時間があるかどうかを判別します。残り時間がある場合は、確認が到着するか再試行時間が切れたと判断されるまで再試行され口ジックが繰り返されます。

Amazon Data Firehose でデータの送信を再試行しない場合は、この値を 0 に設定します。

Parameters - オプション

Amazon Data Firehose では、各 HTTP コールにこれらのキーと値のペアが含まれます。これらのパラメータを使用すると、送信先の識別や整理に役立ちます。

- applicationName: Data Firehose を実行している環境
- subsystemName: Data Firehose 統合の名前
- computerName: 使用中の Firehose ストリームの名前

バッファリングのヒント

Amazon Data Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。宛先に推奨されるバッファリングサイズは、サービスプロバイダーによって異なります。

## Dynatrace の宛先の設定を構成する

このセクションでは、送信先に Dynatrace を使用するためのオプションについて説明します。詳細については、「https://www.dynatrace.com/support/help/technology-support/cloud-platforms/amazon-web-services/integrations/cloudwatch-metric-streams/」を参照してください。

- Firehose ストリームの宛先として Dynatrace を使用するオプションを選択します。

  取り込みタイプ

  さらなる分析と処理のために Dynatrace で [メトリクス] または [ログ] (デフォルト) を配信するかどうかを選択します。

HTTP エンドポイント URL

ドロップダウンメニューから、HTTP エンドポイント URL (Dynatrace US、Dynatrace EU、または Dynatrace Global) を選択します。

認証

API トークンを直接入力するか、 からシークレットを取得 AWS Secrets Manager して Dynatrace にアクセスするかを選択できます。

- API トークン

  Firehose からこのエンドポイントへのデータ配信を有効にするために必要な Dynatrace API トークンを生成します。詳細については、「Dynatrace API - Tokens and authentication」を参照してください。

- シークレット

  Dynatrace の API トークン AWS Secrets Manager を含むシークレットを から選択します。ドロップダウンリストにシークレットが表示されない場合は、 AWS Secrets Manager でシークレットを作成します。詳細については、「Amazon Data Firehose AWS Secrets Manager で を使用して認証する」を参照してください。

API URL

Dynatrace 環境の API URL を指定します。

コンテンツのエンコーディング

リクエストの本文を圧縮するためにコンテンツエンコーディングを有効にするかどうかを選択します。Amazon Data Firehose は、リクエストを宛先に送信する前に、コンテンツのエンコードを使用してリクエストの本文を圧縮します。有効にすると、圧縮したコンテンツが GZIP 形式で表示されます。

再試行の期間

選択された HTTP エンドポイントへのデータの送信を Firehose が再試行する時間を指定します。

データの送信後、Firehose はまず HTTP エンドポイントからの送達確認を待機します。エラーが発生した場合、または確認タイムアウト期間内に確認が到着しない場合、Firehose は、再試行期間カウンターを開始します。再試行期間が終わるまで再試行が続けられます。その後、Firehose はデータ配信が失敗したとみなしてデータを Amazon S3 バケットにバックアップします。

Firehose がデータを HTTP エンドポイントに送信するたびに (初回試行中または再試行後のいずれかにかかわらず)、確認応答タイムアウトカウンターが再度開始され、HTTP エンドポイントからの確認応答を待機します。

再試行期間が切れた場合でも、Firehose は、確認応答が到着するか確認応答タイムアウト期間に達するまで確認応答を待機し続けます。確認がタイムアウトすると、Firehose は再試行カウンターの残り時間があるかどうかを判別します。残り時間がある場合は、確認が到着するか再試行時間が切れたと判断されるまで再試行されロジックが繰り返されます。

Firehose でデータの送信を再試行しない場合は、この値を 0 に設定します。

Parameters - オプション

Amazon Data Firehose では、各 HTTP コールにこれらのキーと値のペアが含まれます。これらのパラメータを使用すると、送信先の識別や整理に役立ちます。

バッファリングのヒント

Amazon Data Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。バッファリングのヒントには、ストリームのバッファリングサイズと間隔が含まれます。宛先に推奨されるバッファリングサイズは、サービスプロバイダーによって異なります。

## LogicMonitor の宛先の設定を構成する

このセクションでは、送信先に LogicMonitor を使用するためのオプションについて説明します。詳細については、「https://www.logicmonitor.com」を参照してください。

- 以下のフィールドに値を入力します。

  HTTP エンドポイント URL

  HTTP エンドポイントの URL を次の形式で指定します。

  ```
  https://ACCOUNT.logicmonitor.com
  ```

認証

API キーを直接入力するか、 からシークレットを取得 AWS Secrets Manager して LogicMonitor にアクセスするかを選択できます。

- API キー

  LogicMonitor に連絡し、Firehose からこのエンドポイントへのデータ配信を有効にするために必要な API キーを取得します。

- シークレット

  LogicMonitor の API キー AWS Secrets Manager を含むシークレットを から選択します。ドロップダウンリストにシークレットが表示されない場合は、 AWS Secrets Manager でシークレットを作成します。詳細については、「Amazon Data Firehose AWS Secrets Manager で を使用して認証する」を参照してください。

コンテンツのエンコーディング

Amazon Data Firehose は、リクエストを宛先に送信する前に、コンテンツのエンコードを使用してリクエストの本文を圧縮します。[GZIP] または [Disabled (無効)] をクリックして、リクエストのコンテンツエンコーディングを有効/無効にします。

再試行の期間

選択された HTTP エンドポイントへのデータの送信を Amazon Data Firehose が再試行する時間を指定します。

データの送信後、Amazon Data Firehose はまず HTTP エンドポイントからの送達確認を待機します。エラーが発生した場合、または確認タイムアウト期間内に確認が到着しない場合、Amazon Data Firehose で再試行期間カウンターが開始されます。再試行期間が終わるまで再試行が続けられます。その後、Amazon Data Firehose はデータ配信が失敗したとみなしてデータを Amazon S3 バケットにバックアップします。

Amazon Data Firehose がデータを HTTP エンドポイントに送信するたびに (初回か再試行かにかかわらず)、確認応答タイムアウトカウンターが再度開始され、HTTP エンドポイントからの確認応答を待機します。

再試行期間が切れた場合でも、Amazon Data Firehose は確認応答が到着するか確認応答タイムアウト期間に達するまで確認応答を待機し続けます。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターの残り時間があるかどうかを判別します。残

り時間がある場合は、確認が到着するか再試行時間が切れたと判断されるまで再試行され口ジックが繰り返されます。

Amazon Data Firehose でデータの送信を再試行しない場合は、この値を 0 に設定します。

Parameters - オプション

Amazon Data Firehose では、各 HTTP コールにこれらのキーと値のペアが含まれます。これらのパラメータを使用すると、送信先の識別や整理に役立ちます。

バッファリングのヒント

Amazon Data Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。送信先の推奨バッファサイズは、サービスプロバイダーによって異なります。

## Logz.io の宛先の設定を構成する

このセクションでは、送信先に Logz.io を使用する方法について説明します。詳細については、https://logz.io/ を参照してください。

> **Note**
>
> 欧州 (ミラノ) リージョンでは、Logz.io は Amazon Data Firehose の宛先としてサポートされていません。

- 以下のフィールドに値を入力します。

  HTTP エンドポイント URL

  HTTP エンドポイントの URL を次の形式で指定します。URL は HTTPS URL である必要があります。

  ```
  https://listener-aws-metrics-stream-<region>.logz.io/
  ```

  例

  ```
  https://listener-aws-metrics-stream-us.logz.io/
  ```

認証

AWS Secrets Manager にアクセスするには、配送トークンを直接入力するか、 からシークレットを取得することができます Logz.io 。

- シッピングトークン

  Firehose からこのエンドポイントへのデータ配信を有効にするために必要なシッピングトークンを取得するには、Logz.io にお問い合わせください。

- シークレット

  Logz.io の配送トークン AWS Secrets Manager を含むシークレットを から選択します。ドロップダウンリストにシークレットが表示されない場合は、 AWS Secrets Manager でシークレットを作成します。詳細については、「Amazon Data Firehose AWS Secrets Manager で を使用して認証する」を参照してください。

再試行の期間

Amazon Data Firehose が Logz.io へのデータ送信を再試行する期間を指定します。

データの送信後、Amazon Data Firehose はまず HTTP エンドポイントからの送達確認を待機します。エラーが発生した場合、または確認タイムアウト期間内に確認が到着しない場合、Amazon Data Firehose で再試行期間カウンターが開始されます。再試行期間が終わるまで再試行が続けられます。その後、Amazon Data Firehose はデータ配信が失敗したとみなしてデータを Amazon S3 バケットにバックアップします。

Amazon Data Firehose がデータを HTTP エンドポイントに送信するたびに (初回か再試行かにかかわらず)、確認応答タイムアウトカウンターが再度開始され、HTTP エンドポイントからの確認応答を待機します。

再試行期間が切れた場合でも、Amazon Data Firehose は確認応答が到着するか確認応答タイムアウト期間に達するまで確認応答を待機し続けます。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターの残り時間があるかどうかを判別します。残り時間がある場合は、確認が到着するか再試行時間が切れたと判断されるまで再試行され口ジックが繰り返されます。

Amazon Data Firehose でデータの送信を再試行しない場合は、この値を 0 に設定します。

Parameters - オプション

Amazon Data Firehose では、各 HTTP コールにこれらのキーと値のペアが含まれます。これらのパラメータを使用すると、送信先の識別や整理に役立ちます。

バッファリングのヒント

Amazon Data Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。送信先の推奨バッファサイズは、サービスプロバイダーによって異なります。

## MongoDB Atlas の送信先の設定を構成する

このセクションでは、送信先に MongoDB Atlas を使用するためのオプションについて説明します。詳細については、「MongoDB Atlas を Amazon Web Services で実行」を参照してください。

- 以下のフィールドに値を入力します。

  API Gateway URL

  HTTP エンドポイントの URL を次の形式で指定します。

  ```
  https://xxxxx.execute-api.region.amazonaws.com/stage
  ```

  URL は HTTPS URL である必要があります。

  認証

  API キーを直接入力するか、 からシークレットを取得 AWS Secrets Manager して MongoDB Atlas にアクセスするかを選択できます。

  - API キー

    「MongoDB Atlas を Amazon Web Services で実行」の指示に従って、Firehose からこのエンドポイントへのデータ配信を有効にするために必要な `APIKeyValue` を取得します。

  - シークレット

    MongoDB Atlas とやり取りする Lambda によってバックアップされる API Gateway の API キー AWS Secrets Manager を含むシークレットを から選択します。ドロップダウンリストにシークレットが表示されない場合は、 AWS Secrets Managerでシークレットを作

成します。詳細については、「Amazon Data Firehose AWS Secrets Manager で を使用して認証する」を参照してください。

コンテンツのエンコーディング

Amazon Data Firehose は、リクエストを宛先に送信する前に、コンテンツのエンコードを使用してリクエストの本文を圧縮します。[GZIP] または [Disabled (無効)] をクリックして、リクエストのコンテンツエンコーディングを有効/無効にします。

再試行の期間

Amazon Data Firehose が、選択したサードパーティープロバイダーへのデータの送信を再試行する期間を指定します。

データの送信後、Amazon Data Firehose はまず HTTP エンドポイントからの送達確認を待機します。エラーが発生した場合、または確認タイムアウト期間内に確認が到着しない場合、Amazon Data Firehose で再試行期間カウンターが開始されます。再試行期間が終わるまで再試行が続けられます。その後、Amazon Data Firehose はデータ配信が失敗したとみなしてデータを Amazon S3 バケットにバックアップします。

Amazon Data Firehose がデータを HTTP エンドポイントに送信するたびに (初回か再試行かにかかわらず)、確認応答タイムアウトカウンターが再度開始され、HTTP エンドポイントからの確認応答を待機します。

再試行期間が切れた場合でも、Amazon Data Firehose は確認応答が到着するか確認応答タイムアウト期間に達するまで確認応答を待機し続けます。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターの残り時間があるかどうかを判別します。残り時間がある場合は、確認が到着するか再試行時間が切れたと判断されるまで再試行されロジックが繰り返されます。

Amazon Data Firehose でデータの送信を再試行しない場合は、この値を 0 に設定します。

バッファリングのヒント

Amazon Data Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。送信先の推奨バッファサイズは、サービスプロバイダーによって異なります。

Parameters - オプション

Amazon Data Firehose では、各 HTTP コールにこれらのキーと値のペアが含まれます。これらのパラメータを使用すると、送信先の識別や整理に役立ちます。

# New Relic の宛先の設定を構成する

このセクションでは、送信先に New Relic を使用するためのオプションについて説明します。詳細については、https://newrelic.com を参照してください。

- 以下のフィールドに値を入力します。

  HTTP エンドポイント URL

  ドロップダウンリストの次のオプションから HTTP エンドポイント URL を選択します。

  - New Relic ログ - 米国
  - New Relic メトリクス - 米国
  - New Relic メトリクス - EU

  認証

  API キーを直接入力するか、 からシークレットを取得 AWS Secrets Manager して New Relic にアクセスするかを選択できます。

  - API キー

    New Relic One アカウント設定から、40 文字の 16 進文字列であるライセンスキーを入力します。Firehose からこのエンドポイントへのデータ配信を有効にするには、この API キーが必要です。

  - シークレット

    New Relic の API キー AWS Secrets Manager を含むシークレットを から選択します。ドロップダウンリストにシークレットが表示されない場合は、 AWS Secrets Manager でシークレットを作成します。詳細については、「Amazon Data Firehose AWS Secrets Manager で を使用して認証する」を参照してください。

  コンテンツのエンコーディング

  Amazon Data Firehose は、リクエストを宛先に送信する前に、コンテンツのエンコードを使用してリクエストの本文を圧縮します。[GZIP] または [Disabled (無効)] をクリックして、リクエストのコンテンツエンコーディングを有効/無効にします。

  再試行の期間

  Amazon Data Firehose が New Relic HTTP エンドポイントへのデータの送信を再試行する時間を指定します。

データの送信後、Amazon Data Firehose はまず HTTP エンドポイントからの送達確認を待機します。エラーが発生した場合、または確認タイムアウト期間内に確認が到着しない場合、Amazon Data Firehose で再試行期間カウンターが開始されます。再試行期間が終わるまで再試行が続けられます。その後、Amazon Data Firehose はデータ配信が失敗したとみなしてデータを Amazon S3 バケットにバックアップします。

Amazon Data Firehose がデータを HTTP エンドポイントに送信するたびに (初回か再試行かにかかわらず)、確認応答タイムアウトカウンターが再度開始され、HTTP エンドポイントからの確認応答を待機します。

再試行期間が切れた場合でも、Amazon Data Firehose は確認応答が到着するか確認応答タイムアウト期間に達するまで確認応答を待機し続けます。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターの残り時間があるかどうかを判別します。残り時間がある場合は、確認が到着するか再試行時間が切れたと判断されるまで再試行され口ジックが繰り返されます。

Amazon Data Firehose でデータの送信を再試行しない場合は、この値を 0 に設定します。

Parameters - オプション

Amazon Data Firehose では、各 HTTP コールにこれらのキーと値のペアが含まれます。これらのパラメータを使用すると、送信先の識別や整理に役立ちます。

バッファリングのヒント

Amazon Data Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。送信先の推奨バッファサイズは、サービスプロバイダーによって異なります。

## Snowflake の宛先の設定を構成する

このセクションでは、宛先に Snowflake を使用するためのオプションについて説明します。

> **Note**
>
> Firehose と Snowflake の統合は、米国東部 (バージニア北部 ) 、米国西部 (オレゴン ) 、欧州 (アイルランド ) 、米国東部 (オハイオ ) 、アジアパシフィック (東京 ) 、欧州 (フランクフルト ) 、アジアパシフィック (シンガポール ) 、アジアパシフィック (ソウル ) 、アジアパシフィック (シドニー ) 、アジアパシフィック (ムンバイ ) 、欧州 (ロンドン ) 、南米 (サン

パウロ）、カナダ (中部）、欧州 (パリ）、アジアパシフィック (大阪）、欧州 (ストックホルム）、アジアパシフィック (ジャカルタ) で利用できます AWS リージョン。

接続の設定

- 以下のフィールドに値を入力します。

  Snowflake アカウント URL

  Snowflake アカウントの URL を指定します。例: `xy12345.us-east-1.aws.snowflakecomputing.com`。アカウント URL を特定する方法については、Snowflake のドキュメントを参照してください。ポート番号を指定してはなりませんが、プロトコル (https://) はオプションであることに留意してください。

  認証

  ユーザーログイン、プライベートキー、パスフレーズを手動で入力するか、 からシークレットを取得 AWS Secrets Manager して Snowflake にアクセスするかを選択できます。

  - ユーザーログイン

    データのロードに使用する Snowflake ユーザーを指定します。Snowflake テーブルにデータを挿入するためのアクセスがユーザーに付与されていることを確認します。

  - プライベートキー

    Snowflake で認証するためのプライベートキーを PKCS8 形式で指定します。さらに、プライベートキーの一部として PEM ヘッダーとフッターを含めないでください。キーが複数の行に分割されている場合は、改行を削除します。プライベートキーがどのように表示されるかの例を次に示します。

    ```
    -----BEGIN PRIVATE KEY-----
    KEY_CONTENT
    -----END PRIVATE KEY-----
    ```

    KEY_CONTENT のスペースを削除し、それを Firehose に提供します。ヘッダー/フッターや改行文字は必要ありません。

  - パスフレーズ

暗号化されたプライベートキーを復号するパスフレーズを指定します。プライベートキーが暗号化されていない場合は、このフィールドを空のままにできます。詳細については、「Using Key Pair Authentication & Key Rotation」を参照してください。

- シークレット

  Snowflake の認証情報 AWS Secrets Manager を含むシークレットを から選択します。ドロップダウンリストにシークレットが表示されない場合は、 AWS Secrets Manager でシークレットを作成します。詳細については、「Amazon Data Firehose AWS Secrets Manager で を使用して認証する」を参照してください。

ロールの設定

デフォルトの Snowflake ロールを使用 – このオプションが選択されている場合、Firehose は Snowflake にロールを渡しません。デフォルトのロールは、データをロードするために引き受けられます。Snowflake テーブルにデータを挿入するための許可がデフォルトのロールに付与されていることを確認してください。

カスタム Snowflake ロールを使用 – Snowflake テーブルにデータをロードする際に Firehose が引き受けるデフォルト以外の Snowflake ロールを入力します。

Snowflake 接続

オプションは、[プライベート] または [パブリック] です。

プライベートVPCE ID (オプション)

Firehose が Snowflake とプライベートに接続するための VPCE ID。ID の形式は com.amazonaws.vpce.[region].vpce-svc-*[id]* です。詳細については、「AWS PrivateLink & Snowflake」を参照してください。

> **Note**
>
> Snowflake クラスターでプライベートリンクが有効になっている場合は、`AwsVpceIds` ベースのネットワークポリシーを使用して Amazon Data Firehose データを許可します。Firehose では、Snowflake アカウントで IP ベースのネットワークポリシーを設定する必要はありません。IP ベースのネットワークポリシーを有効にすると、Firehose の接続が妨げられる可能性があります。IP ベースのポリシーを必要とするエッジケースがある場合は、サポートチケットを送信して

Firehose チームにお問い合わせください。使用できる VPCE ID のリストについては、「VPC での Snowflake へのアクセス」を参照してください。

データベース設定

- Firehose ストリームの宛先として Snowflake を使用するには、次の設定を指定する必要があります。
  - Snowflake データベース - Snowflake のすべてのデータはデータベースに保持されます。
  - Snowflake スキーマ – 各データベースは 1 つ以上のスキーマで構成され、テーブルやビューなどのデータベースオブジェクトの論理グループです。
  - Snowflake テーブル – Snowflake のすべてのデータはデータベーステーブルに保存され、列と行のコレクションとして論理的に構造化されます。

Snowflake テーブルのデータロードオプション

- 列名として JSON キーを使用する
- VARIANT 列を使用する
  - コンテンツ列名 – テーブル内の列名を指定します。ここでは、未処理のデータをロードする必要があります。
  - メタデータ列名 (オプション) – テーブル内の列名を指定します。ここでは、メタデータ情報をロードする必要があります。このフィールドを有効にすると、ソースタイプに基づいて Snowflake テーブルに次の列が表示されます。

    Direct PUT をソースとする場合

```
{
"firehoseDeliveryStreamName" : "streamname",
"IngestionTime" : "timestamp"
}
```

    Kinesis データストリームをソースとする場合

```
{
"kinesisStreamName" : "streamname",
"kinesisShardId" : "Id",
"kinesisPartitionKey" : "key",
```

```
"kinesisSequenceNumber" : "1234",
"subsequenceNumber" : "2334",
"IngestionTime" : "timestamp"
}
```

再試行の期間

Snowflake サービスの問題を理由として、チャネルを開くこと、または Snowflake への配信のいずれかが失敗した場合に、Firehose が再試行する時間 (0 ~ 7,200 秒)。Firehose は、再試行期間が終了するまでエクスポネンシャルバックオフで再試行します。再試行期間を 0 (ゼロ) 秒に設定すると、Firehose は Snowflake の障害発生時に再試行せず、データを Amazon S3 エラーバケットにルーティングします。

バッファリングのヒント

Amazon Data Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。送信先の推奨バッファサイズは、サービスプロバイダーによって異なります。詳細については、「バッファリングのヒントを設定する」を参照してください。

## Splunk の宛先の設定を構成する

このセクションでは、送信先に Splunk を使用するためのオプションについて説明します。

> **Note**
>
> Firehose は、Classic Load Balancer または Application Load Balancer を使用して設定された Splunk クラスターにデータを配信します。

- 以下のフィールドに値を入力します。

  Splunk クラスターエンドポイント

  エンドポイントを確認するには、Splunk ドキュメントの「Configure Amazon Data Firehose to Send Data to the Splunk Platform」を参照してください。

  Splunk エンドポイントタイプ

  ほとんどの場合は `Raw endpoint` を選択します。を使用してデータを前処理 AWS Lambda し、イベントタイプ別に異なるインデックスにデータを送信する`Event endpoint`場合は、

を選択します。使用するエンドポイントについては、Splunk ドキュメントの「Configure Amazon Data Firehose to send data to the Splunk platform」を参照してください。

認証

認証トークンを直接入力するか、 からシークレットを取得 AWS Secrets Manager して Splunk にアクセスするかを選択できます。

- 認証トークン

  Amazon Data Firehose からデータを受信できる Splunk エンドポイントをセットアップするには、Splunk ドキュメントの「Installation and configuration overview for the Splunk Add-on for Amazon Data Firehose」を参照してください。この Firehose ストリームのエンドポイントを設定するときに Splunk から取得したトークンを保存し、ここで追加します。

- シークレット

  Splunk の認証トークン AWS Secrets Manager を含むシークレットを から選択します。ドロップダウンリストにシークレットが表示されない場合は、 AWS Secrets Manager でシークレットを作成します。詳細については、「Amazon Data Firehose AWS Secrets Manager で を使用して認証する」を参照してください。

HEC 送達確認のタイムアウト

Amazon Data Firehose が Splunk からのインデックス確認応答を待機する時間を指定します。Splunk が確認を送信しないままタイムアウトに達すると、Amazon Data Firehose ではデータ配信失敗とみなされます。その後、Amazon Data Firehose は設定された再試行期間値に従って、再試行するか、データを Amazon S3 バケットにバックアップします。

再試行の期間

Amazon Data Firehose が Splunk へのデータ送信を再試行する期間を指定します。

データの送信後、Amazon Data Firehose はまず Splunk からの送達確認を待機します。エラーが発生した場合、または確認タイムアウト期間内に確認が到着しない場合、Amazon Data Firehose で再試行期間カウンターが開始されます。再試行期間が終わるまで再試行が続けられます。その後、Amazon Data Firehose はデータ配信が失敗したとみなしてデータを Amazon S3 バケットにバックアップします。

Amazon Data Firehose がデータを Splunk に送信するたびに (初回か再試行かにかかわらず)、確認応答タイムアウトカウンターが再度開始され、Splunk からの確認応答を待機します。

再試行期間が切れた場合でも、Amazon Data Firehose は確認応答が到着するか確認応答タイムアウト期間に達するまで確認応答を待機し続けます。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターの残り時間があるかどうかを判別します。残り時間がある場合は、確認が到着するか再試行時間が切れたと判断されるまで再試行され口ジックが繰り返されます。

Amazon Data Firehose でデータの送信を再試行しない場合は、この値を 0 に設定します。

バッファリングのヒント

Amazon Data Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。宛先に推奨されるバッファリングサイズは、サービスプロバイダーによって異なります。

## Splunk Observability Cloud の宛先の設定を構成する

このセクションでは、送信先に Splunk Observability Cloud を使用する方法について説明します。詳細については、https://docs.splunk.com/observability/en/gdi/get-data-in/connect/aws/aws-apiconfig.html#connect-to-aws-using-the-splunk-observability-cloud-api を参照してください。

- 以下のフィールドに値を入力します。

  Cloud Ingest のエンドポイント URL

  Splunk Observability Cloud のリアルタイムデータインジェスト URL は、Splunk Observability のコンソールから、[Profile] > [Organizations] > [Profile ] の順にクリックすると見つけることができます。

  認証

  アクセストークンを直接入力するか、 からシークレットを取得 AWS Secrets Manager して Splunk Observability Cloud にアクセスするかを選択できます。

  - アクセストークン

    Splunk Observability コンソールの [設定] の [アクセストークン] から、INGEST 認可スコープを持つ Splunk Observability アクセストークンをコピーします。

  - シークレット

Splunk Observability Cloud のアクセストークン AWS Secrets Manager を含むシークレットを から選択します。ドロップダウンリストにシークレットが表示されない場合は、AWS Secrets Managerでシークレットを作成します。詳細については、「Amazon Data Firehose AWS Secrets Manager で を使用して認証する」を参照してください。

コンテンツのエンコーディング

Amazon Data Firehose は、リクエストを宛先に送信する前に、コンテンツのエンコードを使用してリクエストの本文を圧縮します。[GZIP] または [Disabled (無効)] をクリックして、リクエストのコンテンツエンコーディングを有効/無効にします。

再試行の期間

選択された HTTP エンドポイントへのデータの送信を Amazon Data Firehose が再試行する時間を指定します。

データの送信後、Amazon Data Firehose はまず HTTP エンドポイントからの送達確認を待機します。エラーが発生した場合、または確認タイムアウト期間内に確認が到着しない場合、Amazon Data Firehose で再試行期間カウンターが開始されます。再試行期間が終わるまで再試行が続けられます。その後、Amazon Data Firehose はデータ配信が失敗したとみなしてデータを Amazon S3 バケットにバックアップします。

Amazon Data Firehose がデータを HTTP エンドポイントに送信するたびに (初回か再試行かにかかわらず)、確認応答タイムアウトカウンターが再度開始され、HTTP エンドポイントからの確認応答を待機します。

再試行期間が切れた場合でも、Amazon Data Firehose は確認応答が到着するか確認応答タイムアウト期間に達するまで確認応答を待機し続けます。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターの残り時間があるかどうかを判別します。残り時間がある場合は、確認が到着するか再試行時間が切れたと判断されるまで再試行されロジックが繰り返されます。

Amazon Data Firehose でデータの送信を再試行しない場合は、この値を 0 に設定します。

Parameters - オプション

Amazon Data Firehose では、各 HTTP コールにこれらのキーと値のペアが含まれます。これらのパラメータを使用すると、送信先の識別や整理に役立ちます。

バッファリングのヒント

Amazon Data Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。送信先の推奨バッファサイズは、サービスプロバイダーによって異なります。

## Sumo Logic の宛先の設定を構成する

このセクションでは、送信先に Sumo Logic を使用するためのオプションについて説明します。詳細については、「https://www.sumologic.com」を参照してください。

- 以下のフィールドに値を入力します。

  HTTP エンドポイント URL

  HTTP エンドポイントの URL を次の形式で指定します: `https://deployment name.sumologic.net/receiver/v1/kinesis/dataType/access token`。URL は HTTPS URL であることが必要です。

  コンテンツのエンコーディング

  Amazon Data Firehose は、リクエストを宛先に送信する前に、コンテンツのエンコードを使用してリクエストの本文を圧縮します。[GZIP] または [Disabled (無効)] をクリックして、リクエストのコンテンツエンコーディングを有効/無効にします。

  再試行の期間

  Amazon Data Firehose が Sumo Logic へのデータ送信を再試行する期間を指定します。

  データの送信後、Amazon Data Firehose はまず HTTP エンドポイントからの送達確認を待機します。エラーが発生した場合、または確認タイムアウト期間内に確認が到着しない場合、Amazon Data Firehose で再試行期間カウンターが開始されます。再試行期間が終わるまで再試行が続けられます。その後、Amazon Data Firehose はデータ配信が失敗したとみなしてデータを Amazon S3 バケットにバックアップします。

  Amazon Data Firehose がデータを HTTP エンドポイントに送信するたびに (初回か再試行かにかかわらず)、確認応答タイムアウトカウンターが再度開始され、HTTP エンドポイントからの確認応答を待機します。

  再試行期間が切れた場合でも、Amazon Data Firehose は確認応答が到着するか確認応答タイムアウト期間に達するまで確認応答を待機し続けます。確認がタイムアウトする

と、Amazon Data Firehose は再試行カウンターの残り時間があるかどうかを判別します。残り時間がある場合は、確認が到着するか再試行時間が切れたと判断されるまで再試行され ロジックが繰り返されます。

Amazon Data Firehose でデータの送信を再試行しない場合は、この値を 0 に設定します。

Parameters - オプション

Amazon Data Firehose では、各 HTTP コールにこれらのキーと値のペアが含まれます。これらのパラメータを使用すると、送信先の識別や整理に役立ちます。

バッファリングのヒント

Amazon Data Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。Elastic を送信先とする場合の推奨のバッファサイズは、サービスプロバイダーに応じて異なります。

## Elastic の宛先の設定を構成する

このセクションでは、送信先に Elastic を使用する方法について説明します。

- 以下のフィールドに値を入力します。

  Elastic のエンドポイント URL

  HTTP エンドポイントの URL を次の形式で指定します: `https://<cluster-id>.es.<region>.aws.elastic-cloud.com`。URL は HTTPS URL であることが必要です。

  認証

  API キーを直接入力するか、 からシークレットを取得 AWS Secrets Manager して Elastic にアクセスするかを選択できます。

  - API キー

    Elastic に連絡し、Firehose からこのサービスへのデータ配信を有効にするために必要な API キーを取得します。

  - シークレット

    Elastic の API キー AWS Secrets Manager を含むシークレットを から選択します。ドロップダウンリストにシークレットが表示されない場合は、 AWS Secrets Managerでシーク

レットを作成します。詳細については、「Amazon Data Firehose AWS Secrets Manager で を使用して認証する」を参照してください。

コンテンツのエンコーディング

Amazon Data Firehose は、リクエストを宛先に送信する前に、コンテンツのエンコードを使用してリクエストの本文を圧縮します。[GZIP] (デフォルトによる選択) または [無効] を選択し、リクエストのコンテンツエンコーディングを有効/無効にします。

再試行の期間

Amazon Data Firehose が Elastic へのデータの送信を再試行する期間を指定します。

データの送信後、Amazon Data Firehose はまず HTTP エンドポイントからの送達確認を待機します。エラーが発生した場合、または確認タイムアウト期間内に確認が到着しない場合、Amazon Data Firehose で再試行期間カウンターが開始されます。再試行期間が終わるまで再試行が続けられます。その後、Amazon Data Firehose はデータ配信が失敗したとみなしてデータを Amazon S3 バケットにバックアップします。

Amazon Data Firehose がデータを HTTP エンドポイントに送信するたびに (初回か再試行かにかかわらず)、確認応答タイムアウトカウンターが再度開始され、HTTP エンドポイントからの確認応答を待機します。

再試行期間が切れた場合でも、Amazon Data Firehose は確認応答が到着するか確認応答タイムアウト期間に達するまで確認応答を待機し続けます。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターの残り時間があるかどうかを判別します。残り時間がある場合は、確認が到着するか再試行時間が切れたと判断されるまで再試行され口ジックが繰り返されます。

Amazon Data Firehose でデータの送信を再試行しない場合は、この値を 0 に設定します。

Parameters - オプション

Amazon Data Firehose では、各 HTTP コールにこれらのキーと値のペアが含まれます。これらのパラメータを使用すると、送信先の識別や整理に役立ちます。

バッファリングのヒント

Amazon Data Firehose は着信データをバッファリングしてから、指定された宛先にデータを送信します。Elastic の送信先の推奨バッファサイズは 1 MiB です。

# バックアップの設定を構成する

Amazon Data Firehose は、Amazon S3 を使用して、選択した宛先への配信を試みるすべてのデータまたは失敗したデータのみをバックアップします。

> **Important**
>
> - Backup 設定は、Firehose ストリームのソースが Direct PUT か Kinesis Data Streams である場合のみサポートされます。
> - ゼロバッファリング機能はアプリケーションの宛先のためにのみ使用でき、Amazon S3 バックアップの宛先のためには使用できません。

次のいずれかを選択した場合は、Firehose ストリームの S3 バックアップ設定を指定できます。

- Amazon S3 を Firehose ストリームの送信先として設定し、データレコードを変換する AWS Lambda 関数を指定するか、Firehose ストリームのデータレコード形式を変換することを選択した場合。
- Amazon Redshift を Firehose ストリームの送信先として設定し、データレコードを変換する AWS Lambda 関数を指定することを選択した場合。
- 次のいずれかのサービスを Firehose ストリームの宛先として設定した場合 – Amazon OpenSearch Service、Datadog, Dynatrace、HTTP エンドポイント、LogicMonitor、MongoDB Cloud、New Relic、Splunk、Sumo Logic、Snowflake、Apache Iceberg テーブル。

Firehose ストリームのバックアップ設定は次のとおりです。

- Amazon S3 でのソースレコードバックアップ- S3 または Amazon Redshift が選択した送信先である場合、この設定は、ソースデータのバックアップを有効にするか無効のままにするかを示します。選択した送信先としてサポートされている他のサービス (S3 または Amazon Redshift 以外) が設定されている場合、この設定は、すべてのソースデータまたは失敗したデータのみをバックアップするかどうかを示します。
- S3 バックアップバケット - これは、Amazon Data Firehose がデータをバックアップする S3 バケットです。
- S3 バックアップバケットプレフィックス- これは、Amazon Data Firehose がデータをバックアップするプレフィックスです。

- S3 バックアップバケットエラー出力プレフィックス - すべての失敗したデータは、この S3 バケットエラー出力プレフィックスにバックアップされます。
- バッファリングのヒント、圧縮、およびバックアップの暗号化 - Amazon Data Firehose は、選択した宛先に配信しようとするすべてのデータまたは失敗したデータのみをバックアップするために Amazon S3 を使用します。Amazon Data Firehose は、Amazon S3 に配信 (バックアップ) する前に着信データをバッファリングします。バッファサイズ (1 ~ 128 MB) およびバッファ間隔 (60 ~ 900 秒) を選択できます。最初に満たされた条件は、Amazon S3 へのデータ配信をトリガーします。データ変換を有効にした場合は、変換されたデータが Amazon Data Firehose によって受信されてから Amazon S3 に配信されるまでの時間、バッファリング間隔が適用されます。宛先へのデータ配信が Firehose ストリームへのデータ書き込みよりも遅れると、Amazon Data Firehose はバッファリングサイズを動的に引き上げて遅れを取り戻します。このアクションにより、すべてのデータが送信先に確実に配信されます。
- S3 圧縮 - GZIP、Snappy、Zip、Hadoop 互換の Snappy データ圧縮、またはデータ圧縮なしを選択します。Snappy、Zip、および Hadoop 互換の Snappy 圧縮は、Amazon Redshift を宛先とする Firehose ストリームでは使用できません。
- S3 のファイル拡張子の形式 (オプション) – Amazon S3 宛先バケットに配信されるオブジェクトのファイル拡張子の形式を指定します。この機能を有効にすると、指定されたファイル拡張子は、データ形式変換または .parquet や .gz などの S3 圧縮機能によって付加されたデフォルトのファイル拡張子を上書きします。この機能をデータ形式変換または S3 圧縮で使用する際には、適切なファイル拡張子を設定しているかどうかを確認します。ファイル拡張子はピリオド (.) で始まらなければならず、次の文字を含めることができます: 0-9a-z!-_.*‘()。ファイル拡張子は最大 128 文字です。
- Firehose は、Amazon S3 で配信されたデータを暗号化するための AWS Key Management Service (SSE-KMS) による Amazon S3 サーバー側の暗号化をサポートしています。送信先 S3 バケットで指定されたデフォルトの暗号化タイプを使用するか、所有するキーのリストから AWS KMS キーで暗号化するかを選択できます。 AWS KMS キーでデータを暗号化する場合は、デフォルトの AWS マネージドキー (aws/s3) またはカスタマーマネージドキーを使用できます。詳細については、AWS 「KMS マネージドキーによるサーバー側の暗号化を使用したデータの保護 (SSE-KMS)」を参照してください。

## バッファリングのヒントを設定する

Amazon Data Firehose は、着信ストリーミングデータを、特定のサイズ (バッファリングサイズ)、および特定の期間 (バッファリング間隔) にわたって、メモリにバッファリングしてから、指定された宛先に配信します。バッファリングのヒントは、最適にサイズ設定されたファイルを Amazon S3

に配信し、データ処理アプリケーションのパフォーマンスを改善する場合や、Firehose 配信レートを宛先の速度に合わせて調整する場合に使用します。

新しい Firehose ストリームの作成時にバッファリングサイズとバッファリング間隔を設定したり、既存の Firehose ストリームのバッファリングサイズとバッファリング間隔を更新したりできます。バッファリングサイズは MB 単位で、バッファリング間隔は秒単位です。ただし、一方のパラメータに値を指定する場合は、他方にも値を指定する必要があります。最初に満たされたバッファリング条件によって、Firehose によるデータの配信がトリガーされます。バッファリングの値を設定しない場合、デフォルトの値が使用されます。

Firehose バッファリングヒントは AWS マネジメントコンソール、 AWS Command Line Interface、、または AWS SDKs を使用して設定できます。既存のストリームでは、コンソールの [編集] オプションまたは UpdateDestination API を使用して、ユースケースに適した値でバッファリングのヒントを再設定できます。新しいストリームでは、コンソールまたは CreateDeliveryStream API を使用して、新しいストリーム作成の一部としてバッファリングのヒントを設定できます。バッファリングサイズを調整するには、CreateDeliveryStream または UpdateDestination API の宛先固有の `DestinationConfiguration` パラメータで `SizeInMBs` と `IntervalInSeconds` を設定します。

> **Note**
>
> - バッファリングのヒントはシャードレベルまたはパーティションレベルで適用され、動的パーティショニングバッファリングのヒントはストリームレベルまたはトピックレベルで適用されます。
> - リアルタイムユースケースのより低いレイテンシーに対応するために、ゼロバッファリング間隔ヒントを使用できます。バッファリング間隔を 0 秒として設定すると、Firehose はデータをバッファリングせず、数秒以内にデータを配信します。バッファリングのヒントをより小さい値に変更する前に、宛先用の、Firehose の推奨バッファリングのヒントについてベンダーに確認してください。
> - ゼロバッファリング機能はアプリケーションの宛先のためにのみ使用でき、Amazon S3 バックアップの宛先のためには使用できません。
> - ゼロバッファリング機能は、動的パーティショニングでは使用できません。
> - より低いレイテンシーを提供するために 60 秒未満のバッファリング時間間隔を設定すると、Firehose は S3 の宛先のためにマルチパートアップロードを使用します。S3 の宛先のマルチパートアップロードにより、60 秒未満のバッファリング時間間隔を選択すると、S3 PUT API のコストが若干増加します。

宛先固有のバッファリングのヒントの範囲とデフォルトの値については、次の表を参照してください:

| 目的地 | バッファリングサイズ (MB) (括弧内の値はデフォルト) | バッファリング間隔 (秒) (括弧内の値はデフォルト) |
|---|---|---|
| Amazon S3 | 1-128 (5) | 0-900 (300) |
| Apache Iceberg テーブル | 1-128 (5) | 0-900 (300) |
| Amazon Redshift | 1-128 (5) | 0-900 (300) |
| OpenSearch Serverless | 1-100 (5) | 0-900 (300) |
| OpenSearch | 1-100 (5) | 0-900 (300) |
| Splunk | 1-5 (5) | 0-60 (60) |
| Datadog | 1-4 (4) | 0-900 (60) |
| Coralogix | 1-64 (6) | 0-900 (60) |
| Dynatrace | 1-64 (5) | 0-900 (60) |
| Elastic | 1 | 0-900 (60) |
| Honeycomb | 1-64 (15) | 0-900 (60) |
| HTTP エンドポイント | 1-64 (5) | 0-900 (60) |
| LogicMonitor | 1-64 (5) | 0-900 (60) |
| Logzio | 1-64 (5) | 0-900 (60) |
| mongoDB | 1-16 (5) | 0-900 (60) |
| newRelic | 1-64 (5) | 0-900 (60) |

| 目的地 | バッファリングサイズ (MB) (括弧内の値はデフォルト) | バッファリング間隔 (秒) (括弧内の値はデフォルト) |
|---|---|---|
| sumoLogic | 1-64 (1) | 0-900 (60) |
| Splunk Observability Cloud | 1-64 (1) | 0-900 (60) |
| Snowflake | 1 - 128 (1) | 0 - 900 (0) |

## 詳細設定の設定

次のセクションには、Firehose ストリームの高度な設定に関する詳細が含まれています。

- サーバー側の暗号化 - Amazon Data Firehose はAmazon S3 で配信されたデータを暗号化するための AWS Key Management Service (AWS KMS) による Amazon S3 サーバー側の暗号化をサポートしています。詳細については、AWS 「KMS マネージドキーによるサーバー側の暗号化を使用したデータの保護 (SSE-KMS)」を参照してください。
- エラーログ記録 - Amazon Data Firehose は、処理と配信に関連するエラーを記録します。さらに、データ変換が有効になっている場合は、Lambda 呼び出しをログに記録したりデータ配信エラーを CloudWatch Logs に送信したりできます。詳細については、「CloudWatch Logs を使用して Amazon Data Firehose をモニタリングする」を参照してください。

  > **Important**
  >
  > オプションとして、Firehose ストリームの作成時に Amazon Data Firehose エラーログ記録を有効にすることが強く推奨されています。これにより、レコード処理や配信に失敗した場合でもそのエラーの詳細にアクセスすることができます。

- 許可 - Amazon Data Firehose は、Firehose ストリームに必要なすべての許可のために IAM ロールを使用します。必要な許可が自動的に割り当てられる新しいロールを作成するか、Amazon Data Firehose 用に作成された既存のロールを選択できます。ロールは、S3 バケット、 AWS KMS キー (データ暗号化が有効になっている場合 ) 、Lambda 関数 (データ変換が有効になっている場合) など、さまざまな サービスへのアクセス権を Firehose に付与するために使用されます。コンソールはプレースホルダーを含むロールを作成する可能性があります。詳細については、「IAM とは何ですか?」を参照してください。

> **Note**
>
> IAM ロール (プレースホルダーを含む) は、Firehose ストリームの作成時に選択した設定に基づいて作成されます。Firehose ストリームのソースまたは宛先に変更を加える場合は、IAM ロールを手動で更新する必要があります。

- タグ - タグを追加して、 AWS リソースの整理、コストの追跡、アクセスの制御を行うことができます。

  `CreateDeliveryStream` アクションでタグを指定すると、Amazon Data Firehose は `firehose:TagDeliveryStream` アクションに対して追加の認可を実行し、タグを作成するための許可がユーザーに付与されているかどうかを検証します。この許可を指定しない場合、IAM リソースのタグを使用して新しい Firehose ストリームを作成するリクエストは、次のような `AccessDeniedException` で失敗します。

  ```
  AccessDeniedException
  User: arn:aws:sts::x:assumed-role/x/x is not authorized to perform:
   firehose:TagDeliveryStream on resource: arn:aws:firehose:us-east-1:x:deliverystream/
  x with an explicit deny in an identity-based policy.
  ```

  次の例は、ユーザーが Firehose ストリームを作成してタグを適用できるようにするポリシーを示しています。

バックアップと詳細設定を選択したら、選択内容を確認し、[Firehose ストリームを作成] を選択します。

新しい Firehose ストリームは、使用可能になる前にしばらく [作成中] 状態になります。Firehose ストリームが [アクティブ] 状態になったら、プロデューサーからそこにデータの送信を開始することができます。

# サンプルデータを使用した Firehose ストリームのテスト

を使用して AWS マネジメントコンソール 、シミュレートされた株式ティッカーデータを取り込むことができます。コンソールによってブラウザでスクリプトが実行されて、サンプルレコードが Firehose ストリームに挿入されます。これにより、独自のテストデータを生成することなく、Firehose ストリームの設定をテストできます。

以下に示しているのは、シミュレートされたデータの例です。

```
{"TICKER_SYMBOL":"QXZ","SECTOR":"HEALTHCARE","CHANGE":-0.05,"PRICE":84.51}
```

Firehose ストリームからデータが送信されるときは、標準の Amazon Data Firehose 料金が適用されますが、データが生成されるときは、料金は発生しません。これらの料金の発生を停止するために、いつでもコンソールからサンプルストリームを停止できます。

## 前提条件

開始する前に、Firehose ストリームを作成します。詳細については、「チュートリアル: コンソールから Firehose ストリームを作成する」を参照してください。

## Amazon S3 を使用してテストする

次の手順で、宛先として Amazon Simple Storage Service (Amazon S3) を使用して Firehose ストリームをテストします。

Amazon S3 を使用して Firehose ストリームをテストするには

1. Firehose コンソール (https://console.aws.amazon.com/firehose/) を開きます。
2. アクティブな Firehose ストリームを選択します。データの送信を開始する前に、Firehose ストリームのステータスが [アクティブ] になっている必要があります。
3. [Test with demo data] で、[Start sending demo data] を選択して、サンプル株価データを生成します。
4. 画面の指示に従って、S3 バケットにデータが配信されていることを確認します。バケットのバッファ設定に基づいて、新しいオブジェクトがバケットに表示されるまでに数分かかる場合があります。
5. テストが完了したら、[Stop sending demo data] を選択して、利用料金の発生を停止します。

# Amazon Redshift を使用してテストする

次の手順で、宛先として Amazon Redshift を使用して Firehose ストリームをテストします。

Amazon Redshift を使用して Firehose ストリームをテストするには

1. Firehose ストリームでは、Amazon Redshift クラスターにテーブルがあることが必要です。SQL インターフェイスを使用して Amazon Redshift に接続し、以下のステートメントを実行して、サンプルデータを受け入れるテーブルを作成します。

```
create table firehose_test_table
(
 TICKER_SYMBOL varchar(4),
 SECTOR varchar(16),
 CHANGE float,
 PRICE float
);
```

2. Firehose コンソール (https://console.aws.amazon.com/firehose/) を開きます。
3. アクティブな Firehose ストリームを選択します。データの送信を開始する前に、Firehose ストリームのステータスが [アクティブ] になっている必要があります。
4. 新しく作成した `firehose_test_table` テーブルを参照するように、Firehose ストリームの宛先の詳細を編集します。
5. [Test with demo data] で、[Start sending demo data] を選択して、サンプル株価データを生成します。
6. 画面の指示に従って、テーブルにデータが配信されていることを確認します。バッファ設定に基づいて、新しい行がテーブルに表示されるまでに数分かかる場合があります。
7. テストが完了したら、[Stop sending demo data] を選択して、利用料金の発生を停止します。
8. 別のテーブルを参照するように、Firehose ストリームの宛先の詳細を編集します。
9. (オプション) `firehose_test_table` テーブルを削除します。

# OpenSearch Service を使用してテストする

次の手順で、宛先として Amazon OpenSearch Service を使用して Firehose ストリームをテストします。

OpenSearch Service を使用して Firehose ストリームをテストするには

1. Firehose コンソール (https://console.aws.amazon.com/firehose/) を開きます。
2. アクティブな Firehose ストリームを選択します。データの送信を開始する前に、Firehose ストリームのステータスが [アクティブ] になっている必要があります。
3. [Test with demo data] で、[Start sending demo data] を選択して、サンプル株価データを生成します。
4. 画面の指示に従って、OpenSearch Service ドメインにデータが配信されていることを確認します。詳細については、「Amazon OpenSearch Service デベロッパーガイド」の「Searching Documents in an OpenSearch Service Domain」を参照してください。
5. テストが完了したら、[Stop sending demo data] を選択して、利用料金の発生を停止します。

## Splunk を使用してテストする

次の手順で、宛先として Splunk を使用して Firehose ストリームをテストします。

Splunk を使用して Firehose ストリームをテストするには

1. Firehose コンソール (https://console.aws.amazon.com/firehose/) を開きます。
2. アクティブな Firehose ストリームを選択します。データの送信を開始する前に、Firehose ストリームのステータスが [アクティブ] になっている必要があります。
3. [Test with demo data] で、[Start sending demo data] を選択して、サンプル株価データを生成します。
4. Splunk インデックスにデータが配信されるかどうかを確認します。Splunk の検索用語の例は `sourcetype="aws:firehose:json"` および `index="name-of-your-splunk-index"` です。イベントを検索する方法の詳細については、Splunk のドキュメントの Search Manual を参照してください。

   Splunk インデックスでテストデータが表示されない場合は、Amazon S3 バケットで失敗したイベントを確認します。また、「データが Splunk に配信されない」を参照してください。
5. テストが完了したら、[Stop sending demo data] を選択して、利用料金の発生を停止します。

## Apache Iceberg テーブルを使用してテストする

次の手順で、Apache Iceberg テーブルを宛先として Firehose ストリームをテストします。

Apache Iceberg テーブルを使用して Firehose ストリームをテストするには

1. Firehose コンソール (https://console.aws.amazon.com/firehose/) を開きます。
2. アクティブな Firehose ストリームを選択します。データの送信を開始する前に、Firehose ストリームのステータスが [アクティブ] になっている必要があります。
3. [Test with demo data] で、[Start sending demo data] を選択して、サンプル株価データを生成します。
4. 画面の指示に従って、データが Apache Iceberg テーブルに配信されていることを検証します。バッファリング設定に基づいて、新しいオブジェクトがバケットに表示されるまでに数分かかる場合があります。
5. Apache Iceberg テーブルでテストデータが表示されない場合は、Amazon S3 バケットで失敗したイベントを確認します。
6. テストが完了したら、[Stop sending demo data] を選択して、利用料金の発生を停止します。

# Firehose ストリームにデータを送信する

このセクションでは、さまざまなデータソースを使用して Firehose ストリームにデータを送信する方法について説明します。Amazon Data Firehose を初めて使用する場合、時間を取って「Amazon Data Firehose とは何ですか?」で説明している概念と用語を理解してください。

> **Note**
>
> 一部の AWS サービスは、同じリージョンにある Firehose ストリームにのみメッセージとイベントを送信できます。Amazon CloudWatch Logs、CloudWatch Events、または のターゲットを設定するときに Firehose ストリームがオプションとして表示されない場合は AWS IoT、Firehose ストリームが他の サービスと同じリージョンにあることを確認します。各リージョンのサービスエンドポイントの詳細については、「Amazon Data Firehose endpoints」を参照してください。

次のデータソースから Firehose ストリームにデータを送信できます。

トピック

- データを送信するように Kinesis エージェントを設定する
- AWS SDK でデータを送信する
- CloudWatch Logs を Firehose に送信する
- CloudWatch Events を Firehose に送信する
- Firehose にデータを送信する AWS IoT ように を設定する

## データを送信するように Kinesis エージェントを設定する

Amazon Kinesis エージェントは、Firehose にデータを収集および送信する方法を示すリファレンス実装として機能する、スタンドアロン Java ソフトウェアアプリケーションです。このエージェントは一連のファイルを継続的に監視し、新しいデータを Firehose ストリームに送信します。エージェントは、ファイルのローテーション、チェックポイント、および障害時の再試行を処理する方法を示します。信頼性が高く、適時かつシンプルな態様でデータを配信する方法を示します。また、ストリーミングプロセスをより適切にモニタリングおよびトラブルシューティングするために、CloudWatch メトリクスを発行する方法も示します。詳細については、awslabs/amazon-kinesis-agent を参照してください。

デフォルトでは、レコードは改行文字 ('\n') に基づいて各ファイルから解析されます。しかし、複数行レコードを解析するよう、エージェントを設定することもできます (エージェント構成設定を指定するを参照)。

このエージェントは、ウェブサーバー、ログサーバーおよびデータベースサーバーなど、Linux ベースのサーバー環境にインストールできます。エージェントをインストールした後で、モニタリングするファイルとデータ用の Firehose ストリームを指定して設定します。エージェントが設定されると、ファイルから永続的にデータを収集して、Firehose ストリームに安全にデータを送信します。

## 前提条件

Kinesis Agent の使用を開始する前に、次の前提条件を満たしていることを確認してください。

- オペレーティングシステムは Amazon Linux、または Red Hat Enterprise Linux バージョン 7 以降でなければなりません。
- エージェントバージョン 2.0.0 以降は JRE バージョン 1.8 以降を使用して実行されます。エージェントバージョン 1.1.x は JRE 1.7 以降を使用して実行されます。
- Amazon EC2 を使用してエージェントを実行している場合は、EC2 インスタンスを起動します。
- 指定する IAM ロールまたは AWS 認証情報には、エージェントが Firehose ストリームにデータを送信するために Amazon Data Firehose PutRecordBatch オペレーションを実行するアクセス許可が必要です。エージェントの CloudWatch モニタリングを有効にしている場合は、CloudWatch PutMetricData オペレーションを実行する権限も必要になります。詳細については、「Amazon Data Firehose によるアクセスの制御」、「Kinesis エージェントの状態をモニタリングする」、および「CloudWatch Amazon CloudWatch に対する認証とアクセスコントロール」を参照してください。

## AWS 認証情報の管理

次のいずれかの方法を使用して AWS 認証情報を管理します。

- カスタム認証情報プロバイダーを作成します。詳細については、「the section called “カスタム認証プロバイダーを作成する”」を参照してください。
- EC2 インスタンスを起動する際に IAM ロールを指定します。
- エージェントを設定するときに AWS 認証情報を指定します ( awsSecretAccessKeyの設定テーブルの awsAccessKeyIdおよび のエントリを参照the section called “エージェント構成設定を指定する”)。

- を編集/etc/sysconfig/aws-kinesis-agentして、 AWS リージョンと AWS アクセスキーを指定します。
- EC2 インスタンスが別の AWS アカウントにある場合は、IAM ロールを作成して Amazon Data Firehose サービスへのアクセスを提供します。エージェントを設定するときに、そのロールを指定します (assumeRoleARN と assumeRoleExternalId を参照してください)。前の方法のいずれかを使用して、このロールを引き受けるアクセス許可を持つ他のアカウントのユーザーの AWS 認証情報を指定します。

## カスタム認証プロバイダーを作成する

カスタム認証情報プロバイダーを作成し、そのクラス名と jar パスを Kinesis エージェントに渡すことができます。これらを渡すための構成設定として userDefinedCredentialsProvider.classname と userDefinedCredentialsProvider.location を使用します。これら 2 つの構成設定の説明については、「the section called “エージェント構成設定を指定する”」を参照してください。

カスタム認証情報プロバイダーを作成するには、次の例に示すように、AWS CredentialsProvider インターフェイスを実装するクラスを定義します。

```
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;

public class YourClassName implements AWSCredentialsProvider {
    public YourClassName() {
    }

    public AWSCredentials getCredentials() {
        return new BasicAWSCredentials("key1", "key2");
    }

    public void refresh() {
    }
}
```

このクラスは、引数を取らないコンストラクターを必要とします。

AWS は更新メソッドを定期的に呼び出して、更新された認証情報を取得します。認証情報プロバイダーからその存続期間中に常に異なる認証情報を提供する場合は、このメソッドに認証情報を更新

するためのコードを含めます。または、静的な (変わらない) 認証情報を提供する認証情報プロバイダーを必要とする場合は、このメソッドを空のままにすることもできます。

## エージェントをダウンロードおよびインストールする

最初に、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「Linux インスタンスへの接続」を参照してください。接続できない場合は、「Amazon EC2 ユーザーガイド」の「インスタンスへの接続に関するトラブルシューティング」を参照してください。

次に、次のいずれかの方法を使用して、エージェントをインストールします。

- Amazon Linux リポジトリからエージェントをセットアップするには

  このメソッドは Amazon Linux インスタンスでのみ機能します。以下のコマンドを使用します。

  ```
  sudo yum install -y aws-kinesis-agent
  ```

  エージェント v 2.0.0 以降が、オペレーティングシステム Amazon Linux 2 (AL2) のコンピュータにインストールされます。このエージェントバージョンでは、Java 1.8 以降が必要です。必要な Java バージョンがまだ存在しない場合は、エージェントのインストールプロセスによってインストールされます。Amazon Linux 2 の詳細については、「https://aws.amazon.com/amazon-linux-2/」を参照してください。
- Amazon S3 リポジトリからエージェントをセットアップするには

  このメソッドは、Red Hat Enterprise Linux と Amazon Linux 2 インスタンスでも機能します。これは、パブリックに利用可能なリポジトリからエージェントをインストールするからです。次のコマンドを使用して、エージェントバージョン 2.x.x の最新バージョンをダウンロードしてインストールします。

  ```
  sudo yum install -y https://s3.amazonaws.com/streaming-data-agent/aws-kinesis-agent-latest.amzn2.noarch.rpm
  ```

  特定バージョンのエージェントをインストールするには、そのバージョン番号をコマンドで指定します。たとえば、次のコマンドはエージェント v 2.0.1 をインストールします。

```
sudo yum install -y https://streaming-data-agent.s3.amazonaws.com/aws-kinesis-
agent-2.0.1-1.amzn1.noarch.rpm
```

Java 1.7 を持っていて、それをアップグレードしたくない場合は、Java 1.7 と互換性があるエージェントバージョン 1.x.x をダウンロードできます。たとえば、エージェント v1.1.6 をダウンロードするには、次のコマンドを使用します。

```
sudo yum install -y https://s3.amazonaws.com/streaming-data-agent/aws-kinesis-
agent-1.1.6-1.amzn1.noarch.rpm
```

次のコマンドで最新のエージェントをダウンロードできます。

```
sudo yum install -y https://s3.amazonaws.com/streaming-data-agent/aws-kinesis-agent-
latest.amzn2.noarch.rpm
```

- GitHub リポジトリからエージェントをセットアップするには

  1. まず、エージェントのバージョンに応じて、必要な Java バージョンがインストールされていることを確認します。
  2. 「awslabs/amazon-kinesis-agent」GitHub リポジトリからエージェントをダウンロードしてください。
  3. ダウンロードしたディレクトリまで移動し、次のコマンドを実行してエージェントをインストールします。

     ```
     sudo ./setup --install
     ```

- Docker コンテナにエージェントをセットアップするには

  Kinesis Agent は、amazonlinux コンテナベースを使ってコンテナで実行することもできます。次の Docker ファイルを使用し、`docker build` を実行します。

  ```
  FROM amazonlinux
  ```

```
RUN yum install -y aws-kinesis-agent which findutils
COPY agent.json /etc/aws-kinesis/agent.json

CMD ["start-aws-kinesis-agent"]
```

## エージェントを設定および開始する

エージェントを設定して開始するには

1. (デフォルトのファイルアクセス許可を使用している場合、スーパーユーザーとして) 設定ファイル (/etc/aws-kinesis/agent.json) を開き、編集します。

   この設定ファイルで、エージェントがデータを集めるファイル ("filePattern") とエージェントがデータを送信する Firehose ストリーム ("deliveryStream") を指定します。ファイル名はパターンで、エージェントはファイルローテーションを確認します。1 秒あたりに一度だけ、ファイルを交替または新しいファイルを作成できます。エージェントはファイル作成タイムスタンプを使用して、追跡し、Firehose ストリームに送信するファイルを判断します。新しいファイルを作成したり、1 秒に 1 回を超える頻度でファイルをローテーションしたりしても、エージェント間で適切に区別することはできません。

   ```
   {
      "flows": [
           {
               "filePattern": "/tmp/app.log*",
               "deliveryStream": "yourdeliverystream"
           }
      ]
   }
   ```

   デフォルトの AWS リージョンは ですus-east-1。別のリージョンを使用する場合は、設定ファイルに firehose.endpoint 設定を追加してリージョンのエンドポイントを指定します。詳細については、「エージェント構成設定を指定する」を参照してください。

2. エージェントを手動で開始する:

   ```
   sudo service aws-kinesis-agent start
   ```

3. (オプション) システムスタートアップ時にエージェントを開始するように設定します。

```
sudo chkconfig aws-kinesis-agent on
```

エージェントは、システムのサービスとしてバックグラウンドで実行されます。継続的に指定ファイルをモニタリングし、指定された Firehose ストリームにデータを送信します。エージェント活動は、/var/log/aws-kinesis-agent/aws-kinesis-agent.log に記録されます。

## エージェント構成設定を指定する

エージェントは、2 つの必須設定、filePattern と deliveryStream、さらに追加機能としてオプションの設定をサポートしています。必須およびオプションの設定を /etc/aws-kinesis/agent.json で指定できます。

設定ファイルを変更した場合は、次のコマンドを使用してエージェントを停止および起動する必要があります。

```
sudo service aws-kinesis-agent stop
sudo service aws-kinesis-agent start
```

または、次のコマンドを使用できます。

```
sudo service aws-kinesis-agent restart
```

全般設定は次のとおりです。

| 構成設定 | 説明 |
| --- | --- |
| assumeRoleARN | ユーザーが引き受けるロールの Amazon リソースネーム (ARN)。詳細については、IAM ユーザーガイドの「IAM ロールを使用した AWS アカウント間のアクセスの委任」を参照してください。 |
| assumeRoleExternalId | ロールを引き受けることができるユーザーを決定するオプションの ID。詳細については、IAM ユーザーガイドの外部 ID の使用方法を参照してください。 |
| awsAccessKeyId | AWS デフォルトの認証情報を上書きする アクセスキー ID。この設定は、他のすべての認証情報プロバイダーに優先されます。 |

| 構成設定 | 説明 |
| --- | --- |
| awsSecretAccessKey | AWS デフォルトの認証情報を上書きするシークレットキー。この設定は、他のすべての認証情報プロバイダーに優先されます。 |
| cloudwatch.emitMetrics | エージェントがメトリクスを CloudWatch に出力できるようにします (true に設定された場合)。<br><br>デフォルト: true |
| cloudwatch.endpoint | CloudWatch のリージョンのエンドポイントです。<br><br>デフォルト: monitoring.us-east-1.amazonaws.com |
| firehose.endpoint | Amazon Data Firehose のリージョンのエンドポイントです。<br><br>デフォルト: firehose.us-east-1.amazonaws.com |
| sts.endpoint | AWS セキュリティトークンサービスのリージョンエンドポイント。<br><br>デフォルト: https://sts.amazonaws.com |
| userDefinedCredentialsProvider.classname | カスタム認証情報プロバイダーを定義する場合、この設定を使用してその完全修飾クラス名を指定します。クラス名の末尾に .class を含めないでください。 |
| userDefinedCredentialsProvider.location | カスタム認証情報プロバイダーを定義する場合、この設定を使用して、カスタム認証情報プロバイダーが含まれている jar の絶対パスを指定します。エージェントは、この jar ファイルを /usr/share/aws-kinesis-agent/lib/ でも検索します。 |

フロー設定は次のとおりです。

| 構成設定 | 説明 |
| --- | --- |
| aggregatedRecordSizeBytes | 1 回のオペレーションでエージェントがレコードを集約し、Firehose ストリームに配置するには、この設定を指定します。エージェントが |

| 構成設定 | 説明 |
| --- | --- |
| | Firehose ストリームに配置する前の集約レコードのサイズに設定します。<br><br>デフォルト: 0 (集約なし) |
| dataProcessingOptions | Firehose ストリームに送信される前に解析された各レコードに適用されるの処理オプションの一覧。処理オプションは指定した順序で実行されます。詳細については、「エージェントを使用してデータを事前処理する」を参照してください。 |
| deliveryStream | [必須] Firehose ストリームの名前。 |
| filePattern | [必須] エージェントによってモニタリングされる必要があるファイルの glob このパターンに一致するすべてのファイルは、エージェントによって自動的に選択され、モニタリングされます。このパターンに一致するすべてのファイルで、読み取りアクセス許可を aws-kinesis-agent-user に付与します。ファイルを含むディレクトリで、読み取りアクセス許可と実行アクセス許可を aws-kinesis-agent-user に付与します。<br><br>**Important**<br>エージェントは、このパターンに一致するファイルをすべて取得します。エージェントが意図しないレコードを取得しないように、このパターンは慎重に選択してください。 |
| initialPosition | ファイルの解析が開始される最初の位置。有効な値は、START_OF_FILE および END_OF_FILE です。<br><br>デフォルト: END_OF_FILE |
| maxBufferAgeMillis | エージェントが Firehose ストリームに送信する前にデータをバッファリングする最大時間 (ミリ秒)。<br><br>値の範囲: 1,000 ~ 900,000 (1 秒 ~ 15 分)<br><br>デフォルト: 60,000 (1 分) |

| 構成設定 | 説明 |
| --- | --- |
| maxBuffer SizeBytes | エージェントが Firehose ストリームに送信する前にデータをバッファリングする最大サイズ (バイト)。<br><br>値の範囲: 1 ~ 4,194,304 (4 MB)<br><br>デフォルト: 4,194,304 (4 MB) |
| maxBuffer SizeRecords | エージェントが Firehose ストリームに送信する前にデータをバッファリングするレコードの最大数。<br><br>値の範囲: 1 ~ 500<br><br>デフォルト: 500 |
| minTimeBe tweenFile PollsMillis | エージェントが新しいデータのモニタリング対象ファイルをポーリングし、解析する時間間隔 (ミリ秒単位)。<br><br>値の範囲: 1 以上<br><br>デフォルト: 100 |
| multiLine StartPattern | レコードの開始を識別するパターン。レコードは、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です。デフォルトでは、ログファイルのそれぞれの改行は 1 つのレコードとして解析されます。 |
| skipHeaderLines | モニタリング対象ファイルの始めにエージェントが解析をスキップするの行数。<br><br>値の範囲: 0 以上<br><br>デフォルト: 0 (ゼロ) |
| truncated RecordTer minator | レコードのサイズが Amazon Data Firehose レコードの許容サイズを超えたときに解析されたレコードを切り捨てるために、エージェントが使用する文字列。(1,000 KB)<br><br>デフォルト: '\n' (改行) |

## 複数のファイルディレクトリとストリームを設定する

複数のフロー設定を指定することによって、エージェントが複数のファイルディレクトリを監視し、複数のストリームにデータを送信するように設定できます。次の設定例では、エージェントは 2 つのファイルディレクトリをモニタリングし、それぞれ Kinesis データストリームおよび Firehose ストリームにデータを送信します。Kinesis Data Streams と Amazon Data Firehose に異なるエンドポイントを指定できるため、データストリームと Firehose ストリームが同じリージョンに存在する必要はありません。

```
{
    "cloudwatch.emitMetrics": true,
    "kinesis.endpoint": "https://your/kinesis/endpoint",
    "firehose.endpoint": "https://your/firehose/endpoint",
    "flows": [
        {
            "filePattern": "/tmp/app1.log*",
            "kinesisStream": "yourkinesisstream"
        },
        {
            "filePattern": "/tmp/app2.log*",
            "deliveryStream": "yourfirehosedeliverystream"
        }
    ]
}
```

Amazon Kinesis Data Streams でのエージェントの使用の詳細については、「Kinesis エージェントを使用した Amazon Kinesis Data Streams への書き込み」を参照してください。

## エージェントを使用してデータを事前処理する

エージェントは Firehose ストリームにレコードを送信する前に、モニタリング対象ファイルから解析したレコードを事前処理できます。ファイルフローに `dataProcessingOptions` 設定を追加することで、この機能を有効にできます。処理オプションを 1 つ以上追加することができます。また、指定の順序で実行されます。

エージェントは、次の処理オプションに対応しています。エージェントはオープンソースであるため、処理オプションを開発および拡張できます。Kinesis エージェントからエージェントをダウンロードできます。

処理オプション

SINGLELINE

改行文字、先頭のスペース、末尾のスペースを削除することで、複数行レコードを単一行レコードに変換します。

```
{
    "optionName": "SINGLELINE"
}
```

CSVTOJSON

区切り形式から JSON 形式にレコードを変換します。

```
{
    "optionName": "CSVTOJSON",
    "customFieldNames": [ "field1", "field2", ... ],
    "delimiter": "yourdelimiter"
}
```

customFieldNames

[必須] 各 JSON キー値のペアでキーとして使用されるフィールド名。たとえば、["f1", "f2"] を指定した場合は、レコード「v1、v2」は {"f1":"v1","f2":"v2"} に変換されます。

delimiter

レコードで区切り記号として使用する文字列。デフォルトはカンマ (,) です。

LOGTOJSON

ログ形式から JSON 形式にレコードを変換します。サポートされているログ形式は、Apache Common Log、Apache Combined Log、Apache Error Log、および RFC3164 Syslog です。

```
{
    "optionName": "LOGTOJSON",
    "logFormat": "logformat",
    "matchPattern": "yourregexpattern",
    "customFieldNames": [ "field1", "field2", … ]
}
```

`logFormat`

[必須] ログエントリ形式。以下の値を指定できます。

- `COMMONAPACHELOG` — Apache Common Log 形式。各ログエントリは、デフォルトで次のパターン`%{host} %{ident} %{authuser} [%{datetime}] \"%{request}\" %{response} %{bytes}`になります。
- `COMBINEDAPACHELOG` — Apache Combined Log 形式。各ログエントリは、デフォルトで次のパターン`%{host} %{ident} %{authuser} [%{datetime}] \"%{request}\" %{response} %{bytes} %{referrer} %{agent}`になります。
- `APACHEERRORLOG` — Apache Error Log 形式。各ログエントリは、デフォルトで次のパターン`[%{timestamp}] [%{module}:%{severity}] [pid %{processid}:tid %{threadid}] [client: %{client}] %{message}`になります。
- `SYSLOG` — FC3164 Syslog 形式。各ログエントリは、デフォルトで次のパターン`%{timestamp} %{hostname} %{program}[%{processid}]: %{message}`になります。

`matchPattern`

指定されたログ形式のデフォルトパターンを上書きします。カスタム形式を使用する場合は、この設定を使用してログエンティティから値を抽出します。`matchPattern` を指定する場合は、`customFieldNames` も指定する必要があります。

`customFieldNames`

JSON キー値のペアでキーとして使用されるカスタムフィールド名。`matchPattern` から抽出した値のフィールド名を定義するために、または事前定義されたログ形式のデフォルトのフィールド名を上書きするために、この設定を使用できます。

Example: LOGTOJSON 設定

JSON形式に変換された Apache Common Log エントリの `LOGTOJSON` 設定の一つの例を次に示します。

```
{
    "optionName": "LOGTOJSON",
    "logFormat": "COMMONAPACHELOG"
}
```

変換前:

```
64.242.88.10 - - [07/Mar/2004:16:10:02 -0800] "GET /mailman/listinfo/hsdivision
 HTTP/1.1" 200 6291
```

変換後:

```
{"host":"64.242.88.10","ident":null,"authuser":null,"datetime":"07/
Mar/2004:16:10:02 -0800","request":"GET /mailman/listinfo/hsdivision
 HTTP/1.1","response":"200","bytes":"6291"}
```

Example: カスタムフィールドがある LOGTOJSON 設定

こちらは LOGTOJSON 設定の別の例です。

```
{
    "optionName": "LOGTOJSON",
    "logFormat": "COMMONAPACHELOG",
    "customFieldNames": ["f1", "f2", "f3", "f4", "f5", "f6", "f7"]
}
```

この設定では、前の例からの同じ Apache Common Log エントリは、次のように JSON 形式に変換されます。

```
{"f1":"64.242.88.10","f2":null,"f3":null,"f4":"07/Mar/2004:16:10:02 -0800","f5":"GET /
mailman/listinfo/hsdivision HTTP/1.1","f6":"200","f7":"6291"}
```

Example: Apache Common Log エントリの変換

次のフロー設定は Apache Common Log エントリを JSON 形式の単一行レコードに変換します。

```
{
    "flows": [
        {
            "filePattern": "/tmp/app.log*",
            "deliveryStream": "my-delivery-stream",
            "dataProcessingOptions": [
                {
                    "optionName": "LOGTOJSON",
                    "logFormat": "COMMONAPACHELOG"
                }
            ]
        }
```

```
    ]
}
```

Example: 複数行レコードの変換

次のフロー設定は、最初の行が[SEQUENCE=で開始している複数行レコードを解析します。各レコードはまず単一行レコードに変換されます。次に、値はタブの区切り記号に基づいたレコードから取得されます。取得された値は指定された customFieldNames 値にマッピングされ、JSON 形式の単一行レコードを形成します。

```
{
    "flows": [
        {
            "filePattern": "/tmp/app.log*",
            "deliveryStream": "my-delivery-stream",
            "multiLineStartPattern": "\\[SEQUENCE=",
            "dataProcessingOptions": [
                {
                    "optionName": "SINGLELINE"
                },
                {
                    "optionName": "CSVTOJSON",
                    "customFieldNames": [ "field1", "field2", "field3" ],
                    "delimiter": "\\t"
                }
            ]
        }
    ]
}
```

Example: 一致パターンで LOGTOJSON 設定

こちらは、最後のフィールド (バイト) が省略された JSON 形式に変換された Apache Common Log エントリの LOGTOJSON 設定の一例です。

```
{
    "optionName": "LOGTOJSON",
    "logFormat": "COMMONAPACHELOG",
    "matchPattern": "^([\\d.]+) (\\S+) (\\S+) \\[([\\w:/]+\\s[+\\-]\\d{4})\\] \"(.
+?)\" (\\d{3})",
    "customFieldNames": ["host", "ident", "authuser", "datetime", "request",
 "response"]
```

```
}
```

変換前:

```
123.45.67.89 - - [27/Oct/2000:09:27:09 -0400] "GET /java/javaResources.html HTTP/1.0"
 200
```

変換後:

```
{"host":"123.45.67.89","ident":null,"authuser":null,"datetime":"27/Oct/2000:09:27:09
 -0400","request":"GET /java/javaResources.html HTTP/1.0","response":"200"}
```

## 一般的なエージェント CLI コマンドを使用する

次の表は、 AWS Kinesis エージェントを操作するための一般的なユースケースと対応するコマンドのセットを示しています。

| ユースケース | コマンド |
| --- | --- |
| システムスタートアップ時にエージェントを自動的に開始する | `sudo chkconfig aws-kinesis-agent on` |
| エージェントのステータスを確認する | `sudo service aws-kinesis-agent status` |
| エージェントを停止する | `sudo service aws-kinesis-agent stop` |
| この場所からエージェントのログファイルを読み取る | `/var/log/aws-kinesis-agent/aws-kinesis-agent.log` |
| エージェントのアンインストール | `sudo yum remove aws-kinesis-agent` |

# Kinesis エージェントから送信する際の問題をトラブルシューティングする

この表は、Amazon Kinesis エージェントを使用する際に直面する一般的な問題のトラブルシューティングに関する情報と解決方法を示しています。

| 問題 | ソリューション |
| --- | --- |
| Kinesis エージェントが Windows で機能しないのはなぜですか? | Windows 用 Kinesis Agent は Linux プラットフォーム用 Kinesis Agent とは異なるソフトウェアです。 |
| Kinesis Agent の速度が低下したり、`RecordSendErrors` が増加したりするのはなぜですか? | 通常、これは Kinesis のスロットリングが原因です。Kinesis Data Streams の `WriteProvisionedThroughputExceeded` メトリクス、または Firehose ストリームの `ThrottledRecords` メトリクスをチェックします。これらのメトリクスが 0 を超えている場合は、ストリームの上限を引き上げる必要があることを意味します。詳細については、「Kinesis Data Stream limits」と「Firehose streams」を参照してください。<br><br>スロットリングが原因ではないことがわかったら、Kinesis Agent が大量の小規模ファイルをテーリングするように設定されているかどうかを確認してください。Kinesis Agent が新しいファイルをテーリングするときには遅延が発生するため、少量の大きなファイルをテーリングするようにします。ログファイルを大きなファイルに統合してみてください。 |
| `java.lang.OutOfMemoryError` 例外を解決するにはどうすればよいですか? | これは、Kinesis Agent に、現在のワークロードを処理するための十分なメモリがない場合に発生します。`/usr/bin/start-aws-kinesis-agent` で `JAVA_START_HEAP` と `JAVA_MAX_HEAP` を増やしてエージェントを再起動してみてください。 |
| `IllegalStateException : connection pool shut down` 例外を解決するにはどうすればよいですか? | Kinesis エージェントに、現在のワークロードを処理するための十分な接続がないためです。`/etc/aws-kinesis/agent.json` の一般的なエージェント設定で `maxConnections` と `maxSendingThreads` を増やしてみてください。これらのフィールドのデフォルト |

| 問題 | ソリューション |
| --- | --- |
| | 値は、使用可能なランタイムプロセッサの 12 倍です。高度なエージェント設定については、「AgentConfiguration.java」を参照してください。 |
| Kinesis Agent に関する別の問題をデバッグする方法を教えてください。 | DEBUG レベルログは `/etc/aws-kinesis/log4j.xml` で有効にできます。 |
| Kinesis Agent はどのように設定するとよいですか? | `maxBufferSizeBytes` の値が小さいほど、Kinesis Agent がデータを送信する頻度が高くなります。そのため、レコードの配信時間が短縮されますが、Kinesis への 1 秒あたりのリクエスト数も増えます。 |
| Kinesis Agent が重複レコードを送信するのはなぜですか? | これはファイルテーリングの設定ミスが原因です。各 `fileFlow's filePattern` がそれぞれ 1 つのファイルのみと一致するようにします。また、使用されている `logrotate` モードが `copytruncate` モードになっている場合にも発生することがあります。重複を避けるため、モードをデフォルトか作成モードに変更してみてください。重複レコードの処理に関する詳細は、「Handling Duplicate Records」を参照してください。 |

# AWS SDK でデータを送信する

Amazon Data Firehose API を使用して Firehose ストリームにデータを送信するには、AWS SDK for Java、.NET、Node.js、Python、または Ruby を使用します。Amazon Data Firehose を初めて使用する場合、時間を取って「Amazon Data Firehose とは何ですか?」で説明している概念と用語を理解してください。詳細については、「Amazon Web Services を使用した開発の開始」を参照してください。

これらのサンプルは、すべての例外を確認しているわけではなく、すべてのセキュリティやパフォーマンスの側面を考慮しているわけでもない点で、本稼働環境に使用できるコードを表すものではありません。

Amazon Data Firehose API には、Firehose ストリームにデータを送信するための 2 つのオペレーション (PutRecord および PutRecordBatch) が用意されています。`PutRecord()` が 1 回の呼び出し

内で 1 つのデータレコードを送信し、PutRecordBatch() は 1 回の呼び出し内で複数のデータレコードを送信できます。

## PutRecord を使用した単一の書き込みオペレーション

データの出力では、Firehose ストリーム名とバイトバッファリング (<=1000 KB) のみが必要です。Amazon Data Firehose バッチはファイルを Amazon S3 にロードする前に、複数のレコードをバッチ処理するため、レコードの区切り文字を追加することをお勧めします。レコードを一度にひとつずつ Firehose ストリームに出力するには、次のコードを使用します。

```
PutRecordRequest putRecordRequest = new PutRecordRequest();
putRecordRequest.setDeliveryStreamName(deliveryStreamName);

String data = line + "\n";

Record record = new Record().withData(ByteBuffer.wrap(data.getBytes()));
putRecordRequest.setRecord(record);

// Put record into the DeliveryStream
firehoseClient.putRecord(putRecordRequest);
```

コードコンテキストの詳細については、 AWS SDK に含まれているサンプルコードを参照してください。リクエストとレスポンスの構文の詳細については、「Firehose API Operations」の関連するトピックを参照してください。

## PutRecordBatch を使用したバッチ書き込みオペレーション

データの入力では、Firehose ストリーム名とレコードのリストのみが必要です。Amazon Data Firehose バッチはファイルを Amazon S3 にロードする前に、複数のレコードをバッチ処理するため、レコードの区切り文字を追加することをお勧めします。データレコードをバッチで Firehose ストリームに出力するには、次のコードを使用します。

```
PutRecordBatchRequest putRecordBatchRequest = new PutRecordBatchRequest();
putRecordBatchRequest.setDeliveryStreamName(deliveryStreamName);
putRecordBatchRequest.setRecords(recordList);

// Put Record Batch records. Max No.Of Records we can put in a
// single put record batch request is 500
firehoseClient.putRecordBatch(putRecordBatchRequest);
```

```
recordList.clear();
```

コードコンテキストの詳細については、 AWS SDK に含まれているサンプルコードを参照してください。リクエストとレスポンスの構文の詳細については、「Firehose API Operations」の関連するトピックを参照してください。

# CloudWatch Logs を Firehose に送信する

CloudWatch Logs イベントは、CloudWatch サブスクリプションフィルターを使用して Firehose に送信できます。詳細については、「Subscription filters with Amazon Data Firehose」を参照してください。

CloudWatch Logs イベントは、圧縮 gzip 形式で Firehose に送信されます。解凍されたログイベントを Firehose の宛先に配信する場合は、Firehose の解凍機能を使用して CloudWatch Logs を自動的に解凍できます。

> **Important**
>
> Firehose は現在、Amazon OpenSearch Service の宛先への CloudWatch Logs の配信をサポートしていません。Amazon CloudWatch が複数のログイベントを 1 つの Firehose レコードにまとめても、Amazon OpenSearch Service は 1 つのレコードで複数のログイベントを受け入れることができないためです。代わりの方法として、CloudWatch Logs で Amazon OpenSearch Service のサブスクリプションフィルターを使用することもできます。

## CloudWatch Logs を解凍する

Firehose を使用して CloudWatch Logs を配信し、解凍されたデータを Firehose ストリームの宛先に配信する場合は、Firehose のデータ形式転換 (Parquet、ORC) または動的パーティショニングを使用します。Firehose ストリームのために解凍を有効にする必要があります。

解凍を有効にするには AWS マネジメントコンソール、、、 AWS Command Line Interface または AWS SDKsを使用します。

> **Note**
>
> ストリームで解凍機能を有効にする場合は、そのストリームを CloudWatch Logs サブスクリプションフィルターにのみ使用し、Vended Logs には使用しません。CloudWatch Logs と Vended Logs の両方を取り込むために使用されるストリームで解凍機能を有効にした

場合、Firehose への Vended Logs の取り込みは失敗します。この解凍機能は CloudWatch Logs 専用です。

## CloudWatch Logs の解凍後にメッセージを抽出する

解凍を有効にすると、メッセージ抽出も有効にするオプションを使用できます。メッセージ抽出を使用する場合、Firehose は、解凍された CloudWatch Logs レコードから所有者、ロググループ、ログストリームなど、すべてのメタデータを除外し、メッセージフィールド内のコンテンツのみを配信します。データを Splunk の宛先に配信する場合は、Splunk がデータを解析できるように、メッセージ抽出をオンにする必要があります。解凍後の出力例を次に示します (メッセージ抽出あり/なし)。

図 1: メッセージ抽出なしの解凍後の出力例:

```
{
 "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\":\"Root1\"}"
 },
 {
 "id": "31953106606966983378809025079804211143289615424298221569",
 "timestamp": 1432826855000,
 "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root2\"}"
 },
 {
 "id": "31953106606966983378809025079804211143289615424298221570",
 "timestamp": 1432826855000,
 "message": "{\"eventVersion\":\"1.03\",\"userIdentity\":{\"type\":\"Root3\"}"
 }
 ]
}
```

図 2: メッセージ抽出ありの解凍後の出力例:

```
{"eventVersion":"1.03","userIdentity":{"type":"Root1"}
{"eventVersion":"1.03","userIdentity":{"type":"Root2"}
{"eventVersion":"1.03","userIdentity":{"type":"Root3"}
```

## コンソールから新しい Firehose ストリームでの解凍を有効にする

を使用して新しい Firehose ストリームで解凍を有効にするには AWS マネジメントコンソール

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/kinesis で Kinesis コンソールを開きます。
2. ナビゲーションペインで [Amazon Data Firehose] を選択します。
3. [Firehose ストリームを作成] を選択します。
4. [ソースと送信先を選択] で次のように操作します。

   ソース

   Firehose ストリームのソース。次のソースのいずれかを選択します:

   - Direct PUT – プロデューサーアプリケーションが直接書き込む Firehose ストリームを作成するときは、こちらを選択します。Firehose で Direct PUT と統合される AWS サービス、エージェント、オープンソースサービスのリストについては、こちらのセクションを参照してください。
   - Kinesis ストリーム: Kinesis データストリームをデータソースとして使用する Firehose ストリームを設定するには、このオプションを選択します。これで、Firehose を使用して、既存の Kinesis データストリームからデータを簡単に読み取り、宛先にロードすることができるようになります。詳細については、「Writing to Firehose Using Kinesis Data Streams」を参照してください。

   送信先

   Firehose ストリームの宛先。次のいずれかを選択します。

   - Amazon S3
   - Splunk
5. [Firehose ストリーム名] で、ストリームの名前を入力します。
6. (オプション) [レコードを変換] で次のように操作します。

   - [Amazon CloudWatch Logs からソースレコードを解凍] セクションで、[解凍をオンにする] を選択します。

- 解凍後にメッセージ抽出を使用する場合は、[メッセージ抽出をオンにする] を選択します。

## 既存の Firehose ストリームでの解凍を有効にする

このセクションでは、既存の Firehose ストリームで解凍を有効にする手順について説明します。Lambda 処理が無効になっているストリームと、Lambda 処理がすでに有効になっているストリームの 2 つのシナリオについて説明します。以下のセクションでは、Lambda 関数の作成または変更、Firehose 設定の更新、組み込みの Firehose 解凍機能を正常に実装するための CloudWatch メトリクスのモニタリングなど、ケース別に手順をステップごとに説明します。

### Lambda 処理が無効になっている場合の解凍の有効化

Lambda 処理が無効になっている既存の Firehose ストリームで解凍を有効にするには、まず Lambda 処理を有効にする必要があります。この条件は、既存のストリームに対してのみ有効です。次の手順は、Lambda 処理が有効になっていない既存のストリームで解凍を有効にする方法を示しています。

1. Lambda 関数を作成する。ダミーレコードのパススルーを作成することも、このブループリントを使用して新しい Lambda 関数を作成することもできます。
2. 現在の Firehose ストリームを更新して Lambda 処理を有効にし、処理用に作成した Lambda 関数を使用します。
3. 新しい Lambda 関数でストリームを更新したら、Firehose コンソールに戻り、解凍を有効にします。
4. ステップ 1 で有効にした Lambda 処理を無効にします。これで、ステップ 1 で作成した関数を削除できます。

### Lambda 処理が有効になっている場合の解凍の有効化

Lambda 関数を含む Firehose ストリームがすでにある場合、その関数を Firehose の解凍機能に置き換えると解凍を実行できます。続行する前に、Lambda 関数コードが解凍またはメッセージ抽出のみを実行するようになっていることを確認します。Lambda 関数の出力は、図 1 または図 2 に示す例に類似しているはずです。出力が類似している場合は、次のステップを実行して Lambda 関数を置き換えることができます。

1. 現在の Lambda 関数を、このブループリント に置き換えます。新しいブループリント Lambda 関数は、着信データが圧縮されているか、解凍されているかを自動的に検出します。入力データが圧縮されている場合にのみ、解凍が実行されます。

2. 解凍のために、組み込み Firehose オプションを使用して解凍をオンにします。
3. Firehose ストリームのために CloudWatch メトリクスを有効にします (まだ有効になっていない場合)。メトリクス `CloudWatchProcessorLambda_IncomingCompressedData` をモニタリングし、このメトリクスがゼロに変わるまで待機します。これにより、Lambda 関数に送信された入力データがすべて解凍され、Lambda 関数が不要になります。
4. ストリームを解凍する必要がなくなったため、Lambda データ変換を削除します。

## Firehose ストリームでの解凍を無効にする

を使用してデータストリームの解凍を無効にするには AWS マネジメントコンソール

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/kinesis で Kinesis コンソールを開きます。
2. ナビゲーションペインで [Amazon Data Firehose] を選択します。
3. 編集する Firehose ストリームを選択します。
4. [Firehose ストリームの詳細] ページで、[設定] タブを選択します。
5. [レコードを変換および転換] セクションで、[編集] を選択します。
6. [Amazon CloudWatch Logs からソースレコードを解凍] で、[解凍をオンにする] をクリアし、[変更を保存] を選択します。

## Firehose での解凍をトラブルシューティングする

次の表は、Firehose がデータの解凍および処理中にエラーを処理する方法を示しています。これには、エラー S3 バケットへのレコードの配信、エラーのログ記録、メトリクスの出力が含まれます。また、不正なデータプットオペレーションについて返されるエラーメッセージに関しても説明します。

| 問題 | ソリューション |
| --- | --- |
| 解凍中にエラーが発生した場合、ソースデータはどうなりますか? | Amazon Data Firehose がレコードを解凍できない場合、レコードはそのまま (圧縮形式で) Firehose ストリームの作成時に指定したエラー S3 バケットに配信されます。レコードとともに、配信されるオブジェクトにはエラーコードとエラーメッセージも含まれ、これらのオブジェ |

| 問題 | ソリューション |
| --- | --- |
| | クトは decompression-failed という S3 バケットプレフィックスに配信されます。Firehose は、レコードの解凍に失敗した後も、他のレコードの処理を継続します。 |
| 解凍が成功した後に処理パイプラインにエラーが発生した場合、ソースデータはどうなりますか? | 動的パーティショニングやデータ形式転換などの解凍後に、処理ステップにおいて Amazon Data Firehose でエラーが発生した場合、レコードは、Firehose ストリームの作成時に指定したエラー S3 バケットに圧縮形式で配信されます。レコードとともに、配信されるオブジェクトにはエラーコードとエラーメッセージも含まれます。 |
| エラーや例外が発生した場合はどのように通知されますか? | 解凍中にエラーまたは例外が発生した場合、CloudWatch Logs を設定すると、Firehose は、エラーメッセージを CloudWatch Logs にログ記録します。さらに、Firehose は、モニタリングできる CloudWatch メトリクスにメトリクスを送信します。また、オプションで、Firehose によって発行されたメトリクスに基づいてアラームを作成することもできます。 |
| put オペレーションが CloudWatch Logs から取得されない場合はどうなりますか? | お客様の puts が CloudWatch Logs からのものではない場合、次のエラーメッセージが返されます:<br>`Put to Firehose failed for AccountId: <accountID>, FirehoseName: <firehosename> because the request is not originating from allowed source types.` |
| Firehose は解凍機能についてどのようなメトリクスを出力しますか? | Firehose は、あらゆるレコードの解凍についてメトリクスを出力します。DecompressedRecords の失敗もしくは成功、または DecompressedBytes の失敗もしくは成功の数を取得するために、期間 (1 分)、統計 (合計)、日付範囲を選択する必要があります。詳細については、「CloudWatch Logs の解凍メトリクス」を参照してください。 |

# CloudWatch Events を Firehose に送信する

CloudWatch Events ルールにターゲットを追加することで、Amazon CloudWatch が Firehose ストリームにイベントを送信するよう設定できます。

既存の Firehose ストリームにイベントを送信するターゲットを CloudWatch Events ルールで作成するには

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. [ルールを作成] を選択します。
3. [ステップ 1: ルールを作成] ページの [ターゲット] で、[ターゲットの追加]、[Firehose ストリーム] の順に選択します。
4. 既存の Firehose ストリームを選択します。

CloudWatch Events ルールの作成の詳細については、「Amazon CloudWatch Events の開始方法」を参照してください。

# Firehose にデータを送信する AWS IoT ように を設定する

アクションを追加することで、Firehose ストリームに情報を送信する AWS IoT ように を設定できます。

既存の Firehose ストリームにイベントを送信するアクションを作成するには

1. AWS IoT コンソールでルールを作成するときは、ルールの作成ページの「1 つ以上のアクションを設定する」で、「アクションの追加」を選択します。
2. [Amazon Kinesis Firehose ストリームにメッセージを送信する] を選択します。
3. アクションの設定を選択します。
4. [ストリーム名] で、既存の Firehose ストリームを選択します。
5. [Separator] には、レコード間に挿入するための区切り記号を選択します。
6. [IAM role name] には、既存の IAM ロールを選択するか、または [Create a new role] を選択します。
7. [アクションを追加] を選択します。

AWS IoT ルールの作成の詳細については、AWS 「 IoT ルールのチュートリアル」を参照してください。

# Amazon Data Firehose でソースデータを変換する

Amazon Data Firehoseでは、Lambda 関数を呼び出して、着信ソースデータを変換してから宛先に配信できます。Amazon Data Firehose のデータ変換は、Firehose ストリームの作成時に有効にすることができます。

## データ変換フローを理解する

Firehose のデータ変換を有効にすると、Firehose は着信データをバッファリングします。バッファリングサイズのヒントの範囲は 0.2 MB ~ 3 MB です。デフォルトの Lambda バッファリングサイズのヒントは、Splunk と Snowflake を除くすべての宛先で 1 MB です。Splunk と Snowflake の場合、デフォルトのバッファリングのヒントは 256 KB です。Lambda バッファリング間隔のヒントの範囲は 0 ~ 900 秒です。デフォルトの Lambda バッファリング間隔のヒントは、Snowflake を除くすべての宛先で 60 秒です。Snowflake の場合、デフォルトのバッファリングのヒントの間隔は 30 秒です。バッファリングサイズを調整するには、CreateDeliveryStream または UpdateDestination API の ProcessingConfiguration パラメータを、`BufferSizeInMBs` および `IntervalInSeconds` という ProcessorParameter で設定します。次に、Firehose は、同期呼び出しモードを使用して、バッファされた各バッチと同期的に指定された Lambda AWS Lambda 関数を呼び出します。変換されたデータは、Lambda から Firehose に送信されます。その後、変換されたデータは、指定された宛先のバッファリングサイズとバッファリング間隔のいずれかに到達したときに、Firehose より宛先に配信されます。到達順序は関係ありません。

> **Important**
>
> Lambda 同期呼び出しモードには、リクエストとレスポンスの両方について、ペイロードサイズに 6 MB の制限があります。関数にリクエストを送信するためのバッファサイズが 6 MB 以下であることを確認してください。また、関数より返るレスポンスが 6 MB を超えないことを確認します。

## Lambda の呼び出し期間

Amazon Data Firehose では、最大 5 分の Lambda の呼び出し時間がサポートされます。Lambda 関数が完了するまでに 5 分以上かかると、次のエラーが発生します。Firehose は AWS Lambda を呼び出すときにタイムアウトエラーを検出しました。サポートされている関数のタイムアウトは最大 5 分です。

このようなエラーが発生した場合の Amazon Data Firehose による処理の詳細については、「the section called “データ変換の失敗を処理する”」を参照してください。

# データ変換に必要なパラメータ

Lambda からのすべての変換されたレコードには、次のパラメータが含まれる必要があります。含まれない場合、Amazon Data Firehose はそれらのレコードを拒否し、データ変換の失敗として処理します。

For Kinesis Data Streams and Direct PUT

Lambda から変換されたレコードすべてで、次のパラメータが必要です。

- `recordId` – レコード ID は呼び出し時に Amazon Data Firehose から Lambda に渡されます。変換されたレコードには、同じレコード ID が含まれる必要があります。元のレコードの ID と変換されたレコードの ID との不一致は、データ変換失敗として扱われます。
- `result` – レコードのデータ変換のステータス。指定できる値は次のとおりです: `Ok` (レコードが正常に変換された)、`Dropped` (レコードが処理ロジックによって意図的に削除された)、`ProcessingFailed` (レコードを変換できなかった)。レコードのステータスが `Ok` または `Dropped` の場合、Amazon Data Firehose はレコードが正常に処理されたとみなします。それ以外の場合、Amazon Data Firehose はそれが正常に処理できなかったとみなします。
- `data` – base64 エンコード後の変換されたデータペイロード。

  以下は、Lambda の結果の出力例です。

```
{
    "recordId": "<recordId from the Lambda input>",
    "result": "Ok",
    "data": "<Base64 encoded Transformed data>"
}
```

For Amazon MSK

Lambda から変換されたレコードすべてで、次のパラメータが必要です。

- `recordId` – レコード ID は呼び出し時に Firehose から Lambda に渡されます。変換されたレコードには、同じレコード ID が含まれる必要があります。元のレコードの ID と変換されたレコードの ID との不一致は、データ変換失敗として扱われます。

- `result` – レコードのデータ変換のステータス。指定できる値は次のとおりです: `Ok` (レコードが正常に変換された)、`Dropped` (レコードが処理ロジックによって意図的に削除された)、`ProcessingFailed` (レコードを変換できなかった)。レコードのステータスが `Ok` または `Dropped` の場合、Firehose はレコードが正常に処理されたとみなします。それ以外の場合、Firehose はそれが正常に処理できなかったとみなします。
- `KafkaRecordValue` – base64 エンコード後の変換されたデータペイロード。

  以下は、Lambda の結果の出力例です。

```
{
    "recordId": "<recordId from the Lambda input>",
    "result": "Ok",
    "kafkaRecordValue": "<Base64 encoded Transformed data>"
}
```

# サポートされている Lambda ブループリント

これらの設計図は、 AWS Lambda 関数を作成して使用し、Amazon Data Firehose データストリーム内のデータを変換する方法を示しています。

AWS Lambda コンソールで使用可能なブループリントを表示するには

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/lambda/ で AWS Lambda コンソールを開きます。
2. [関数の作成]、[Use a blueprint (設計図の使用)] の順に選択します。
3. [ブループリント] フィールドで、キーワード `firehose` で検索して Amazon Data Firehose Lambda ブループリントを見つけます。

ブループリントのリスト:

- Amazon Data Firehose ストリームに送信されたレコードを処理する (Node.js、Python)

  このブループリントは、 AWS Lambda を使用して Firehose データストリーム内のデータを処理する方法の基本的な例を示しています。

  最終リリース日: 2016 年 11 月

  リリースノート: なし

- Firehose に送信された CloudWatch Logs を処理する

  このブループリントは非推奨です。このブループリントは使用しないでください。解凍された CloudWatch Logs データが 6 MB (Lambda の制限) を超えると、高額な料金が発生する可能性があります。Firehose に送信された CloudWatch Logs の処理については、「Writing to Firehose Using CloudWatch Logs」を参照してください。

- Syslog 形式の Amazon Data Firehose ストリームレコードを JSON (Node.js) に変換する

  このブループリントは、RFC3164 Syslog 形式の入力レコードを JSON に変換する方法を示しています。

  最終リリース日: 2016 年 11 月

  リリースノート: なし

で使用できるブループリントを表示するには AWS Serverless Application Repository

1. AWS Serverless Application Repository に移動します。
2. [すべてのアプリケーションを参照] を選択します。
3. [アプリケーション] フィールドで、キーワード `firehose` を検索します。

設計図を使用せずに Lambda 関数を作成することもできます。AWS 「Lambda の開始方法」を参照してください。

# データ変換の失敗を処理する

ネットワークタイムアウトのために、または Lambda 呼び出しの制限に達したために、Lambda 関数呼び出しが失敗した場合、Amazon Data Firehose は呼び出しをデフォルトで 3 回再試行します。呼び出しが成功しなければ、Amazon Data Firehose はそのレコードのバッチをスキップします。スキップされたレコードは処理失敗として扱われます。CreateDeliveryStream または `UpdateDestination` API を使用して、再試行オプションを指定または上書きできます。このタイプの失敗の場合、呼び出しエラーログを Amazon CloudWatch Logs に出力できます。詳細については、「CloudWatch Logs を使用して Amazon Data Firehose をモニタリングする」を参照してください。

レコードのデータ変換のステータスが `ProcessingFailed` の場合、Amazon Data Firehose はそのレコードを処理失敗として扱います。このタイプの失敗の場合、エラーログを Lambda 関数から

Amazon CloudWatch Logs に出力できます。詳細については、AWS Lambda 開発者ガイドの「AWS Lambdaの Amazon CloudWatch Logs にアクセス」を参照してください。

データ変換が失敗した場合、処理に失敗したレコードは S3 バケットの processing-failed フォルダに配信されます。レコードの形式は以下のとおりです。

```
{
    "attemptsMade": "count",
    "arrivalTimestamp": "timestamp",
    "errorCode": "code",
    "errorMessage": "message",
    "attemptEndingTimestamp": "timestamp",
    "rawData": "data",
    "lambdaArn": "arn"
}
```

`attemptsMade`

呼び出しリクエストの試行回数。

`arrivalTimestamp`

Amazon Data Firehose がレコードを受信した時間。

`errorCode`

Lambda から返された HTTP エラーコード。

`errorMessage`

Lambda から返されたエラーメッセージ。

`attemptEndingTimestamp`

Amazon Data Firehose が Lambda 呼び出しの試行を停止した時間。

`rawData`

base64 エンコード後のレコードデータ。

`lambdaArn`

Lambda 関数の Amazon リソースネーム (ARN)。

# ソースレコードのバックアップ

Amazon Data Firehose は、変換されたレコードを宛先に配信すると同時に、変換されなかったすべてのレコードを S3 バケットにバックアップできます。ソースレコードのバックアップは、Firehose ストリームの作成または更新時に有効にすることができます。ソースレコードのバックアップは、有効にした後で無効にすることはできません。

# Amazon Data Firehose のストリーミングデータのパーティショニング

動的パーティショニングを使用すると、データ内のキーを使用して (例えば、`customer_id` または `transaction_id`) Firehose でストリーミングデータを継続的にパーティショニングし、これらのキーでグループ化されたデータを、対応する Amazon Simple Storage Service (Amazon S3) プレフィックスに配信できます。これにより、Amazon Athena、Amazon EMR、Amazon Redshift Spectrum、Amazon QuickSight などのさまざまなサービスを使用して、Amazon S3 のストリーミングデータに対して高性能でコスト効率の高い分析を簡単に実行できるようになります。さらに、AWS Glue は、動的にパーティション分割されたストリーミングデータが Amazon S3 に配信された後に、追加の処理が必要なユースケースで、より高度な抽出、変換、ロード (ETL) ジョブを実行できます。

データをパーティショニングすることで、スキャンされるデータ量が最小限に抑えられ、パフォーマンスが最適化され、Amazon S3 での分析クエリのコストが削減されます。また、データへのきめ細かいアクセスも向上します。Firehose ストリームは、データをキャプチャして Amazon S3 にロードするために従来より使用されています。Amazon S3 ベースの分析用にストリーミングデータセットをパーティショニングするには、データを分析に使用できるようにする前に、Amazon S3 バケット間でパーティショニングアプリケーションを実行する必要がありますが、これは複雑になるか、費用がかかる場合があります。

動的パーティショニングでは、Firehose は、動的または静的に定義されたデータキーを使用して送信中のデータを継続的にグループ化し、キーごとに個々の Amazon S3 プレフィックスにデータを配信します。これにより、洞察に要する時間が数分または数時間短縮されます。また、コストを削減し、アーキテクチャを簡素化します。

トピック

# Amazon Data Firehose で動的パーティショニングを有効にする

Amazon Data Firehose マネジメントコンソール、CLI、または API を使用して、Firehose ストリームの動的パーティショニングを設定できます。

> **Important**
>
> 動的パーティショニングは、新しい Firehose ストリームを作成する場合にのみ有効にできます。動的パーティショニングがまだ有効になっていない既存の Firehose ストリームでは、動的パーティショニングを有効にすることはできません。

新しい Firehose ストリームの作成中に Firehose マネジメントコンソールを使用して動的パーティショニングを有効にして設定する方法の詳細なステップについては、「Creating an Amazon Firehose stream」を参照してください。Firehose ストリームの送信先を指定する作業に進むときは、「宛先の設定を構成する」セクションの手順に従ってください。現在、動的パーティショニングは、Amazon S3 を送信先として使用する Firehose ストリームでのみサポートされているためです。

アクティブな Firehose ストリームで動的パーティショニングを有効にすると、新しいパーティショニングキーと S3 プレフィックス式を追加するか、既存のそれらのキーと式を削除または更新することで、設定を更新できます。更新されると、Firehose は新しいキーと新しい S3 プレフィックス式の使用を開始します。

> **Important**
>
> Firehose ストリームで動的パーティショニングを有効にすると、この Firehose ストリームでは無効にできません。

## パーティショニングキーを理解する

動的パーティショニングでは、パーティショニングキーに基づいてデータをパーティショニングすることで、ストリーミング S3 データからターゲットデータセットを作成します。パーティショニングキーを使用すると、特定の値に基づいてストリーミングデータをフィルタリングできます。たとえば、顧客 ID と国に基づいてデータをフィルタリングする必要がある場合は、1 つのパーティショニングキーとして `customer_id` のデータフィールドを、また別のパーティショニングキーとして `country` のデータフィールドを指定できます。次に、(サポートされている形式を使用して) 式を指

定し、動的にパーティショニングされたデータレコードの配信先となる S3 バケットプレフィックスを定義します。

パーティショニングキーは、次の方法で作成できます。

- Inline parsing – このメソッドは、JSON 形式のデータレコードからパーティショニングするためのキーの抽出で、Firehose 組み込みサポートメカニズムである jq パーサーを使用します。現在、jq 1.6 バージョンのみをサポートしています。
- AWS Lambda 関数 – このメソッドは、指定された AWS Lambda 関数を使用して、パーティショニングに必要なデータフィールドを抽出して返します。

> **Important**
>
> 動的パーティショニングを有効にする場合、データをパーティショニングするには、これらのメソッドの少なくとも 1 つを設定する必要があります。これらのメソッドのいずれかを設定して、パーティショニングキーを指定することも、両方を同時に指定することもできます。

## インライン解析でパーティショニングキーを作成する

ストリーミングデータの動的パーティショニングメソッドとしてインライン解析を設定するには、パーティショニングキーとして使用するデータレコードパラメータを選択し、それぞれの指定したパーティショニングキーの値を提供する必要があります。

次のサンプルデータレコードは、インライン解析を使用してパーティションキーを定義する方法を示しています。データは Base64 形式でエンコードされる必要があることに留意してください。CLI の例を参照することもできます。

```
{
   "type": {
    "device": "mobile",
    "event": "user_clicked_submit_button"
  },
  "customer_id": "1234567890",
  "event_timestamp": 1565382027,   #epoch timestamp
  "region": "sample_region"
}
```

たとえば、customer_id パラメータまたは event_timestamp パラメータに基づいてデータをパーティショニングすることを選択できます。これは、レコードが配信される S3 プレフィックスの決定に使用される各レコードの customer_id パラメータまたは event_timestamp パラメータの値が必要であることを意味します。また、式 .type.device を用いた device のように、ネストされたパラメータを選択することもできます。動的パーティショニングロジックは、複数のパラメータに依存する可能性があります。

パーティショニングキーのデータパラメータを選択した後、各パラメータを有効な jq 式にマップします。次のテーブルに、jq 式へのパラメータのマッピングを示します。

| パラメータ | ip 式 |
| --- | --- |
| customer_id | .customer_id |
| device | .type.device |
| year | .event_timestamp\| strftime("%Y") |
| month | .event_timestamp\| strftime("%m") |
| day | .event_timestamp\| strftime("%d") |
| hour | .event_timestamp\| strftime("%H") |

実行時に、Firehose は上の右の列を使用して、各レコードのデータに基づいてパラメータを評価します。

## AWS Lambda 関数を使用してパーティショニングキーを作成する

圧縮または暗号化されたデータレコード、または JSON 以外のファイル形式のデータの場合、統合 AWS Lambda 関数を独自のカスタムコードとともに使用してレコードを解凍、復号、または変換し、パーティショニングに必要なデータフィールドを抽出して返すことができます。これは、Firehose で現在利用できる既存の変換 Lambda 関数の拡張です。同じ Lambda 関数を使用して、動的パーティショニングに使用できるデータフィールドを変換、解析、および返すことができます。

入力から出力までのすべての読み取りレコードを再生し、レコードからパーティショニングキーを抽出する Python の Lambda 関数を処理する Firehose ストリームの例を次に示します。

```
from __future__ import print_function
import base64
import json
import datetime

# Signature for all Lambda functions that user must implement
def lambda_handler(firehose_records_input, context):
    print("Received records for processing from DeliveryStream: " +
 firehose_records_input['deliveryStreamArn']
          + ", Region: " + firehose_records_input['region']
          + ", and InvocationId: " + firehose_records_input['invocationId'])

    # Create return value.
    firehose_records_output = {'records': []}

    # Create result object.
    # Go through records and process them

    for firehose_record_input in firehose_records_input['records']:
        # Get user payload
        payload = base64.b64decode(firehose_record_input['data'])
        json_value = json.loads(payload)

        print("Record that was received")
        print(json_value)
        print("\n")
        # Create output Firehose record and add modified payload and record ID to it.
        firehose_record_output = {}
        event_timestamp = datetime.datetime.fromtimestamp(json_value['eventTimestamp'])
        partition_keys = {"customerId": json_value['customerId'],
                          "year": event_timestamp.strftime('%Y'),
                          "month": event_timestamp.strftime('%m'),
                          "day": event_timestamp.strftime('%d'),
                          "hour": event_timestamp.strftime('%H'),
                          "minute": event_timestamp.strftime('%M')
                          }

        # Create output Firehose record and add modified payload and record ID to it.
        firehose_record_output = {'recordId': firehose_record_input['recordId'],
                                  'data': firehose_record_input['data'],
                                  'result': 'Ok',
                                  'metadata': { 'partitionKeys': partition_keys }}
```

```
        # Must set proper record ID
        # Add the record to the list of output records.

        firehose_records_output['records'].append(firehose_record_output)

    # At the end return processed records
    return firehose_records_output
```

入力から出力までのすべての読み取りレコードを再生し、レコードからパーティショニングキーを抽出する Go の Lambda 関数を処理する Firehose ストリームの例を次に示します。

```
package main

import (
 "fmt"
 "encoding/json"
 "time"
 "strconv"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

type DataFirehoseEventRecordData struct {
 CustomerId string `json:"customerId"`
}

func handleRequest(evnt events.DataFirehoseEvent) (events.DataFirehoseResponse, error)
 {

 fmt.Printf("InvocationID: %s\n", evnt.InvocationID)
 fmt.Printf("DeliveryStreamArn: %s\n", evnt.DeliveryStreamArn)
 fmt.Printf("Region: %s\n", evnt.Region)

 var response events.DataFirehoseResponse

 for _, record := range evnt.Records {
  fmt.Printf("RecordID: %s\n", record.RecordID)
  fmt.Printf("ApproximateArrivalTimestamp: %s\n", record.ApproximateArrivalTimestamp)

  var transformedRecord events.DataFirehoseResponseRecord
```

```
  transformedRecord.RecordID = record.RecordID
  transformedRecord.Result = events.DataFirehoseTransformedStateOk
  transformedRecord.Data = record.Data

  var metaData events.DataFirehoseResponseRecordMetadata
  var recordData DataFirehoseEventRecordData
  partitionKeys := make(map[string]string)

  currentTime := time.Now()
  json.Unmarshal(record.Data, &recordData)
  partitionKeys["customerId"] = recordData.CustomerId
  partitionKeys["year"] = strconv.Itoa(currentTime.Year())
  partitionKeys["month"] = strconv.Itoa(int(currentTime.Month()))
  partitionKeys["date"] = strconv.Itoa(currentTime.Day())
  partitionKeys["hour"] = strconv.Itoa(currentTime.Hour())
  partitionKeys["minute"] = strconv.Itoa(currentTime.Minute())
  metaData.PartitionKeys = partitionKeys
  transformedRecord.Metadata = metaData

  response.Records = append(response.Records, transformedRecord)
 }

 return response, nil
}

func main() {
 lambda.Start(handleRequest)
}
```

# Amazon S3 バケットプレフィックスを使用してデータを配信する

Amazon S3 を宛先として使用する Firehose ストリームを作成する場合は、Firehose がデータを配信する Amazon S3 バケットを指定する必要があります。Amazon S3 バケットプレフィックスを使用して、S3 バケットに保存するデータを整理できます。Amazon S3 バケットプレフィックスは、類似オブジェクトをグループ化できるディレクトリと同様のものです。

動的パーティショニングでは、パーティショニングされたデータは、指定された Amazon S3 プレフィックスに配信されます。動的パーティショニングを有効にしない場合、Firehose ストリームの S3 バケットプレフィックスの指定はオプションです。ただし、動的パーティショニングを有効にす

る場合は、Firehose がパーティショニングされたデータを配信する S3 バケットプレフィックスを指定する必要があります。

動的パーティショニングを有効にするすべての Firehose ストリームで、S3 バケットプレフィックス値は、その Firehose ストリームのために指定されたパーティショニングキーに基づく式で構成されます。上記のデータレコードの例を再度使用して、上記で定義したパーティショニングキーに基づく式で構成される次の S3 プレフィックス値を構築できます。

```
"ExtendedS3DestinationConfiguration": {
"BucketARN": "arn:aws:s3:::my-logs-prod",
"Prefix": "customer_id=!{partitionKeyFromQuery:customer_id}/
    device=!{partitionKeyFromQuery:device}/
    year=!{partitionKeyFromQuery:year}/
    month=!{partitionKeyFromQuery:month}/
    day=!{partitionKeyFromQuery:day}/
    hour=!{partitionKeyFromQuery:hour}/"
}
```

Firehose は、実行時に上記の式を評価します。同じ評価された S3 プレフィックス式に一致するレコードを 1 つのデータセットにグループ化します。Firehose は、評価された S3 プレフィックスに各データセットを配信します。S3 へのデータセット配信の頻度は、Firehose ストリームバッファリング設定によって決まります。その結果、この例のレコードは次の S3 オブジェクトキーに配信されます。

```
s3://my-logs-prod/customer_id=1234567890/device=mobile/year=2019/month=08/day=09/
hour=20/my-delivery-stream-2019-08-09-23-55-09-a9fa96af-e4e4-409f-bac3-1f804714faaa
```

動的パーティショニングでは、S3 バケットプレフィックスで次の式形式を使用する必要があります: `!{namespace:value}`。ここで、名前空間は `partitionKeyFromQuery` または `partitionKeyFromLambda`、またはその両方です。インライン解析を使用してソースデータのパーティショニングキーを作成している場合は、次の形式で指定された式で構成される S3 バケットプレフィクス値を指定する必要があります: `"partitionKeyFromQuery:keyID"`。AWS Lambda 関数を使用してソースデータのパーティショニングキーを作成している場合は、次の形式で指定された式で構成される S3 バケットプレフィックス値を指定する必要があります。`"partitionKeyFromLambda:keyID"`。

> **Note**
>
> また、Hive スタイルの形式 (customer_id=!{partitionKeyFromQuery:customer_id} など) を使用して S3 バケットプレフィクス値を指定することもできます。

詳細については、「Creating an Amazon Firehose stream」および「Custom Prefixes for Amazon S3 Objects」の「Choose Amazon S3 for Your Destination」を参照してください。

## Amazon S3 にデータを配信する際に改行区切り文字を追加する

[改行区切り文字] を有効にして、Amazon S3 に配信されるオブジェクト内のレコード間に改行区切り文字を追加できます。これは、Amazon S3 のオブジェクトの解析に役立ちます。これは、動的パーティショニングが集約データに適用される場合にも特に便利です。それは、マルチレコードの集約解除 (動的にパーティション化する前に集約データに適用する必要があります) は、解析プロセスの一環としてレコードから新しい行を削除するからです。

# 集約データに動的パーティショニングを適用する

動的パーティショニングを集約データ (たとえば、複数のイベント、ログ、または単一の `PutRecord` および `PutRecordBatch` API コールに集約されたレコードなど) に適用できますが、このデータはまず集約解除する必要があります。マルチレコードの集約解除を有効にすることで、データを集約解除できます。これは、Firehose ストリーム内のレコードを解析して分離するプロセスです。

マルチレコードの集約解除は、`JSON` タイプのいずれかになります。これは、レコードの分離が連続する JSON オブジェクトに基づいていることを意味します。集約解除はタイプ `Delimited` にすることもできます。これは、指定されたカスタム区切り文字に基づいてレコードの分離が実行されることを意味します。このカスタム区切り文字は base-64 でエンコードされた文字列である必要があります。例えば、カスタム区切り記号 #### として次の文字列を使用する場合、base-64 のエンコード形式で指定する必要があります。これにより、`IyMjIw==` に変換されます。JSON または区切り文字によるレコードの集約解除は、レコードごとに 500 に制限されます。

> **Note**
>
> JSON レコードを集約解除する際には、サポートされている JSON 形式で入力がまだ表示されていることを確認してください。JSON オブジェクトは、区切り文字や改行区切り

(JSONL) のない、単一の行にのみ配置される必要があります。JSON オブジェクトの配列は有効な入力ではありません。
これらは正しい入力の例です: `{"a":1}{"a":2}` and `{"a":1}\n{"a":2}`
これは間違った入力の例です: `[{"a":1}, {"a":2}]`

集約データでは、動的パーティショニングを有効にすると、Firehose はレコードを解析し、指定されたマルチレコードの集約解除タイプに基づいて、各 API コール内で有効な JSON オブジェクトまたは区切り付きレコードを検索します。

**Important**

データが集約されている場合、動的パーティショニングは、データが最初に集約解除された場合にのみ適用できます。

**Important**

Firehose のデータ変換機能を使用すると、データ変換の前にディスアグリゲーションが適用されます。Firehose に入力されるデータは、ディスアグリゲーション、Lambda によるデータ変換、パーティショニングキー、の順で処理されます。

## 動的パーティショニングエラーをトラブルシューティングする

Amazon Data Firehose が Firehose ストリーム内のデータレコードを解析できない場合、または指定したパーティショニングキーを抽出できない場合、または S3 プレフィックス値に含まれる式を評価できない場合、これらのデータレコードは S3 エラーバケットプレフィックスに配信されます。このプレフィックスは、動的パーティショニングが有効にされる Firehose ストリームを作成するときに指定する必要があります。S3 エラーバケットプレフィックスには、Firehose が指定された S3 宛先に配信できないすべてのレコードが含まれています。これらのレコードは、エラータイプに基づいて整理されます。レコードとともに、配信されたオブジェクトには、エラーの理解と解決に役立つエラーに関する情報も含まれます。

Firehose ストリームの動的パーティショニングを有効にするには、この Firehose ストリームに S3 エラーバケットプレフィックスを指定する必要があります。Firehose ストリームの動的パーティショニングを有効にしない場合は、S3 エラーバケットプレフィックスの指定はオプションです。

# 動的パーティショニングのバッファリングデータ

Amazon Data Firehose は特定の期間、着信ストリーミングデータを特定のサイズにバッファリングしてから、指定の宛先に配信します。新しい Firehose ストリームの作成時にバッファリングサイズとバッファリング間隔を設定したり、既存の Firehose ストリームのバッファリングサイズとバッファリング間隔を更新したりできます。バッファサイズは MB 単位で、バッファ間隔は秒単位です。

> **Note**
>
> ゼロバッファリング機能は、動的パーティショニングでは使用できません。

動的パーティショニングが有効になっている場合、Firehose は、設定されたバッファリングのヒント (サイズと時間) に基づいて特定のパーティションに属するレコードを内部的にバッファリングしてから、これらのレコードを Amazon S3 バケットに配信します。最大サイズのオブジェクトを配信するため、Firehose は、内部でマルチステージバッファリングを使用します。そのため、レコードのバッチのエンドツーエンドの遅延が、設定したバッファリングヒント時間の 1.5 倍程度かかる可能性があります。この時間は、Firehose ストリームのデータ鮮度に影響します。

アクティブパーティション数は、配信バッファ内のアクティブパーティションの総数です。たとえば、動的パーティショニングクエリが 1 秒あたり 3 つのパーティションを構築し、60 秒ごとに配信をトリガーするバッファのヒント設定がある場合、平均して 180 個のアクティブパーティションが作成されます。Firehose がパーティション内のデータを宛先に配信できない場合、このパーティションは、配信可能になるまで配信バッファ内でアクティブとしてカウントされます。

レコードデータフィールドと S3 プレフィックス式に基づいて S3 プレフィクスが新しい値に評価されると、新しいパーティションが作成されます。アクティブパーティションごとに新しいバッファが作成されます。同じ評価された S3 プレフィックスを持つ後続のすべてのレコードが、そのバッファに配信されます。

バッファがバッファリングサイズ制限またはバッファ時間間隔を満たすと、Firehose はバッファリングデータを含むオブジェクトを作成し、指定した Amazon S3 プレフィックスに配信します。オブジェクトが配信された後、そのパーティションのバッファとパーティション自体が削除され、アクティブなパーティション数から除外されます。

Firehose は、各パーティションのバッファリングサイズまたは間隔が個別に満たされると、各バッファリングデータを 1 つのオブジェクトとして配信します。アクティブパーティションの数が

Firehose ストリームごとの 500 個の制限に達すると、Firehose ストリームの残りのレコードは、指定された S3 エラーバケットプレフィックス (activePartitionExceeded) に配信されます。Amazon Data Firehose の制限フォームを使用して、このクォータを、指定された Firehose ストリームごとに最大 5,000 のアクティブパーティションに増やすようリクエストすることができます。さらにパーティションが必要な場合は、Firehose ストリームをさらに作成することでアクティブパーティションをそれらに分散させることができます。

# Amazon Data Firehose で入力データ形式を変換する

Amazon Data Firehose は、データを Amazon S3 に保存する前に、入力データ形式を JSON から Apache Parquet または Apache ORC に変換できます。Parquet と ORC は列指向のデータ形式であり、容量を節約するだけでなく、JSON のような行指向の形式と比較してより高速なクエリを可能にします。カンマ区切り値 (CSV) や構造化テキストなど、JSON 以外の入力形式を変換する場合は、AWS Lambda を使用して最初に JSON に変換できます。詳細については、「ソースデータを変換する」を参照してください。

Amazon Data Firehose に送信する前にレコードを集約しても、データの形式を変換できます。

Amazon Data Firehose でレコードデータの形式を変換するためには、次の 3 つの要素が必要です。

## Deserializer

Amazon Data Firehose では、入力データの JSON を読み取るためにデシリアライザーが必要です。次の 2 種類のデシリアライザーのいずれかを選択できます。

複数の JSON ドキュメントを同じレコードに結合する場合は、サポートされている JSON 形式で入力が表示されていることを確認してください。JSON ドキュメントの配列は有効な入力ではありません。

例えば、`{"a": 1}{"b": 1}` は正しい入力で、`[{"a":1}, {"a":2}]` は誤った入力になります。

- Apache Hive JSON SerDe
- OpenX JSON SerDe

### JSON デシリアライザーを選択する

入力 JSON に次の形式のタイムスタンプが含まれている場合は、OpenX JSON SerDe を選択します。

- yyyy-MM-dd'T'HH:mm:ss[.S]'Z'。小数は最大 9 桁まで使用できます – 例: `2017-02-07T15:13:01.39256Z`。
- yyyy-[M]M-[d]d HH:mm:ss[.S]。小数は最大 9 桁まで使用できます – 例: `2017-02-07 15:13:01.14`。
- エポック秒 – たとえば、`1518033528` です。

- エポックミリ秒 – たとえば、1518033528123 です。
- 浮動小数点エポック秒 – たとえば、1518033528.123 です。

OpenX JSON SerDe はピリオド (.) をアンダースコア (_) に変換できます。デシリアライズする前に、JSON キーを小文字に変換することもできます。Amazon Data Firehose を介したこのデシリアライザーで利用可能になるオプションの詳細については、「OpenXJsonSerDe」を参照してください。

どのデシリアライザーを選択するかわからない場合は、サポートされていないタイムスタンプがない限り、OpenX JSON SerDe を使用します。

前述の形式以外のタイムスタンプがある場合は、Apache Hive JSON SerDe を使用します。このデシリアライザーを選択すると、使用するタイムスタンプ形式を指定できます。指定するには、Joda-Time `DateTimeFormat` 形式の文字列のパターン構文に従います。詳細については、「Class DateTimeFormat」を参照してください。

特殊な値 `millis` を使用して、エポックミリ秒でタイムスタンプを解析することもできます。形式を指定していない場合は、Amazon Data Firehose はデフォルトで `java.sql.Timestamp::valueOf` を使用します。

Hive JSON SerDe は以下を許可しません。

- 列名のピリオド (.)。
- タイプが `uniontype` のフィールド。
- スキーマに数値型を持つフィールドですが、JSON 形式の文字列です。たとえば、スキーマが (int) で JSON が `{"a":"123"}` の場合、Hive SerDe ではエラーが発生します。

Hive SerDe はネストされた JSON を文字列に変換しません。たとえば、`{"a":{"inner":1}}` がある場合、`{"inner":1}` は文字列として扱われません。

## Schema

Amazon Data Firehose では、そのデータを解釈する方法を決定するスキーマが必要です。AWS Glue を使用して、 AWS Glue Data Catalogでスキーマを作成します。Amazon Data Firehose はそのスキーマを参照し、使用して入力データを解釈します。同じスキーマを使用して、Amazon Data Firehose ソフトウェアと分析ソフトウェアの両方を設定できます。詳細については、「 AWS Glue デベロッパーガイドAWS 」の「 Glue データカタログの入力」を参照してください。

> **Note**
>
> AWS Glue Data Catalog で作成されたスキーマは、入力データ構造と一致する必要があります。一致していないと、変換されたデータに、スキーマで指定されていない属性が含まれなくなります。ネストされた JSON を使用する場合は、STRUCT タイプを JSON データの構造を反映したスキーマで使用します。STRUCT タイプを使ってネストされた JSON を処理する方法については、こちらの例を参照してください。

> **Important**
>
> サイズ制限を指定しないデータ型については、単一の行内にあるすべてのデータのために 32 MB という実用的制限が設定されています。
> CHAR または VARCHAR の長さを指定すると、Firehose は入力データを読み取る際に、指定された長さで文字列を切り詰めます。基盤となるデータ文字列の方が長い場合は、変更されません。

## Serializer

Firehose では、データをターゲットの列指向ストレージ形式 (Parquet または ORC) に変換するためのシリアライザーが必要です – 次の 2 種類のシリアライザーのいずれかを選択できます。

- ORC SerDe
- Parquet SerDe

### シリアライザーを選択する

選択するシリアライザーは、ビジネスニーズに応じて異なります。シリアライザーの 2 つのオプションの詳細については、「ORC SerDe」および「Parquet SerDe」を参照してください。

# レコード形式の変換を有効にする

レコード形式の変換を有効にすると、Amazon Data Firehose の宛先を Amazon OpenSearch Service、Amazon Redshift、または Splunk に設定することはできません。形式の変換を有効にすると、Amazon S3 が唯一 Firehose ストリームに使用できる宛先になります。次のセ

クションは、コンソールおよび Firehose API オペレーションからレコード形式変換を有効にする方法を示しています。を使用してレコード形式の変換を設定する方法の例については CloudFormation、AWS「::DataFirehose::DeliveryStream」を参照してください。

## コンソールからのレコード形式変換を有効にする

Firehose ストリームを作成または更新するときに、コンソールでデータ形式の変換を有効にできます。データ形式の変換を有効にすると、Amazon S3 が Firehose ストリームに設定できる唯一の宛先になります。また、形式変換を有効にすると Amazon S3 圧縮が無効化されます。ただし、変換プロセスの一部として Snappy 圧縮が自動的に実行されます。この場合に Amazon Data Firehose が使用する Snappy のフレーミング形式は Hadoop と互換性があります。つまり、Snappy 圧縮の結果を使用して、Athena でこのデータに対するクエリを実行できます。Hadoop が依存する Snappy のフレーミング形式については、「BlockCompressorStream.java」を参照してください。

データ Firehose ストリームのデータ形式の変換を有効にするには

1. にサインインし AWS マネジメントコンソール、https://console.aws.amazon.com/firehose/ で Amazon Data Firehose コンソールを開きます。
2. 更新する Firehose ストリームを選択するか、「チュートリアル: コンソールから Firehose ストリームを作成する」のステップに従って新しい Firehose ストリームを作成します。
3. [Convert record format (レコード形式を変換)] で、[Record format conversion (レコード形式の変換)] を [Enabled (有効)] に設定します。
4. 目的の出力形式を選択します。2 つのオプションの詳細については、Apache Parquet および Apache ORC を参照してください。
5. AWS Glue テーブルを選択して、ソースレコードのスキーマを指定します。リージョン、データベース、テーブル、テーブルバージョンを設定します。

## Firehose API からのレコード形式変換を管理する

Amazon Data Firehose で入力データの形式を JSON から Parquet または ORC に変換する場合、ExtendedS3DestinationConfiguration または ExtendedS3DestinationUpdate で、オプションの DataFormatConversionConfiguration 要素を指定します。DataFormatConversionConfiguration を指定する場合は、次の制限が適用されます。

- BufferingHints では、レコード形式の変換を有効にすると、`SizeInMBs` を 64 未満の値に設定できません。また、形式の変換が有効でない場合、デフォルト値は 5 です。有効にすると、この値は 128 になります。

- ExtendedS3DestinationConfiguration または ExtendedS3DestinationUpdate の CompressionFormat を UNCOMPRESSED に設定する必要があります。CompressionFormat のデフォルト値は UNCOMPRESSED です。したがって、ExtendedS3DestinationConfiguration で指定しないままにすることもできます。その場合もデータは、デフォルトで Snappy 圧縮を使用して、シリアル化プロセスの一環として圧縮されます。この場合に Amazon Data Firehose が使用する Snappy のフレーミング形式は Hadoop と互換性があります。つまり、Snappy 圧縮の結果を使用して、Athena でこのデータに対するクエリを実行できます。Hadoop が依存する Snappy のフレーミング形式については、「BlockCompressorStream.java」を参照してください。シリアライザーを構成する場合は、他のタイプの圧縮を選択できます。

# データ形式変換のエラーの処理

Amazon Data Firehose がレコードを解析またはデシリアライズできない場合 (例えば、データがスキーマと一致しない場合)、エラープレフィックスを付けて Amazon S3 に書き込みます。この書き込みが失敗した場合、Amazon Data Firehose はこの書き込みを永久に再試行し、追加で配信されないようにします。失敗したレコードごとに、Amazon Data Firehose は次のスキーマを持つ JSON ドキュメントを書き込みます。

```
{
  "attemptsMade": long,
  "arrivalTimestamp": long,
  "ErrorCode": string,
  "ErrorMessage": string,
  "attemptEndingTimestamp": long,
  "rawData": string,
  "sequenceNumber": string,
  "subSequenceNumber": long,
  "dataCatalogTable": {
    "catalogId": string,
    "databaseName": string,
    "tableName": string,
    "region": string,
    "versionId": string,
    "catalogArn": string
  }
}
```

# Amazon Data Firehose でのデータ配信を理解する

Firehose ストリームにデータを送信すると、そのデータは選択した送信先に自動的に配信されます。次の表は、さまざまな宛先へのデータ配信について説明するものです。

| 目的地 | Details |
| --- | --- |
| Amazon S3 | Amazon S3 へのデータ配信の場合、Firehose は、Firehose ストリームのバッファリング設定に基づいて、複数の着信レコードを連結します。次に、Amazon S3 オブジェクトとしてレコードを Amazon S3 に配信します。デフォルトでは、Firehose は区切り文字なしでデータを連結します。レコード間に改行区切り文字を使用する場合は、[Firehose コンソール設定] または [API パラメータ] でこの機能を有効にすることで、改行区切り文字を追加できます。Firehose と Amazon S3 の宛先間のデータ配信は TLS (HTTPS) で暗号化されます。 |
| Amazon Redshift | Amazon Redshift へのデータ配信では、Firehose は最初に着信データを前に説明した形式で S3 バケットに配信します。次に Firehose は、Amazon Redshift COPY コマンドを発行して、S3 バケットから Amazon Redshift プロビジョンドクラスターまたは Amazon Redshift Serverless ワークグループにデータをロードします。Amazon Data Firehose が複数の着信レコードを Amazon S3 オブジェクトに連結した後に、Amazon S3 オブジェクトを Amazon Redshift プロビジョンドクラスターまたは Amazon Redshift Serverless ワークグループにコピーできることを確認します。詳細については、「Amazon Redshift COPY コマンドのデータ形式パラメータ」を参照してください。 |
| OpenSearch Service と OpenSearch Serverless | OpenSearch Service と OpenSearch Serverless へのデータ配信では、Amazon Data Firehose は Firehose ストリームのバッファリング設定に基づいて着信レコードをバッファリングします。次に、OpenSearch Service または OpenSearch Serverless への一括リクエストを生成して、OpenSearch Service クラスターまたは OpenSearch Serverless コレクションに複数のレコードのインデックスを作成します。Amazon Data Firehose に送信する前に、レコードが UTF-8 でエンコードされ、1 行の JSON オブジェクトにフラット化されてい |

| 目的地 | Details |
| --- | --- |
| | ることを確認します。また、1 秒ごとに設定される明示的なインデックスを使用して一括リクエストを行うには、OpenSearch Service クラスターの `rest.action.multi.allow_explicit_index` オプションを true (デフォルト) に設定する必要があります。詳細については、「Amazon OpenSearch Service デベロッパーガイド」の「OpenSearch Service Configure Advanced Options」を参照してください。 |
| Splunk | Splunk へのデータ配信の場合は、Amazon Data Firehose は送信するバイト数を連結します。データを改行文字などで区切る場合は、自分で挿入する必要があります。Splunk がそのような区切り記号を解析するように設定されていることを確認してください。S3 エラーバケット (S3 バックアップ) に配信されたデータを Splunk にリドライブするには、Splunk ドキュメントに記載されているステップに従います。 |
| HTTP エンドポイント | サポートされているサードパーティーサービスプロバイダーが所有する HTTP エンドポイントへのデータ配信のために、統合された Amazon Lambda サービスを使用して、着信レコードをサービスプロバイダーの統合が想定している形式に一致する形式に変換する関数を作成できます。受信したレコード形式の詳細については、送信先に HTTP エンドポイントを選択したサードパーティーサービスプロバイダーにお問い合わせください。 |

| 目的地 | Details |
| --- | --- |
| Snowflake | Snowflake へのデータ配信のために、Amazon Data Firehose は内部的に 1 秒間データをバッファリングし、Snowflake ストリーミング API オペレーションを使用して Snowflake にデータを挿入します。デフォルトでは、挿入したレコードは 1 秒ごとにフラッシュされ、Snowflake テーブルにコミットされます。挿入呼び出しを実行すると、Firehose は、データが Snowflake にコミットされるまでにかかる時間を測定する CloudWatch メトリクスを出力します。Firehose は現在、レコードペイロードとして単一の JSON 項目のみをサポートしており、JSON 配列をサポートしていません。入力ペイロードが有効な JSON オブジェクトであり、余分な二重引用符、引用符、エスケープ文字なしで適切に形成されていることを確認してください。 |

Firehose の宛先ごとに、独自のデータ配信の頻度があります。詳細については、「バッファリングのヒントを設定する」を参照してください。

重複レコード

Amazon Data Firehose は、データ配信に最低 1 回セマンティクスを使用します。データ配信がタイムアウトした場合など、状況によっては、元のデータ配信リクエストが最終的に通過すると、Amazon Data Firehose による配信の再試行が重複する可能性があります。これは、Amazon S3 送信先、Apache Iceberg テーブル、および Snowflake 送信先を除く、Amazon Data Firehose がサポートするすべての送信先タイプに適用されます。

トピック

- AWS アカウントとリージョン間の配信を理解する
- HTTP エンドポイント配信リクエストとレスポンスの仕様を理解する
- データ配信の失敗を処理する
- Amazon S3 オブジェクト名の形式を設定する
- OpenSearch Service のインデックスローテーションを設定する
- データ配信を一時停止および再開する

# AWS アカウントとリージョン間の配信を理解する

Amazon Data Firehose は AWS 、アカウント間の HTTP エンドポイント送信先へのデータ配信をサポートしています。Firehose ストリームと送信先として選択した HTTP エンドポイントは、異なる AWS アカウントに属することができます。

Amazon Data Firehose は、 AWS リージョン間の HTTP エンドポイント送信先へのデータ配信もサポートしています。ある AWS リージョンの Firehose ストリームから別の AWS リージョンの HTTP エンドポイントにデータを配信できます。Firehose ストリームから AWS リージョン外の HTTP エンドポイント宛先にデータを配信することもできます。たとえば、HTTP エンドポイント URL を目的の宛先に設定することで、独自のオンプレミスサーバーにデータを配信できます。これらのシナリオでは、配信コストに追加のデータ転送料金が加算されます。詳細については、「オンデマンド料金」ページの「データ転送」セクションを参照してください。

# HTTP エンドポイント配信リクエストとレスポンスの仕様を理解する

Amazon Data Firehose がカスタム HTTP エンドポイントに正常にデータを配信するには、これらのエンドポイントがリクエストを受け入れ、特定の Amazon Data Firehose リクエストおよびレスポンス形式を使用してレスポンスを送信する必要があります。このセクションでは、Amazon Data Firehose サービスがカスタム HTTP エンドポイントに送信する HTTP リクエストの形式の仕様と、Amazon Data Firehose サービスが期待する HTTP レスポンスの形式の仕様について説明します。HTTP エンドポイントは、Amazon Data Firehose がそのリクエストをタイムアウトする前の 3 分以内にリクエストに応答します。Amazon Data Firehose は、適切な形式に従わないレスポンスを配信の失敗として扱います。

## リクエストの形式

### パスと URL パラメータ

これらは、単一の URL フィールドの一部として直接設定されます。Amazon Data Firehose は、変更なしに設定されたとおりにそれらを送信します。https 送信先のみがサポートされます。URL 制限は、配信ストリーム設定時に適用されます。

> **Note**
>
> 現在、HTTP エンドポイントデータ配信では、ポート 443 のみがサポートされています。

HTTP ヘッダー - X-Amz-Firehose-Protocol-Version

このヘッダーは、リクエスト/レスポンス形式のバージョンを示すために使用されます。現在バージョンは 1.0 のみです。

HTTP ヘッダー - X-Amz-Firehose-Request-Id

このヘッダーの値は不透明な GUID であり、デバッグや重複排除に使用できます。エンドポイントの実装では、成功したリクエストと失敗したリクエストの両方について、可能であれば、このヘッダーの値を口グ記録する必要があります。リクエスト ID は、同じリクエストを複数回試行しても同じに保たれます。

HTTP ヘッダー - Content-Type

Content-Type ヘッダーの値は常に `application/json` です。

HTTP ヘッダー - Content-Encoding

Firehose ストリームは、リクエストを送信するときに GZIP を使用して本文を圧縮するように設定できます。この圧縮を有効にすると、標準的な方法に従って Content-Encoding ヘッダーの値は gzip に設定されます。圧縮が有効にされない場合、Content-Encoding ヘッダーはまったく存在しません。

HTTP ヘッダー - Content-Length

これは標準的な方法で使用されます。

HTTP ヘッダー-X-Amz-Firehose-Source-Arn:

ASCII 文字列形式で表される Firehose ストリームの ARN。ARN は、リージョン、 AWS アカウント ID、ストリーム名をエンコードします。例えば、`arn:aws:firehose:us-east-1:123456789:deliverystream/testStream`。

HTTP ヘッダー - X-Amz-Firehose-Access-Key

このヘッダーは API キーまたはその他の認証情報を運びます。delivery-stream を作成または更新するときに、API キー (別名、認可トークン) を作成または更新できます。Amazon Data Firehose では、アクセスキーのサイズが 4096 バイトに制限されます。Amazon Data Firehose は、このキーを一切解釈しようとしません。設定されたキーは、このヘッダーの値に逐語的にコピーされ

ます。ただし、Secrets Manager を使用してキーを設定する場合、シークレットは特定の JSON オブジェクト形式 {"api_key": "..."} に従う必要があります。

コンテンツは任意であり、JWT トークンまたは ACCESS_KEY を表すことができます。エンドポイントで複数フィールドの認証情報 (ユーザー名やパスワードなど) が必要な場合は、すべてのフィールドの値を、エンドポイントが認識できる形式 (JSON または CSV) で 1 つのアクセスキー内にまとめて保存する必要があります。元の内容がバイナリである場合、このフィールドは base-64 でエンコードできます。Amazon Data Firehose は、設定された値を変更やエンコードせず、コンテンツをそのまま使用します。

HTTP ヘッダー - X-Amz-Firehose-Common-Attributes

このヘッダーは、リクエスト全体やリクエスト内のすべてのレコードに関連する共通の属性 (メタデータ) を保持します。これらは、Firehose ストリームの作成時にユーザーが直接設定します。この属性の値は、次のスキーマを使用して JSON オブジェクトとしてエンコードされます。

```
"$schema": http://json-schema.org/draft-07/schema#

properties:
  commonAttributes:
    type: object
    minProperties: 0
    maxProperties: 50
    patternProperties:
      "^.{1,256}$":
        type: string
        minLength: 0
        maxLength: 1024
```

例を示します。

```
"commonAttributes": {
    "deployment -context": "pre-prod-gamma",
    "device-types": ""
  }
```

本文 - 最大サイズ

最大本文サイズはユーザーによって設定され、圧縮前に最大 64 MiB まで設定できます。

### 本文 - スキーマ

本文には、次の JSON スキーマ (YAML で記述) を持つ 1 つの JSON ドキュメントが含まれます。

```
"$schema": http://json-schema.org/draft-07/schema#

title: FirehoseCustomHttpsEndpointRequest
description: >
  The request body that the Firehose service sends to
  custom HTTPS endpoints.
type: object
properties:
  requestId:
    description: >
      Same as the value in the X-Amz-Firehose-Request-Id header,
      duplicated here for convenience.
    type: string
  timestamp:
    description: >
      The timestamp (milliseconds since epoch) at which the Firehose
      server generated this request.
    type: integer
  records:
    description: >
      The actual records of the Firehose stream, carrying
      the customer data.
    type: array
    minItems: 1
    maxItems: 10000
    items:
      type: object
      properties:
        data:
          description: >
            The data of this record, in Base64. Note that empty
            records are permitted in Firehose. The maximum allowed
            size of the data, before Base64 encoding, is 1024000
            bytes; the maximum length of this field is therefore
            1365336 chars.
          type: string
          minLength: 0
          maxLength: 1365336
```

```
required:
  - requestId
  - records
```

例を示します。

```
{
  "requestId": "ed4acda5-034f-9f42-bba1-f29aea6d7d8f",
  "timestamp": 1578090901599
  "records": [
    {
      "data": "aGVsbG8="
    },
    {
      "data": "aGVsbG8gd29ybGQ="
    }
  ]
}
```

## レスポンスの形式

### エラー時のデフォルトの動作

レスポンスが以下の要件に適合しない場合、Firehose サーバーは本文なしで 500 ステータスコードがあるかのように処理します。

### ステータスコード

HTTP ステータスコードは 2XX、4XX、または 5XX でなければなりませんす。

Amazon Data Firehose サーバーはリダイレクト (3XX ステータスコード) に従いません。HTTP/EP へのレコードの正常な配信と見なされるのは、レスポンスコード 200 のみです。レスポンスコード 413 (サイズを超過) は永続的な障害と見なされ、レコードバッチが設定されている場合、エラーバケットに送信されません。その他のすべてのレスポンスコードは、再試行可能なエラーとみなされ、後で説明するバックオフ再試行アルゴリズムの対象となります。

ヘッダー - コンテンツタイプ

許容できるコンテンツタイプは application/json です。

HTTP ヘッダー - Content-Encoding

Content-Encoding は使用しないでください。本文は圧縮解除しなければなりません。

HTTP ヘッダー - Content-Length

Content-Length ヘッダーは、レスポンスに本文がある場合、存在しなければなりません。

本文 - 最大サイズ

レスポンス本文のサイズは 1 MiB 以下である必要があります。

```
"$schema": http://json-schema.org/draft-07/schema#

title: FirehoseCustomHttpsEndpointResponse

description: >
  The response body that the Firehose service sends to
  custom HTTPS endpoints.
type: object
properties:
  requestId:
    description: >
      Must match the requestId in the request.
    type: string

  timestamp:
    description: >
      The timestamp (milliseconds since epoch) at which the
      server processed this request.
    type: integer

  errorMessage:
    description: >
      For failed requests, a message explaining the failure.
      If a request fails after exhausting all retries, the last
      Instance of the error message is copied to error output
      S3 bucket if configured.
    type: string
    minLength: 0
```

```
    maxLength: 8192
required:
  - requestId
  - timestamp
```

例を示します。

```
Failure Case (HTTP Response Code 4xx or 5xx)
{
  "requestId": "ed4acda5-034f-9f42-bba1-f29aea6d7d8f",
  "timestamp": "1578090903599",
  "errorMessage": "Unable to deliver records due to unknown error."
}
Success case (HTTP Response Code 200)
{
  "requestId": "ed4acda5-034f-9f42-bba1-f29aea6d7d8f",
  "timestamp": 1578090903599
}
```

エラーレスポンスの処理

すべてエラーの場合、Amazon Data Firehose サーバーは指数バックオフアルゴリズムを使用して同じレコードのバッチの配信を再試行します。再試行は、ジッタ係数 (15%) の初期バックオフ時間 (1 秒) を使用してバックオフされ、その後の各再試行はジッタを追加した式 (initial-backoff-time * (multiplier(2) ^ retry_count)) を使用してバックオフされます。バックオフ時間は最大 2 分間隔で制限されます。例えば、「n」回目の再試行では、バックオフ時間は = MAX(120, 2^n) * random(0.85, 1.15) となります。

前の数式で指定されたパラメータは変更の対象となります。エクスポネンシャルバックオフアルゴリズムで使用される正確な初期バックオフ時間、最大バックオフ時間、乗数、ジッターの割合については、 AWS Firehose のドキュメントを参照してください。

その後の再試行ごとに、レコードが配信されるアクセスキーや宛先が、Firehose ストリームの更新された設定に基づいて変更される場合があります。Amazon Data Firehose サービスは、再試行全体で同じリクエスト ID をベストエフォート方式で使用します。この最後の機能は、HTTP エンドポイントサーバーによる重複排除の目的で使用できます。許容される最大時間 (Firehose ストリーム設定に基づく) 後にリクエストが配信されない場合は、ストリーム設定に基づいてレコードのバッチをオプションでエラーバケットに配信できます。

## 例

CWLog ソースリクエストの例。

```
{
  "requestId": "ed4acda5-034f-9f42-bba1-f29aea6d7d8f",
  "timestamp": 1578090901599,
  "records": [
   {
    "data": {
      "messageType": "DATA_MESSAGE",
      "owner": "123456789012",
      "logGroup": "log_group_name",
      "logStream": "log_stream_name",
      "subscriptionFilters": [
        "subscription_filter_name"
      ],
      "logEvents": [
        {
          "id": "01234567890123456789012345678901234567890123456789012345",
          "timestamp": 1510109208016,
          "message": "log message 1"
        },
        {
          "id": "01234567890123456789012345678901234567890123456789012345",
          "timestamp": 1510109208017,
          "message": "log message 2"
        }
      ]
    }
   }
  ]
}
```

# データ配信の失敗を処理する

Amazon Data Firehose の宛先ごとに、独自のデータ配信の失敗処理があります。

Firehose ストリームを設定するときは、OpenSearch、Splunk、HTTP エンドポイントなどの多くの宛先のために、配信に失敗したデータをバックアップできる S3 バケットも設定します。Firehose が

配信に失敗した場合にデータをバックアップする方法の詳細については、このページの関連する宛先セクションを参照してください。配信に失敗したデータをバックアップできる S3 バケットへのアクセスを許可する方法については「Amazon S3 の宛先へのアクセスを Firehose に付与する」を参照してください。Firehose が、(a) ストリーム宛先へのデータ配信に失敗し、(b) 配信に失敗したためにバックアップ S3 バケットにデータを書き込みできない場合は、配信先へのデータ配信またはバックアップ用の S3 へのデータ書き込みが可能になるまで、ストリーム配信を効果的に一時停止します。

## Amazon S3

S3 バケットへのデータ配信は、さまざまな理由で失敗する場合があります。例えば、バケットが存在しない、Amazon Data Firehose が引き受ける IAM ロールにバケットへのアクセスがない、ネットワークの障害、類似したイベントなどの理由があります。そのような状況では、Amazon Data Firehose は配信が成功するまで最大 24 時間にわたり再試行し続けます。Amazon Data Firehose の最大データ保存時間は 24 時間です。データ配信が 24 時間を超えて失敗した場合、データは失われます。

S3 バケットへのデータ配信は、次のようなさまざまな理由で失敗する可能性があります。

- バケットがもう存在しない。
- Amazon Data Firehose が引き受けた IAM ロールにバケットへのアクセス権がない。
- ネットワークに問題がある。
- S3 エラー (HTTP 500 やその他の API 障害など) が発生した。

以下の場合、Amazon Data Firehose は配信を再試行します。

- DirectPut ソース: 再試行は最大 24 時間継続されます。
- Kinesis Data Streams または Amazon MSK ソース: 再試行は、ストリームで定義された保持ポリシーに達するまで無期限に継続されます。

Amazon Data Firehose は、Lambda 処理または Parquet 変換が失敗した場合にのみ、失敗したレコードを S3 エラーバケットに配信します。その他の障害シナリオでは、保持期間に達するまで S3 への再試行が継続的に行われます。Firehose はレコードを S3 に正常に配信すると、S3 オブジェクトファイルを作成し、部分的なレコード障害が発生した場合は配信を自動的に再試行し、正常に処理されたレコードで同じ S3 オブジェクトファイルを更新します。

## Amazon Redshift

Amazon Redshift が宛先の場合、Firehose ストリームの作成時に、再試行の期間 (0 ~ 7200 秒) を指定できます。

Amazon Redshift プロビジョンドクラスターまたは Amazon Redshift Serverless ワークグループへのデータ配信は、いくつかの理由で失敗する場合があります。例えば、Firehose ストリームのクラスター設定が正しくない、クラスターまたはワークグループがメンテナンス中である、ネットワークの障害が発生している場合などです。そのような状況では、Amazon Data Firehose は指定された期間にわたって、その特定のバッチの Amazon S3 オブジェクトをスキップします。スキップされたオブジェクトの情報は、マニフェストファイルとして `errors/` フォルダーの S3 バケットに配信されます。この情報は手動のバックフィルに使用できます。データを手動でマニフェストファイルにコピーする方法の詳細については、「マニフェストを使用し、データファイルを指定する」を参照してください。

## Amazon OpenSearch Service と OpenSearch Serverless

OpenSearch Service と OpenSearch Serverless が宛先である場合、Firehose ストリームの作成中に、再試行の期間 (0 ~ 7,200 秒) を指定できます。

OpenSearch Service クラスターまたは OpenSearch Serverless コレクションへのデータ配信は、いくつかの理由で失敗する場合があります。例えば、Firehose ストリームの OpenSearch Service クラスターまたは OpenSearch Serverless コレクションの設定が正しくない、OpenSearch Service クラスターまたは OpenSearch Serverless コレクションがメンテナンス中である、ネットワークの障害が発生している、または同様のイベントが発生している場合などです。そのような状況では、Amazon Data Firehose は指定された期間にわたって再試行してから、その特定のインデックスリクエストをスキップします。スキップされたドキュメントは `AmazonOpenSearchService_failed/` フォルダーの S3 バケットに配信され、手動のバックフィルに使用できます。

OpenSearch Service では、各ドキュメントは以下の JSON 形式で書かれています。

```
{
    "attemptsMade": "(number of index requests attempted)",
    "arrivalTimestamp": "(the time when the document was received by Firehose)",
    "errorCode": "(http error code returned by OpenSearch Service)",
    "errorMessage": "(error message returned by OpenSearch Service)",
    "attemptEndingTimestamp": "(the time when Firehose stopped attempting index
 request)",
```

```
    "esDocumentId": "(intended OpenSearch Service document ID)",
    "esIndexName": "(intended OpenSearch Service index name)",
    "esTypeName": "(intended OpenSearch Service type name)",
    "rawData": "(base64-encoded document data)"
}
```

OpenSearch Serverless では、各ドキュメントは以下の JSON 形式で書かれています。

```
{
    "attemptsMade": "(number of index requests attempted)",
    "arrivalTimestamp": "(the time when the document was received by Firehose)",
    "errorCode": "(http error code returned by OpenSearch Serverless)",
    "errorMessage": "(error message returned by OpenSearch Serverless)",
    "attemptEndingTimestamp": "(the time when Firehose stopped attempting index
 request)",
    "osDocumentId": "(intended OpenSearch Serverless document ID)",
    "osIndexName": "(intended OpenSearch Serverless index name)",
    "rawData": "(base64-encoded document data)"
}
```

## Splunk

Amazon Data Firehose がデータを Splunk に送信すると、Splunk からの送達確認を待機します。エラーが発生した場合、または確認タイムアウト期間内に確認が到着しない場合、Amazon Data Firehose で再試行期間カウンターが開始されます。再試行期間が終わるまで再試行が続けられます。その後、Amazon Data Firehose はデータ配信が失敗したとみなしてデータを Amazon S3 バケットにバックアップします。

Amazon Data Firehose がデータを Splunk に送信するたびに、初回か再試行かにかかわらず、確認応答タイムアウトカウンターが再度開始されます。Splunk から送達確認が来るのを待機します。再試行期間が切れた場合でも、Amazon Data Firehose は確認応答が到着するか、確認応答タイムアウトに達するまで確認応答を待機し続けます。送達確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターの残り時間があるかどうかをチェックして判別します。残り時間がある場合は、確認が到着するか再試行時間が切れたと判断されるまで再試行され口ジックが繰り返されます。

送達確認の受信失敗だけが、発生する可能性のあるデータ配信エラーのタイプではありません。他のタイプのデータ配信エラーの詳細については、「Splunk データ配信エラー」を参照してくださ

い。再試行期間が 0 より大きい場合、すべてのデータ配信エラーで再試行ロジックがトリガーされます。

以下に、エラーレコードの例を示します。


```
{
  "attemptsMade": 0,
  "arrivalTimestamp": 1506035354675,
  "errorCode": "Splunk.AckTimeout",
  "errorMessage": "Did not receive an acknowledgement from HEC before the HEC
 acknowledgement timeout expired. Despite the acknowledgement timeout, it's possible
 the data was indexed successfully in Splunk. Amazon Data Firehose backs up in Amazon
 S3 data for which the acknowledgement timeout expired.",
  "attemptEndingTimestamp": 13626284715507,
  "rawData":
 "MiAyNTE2MjAyNzIyMDkgZW5pLTA1ZjMyMmQ1IDIxOC45Mi4xODguMjE0IDE3Mi4xNi4xLjE2NyAyNTIzMyAxNDMzIDYgM
  "EventId": "49577193928114147339600778471082492393164139877200035842.0"
}
```


## HTTP エンドポイント送信先

Amazon Data Firehose が HTTP エンドポイントの宛先にデータを送信すると、この宛先からのレスポンスを待ちます。エラーが発生した場合、またはレスポンスタイムアウト期間内にレスポンスが到着しない場合、Amazon Data Firehose で再試行の期間カウンターが開始されます。再試行期間が終わるまで再試行が続けられます。その後、Amazon Data Firehose はデータ配信が失敗したとみなしてデータを Amazon S3 バケットにバックアップします。

Amazon Data Firehose がデータを HTTP エンドポイントの宛先に送信するたびに、初回か再試行かにかかわらず、レスポンスタイムアウトカウンターを再起動します。次に、HTTP エンドポイントの送信先から応答が到着するのを待ちます。再試行期間が切れた場合でも、Amazon Data Firehose は引き続きレスポンスが到着するかレスポンスタイムアウトに達するまでレスポンスを待機し続けます。レスポンスがタイムアウトすると、Amazon Data Firehose は再試行カウンターの残り時間があるかどうかをチェックして判別します。残り時間がある場合は、応答が到着するか再試行時間が切れたと判断されるまで再試行されロジックが繰り返されます。

応答の受信失敗だけが、発生する可能性のあるデータ配信エラーのタイプではありません。他のタイプのデータ配信エラーの詳細については、「HTTP エンドポイントデータ配信エラー」を参照してください。

以下に、エラーレコードの例を示します。

```
{
 "attemptsMade":5,
 "arrivalTimestamp":1594265943615,
 "errorCode":"HttpEndpoint.DestinationException",
 "errorMessage":"Received the following response from the endpoint destination.
 {"requestId": "109777ac-8f9b-4082-8e8d-b4f12b5fc17b", "timestamp": 1594266081268,
 "errorMessage": "Unauthorized"}",
 "attemptEndingTimestamp":1594266081318,
 "rawData":"c2FtcGxlIHJhdyBkYXRh",
 "subsequenceNumber":0,
 "dataId":"49607357361271740811418664280693044274821622880012337186.0"
}
```

## Snowflake

Snowflake の宛先の場合、Firehose ストリームを作成する際に、オプションの再試行期間 (0 ~ 7,200 秒) を指定できます。再試行期間のデフォルト値は 60 秒です。

Snowflake テーブルへのデータ配信は、Snowflake の宛先設定の誤り、Snowflake の停止、ネットワーク障害など、いくつかの理由で失敗する可能性があります。再試行ポリシーは、取得不能なエラーには適用されません。例えば、JSON ペイロードに余分な列があったが、テーブルにはないために、Snowflake がその JSON ペイロードを拒否した場合、Firehose は再配信を試行しません。代わりに、JSON ペイロードの問題を理由とするすべての挿入失敗のバックアップを、S3 エラーバケットに作成します。

同様に、ロール、テーブル、またはデータベースが正しくないために配信に失敗した場合、Firehose は再試行せず、S3 バケットにデータを書き込みます。再試行期間は、Snowflake サービスの問題、一時的なネットワークグリッチなどを理由とする失敗にのみ適用されます。これらの条件下で、Firehose は指定された期間にわたって再試行してから、S3 に配信します。失敗したレコードは snowflake-failed/ フォルダに配信され、手動バックフィルのために使用できます。

S3 に配信する各レコードの JSON の例を次に示します。

```
{
    "attemptsMade": 3,
    "arrivalTimestamp": 1594265943615,
    "errorCode": "Snowflake.InvalidColumns",
    "errorMessage": "Snowpipe Streaming does not support columns of type AUTOINCREMENT,
 IDENTITY, GEO, or columns with a default value or collation",
    "attemptEndingTimestamp": 1712937865543,
    "rawData": "c2FtcGxlIHJhdyBkYXRh"
```

```
}
```

# Amazon S3 オブジェクト名の形式を設定する

Firehose がデータを Amazon S3 に配信する場合、S3 オブジェクトキー名は <評価されたプレフィックス><サフィックス> の形式に従います (ここで、サフィックスの形式は <Firehose ストリーム名>-<Firehose ストリームバージョン>-<年>-<月>-<日>-<時>-<分>-<秒>-<UUID><ファイル拡張子> <Firehose ストリームバージョン>です)。これは 1 から始まり、Firehose ストリームの設定が変更されるたびに 1 ずつ増加します。Firehose ストリーム設定 (例: S3 バケットの名前、バッファリングのヒント、圧縮、暗号化) は変更できます。これを行うには、Firehose コンソールまたは UpdateDestination API オペレーションを使用します。

<evaluated prefix> の場合、Firehose はデフォルトの時刻プレフィックスを YYYY/MM/dd/HH の形式で追加します。このプレフィックスはバケットで論理的階層を作成します。それぞれのフォワードスラッシュ (/) は階層でレベルを作成します。この構造は、実行時に評価される式を含むカスタムプレフィックスを指定することで変更できます。カスタムプレフィックスを指定する方法については、「Amazon Simple Storage Service オブジェクトのカスタムプレフィックス」を参照してください。

デフォルトでは、時刻プレフィックスおよびサフィックスで使用されるタイムゾーンは UTC ですが、任意のタイムゾーンに変更できます。たとえば、UTC の代わりに日本標準時を使用するには、AWS マネジメントコンソール または API パラメータ設定 (CustomTimeZone) でタイムゾーンをアジア/東京に設定できます。次のリストには、Firehose が S3 プレフィックス設定のためにサポートするタイムゾーンが含まれています。

## サポートされているタイムゾーン

Firehose が S3 プレフィックス設定のためにサポートするタイムゾーンのリストを次に示します。

Africa

```
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmera
Africa/Bangui
Africa/Banjul
Africa/Bissau
Africa/Blantyre
Africa/Bujumbura
```

```
Africa/Cairo
Africa/Casablanca
Africa/Conakry
Africa/Dakar
Africa/Dar_es_Salaam
Africa/Djibouti
Africa/Douala
Africa/Freetown
Africa/Gaborone
Africa/Harare
Africa/Johannesburg
Africa/Kampala
Africa/Khartoum
Africa/Kigali
Africa/Kinshasa
Africa/Lagos
Africa/Libreville
Africa/Lome
Africa/Luanda
Africa/Lubumbashi
Africa/Lusaka
Africa/Malabo
Africa/Maputo
Africa/Maseru
Africa/Mbabane
Africa/Mogadishu
Africa/Monrovia
Africa/Nairobi
Africa/Ndjamena
Africa/Niamey
Africa/Nouakchott
Africa/Ouagadougou
Africa/Porto-Novo
Africa/Sao_Tome
Africa/Timbuktu
Africa/Tripoli
Africa/Tunis
Africa/Windhoek
```

## America

```
America/Adak
America/Anchorage
```

```
America/Anguilla
America/Antigua
America/Aruba
America/Asuncion
America/Barbados
America/Belize
America/Bogota
America/Buenos_Aires
America/Caracas
America/Cayenne
America/Cayman
America/Chicago
America/Costa_Rica
America/Cuiaba
America/Curacao
America/Dawson_Creek
America/Denver
America/Dominica
America/Edmonton
America/El_Salvador
America/Fortaleza
America/Godthab
America/Grand_Turk
America/Grenada
America/Guadeloupe
America/Guatemala
America/Guayaquil
America/Guyana
America/Halifax
America/Havana
America/Indianapolis
America/Jamaica
America/La_Paz
America/Lima
America/Los_Angeles
America/Managua
America/Manaus
America/Martinique
America/Mazatlan
America/Mexico_City
America/Miquelon
America/Montevideo
America/Montreal
America/Montserrat
```

```
America/Nassau
America/New_York
America/Noronha
America/Panama
America/Paramaribo
America/Phoenix
America/Port_of_Spain
America/Port-au-Prince
America/Porto_Acre
America/Puerto_Rico
America/Regina
America/Rio_Branco
America/Santiago
America/Santo_Domingo
America/Sao_Paulo
America/Scoresbysund
America/St_Johns
America/St_Kitts
America/St_Lucia
America/St_Thomas
America/St_Vincent
America/Tegucigalpa
America/Thule
America/Tijuana
America/Tortola
America/Vancouver
America/Winnipeg
```

### Antarctica

```
Antarctica/Casey
Antarctica/DumontDUrville
Antarctica/Mawson
Antarctica/McMurdo
Antarctica/Palmer
```

### Asia

```
Asia/Aden
Asia/Almaty
Asia/Amman
Asia/Anadyr
Asia/Aqtau
```

```
Asia/Aqtobe
Asia/Ashgabat
Asia/Ashkhabad
Asia/Baghdad
Asia/Bahrain
Asia/Baku
Asia/Bangkok
Asia/Beirut
Asia/Bishkek
Asia/Brunei
Asia/Calcutta
Asia/Colombo
Asia/Dacca
Asia/Damascus
Asia/Dhaka
Asia/Dubai
Asia/Dushanbe
Asia/Hong_Kong
Asia/Irkutsk
Asia/Jakarta
Asia/Jayapura
Asia/Jerusalem
Asia/Kabul
Asia/Kamchatka
Asia/Karachi
Asia/Katmandu
Asia/Krasnoyarsk
Asia/Kuala_Lumpur
Asia/Kuwait
Asia/Macao
Asia/Magadan
Asia/Manila
Asia/Muscat
Asia/Nicosia
Asia/Novosibirsk
Asia/Phnom_Penh
Asia/Pyongyang
Asia/Qatar
Asia/Rangoon
Asia/Riyadh
Asia/Saigon
Asia/Seoul
Asia/Shanghai
Asia/Singapore
```

```
Asia/Taipei
Asia/Tashkent
Asia/Tbilisi
Asia/Tehran
Asia/Thimbu
Asia/Thimphu
Asia/Tokyo
Asia/Ujung_Pandang
Asia/Ulaanbaatar
Asia/Ulan_Bator
Asia/Vientiane
Asia/Vladivostok
Asia/Yakutsk
Asia/Yekaterinburg
Asia/Yerevan
```

### Atlantic

```
Atlantic/Azores
Atlantic/Bermuda
Atlantic/Canary
Atlantic/Cape_Verde
Atlantic/Faeroe
Atlantic/Jan_Mayen
Atlantic/Reykjavik
Atlantic/South_Georgia
Atlantic/St_Helena
Atlantic/Stanley
```

### Australia

```
Australia/Adelaide
Australia/Brisbane
Australia/Broken_Hill
Australia/Darwin
Australia/Hobart
Australia/Lord_Howe
Australia/Perth
Australia/Sydney
```

### Europe

```
Europe/Amsterdam
```

```
Europe/Andorra
Europe/Athens
Europe/Belgrade
Europe/Berlin
Europe/Brussels
Europe/Bucharest
Europe/Budapest
Europe/Chisinau
Europe/Copenhagen
Europe/Dublin
Europe/Gibraltar
Europe/Helsinki
Europe/Istanbul
Europe/Kaliningrad
Europe/Kiev
Europe/Lisbon
Europe/London
Europe/Luxembourg
Europe/Madrid
Europe/Malta
Europe/Minsk
Europe/Monaco
Europe/Moscow
Europe/Oslo
Europe/Paris
Europe/Prague
Europe/Riga
Europe/Rome
Europe/Samara
Europe/Simferopol
Europe/Sofia
Europe/Stockholm
Europe/Tallinn
Europe/Tirane
Europe/Vaduz
Europe/Vienna
Europe/Vilnius
Europe/Warsaw
Europe/Zurich
```

### Indian

```
Indian/Antananarivo
```

```
Indian/Chagos
Indian/Christmas
Indian/Cocos
Indian/Comoro
Indian/Kerguelen
Indian/Mahe
Indian/Maldives
Indian/Mauritius
Indian/Mayotte
Indian/Reunion
```

### Pacific

```
Pacific/Apia
Pacific/Auckland
Pacific/Chatham
Pacific/Easter
Pacific/Efate
Pacific/Enderbury
Pacific/Fakaofo
Pacific/Fiji
Pacific/Funafuti
Pacific/Galapagos
Pacific/Gambier
Pacific/Guadalcanal
Pacific/Guam
Pacific/Honolulu
Pacific/Kiritimati
Pacific/Kosrae
Pacific/Majuro
Pacific/Marquesas
Pacific/Nauru
Pacific/Niue
Pacific/Norfolk
Pacific/Noumea
Pacific/Pago_Pago
Pacific/Palau
Pacific/Pitcairn
Pacific/Ponape
Pacific/Port_Moresby
Pacific/Rarotonga
Pacific/Saipan
Pacific/Tahiti
```

```
Pacific/Tarawa
Pacific/Tongatapu
Pacific/Truk
Pacific/Wake
Pacific/Wallis
```

<ファイル拡張子> 以外のサフィックスフィールドを変更することはできません。データ形式変換または圧縮を有効にすると、Firehose は設定に基づいてファイル拡張子を付加します。次の表は、Firehose によって付加されるデフォルトのファイル拡張子を説明するものです。

| 設定 | ファイル拡張子 |
| --- | --- |
| データ形式変換: Parquet | .parquet |
| データ形式変換: ORC | .orc |
| 圧縮: Gzip | .gz |
| 圧縮: Zip | .zip |
| 圧縮: Snappy | .snappy |
| 圧縮: Hadoop-Snappy | .hsnappy |

Firehose コンソールまたは API で、希望するファイル拡張子を指定することもできます。ファイル拡張子はピリオド (.) で始まらなければならず、次の文字を含めることができます: 0-9a-z!-_.*‘()。ファイル拡張子は最大 128 文字です。

> **Note**
>
> ファイル拡張子を指定すると、データ形式変換または圧縮が有効になっている場合に Firehose が追加するデフォルトのファイル拡張子が上書きされます。

# Amazon S3 オブジェクトのカスタムプレフィックスを理解する

Amazon S3 に配信されるオブジェクトは、<evaluated prefix><suffix> の名前形式に従います。実行時に評価される式を含むカスタムプレフィックスを指定できます。指定するカスタムプレフィックスは、yyyy/MM/dd/HH のデフォルトのプレフィックスを上書きします。

カスタムプレフィックスでは、フォーム !{namespace:*value*} の式を使用できます。ここで、namespace は、以下のセクションで説明されているとおり、以下のいずれかです。

- firehose
- timestamp
- partitionKeyFromQuery
- partitionKeyFromLambda

プレフィックスの最後が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。詳細については、「Amazon Data Firehose デベロッパーガイド」の「Amazon S3 オブジェクト名の形式」を参照してください。

## timestamp 名前空間

この名前空間の有効な値は、有効な Java DateTimeFormatter 文字列である文字列です。例としては、2018 年には、式 !{timestamp:yyyy} は 2018 として評価されます。

タイムスタンプを評価するとき、Firehose は、書き込まれる Amazon S3 オブジェクトに含まれる最も古いレコードのおおよその到達タイムスタンプを使用します。

デフォルトでは、タイムスタンプは UTC です。ただし、希望するタイムゾーンを指定できます。たとえば、UTC の代わりに日本標準時を使用する場合は、 AWS マネジメントコンソール または API パラメータ設定 (CustomTimeZone) でタイムゾーンをアジア/東京に設定できます。サポートされているタイムゾーンのリストを確認するには、「Amazon S3 Object Name Format」を参照してください。

timestamp 名前空間を同じプレフィックス式で複数回使用した場合、すべてのインスタンスが同じ時点として評価されます。

## firehose 名前空間

この名前空間では、2 つの値 error-output-type および random-string を使用できます。 次の表は、これらの値の使用方法を説明しています。

**firehose** 名前空間の値

| 変換 | 説明 | 入力例 | 出力例 | 注意事項 |
| --- | --- | --- | --- | --- |
| error-output-type | Firehose ストリームと失敗の理由に応じて、次の文字列のいずれかとして評価されます: {processing-failed、AmazonOpenSearchService-failed、splunk-failed、format-conversion-failed、http-endpoint-failed}。<br><br>同じ式で複数回使用した場合、すべてのインスタンスが同じエラー文字列として評価されます。 | myPrefix/result=!{firehose:error-output-type}/!{timestamp:yyyy/MM/dd} | myPrefix/result=processing-failed/2018/08/03 | error-output-type 値は、ErrorOutputPrefix フィールドでのみ使用できます。 |
| random-string | 11 文字のランダムな文字列として評価されます。同じ式で複数回使用した場合、すべてのインスタンスが新しいランダム文 | myPrefix/!{firehose:random-string}/ | myPrefix/046b6c7f-0b/ | 両方のプレフィックスタイプで使用できます。<br><br>形式の文字列の先頭にこれを配置すると、ラン |

| 変換 | 説明 | 入力例 | 出力例 | 注意事項 |
| --- | --- | --- | --- | --- |
| | 字列として評価されます。 | | | ダム化されたプレフィックスを取得できます。これは、Amazon S3 で非常に高いスループットを実現するために必要になることがあります。 |

### partitionKeyFromLambda および partitionKeyFromQuery 名前空間

動的パーティショニングでは、S3 バケットプレフィックスで次の式形式を使用する必要があります: `!{namespace:value}`。ここで、名前空間は `partitionKeyFromQuery` または `partitionKeyFromLambda`、またはその両方です。インライン解析を使用してソースデータのパーティショニングキーを作成している場合は、次の形式で指定された式で構成される S3 バケットプレフィクス値を指定する必要があります: `"partitionKeyFromQuery:keyID"`。AWS Lambda 関数を使用してソースデータのパーティショニングキーを作成している場合は、次の形式で指定された式で構成される S3 バケットプレフィックス値を指定する必要があります。`"partitionKeyFromLambda:keyID"`。詳細については、「Creating an Amazon Firehose stream」の「Choose Amazon S3 for Your Destination」を参照してください。

### セマンティックルール

`Prefix` および `ErrorOutputPrefix` 式には、以下のルールが制限されます。

- `timestamp` 名前空間では、一重引用符で囲まれていないすべての文字が評価されます。言い換えると、値フィールドで一重引用符によりエスケープされたすべての文字列が文字どおりに解釈されます。
- タイムスタンプ名前空間式が含まれないプレフィックスを指定した場合、Firehose は式 `!{timestamp:yyyy/MM/dd/HH/}` を `Prefix` フィールドの値に追加します。
- シーケンス `!{` は、`!{namespace:value}` 式にのみ現れます。
- `Prefix` に式が含まれていない場合、`ErrorOutputPrefix` は null にのみすることができます。この場合、`Prefix` は `<specified-prefix>yyyy/MM/DDD/HH/` と評価さ

れ、ErrorOutputPrefix は <specified-prefix><error-output-type>yyyy/MM/DDD/HH/ と評価されます。DDD は日を表します。

- ErrorOutputPrefix の式を指定した場合、!{firehose:error-output-type} のインスタンスを少なくとも 1 つ含める必要があります。
- Prefix に !{firehose:error-output-type} を含めることはできません。
- Prefix と ErrorOutputPrefix のどちらも、評価後に 512 文字を超えることはできません。
- 送信先が Amazon Redshift の場合、Prefix に式を含めることはできず、ErrorOutputPrefix は null にする必要があります。
- 宛先が Amazon OpenSearch Service または Splunk であり、ErrorOutputPrefix が指定されていない場合、Firehose は失敗したレコードに Prefix フィールドを使用します。
- 送信先が Amazon S3 の場合、Amazon S3 送信先設定の Prefix および ErrorOutputPrefix は、それぞれ成功したレコードと失敗したレコードに使用されます。 AWS CLI または API を使用する場合は、ExtendedS3DestinationConfiguration を使用して Amazon S3 バックアップ設定をそれ自身の Prefix と ErrorOutputPrefix を用いて指定できます。
- を使用して送信先を Amazon S3 に設定する AWS マネジメントコンソール と、Firehose は送信先設定ErrorOutputPrefixで Prefixと をそれぞれ成功レコードと失敗レコードに使用します。式を使用してプレフィックスを指定する場合は、!{firehose:error-output-type} を含むエラープレフィックスを指定する必要があります。
- ExtendedS3DestinationConfiguration を AWS CLI、 API、または で CloudFormation 使用する場合S3BackupConfiguration、Firehose はデフォルトの を提供しませんErrorOutputPrefix。
- ErrorOutputPrefix 式を作成するときに、partitionKeyFromLambda および partitionKeyFromQuery 名前空間を使用することはできません。

## プレフィックスの例

### **Prefix** と **ErrorOutputPrefix** の例

| Input | 評価されるプレフィックス (2018 年 8 月 27 日の午前 10:30 UTC) |
|---|---|
| Prefix: 未指定 | Prefix: 2018/08/27/10<br><br>ErrorOutputPrefix ：myFirehoseFailures/processing-failed/ |

| Input | 評価されるプレフィックス (2018 年 8 月 27 日の午前 10:30 UTC) |
| --- | --- |
| ErrorOutputPrefix : myFirehoseFailures/!{firehose:error-output-type}/ | |
| Prefix: !{timestamp:yyyy/MM/dd}<br><br>ErrorOutputPrefix : 未指定 | 無効な入力: Prefix に式が含まれている場合、ErrorOutputPrefix を null にすることはできません。 |
| Prefix: myFirehose/DeliveredYear=!{timestamp:yyyy}/anyMonth/rand=!{firehose:random-string}<br><br>ErrorOutputPrefix : myFirehoseFailures/!{firehose:error-output-type}/!{timestamp:yyyy}/anyMonth/!{timestamp:dd} | Prefix: myFirehose/DeliveredYear=2018/anyMonth/rand=5abf82daaa5<br><br>ErrorOutputPrefix : myFirehoseFailures/processing-failed/2018/anyMonth/10 |
| Prefix: myPrefix/year=!{timestamp:yyyy}/month=!{timestamp:MM}/day=!{timestamp:dd}/hour=!{timestamp:HH}/<br><br>ErrorOutputPrefix : myErrorPrefix/year=!{timestamp:yyyy}/month=!{timestamp:MM}/day=!{timestamp:dd}/hour=!{timestamp:HH}/!{firehose:error-output-type} | Prefix: myPrefix/year=2018/month=07/day=06/hour=23/<br><br>ErrorOutputPrefix : myErrorPrefix/year=2018/month=07/day=06/hour=23/processing-failed |
| Prefix: myFirehosePrefix/<br><br>ErrorOutputPrefix : 未指定 | Prefix: myFirehosePrefix/2018/08/27/<br><br>ErrorOutputPrefix : myFirehosePrefix/processing-failed/2018/08/27/ |

# OpenSearch Service のインデックスローテーションを設定する

OpenSearch Service が送信先の場合、[NoRotation]、[OneHour]、[OneDay]、[OneWeek]、[OneMonth] の 5 つのオプションの 1 つから、時間ベースのインデックスのローテーションオプションを指定できます。

選択するローテーションオプションに基づき、Amazon Data Firehose によって、UTC の到着タイプスタンプが指定されたインデックス名に追加されます。追加されたタイムスタンプは、それに応じてローテーションされます。以下の例は、各インデックスのローテーションオプションに対する OpenSearch Service で作成されるインデックス名を示しています。指定されたインデックス名は myindex で、到着タイムスタンプは `2016-02-25T13:00:00Z` です。

| RotationPeriod | IndexName |
|---|---|
| `NoRotation` | `myindex` |
| `OneHour` | `myindex-2016-02-25-13` |
| `OneDay` | `myindex-2016-02-25` |
| `OneWeek` | `myindex-2016-w08` |
| `OneMonth` | `myindex-2016-02` |

> **Note**
>
> `OneWeek` オプションでは、Data Firehose は <YEAR>-w<WEEK NUMBER> の形式を使用してインデックスを自動作成します (たとえば、`2020-w33`)。ここで、週数は UTC 時間を使用し、次の米国の表記規則に従って計算されます。
>
> - 週は日曜日から始まります
> - その年の最初の週は、今年の土曜日を含む最初の週です。

# データ配信を一時停止および再開する

Firehose ストリームを設定すると、ストリームソースで使用できるデータが継続的に宛先に配信されます。ストリームの送信先が一時的に使用できない状況 (計画的メンテナンスなど) になった場合は、データ配信を一時的に停止し、送信先が再び使用できるようになったら再開してください。

> **Important**
>
> 以下で説明するアプローチを使用してストリームを一時停止および再開すると、ストリームを再開した後、Amazon S3 のエラーバケットに配信されるレコードはほとんどなく、残りのストリームは引き続き宛先に配信されているのがわかります。これはこのアプローチの既知の制限であり、以前、複数回再試行しても宛先に配信できなかった少数のレコードが失敗として追跡されるために発生します。

## Firehose ストリームを一時停止する

Firehose でストリーム配信を一時停止するには、まず、Firehose が配信に失敗したデータを S3 のバックアップ場所に書き込むため許可を削除します。例えば、OpenSearch の宛先への Firehose ストリームを一時停止する場合は、許可を更新することでそれを実行できます。詳細については、「Firehose に公開 OpenSearch Service の宛先へのアクセスを付与する」を参照してください。

アクション `s3:PutObject` のアクセス許可 `"Effect": "Allow"` を削除し、アクション `s3:PutObject` のアクセス許可 `Effect": "Deny"` を、配信に失敗した場合に使用する S3 バケットに適用するステートメントを明示的に追加します。次に、ストリームの宛先をオフにする (宛先の OpenSearch ドメインをオフにするなど) か、Firehose が宛先に書き込みする許可を削除します。他の宛先への許可を更新するには、「Controlling Access with Amazon Data Firehose」に記載されている宛先に関するセクションを参照してください。この 2 つのアクションを完了すると、Firehose はストリームの配信を停止します。Firehose の CloudWatch メトリクスを使用してこれをモニタリングできます。

> **Important**
>
> Firehose でストリーム配信を一時停止するときは、ストリームのソース (Kinesis Data Streams や Managed Service for Kafka など) で、ストリーム配信が再開されてデータが宛先に配信されるまでデータを保持するように設定されていることを確認する必要があります。

ソースが DirectPUT である場合、Firehose はデータを 24 時間保持します。データ保持期間内にストリームを再開してデータを配信しなければ、データが失われる可能性があります。

## Firehose ストリームを再開する

配信を再開するには、まず宛先を有効にし、Firehose にストリームを配信する許可があることを確認して、ストリームの宛先に対して以前行った変更を元に戻します。次に、失敗した配信をバックアップする S3 バケットに適用されたアクセス許可に対して、以前に行った変更を元に戻します。つまり、アクション `s3:PutObject` のアクセス許可 `"Effect": "Allow"` を適用し、配信に失敗した場合に使用する S3 バケットに対するアクション `s3:PutObject` のアクセス許可 `"Effect": "Deny"` を削除します。最後に、Firehose の CloudWatch メトリクスを使用してモニタリングし、ストリームが宛先に配信されていることを確認します。エラーを確認してトラブルシューティングするには、「Amazon CloudWatch Logs monitoring for Firehose」を参照してください。

# Amazon Data Firehose を使用して Apache Iceberg テーブルにデータを配信する

Apache Iceberg は、ビッグデータ分析を実行するための高性能オープンソーステーブル形式です。Apache Iceberg は、SQL テーブルの信頼性とシンプルさを Amazon S3 データレイクにもたらし、Spark、Flink、Trino、Hive、Impala などのオープンソース分析エンジンが同じデータを同時に操作することを可能にします。詳細については、「Apache Iceberg」と「考慮事項と制限事項」を参照してください。

Firehose を使用すると、Amazon S3 の Apache Iceberg テーブルにストリーミングデータを配信できます。Apache Iceberg テーブルは、Amazon S3 でセルフマネージド型にすることも、Amazon S3 Tables でホストすることもできます。セルフマネージド型 Iceberg テーブルでは、圧縮やスナップショットの有効期限など、すべてのテーブルの最適化を管理できます。Amazon S3 Tables は、大規模な分析ワークロード用に最適化されたストレージを提供し、クエリのパフォーマンスを継続的に向上させ、表形式データのストレージコストを削減する機能を備えています。Amazon S3 Tables の詳細については、「Amazon S3 Tables」を参照してください。

この機能を使用すると、単一のストリームのレコードを異なる Apache Iceberg テーブルにルーティングできます。これらのテーブル内のレコードに対して、挿入、更新、および削除オペレーションを自動的に適用できます。また、Amazon S3 の Apache Iceberg テーブルに対するきめ細かなデータアクセスコントロールもサポートしています AWS Lake Formation。アクセスコントロールを で一元的に指定 AWS Lake Formation し、Firehose のより詳細なテーブルレベルおよび列レベルのアクセス許可を提供できます。

## 考慮事項と制限事項

> **Note**
>
> Firehose は、中国リージョン、アジアパシフィック (台北 ) 、アジアパシフィック (マレーシア ) 、アジアパシフィック (ニュージーランド) AWS GovCloud (US) Regions、メキシコ (中部) AWS リージョンを除くすべての で Apache Iceberg Tables を送信先としてサポートしています。

Apache Iceberg テーブルに対する Firehose のサポートには、次の考慮事項と制限があります。

- スループット – Direct PUT をソースとして使用して Apache Iceberg テーブルにデータを配信する場合、ストリームあたりの最大スループットは、米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド) リージョンでは 5 MiB/秒、その他のすべての AWS リージョンでは 1 MiB/秒になります。データを更新や削除をせずに Iceberg テーブルに挿入し、ストリームのスループットを増やしたい場合は、Firehose の制限フォームを使用して、スループットの上限の引き上げをリクエストできます。

  データの挿入のみで、更新や削除を行わない場合、`AppendOnly` フラグを `True` に設定することもできます。`AppendOnly` フラグを `True` に設定すると、Firehose はスループットに合わせて自動的にスケーリングします。現在、このフラグは CreateDeliveryStream API オペレーションでのみ設定できます。

  データ取り込み量が多くなり、Firehose ストリームのスループットキャパシティを超えたために Direct PUT ストリームでスロットリングが発生した場合、Firehose はスロットリングが抑制されるまでストリームのスループットの上限を自動的に引き上げます。スループットの増加とスロットリングの状況によっては、Firehose がストリームのスループットを所望のレベルに引き上げるまでに時間がかかる場合があります。このため、失敗したデータ取り込みレコードの再試行を継続します。データ量が突発的に増加することが予想される場合、または新しいストリームでデフォルトのスループット制限よりも高いスループットが必要な場合は、スループットの上限の引き上げをリクエストしてください。

- S3 トランザクション/秒 (TPS) – S3 パフォーマンスを最適化するためには、Kinesis Data Streams または Amazon MSK をソースとして使用している場合、適切なパーティションキーを使用してソースレコードをパーティション化することをお勧めします。このようにして、同じ Iceberg テーブルにルーティングされるデータレコードは、シャードと呼ばれる 1 つまたは複数のソースパーティションにマッピングされます。可能であれば、異なるターゲット Iceberg テーブルに含まれるデータレコードを異なるパーティション/シャードに分散させます。そうすることで、ソーストピック/ストリームのすべてのパーティション/シャードで使用できるすべての集約スループットを使用できるようになります。

- [列] – 列名と値については、Firehose はマルチレベルでネストされた JSON の最初のレベルのノードのみを取得します。例えば、Firehose は、位置フィールドを含む最初のレベルで使用可能なノードを選択します。Firehose が正常に配信するには、ソースデータの列名とデータ型がターゲットテーブルの列名とデータ型と完全に一致している必要があります。この場合、Firehose は、データ型列が位置フィールドと一致するように、Iceberg テーブルで構築またはマッピングされていることを想定します。Firehose は 16 レベルのネストをサポートしています。ネストされた JSON の例を次に示します。

```
{
```

```
   "version":"2016-04-01",
   "deviceId":"<solution_unique_device_id>",
   "sensorId":"<device_sensor_id>",
   "timestamp":"2024-01-11T20:42:45.000Z",
   "value":"<actual_value>",
   "position":{
      "x":143.595901,
      "y":476.399628,
      "z":0.24234876
   }
}
```

列名またはデータ型が一致しない場合、Firehose はエラーをスローし、データを S3 エラーバケットに配信します。すべての列名とデータ型が Apache Iceberg テーブルで一致しても、ソースレコードに追加のフィールドが存在している場合、Firehose は新しいフィールドをスキップします。

- レコードごとに 1 つの JSON オブジェクト – 1 つの Firehose レコードに 1 つの JSON オブジェクトのみを送信できます。レコード内で複数の JSON オブジェクトを集約して送信すると、Firehose はエラーをスローし、データを S3 エラーバケットに配信します。KPL でレコードを集約し、Amazon Kinesis Data Streams をソースとして Firehose にデータを取り込むと、Firehose はレコードごとに 1 つの JSON オブジェクトを自動的に集約解除して使用します。
- 圧縮とストレージの最適化 – Firehose を使用して Iceberg テーブルに書き込むたびに、スナップショット、データファイル、および削除ファイルをコミットして生成します。データファイルの数が多いと、メタデータのオーバーヘッドが増加し、読み取りパフォーマンスに影響します。効率的なクエリパフォーマンスを実現するには、小さなデータファイルを定期的に取得し、それらをまとめて少数の大きなデータファイルに書き換えるソリューションを検討することをお勧めします。このプロセスは compaction と呼ばれます。 は、Apache Iceberg テーブルの自動圧縮 AWS Glue Data Catalog をサポートしています。詳細については、「AWS Glue ユーザーガイド」の「Compaction management」を参照してください。詳細については、「Automatic compaction of Apache Iceberg Tables」を参照してください。別の方法として、Athena Optimize コマンドを実行して、圧縮を手動で実行することも可能です。Optimize コマンドの詳細については、「Athena Optimize」を参照してください。

  データファイルの圧縮に加えて、スナップショットの有効期限や未参照ファイルの削除など、Apache Iceberg テーブルでテーブルメンテナンスを実行する VACUUM ステートメントを使用して、ストレージの消費を最適化することもできます。または、 AWS Glue Data Catalog を使用して、データファイルや孤立したファイルを自動的に削除し、不要になったスナップショットを期限切れにすることで、Apache Iceberg テーブルのマネージドテーブル最適化をサポートするこ

ともできます。詳細については、Apache Iceberg テーブルのストレージ最適化に関するこのブログ記事を参照してください。

- Apache Iceberg テーブルの送信先としての Amazon MSK Serverless ソースはサポートされていません。
- 更新オペレーションの場合、Firehose は削除ファイルを配置し、その後に挿入オペレーションを実行します。削除ファイルを配置すると、Amazon S3 PUT 料金が発生します。
- Firehose では、複数の Firehose ストリームを使用して同じ Apache Iceberg テーブルにデータを書き込むことは推奨されません。これは、Apache Iceberg がオプティミスティック同時実行制御 (OCC) に依存しているためです。複数の Firehose ストリームで 1 つの Iceberg テーブルへの同時書き込みを行おうとした場合、特定の時点でデータのコミットに成功するのは 1 つのストリームのみです。コミットに失敗した他のストリームはバックオフし、設定した再試行期間が終了するまでコミットオペレーションを再試行します。再試行期間が過ぎると、設定した Amazon S3 エラープレフィックスにデータと削除ファイルキー (Amazon S3 パス) が送信されます。
- Firehose がサポートする現在の Iceberg Library バージョンは 1.5.2 です。
- 暗号化されたデータを Amazon S3 Tables に配信するには、Firehose 設定ではなく、Amazon S3 Tables で AWS Key Management Service パラメータを設定する必要があります。暗号化されたデータを Amazon S3 Tables に配信するように Firehose で AWS Key Management Service パラメータを設定した場合、Firehose はこれらのパラメータを使用して暗号化することはできません。詳細については、「 AWS KMS キーによるサーバー側の暗号化の使用」を参照してください。
- Firehose ストリームは、Iceberg の GlueCatalog API を介して作成されたデータベースとテーブルへの配信のみをサポートします。Glue SDK を使用して作成されたデータベースやテーブルへの配信はサポートされていません。ハイフン (-) 文字は、Iceberg ライブラリのデータベース名とテーブル名ではサポートされていません。詳細については、Iceberg ライブラリでサポートされている Glue Database Regex と Glue Table Regex を参照してください。
- Firehose によって書き込まれたすべてのファイルは、レコードに存在するパーティションを使用して計算されます。これは、削除されたファイルにも適用されます。パーティション化されたテーブルに対するパーティション化されていない削除ファイルの書き込みなどのグローバル削除はサポートされていません。

# Apache Iceberg テーブルを宛先として使用するための前提条件

以下のオプションから選択し、必要な前提条件を完了します。

トピック

## Amazon S3 で Iceberg テーブルに配信するための前提条件

開始する前に、次の前提条件を完了します。

- Amazon S3 バケットを作成する – テーブルの作成中にメタデータファイルパスを追加するには、Amazon S3 バケットを作成する必要があります。詳細については、「Create an S3 bucket」を参照してください。
- 必要な許可を持つ IAM ロールを作成する – Firehose には、 AWS Glue テーブルにアクセスしてデータを Amazon S3 に書き込むための特定の許可を持つ IAM ロールが必要です。同じロールを使用して、Amazon S3 バケット AWS Glue へのアクセスを許可します。Iceberg テーブルと Firehose ストリームを作成するときに、この IAM ロールが必要になります。詳細については、「Firehose に Amazon S3 Tables へのアクセスを許可する」を参照してください。
- Apache Iceberg テーブルを作成する – 更新と削除のために Firehose ストリームで一意のキーを設定する場合、Firehose はテーブルと一意のキーがストリーム作成の一部として存在しているかどうかを検証します。このシナリオでは、Firehose ストリームを作成する前にテーブルを作成する必要があります。 AWS Glue を使用して Apache Iceberg テーブルを作成できます。詳細については、「Apache Iceberg テーブルの作成」を参照してください。Firehose ストリームで一意のキーを設定していない場合、Firehose ストリームを作成する前に Iceberg テーブルを作成する必要はありません。

> **Note**
>
> Firehose は、Apache Iceberg テーブルのために、次のテーブルバージョンと形式をサポートします。
>
> - テーブル形式バージョン – Firehose は V2 テーブル形式のみをサポートします。V1 形式でテーブルを作成しないでください。V1 形式でテーブルを作成するとエラーが発生し、代わりにデータが S3 エラーバケットに配信されます。
> - データストレージ形式 – Firehose は、Parquet 形式でデータを Apache Iceberg テーブルに書き込みます。
> - 行レベルのオペレーション – Firehose は、Apache Iceberg テーブルへのデータの書き込みにおいて Merge-on-Read (MOR) モードをサポートします。

## Amazon S3 Tables に配信するための前提条件

Amazon S3 テーブルバケットにデータを配信するには、次の前提条件を完了します。

- 「Getting started with Amazon S3 Tables」で説明されている、Amazon S3 バケット、名前空間、およびテーブルバケット内のテーブルの作成、およびその他の統合手順を実行します。「S3 tables catalog integration limitations」で指定されているように、S3 Tables カタログ統合によって課された制限があるため、列名は小文字にする必要があります。
- 必要な許可を持つ IAM ロールを作成する – Firehose には、 AWS AWS Glue テーブルにアクセスしてデータを Amazon S3 テーブルバケットに書き込むための特定の許可を持つ IAM ロールが必要です。Amazon S3 テーブルバケット内のテーブルに書き込むには、IAM ロールに必要なアクセス許可も提供する必要があります。Amazon S3 Tables カタログに必要なアクセス許可は、使用するアクセスコントロールモードによって異なります。
  - IAM アクセスコントロール – Firehose 配信ロールには、Amazon S3 Tables リソースに対する IAM アクセス許可が直接必要です。
  - Lake Formation アクセスコントロール – Firehose 配信ロールには、テーブルリソースへのアクセスを管理するためのアクセス AWS AWS Lake Formation 許可が必要です。 は、Data Catalog リソースのきめ細かなアクセスコントロールを可能にする独自のアクセス許可モデル AWS Lake Formation を使用します。

  Firehose ストリームを作成するときに、この IAM ロールを設定します。詳細については、「Grant Firehose access to Amazon S3 Tables」を参照してください。

段階的な統合については、ブログ「Build a data lake for streaming data with Amazon S3 Tables and Amazon Data Firehose」を参照してください。詳細については、AWS 「分析サービスでの Amazon S3 Tables の使用」も参照してください。

## Firehose ストリームを設定する

Apache Iceberg テーブルを送信先として Firehose ストリームを作成するには、次を設定する必要があります。

> **Note**
>
> S3 テーブルバケット内のテーブルに配信するための Firehose ストリームの設定は、Amazon S3 の Apache Iceberg テーブルと同じです。

## ソースと宛先を設定する

Apache Iceberg テーブルにデータを配信するには、ストリームのソースを選択します。

ストリームのソースを設定するには、「Configure source settings」を参照してください。

次に、宛先として [Apache Iceberg テーブル] を選択し、Firehose ストリーム名を指定します。

## データ変換を設定する

着信ストリーム内のレコードの追加や変更など、データに対してカスタム変換を実行するには、Firehose ストリームに Lambda 関数を追加できます。Firehose ストリームでの Lambda を使用したデータ変換の詳細については、「Amazon Data Firehose でソースデータを変換する」を参照してください。

Apache Iceberg テーブルでは、着信レコードを異なる送信先テーブルにルーティングする方法と、実行するオペレーションを指定する必要があります。必要なルーティング情報を Firehose に提供する方法の 1 つは、Lambda 関数を使用することです。

詳細については、「Route records to different Iceberg tables」を参照してください。

## データカタログを接続する

Apache Iceberg には、Apache Iceberg テーブルに書き込むデータカタログが必要です。Firehose は AWS Glue Data Catalog for Apache Iceberg Tables と統合されています。

Firehose ストリームと同じアカウント AWS Glue Data Catalog 、クロスアカウント、および Firehose ストリームと同じリージョン (デフォルト ) 、または別のリージョンで を使用できます。

Amazon S3 テーブルに配信している場合、およびコンソールを使用して Firehose ストリームを設定している場合は、Amazon S3 テーブルカタログに対応するカタログを選択します。CLI を使用して Firehose ストリームを設定している場合は、`CatalogConfiguration` 入力で `arn:aws:glue:<region>:<account-id>:catalog/s3tablescatalog/<s3 table bucket name>` という形式の `CatalogARN` を使用します。詳細については、「Setting up a Firehose stream to Amazon S3 tables」を参照してください。

> **Note**
>
> Firehose は Iceberg テーブルに対して、挿入、更新、削除の 3 種類のオペレーションをサポートしています。オペレーションが指定されていない場合、Firehose はデフォルトで挿入を実行し、各着信レコードを新しい行として追加し、重複を保持します。既存のレコードを

変更するには、プライマリキーを使用して既存の行を検索および変更する「更新」オペレーションを指定します。
例:

- デフォルト (挿入): 同一のカスタマーレコードが複数存在すると、重複行が作成されます。
- 指定された更新: 新しいカスタマーアドレスによって既存のレコードが更新されます。

## JQ 式を設定する

Apache Iceberg テーブルでは、着信レコードを異なる宛先テーブルにルーティングする方法と、挿入、更新、削除などの実行するオペレーションを指定する必要があります。これを実行するには、Firehose のために JQ 式を設定して、必要な情報を解析および取得します。詳細については、「???」を参照してください。

## 一意のキーを設定する

複数のテーブルを使用した更新と削除 – 一意のキーは、ソースレコード内の 1 つ以上のフィールドであり、Apache Iceberg テーブルの行を一意に識別します。複数のテーブルを含むシナリオのみを挿入する場合は、一意のキーを設定する必要はありません。特定のテーブルで更新と削除を実行する場合は、必要なテーブルのために一意のキーを設定する必要があります。テーブルの行が欠落している場合、更新によって行が自動的に挿入されることに留意してください。テーブルが 1 つしかない場合は、一意のキーを設定できます。更新オペレーションの場合、Firehose は削除ファイルを配置し、その後に挿入を実行します。

Firehose ストリーム作成の一環としてテーブルごとに一意のキーを設定するか、または create table もしくは alter table オペレーション中に Iceberg で identifier-field-ids をネイティブに設定できます。ストリームの作成中にテーブルごとに一意のキーを設定することはオプションです。ストリームの作成中にテーブルごとに一意のキーを設定しない場合、Firehose は必要なテーブルについて `identifier-field-ids` をチェックし、それらを一意のキーとして使用します。両方が設定されていない場合、更新オペレーションと削除オペレーションを使用したデータの配信は失敗します。

このセクションを設定するには、データを更新または削除するテーブルのデータベース名、テーブル名、一意のキーを入力します。設定内の各テーブルのためにのみエントリを持つことができます。追加のみのシナリオでは、このセクションを設定する必要はありません。オプションで、次の例に示すように、テーブルからのデータが配信に失敗した場合に備えてエラーバケットプレフィックスを指定することもできます。

```
[
```

```
  {
    "DestinationDatabaseName": "MySampleDatabase",
    "DestinationTableName": "MySampleTable",
    "UniqueKeys": [
      "COLUMN_PLACEHOLDER"
    ],
    "S3ErrorOutputPrefix": "OPTIONAL_PREFIX_PLACEHOLDER"
  }
]
```

指定された列名がテーブル全体で一意である場合、Firehose は一意のキーの設定をサポートします。ただし、完全修飾列名は一意のキーとしてサポートされていません。例えば、top._id という名前のキーは、列名 _id が最上位レベルにも存在する場合、一意のキーとは見なされません。_id がテーブル全体で一意である場合、テーブル構造内の場所に関係なく使用されます。つまり、最上位列であるか、ネストされた列であるかは関係ありません。次の例では、列名がスキーマ全体で一意であるため、_id はスキーマの有効な一意のキーです。

```
[
 "schema": {
  "type": "struct",
  "fields": [
    {
      "name": "top",
      "type": {
        "type": "struct",
        "fields": [
          { "name": "_id", "type": "string" },
          { "name": "name", "type": "string" }
        ]
      }
    },
    { "name": "user", "type": "string" }
  ]
}
]
```

次の例では、_id は最上位列とネストされた構造体の両方で使用されるため、スキーマの有効な一意のキーではありません。

```
[
"schema": {
  "type": "struct",
```

```
  "fields": [
    {
      "name": "top",
      "type": {
        "type": "struct",
        "fields": [
          { "name": "_id", "type": "string" },
          { "name": "name", "type": "string" }
        ]
      }
    },
    { "name": "_id", "type": "string" }
  ]
}
]
```

## 再試行期間を指定する

この設定を使用して、Amazon S3 の Apache Iceberg テーブルへの書き込みで障害が発生した場合に Firehose が再試行を試みる期間 (秒) を指定できます。再試行を実行するには、0 ~ 7,200 秒の任意の値を設定できます。デフォルトでは、Firehose は 300 秒間再試行します。

## 失敗した配信または処理に対応する

再試行期間の経過後にストリームの処理または配信に障害が発生した場合に備えて、レコードを S3 バックアップバケットに配信するように Firehose を設定する必要があります。これを実行するには、コンソールの [バックアップ設定] から、S3 バックアップバケットと S3 バックアップバケットのエラー出力プレフィックスを設定します。

## エラー処理

Firehose は、すべての配信エラーを CloudWatch Logs と Amazon S3 エラーバケットに送信します。

エラーのリスト:

| エラーメッセージ | 説明 |
| --- | --- |
| `Iceberg.NoSuchTable` | Firehose は存在しないテーブルに書き込んでいるか、テーブルが V2 形式ではありません |

| エラーメッセージ | 説明 |
| --- | --- |
| | 。Firehose は V1 形式のテーブルをサポートしていません。 |
| `Iceberg.InvalidTableName` | null または空のテーブル名が渡されたか、テーブルが V2 形式ではありません。Firehose は V1 形式のテーブルをサポートしていません。 |
| `S3.AccessDenied` | 前提条件のステップで作成された IAM ロールに必要なアクセス許可と信頼ポリシーがあることを確認します。 |
| `Glue.AccessDenied` | 前提条件のステップで作成された IAM ロールに必要なアクセス許可と信頼ポリシーがあることを確認します。 |

## バッファリングのヒントを設定する

Firehose は、着信ストリーミングデータを、特定のサイズ (バッファリングサイズ)、および特定の期間 (バッファリング間隔) にわたって、メモリにバッファリングしてから、Apache Iceberg テーブルに配信します。バッファリングサイズ (1 ~ 128 MB) およびバッファリング間隔 (0 ~ 900 秒) を選択できます。バッファリングのヒントが大きいほど、S3 書き込みの回数が少なくなり、データファイルがより大きいために圧縮コストが低減され、クエリランタイムはより高速になります。ただし、レイテンシーは高くなります。バッファリングのヒントの値が小さいと、レイテンシーが低くなります。

## 詳細設定の設定

Apache Iceberg テーブルに対して、サーバー側の暗号化、エラーログ記録、アクセス許可、およびタグを設定できます。詳細については、「詳細設定の設定」を参照してください。??? の一部として作成した IAM ロールを追加する必要があります。Firehose は、 AWS Glue テーブルにアクセスして Amazon S3 バケットに書き込むロールを引き受けます。

Firehose ストリームの作成が完了するまでに数分かかる場合があります。Firehose ストリームが正常に作成されたら、そのストリームへのデータの取り込みを開始し、Apache Iceberg テーブルにデータを表示できます。

# 着信レコードを単一の Iceberg テーブルにルーティングする

Firehose が単一の Iceberg テーブルにデータを挿入する場合に必要なのは、次の JSON の例に示すように、ストリーム設定で単一のデータベースとテーブルを設定することだけです。単一テーブルでは、Firehose にルーティング情報を提供するために JQ 式と Lambda 関数は必要ありません。これらのフィールドを JQ または Lambda とともに指定すると、Firehose は JQ または Lambda から入力を受け取ります。

```
[
  {
    "DestinationDatabaseName": "UserEvents",
    "DestinationTableName": "customer_id",
    "UniqueKeys": [
      "COLUMN_PLACEHOLDER"
    ],
    "S3ErrorOutputPrefix": "OPTIONAL_PREFIX_PLACEHOLDER"
  }
]
```

この例では、Firehose はすべての入力レコードを `UserEvents` データベース内の `customer_id` テーブルにルーティングします。単一テーブルで更新または削除オペレーションを実行する場合は、JSONQuery メソッドまたは Lambda メソッドのいずれかを使用して、各着信レコードのオペレーションを Firehose に提供する必要があります。

# 着信レコードを異なる Iceberg テーブルにルーティングする

Amazon Data Firehose は、ストリーム内の着信レコードを、レコードの内容に基づいて異なる Iceberg テーブルにルーティングできます。Amazon Data Firehose から配信された場合、レコードの順番は保持されません。次のサンプル入力レコードを考えてみましょう。

```
{
  "deviceId": "Device1234",
  "timestamp": "2024-11-28T11:30:00Z",
  "data": {
    "temperature": 21.5,
    "location": {
      "latitude": 37.3324,
      "longitude": -122.0311
    }
```

```
  },
  "powerlevel": 84,
  "status": "online"
}
```

```
{
  "deviceId": "Device4567",
  "timestamp": "2023-11-28T10:40:00Z",
  "data": {
    "pressure": 1012.4,
    "location": {
      "zipcode": 24567
    }
  },
  "powerlevel": 82,
  "status": "online"
}
```

この例では、**deviceId** フィールドには Device1234 と Device4567 という 2 つの可能な値があります。着信レコードの **deviceId** フィールドが Device1234 である場合、レコードを Device1234 という名前の Iceberg テーブルに書き込み、着信レコードの **deviceId** フィールドが Device4567 である場合、レコードを Device4567 という名前のテーブルに書き込みます。

Device1234 および Device4567 のレコードには、対応する Iceberg テーブルの異なる列にマッピングされるフィールドの異なるセットがある場合があることに留意してください。着信レコードには、JSON レコード内に **deviceId** をネストできる、ネストされた JSON 構造がある場合があります。この後のセクションでは、このようなシナリオで Firehose に適切なルーティング情報を提供することによって、レコードを異なるテーブルにルーティングする方法について説明します。

## JSONQuery 式を使用して Firehose にルーティング情報を提供する

Firehose にレコードルーティング情報を提供する極めてシンプルでコスト効率の高い方法は、JSONQuery 式を提供することです。このアプローチでは、Database Name、Table Name、および (オプションで) Operation という 3 つのパラメータで JSONQuery 式を指定します。Firehose は、指定した式を使用して着信ストリームレコードから情報を抽出し、レコードをルーティングします。

Database Name パラメータは、送信先データベースの名前を指定します。Table Name パラメータは送信先テーブルの名前を指定し、Operation は、着信ストリームレコードを新しいレコードとして送信先テーブルに挿入するか、または送信先テーブル内の既存のレコードを変

更もしくは削除するのかを示すオプションのパラメータです。[オペレーション] フィールドには、insert、update、または delete のいずれかの値が必要です。

これらの 3 つのパラメータのそれぞれについて、静的な値、または値が着信レコードから取得される動的な式を指定できます。例えば、すべての着信ストリームレコードを IoTevents という名前の単一のデータベースに配信する場合、[データベース名] は静的な値 “IoTevents” となります。着信レコードのフィールドから送信先テーブル名を取得する必要がある場合、[テーブル名] は、送信先テーブル名を取得する必要がある着信レコードのフィールドを指定する動的な式になります。

次の例では、[データベース名] には静的な値、[テーブル名] には動的な値、[オペレーション] には静的な値を使用します。[オペレーション] の指定はオプションであることに留意してください。オペレーションが指定されていない場合、Firehose はデフォルトで、着信レコードを新しいレコードとして宛先テーブルに挿入します。

```
Database Name : "IoTevents"
Table Name : .deviceId
Operation : "insert"
```

deviceId フィールドが JSON レコード内にネストされている場合は、ネストされたフィールド情報を .event.deviceId として [テーブル名] に指定します。

> **Note**
>
> - オペレーションを update または delete として指定する場合、Firehose ストリームを設定する際に送信先テーブルのために一意のキーを指定するか、または Iceberg で create table もしくは alter table オペレーションを実行する際に Iceberg で identifier-field-ids を設定する必要があります。これを指定しない場合、Firehose はエラーをスローし、S3 エラーバケットにデータを配信します。
> - Database Name および Table Nameの値は、宛先データベースおよびテーブル名と完全に一致する必要があります。一致しない場合、Firehose はエラーをスローし、データを S3 エラーバケットに配信します。

## AWS Lambda 関数を使用してルーティング情報を提供する

着信レコードを宛先テーブルにルーティングする方法を決定する複雑なルールが存在するシナリオがある場合があります。例えば、TableX という名前の宛先テーブルにルーティングされるべき

値 A、B、または F がフィールドに含まれているかどうかを定義したルールを使用したり、属性をさらに追加することで着信ストリームレコードを補強したいと考えたりする場合があります。例えば、レコードに値が 1 のフィールド device_id が含まれている場合、値が「modem」である別のフィールド device_type を追加し、送信先テーブル列にその追加のフィールドを書き込むことができます。このような場合、Firehose の AWS Lambda 関数を使用してソースストリームを変換し、Lambda 変換関数の出力の一部としてルーティング情報を提供できます。Firehose で AWS Lambda 関数を使用してソースストリームを変換する方法を理解するには、「Amazon Data Firehose でソースデータを変換する」を参照してください。

Firehose でソースストリームの変換のために Lambda を使用する場合、出力には、recordId、result、および data または KafkaRecordValue パラメータが含まれている必要があります。パラメータ recordId には入力ストリームレコードが含まれ、result は変換が成功したかどうかを示し、data には Lambda 関数の、Base64 でエンコードされ、変換された出力が含まれます。詳細については、「???」を参照してください。

```
{
  "recordId": "49655962066601463032522589543535113056108699331451682818000000",
  "result": "Ok",
  "data": "1IiwiI6ICJmYWxsIiwgImdgU21IiwiI6ICJmYWxsIiwg==tcHV0ZXIgU2NpZW5jZSIsICJzZW1"
}
```

Lambda 関数の一部としてストリームレコードを宛先テーブルにルーティングする方法に関するルーティング情報を Firehose に対して指定するには、Lambda 関数の出力には、metadata についての追加セクションが含まれている必要があります。次の例は、Kinesis Data Streams をデータソースとして使用して、データベース IoTevents の Device1234 という名前の新しいレコードとしてレコードを挿入する必要があることを Firehose に指示する Firehose ストリームについての Lambda 出力にメタデータセクションを追加する方法を示しています。

```
 {
"recordId": "49655962066601463032522589543535113056108699331451682818000000",
    "result": "Ok",
    "data":
 "1IiwiI6ICJmYWxsIiwgImdgU21IiwiI6ICJmYWxsIiwg==tcHV0ZXIgU2NpZW5jZSIsICJzZW1",

    "metadata":{
"otfMetadata":{
            "destinationTableName":"Device1234",
            "destinationDatabaseName":"IoTevents",
            "operation":"insert"
        }
```

```
    }
  }
```

同様に、次の例は、Amazon Managed Streaming for Apache Kafka をデータソースとして使用する Firehose に対して、Lambda の出力にメタデータセクションを追加する方法を示しています。これにより、Firehose に対して、データベース `IoTevents` 内の `Device1234` という名前のテーブルに新しいレコードとして挿入するよう指示できます。

```
{
"recordId": "49655962066601463032522589543535113056108699331451682818000000",
    "result": "Ok",
    "kafkaRecordValue":
 "1IiwiI6ICJmYWxsIiwgImdgU21IiwiI6ICJmYWxsIiwg==tcHV0ZXIgU2NpZW5jZSIsICJzZW1",

    "metadata":{
"otfMetadata":{
            "destinationTableName":"Device1234",
            "destinationDatabaseName":"IoTevents",
            "operation":"insert"
        }
    }
  }
```

この例では、次のとおりです。

- `destinationDatabaseName` はターゲットデータベースの名前を参照し、必須フィールドです。
- `destinationTableName` はターゲットテーブルの名前を参照し、必須フィールドです。
- `operation` は、`insert`、`update`、および `delete` の可能な値を持つオプションのフィールドです。値を指定しない場合、デフォルトのオペレーションは `insert` になります。

> **Note**
>
> - オペレーションを `update` または `delete` として指定する場合、Firehose ストリームを設定する際に送信先テーブルのために一意のキーを指定するか、または Iceberg で create table もしくは alter table オペレーションを実行する際に Iceberg で identifier-field-ids を設定する必要があります。これを指定しない場合、Firehose はエラーをスローし、S3 エラーバケットにデータを配信します。

- `Database Name` および `Table Name`の値は、宛先データベースおよびテーブル名と完全に一致する必要があります。一致しない場合、Firehose はエラーをスローし、データを S3 エラーバケットに配信します。
- Firehose ストリームに Lambda 変換関数と JSONQuery 式の両方がある場合、Firehose はまず Lambda 出力のメタデータフィールドをチェックして、レコードを適切な送信先テーブルにルーティングする方法を決定し、次に JSONQuery 式の出力を参照して欠落しているフィールドの有無を確認します。

  Lambda または JSONQuery 式が必要なルーティング情報を提供しない場合、Firehose はこれを単一テーブルシナリオとみなし、一意のキー設定で単一テーブルの情報を検索します。

  詳細については、「Route incoming records to a single Iceberg table」を参照してください。Firehose がルーティング情報を決定せず、レコードを指定された送信先テーブルに一致させることができない場合、指定された S3 エラーバケットにデータが配信されます。

## サンプル Lambda 関数

この Lambda 関数は、着信ストリームレコードを解析し、特定のテーブルにデータを書き込む方法を指定する必須フィールドを追加するサンプル Python コードです。このサンプルコードを使用して、ルーティング情報のメタデータセクションを追加できます。

```
import json
import base64


def lambda_handler(firehose_records_input, context):
    print("Received records for processing from DeliveryStream: " +
 firehose_records_input['deliveryStreamArn'])

    firehose_records_output = {}
    firehose_records_output['records'] = []


    for firehose_record_input in firehose_records_input['records']:

        # Get payload from Lambda input, it could be different with different sources
        if 'kafkaRecordValue' in firehose_record_input:
```

```
            payload_bytes =
base64.b64decode(firehose_record_input['kafkaRecordValue']).decode('utf-8')
        else
            payload_bytes =
base64.b64decode(firehose_record_input['data']).decode('utf-8')

        # perform data processing on customer payload bytes here

        # Create output with proper record ID, output data (may be different with
different sources), result, and metadata
        firehose_record_output = {}

        if 'kafkaRecordValue' in firehose_record_input:
            firehose_record_output['kafkaRecordValue'] =
base64.b64encode(payload_bytes.encode('utf-8'))
        else
            firehose_record_output['data'] =
base64.b64encode(payload_bytes.encode('utf-8'))

        firehose_record_output['recordId'] = firehose_record_input['recordId']
        firehose_record_output['result'] =  'Ok'
        firehose_record_output['metadata'] = {
            'otfMetadata': {
                'destinationDatabaseName': 'your_destination_database',
                'destinationTableName': 'your_destination_table',
                'operation': 'insert'
                }
            }
        firehose_records_output['records'].append(firehose_record_output)
    return firehose_records_output
```

# メトリクスをモニタリングする

Apache Iceberg テーブルへのデータ配信のために、Firehose はストリームレベルで次の CloudWatch メトリクスを出力します。

| メトリクス | 説明 |
| --- | --- |
| DeliveryToIceberg.Bytes | 指定された期間に Apache Iceberg テーブルに配信されたバイト数。 |

| メトリクス | 説明 |
| --- | --- |
| | 単位: バイト |
| DeliveryToIceberg.IncomingRowCount | Firehose が Apache Iceberg テーブルへの配信を試みるレコードの数。<br><br>単位: カウント |
| DeliveryToIceberg.SuccessfulRowCount | Apache Iceberg テーブルに配信された、成功した行の数。<br><br>単位: カウント |
| DeliveryToIceberg.FailedRowCount | S3 バックアップバケットに配信された、失敗した行の数。<br><br>単位: カウント |
| DeliveryToIceberg.DataFreshness | Firehose の最も古いレコードの経過期間 (Firehose に入ってから現在まで)。経過期間がこれより長いレコードはすべて Apache Iceberg テーブルに配信されています。<br><br>単位: 秒 |
| DeliveryToIceberg.Success | Apache Iceberg テーブルへの正常なコミットの合計。 |
| JQProcessing.Duration | JQ 式の実行に要した時間。<br><br>単位: ミリ秒 |

# サポートされているデータ型を理解する

Firehose は、Apache Iceberg がサポートするすべてのプリミティブおよび複雑なデータ型をサポートします。詳細については、「Schemas and Data Types」を参照してください。バイナリデータを文字列として送信する場合、Firehose でサポートされているエンコーディングタイプ (Basic Base64、MIME Base64、url-and-filename-safe Base64、および Hex) を使用する必要があります。Timestamp データ型の場合、常にマイクロ秒で送信する必要があります。

# データ型の例

次のセクションでは、さまざまなデータ型の例を示します。

### MapType


```
{
    "destination_column_0":
 {"WP5o0JOkuIQcDPcsvpJJygF1xzaOSq0wUlgTwuIeCEzgVneGxA":"PO3ReF3auyDqbfonx9Cd8NTmcQnqnw7JuZOCWwI
    "destination_column_1": "{\"{\\\"destination_nested_column_0\\\": \
\\"18:56:14.974\\\", \\\"destination_nested_column_1\\\": 241.86246}\":
\"M07kAvYdHvBh61F7RzfxtEd39YQI33LnM2NbGS67DOFFsRUyUUujKT5VnK7Wtfz1mHNeIix6FAY9cYpwTdedgr9XnFwG0
\",\"{\\\"destination_nested_column_0\\\": \\\"18:56:14.974\
\\", \\\"destination_nested_column_1\\\": 562.56384}\":
\"9GlxhDCt95LxBo51HybBZihqOqf6EU8jrDu7NMpxtGB2dY6q6kXpvxIrFuMdqHCJKIZIcDikwggLniUm8kgE4d
\",\"{\\\"destination_nested_column_0\\\": \\\"18:56:14.974\
\\", \\\"destination_nested_column_1\\\": 496.03268}\":
\"keTJZYLNvLRB50DMKzEI6M0AM4mueyNnA1m2YVnYdDwyxUpPqkb72Q6LiX0B9s8gCjZ6trW6C1PFk9KNBIpxYsj5Tc5Xs
\",\"{\\\"destination_nested_column_0\\\": \\\"18:56:14.974\\\", \\
\"destination_nested_column_1\\\": 559.0878}\":
\"mG0ZET84BUF28E312UCIWgmypyQFSUODH9NAMAnF3LJEutbooZWcBt97PP5AHaopNvC8pQZ4mGXB9hmVmjUNmuj5Qanyx
\",\"{\\\"destination_nested_column_0\\\": \\\"18:56:14.974\
\\", \\\"destination_nested_column_1\\\": 106.845245}\":
\"aidoVYrzu8gcLRkVVUyTKCN9gqTUFYi8uJQsrXEFEYl1f9ool7JhAtg9QKG5BBu67Ngb95ENsNKQyCHNImSu5x4hMnmHU
\"}"
}
```


### DecimalType


```
{
    "destination_column_0": 9455262425851.1342772,
    "destination_column_1": "9455262425851.1342772",
    "destination_column_2": 9455262425852
}
```


### BinaryType (base64-default、base64-mime、base64-url-safe、hex)


```
{
    "destination_column_0": "AsYhnHD\/Ra54hITl1daNV9glOjtWPEfopH
+PjgUKHYB6K7UcYi4K19b80wD4J\/93x5tyh+0y
+k5cMljVRlmfIkIuLxl9ERBiPPLhf4+yoJ2k70VavPnYWmNLs1hLDHlfeEMIfVhrqOGzJMoA
+CBAWXfIuiG420JSQP5iAx5xFG\/
mOfkM5zYothje8OGXltdthcCL6WYBiP0SlwXcE0uMeRfwclAc9fTOBz6RzdJlHhUDjoAXg
```

```
+4cvly27F82XpuGMNwpUj98AOrgbh2MoU9yvsM9ZrjD0eGVgOZP8Ky7Za4oE\/oK8j
+qABF6XV712iA6pVtTNJFvX6Ey3ssNYvno+LYF5ZsySs2rB5AbVM73RfOPqdS\/c\/
r3MEqoEqt+nPx6eGam4WSA+0swztt7aLdrlX6yK7xJeIJ0rTlIDBo0ZUaw011ykY
\/8Bvy+4byoPlmr4Z5yhN1z3ZTOkx7eDR6xMv+vDVsDBtItVazDwHgDy41r
\/hQNeNedPKrozc8TY9k7wZre\/6V2lCa3BmT8Uu9b9ydjR9z+fCSdG
+VRv35nz5kdqdKy8YIrynYs4eOcjh8jH3UwVYrYQcnWkBAfF7Xk9CoPVnL3ciHZtyiZOaTGIj9rO0xX\/
W5dGe9\/4YChs6LbD584kxLTxvHgSl4vadaTGNKci3SvNmZNsz8ducxtNXF\/Tv2DUub465hzgpaLPur3+MB
+kfdN2YXUfqb
+xJAgxThWfUe151nrH0EPow9lgSlp21rUBGznJAvPRl1ExGIAuc7JYAoUrJUkx5Hfl6PekPDhqt7+yJwCB8qxhTTryxo
+bjtai4ndRCGcuCaxT8KkOcXsS37urd3YGSDMinZdMNVc646s25415qK6nBRlqqAY8+EYmcUIVB9XcNdke4zoUfhVQoruwi
\/kFafoulo5DEoMOyaH1N2HCSxG5tZXNQocSZPaY8efZYMCpmDXsPAzkmgSkYRDSu\/r3wUqROa2tGK5\/
pQY24v+Jq0U\/jQ99GShlU283nZ85ot2ocbtMAgD\/WsrSEh6lNt9RaI3HfA7\/HcH\/
fgr9jsTtxDgZhabTBwwDwX0zjWGx1bCuTLKBN7byxg9ZvAVgqwPS4HERLer5T5UkKf74zn9Eq3HYH1Q5JpyDUx
+im7mte1sprf1+A24kksVU\/MD9aP9N8\/QDsQ13gkhOn5KwFMz3BC2Vw5gL
+gGNHFKDRL6wGIfhuYcx9LucolZ1yNy9Gbb3ioWSSufyFpyXqtndDLPI5QS1SJpJm2KDyqcH1SmRLIhd9MNRUC73EAEm
+NO5wxPzBRSjhCHZpf8SrYITWJl7K3XzGOfPFh2NgES3jMP9cvSXO6yyICcep2HBYGbFflni89+Rw==",
    "destination_column_1": "AsYhnHD\/Ra54hITl1daNV9glOjtWPEfopH
+PjgUKHYB6K7UcYi4K19b80wD4J\/93x5tyh+0y+k5c\r
\nMljVRlmfIkIuLxl9ERBiPPLhf4+yoJ2k70VavPnYWmNLs1hLDHlfeEMIfVhrqOGzJMoA+CBAWXfI\r
\nuiG420JSQP5iAx5xFG\/mOfkM5zYothje8OGXltdthcCL6WYBiP0SlwXcE0uMeRfwclAc9fTOBz6R\r
\nzdJlHhUDjoAXg+4cvly27F82XpuGMNwpUj98AOrgbh2MoU9yvsM9ZrjD0eGVgOZP8Ky7Za4oE\/oK\r
\n8j+qABF6XV712iA6pVtTNJFvX6Ey3ssNYvno+LYF5ZsySs2rB5AbVM73RfOPqdS\/c\/r3MEqoEqt+
\r\nnPx6eGam4WSA+0swztt7aLdrlX6yK7xJeIJ0rTlIDBo0ZUaw011ykY\/8Bvy+4byoPlmr4Z5yhN1z
\r\n3ZTOkx7eDR6xMv+vDVsDBtItVazDwHgDy41r\/hQNeNedPKrozc8TY9k7wZre\/6V2lCa3BmT8Uu9b
\r\n9ydjR9z+fCSdG+VRv35nz5kdqdKy8YIrynYs4eOcjh8jH3UwVYrYQcnWkBAfF7Xk9CoPVnL3ciHZ
\r\ntyiZOaTGIj9rO0xX\/W5dGe9\/4YChs6LbD584kxLTxvHgSl4vadaTGNKci3SvNmZNsz8ducxtNXF
\/\r\nTv2DUub465hzgpaLPur3+MB+kfdN2YXUfqb+xJAgxThWfUe151nrH0EPow9lgSlp21rUBGznJAvP
\r\nRl1ExGIAuc7JYAoUrJUkx5Hfl6PekPDhqt7+yJwCB8qxhTTryxo+bjtai4ndRCGcuCaxT8KkOcXs\r
\nS37urd3YGSDMinZdMNVc646s25415qK6nBRlqqAY8+EYmcUIVB9XcNdke4zoUfhVQoruwidzDU\/k\r
\nFafoulo5DEoMOyaH1N2HCSxG5tZXNQocSZPaY8efZYMCpmDXsPAzkmgSkYRDSu\/r3wUqROa2tGK5\r
\n\/pQY24v+Jq0U\/jQ99GShlU283nZ85ot2ocbtMAgD\/WsrSEh6lNt9RaI3HfA7\/HcH\/fgr9jsTtxDg
\r\nZhabTBwwDwX0zjWGx1bCuTLKBN7byxg9ZvAVgqwPS4HERLer5T5UkKf74zn9Eq3HYH1Q5JpyDUx+\r
\nim7mte1sprf1+A24kksVU\/MD9aP9N8\/QDsQ13gkhOn5KwFMz3BC2Vw5gL+gGNHFKDRL6wGIfhuYc
\r\nx9LucolZ1yNy9Gbb3ioWSSufyFpyXqtndDLPI5QS1SJpJm2KDyqcH1SmRLIhd9MNRUC73EAEm+NO\r
\n5wxPzBRSjhCHZpf8SrYITWJl7K3XzGOfPFh2NgES3jMP9cvSXO6yyICcep2HBYGbFflni89+Rw==",
    "destination_column_2": "AsYhnHD_Ra54hITl1daNV9glOjtWPEfopH-
PjgUKHYB6K7UcYi4K19b80wD4J_93x5tyh-0y-k5cMljVRlmfIkIuLxl9ERBiPPLhf4-
yoJ2k70VavPnYWmNLs1hLDHlfeEMIfVhrqOGzJMoA-
CBAWXfIuiG420JSQP5iAx5xFG_mOfkM5zYothje8OGXltdthcCL6WYBiP0SlwXcE0uMeRfwclAc9fTOBz6RzdJlHhUDjoAX
qABF6XV712iA6pVtTNJFvX6Ey3ssNYvno-LYF5ZsySs2rB5AbVM73RfOPqdS_c_r3MEqoEqt-
nPx6eGam4WSA-0swztt7aLdrlX6yK7xJeIJ0rTlIDBo0ZUaw011ykY_8Bvy-4byoPlmr4Z5yhN1z3ZTOkx7eDR6xMv-
vDVsDBtItVazDwHgDy41r_hQNeNedPKrozc8TY9k7wZre_6V2lCa3BmT8Uu9b9ydjR9z-fCSdG-
VRv35nz5kdqdKy8YIrynYs4eOcjh8jH3UwVYrYQcnWkBAfF7Xk9CoPVnL3ciHZtyiZOaTGIj9rO0xX_W5dGe9_4YChs6LbD
MB-kfdN2YXUfqb-
```

```
xJAgxThWfUe151nrH0EPow9lgSlp21rUBGznJAvPRl1ExGIAuc7JYAoUrJUkx5Hfl6PekPDhqt7-
yJwCB8qxhTTryxo-bjtai4ndRCGcuCaxT8KkOcXsS37urd3YGSDMinZdMNVc646s25415qK6nBRlqqAY8-
EYmcUIVB9XcNdke4zoUfhVQoruwidzDU_kFafoulo5DEoMOyaH1N2HCSxG5tZXNQocSZPaY8efZYMCpmDXsPAzkmgSkYRDS
Jq0U_jQ99GShlU283nZ85ot2ocbtMAgD_WsrSEh6lNt9RaI3HfA7_HcH_fgr9jsTtxDgZhabTBwwDwX0zjWGx1bCuTLKBN7
im7mte1sprf1-A24kksVU_MD9aP9N8_QDsQ13gkhOn5KwFMz3BC2Vw5gL-
gGNHFKDRL6wGIfhuYcx9LucolZ1yNy9Gbb3ioWSSufyFpyXqtndDLPI5QS1SJpJm2KDyqcH1SmRLIhd9MNRUC73EAEm-
NO5wxPzBRSjhCHZpf8SrYITWJl7K3XzGOfPFh2NgES3jMP9cvSXO6yyICcep2HBYGbFflni89-Rw==",
    "destination_column_3":
 "02c6219c70ff45ae788484e5d5d68d57d8253a3b563c47e8a47f8f8e050a1d807a2bb51c622e0ad7d6fcd300f827f
}
```

TimeType (マイクロ秒のエポック、LocalTime Java オブジェクト)

```
{
    "destination_column_0": 68175096000,
    "destination_column_1": "18:56:15.096"
}
```

TimestampType.withZone (マイクロ秒のエポック、OffsetDateTime Java オブジェクト、LocalDateTime Java オブジェクト)

```
{
    "destination_column_0": 1725476175099000,
    "destination_column_1": "2024-09-04T18:56:15.099Z",
    "destination_column_2": "2024-09-04T18:56:15.099"
}
```

DoubleType

```
{
    "destination_column_0": 9.18477568715142,
    "destination_column_1": "9.18477568715142"
}
```

BooleanType

```
{
    "destination_column_0": true,
    "destination_column_1": "false",
    "destination_column_2": 1,
    "destination_column_3": 0
```

```
}
```

FloatType

```
{
    "destination_column_0": 0.6242226,
    "destination_column_1": "0.6242226"
}
```

IntegerType

```
{
    "destination_column_0": 7,
    "destination_column_1": "7"
}
```

TimestampType.withoutZone (マイクロ秒のエポック、LocalDateTime Java オブジェクト、OffsetDateTime Java オブジェクト、ZonedDateTime Java オブジェクト)

```
{
    "destination_column_0": 1725476175114000,
    "destination_column_1": "2024-09-04T18:56:15.114",
    "destination_column_2": "2024-09-04T18:56:15.114Z",
    "destination_column_3": "2024-09-04T18:56:15.114-07:00"
}
```

DateType

```
{
    "destination_column_0": 19970,
    "destination_column_1": "2024-09-04"
}
```

LongType

```
{
    "destination_column_0": 8,
    "destination_column_1": "8"
}
```

UUIDType (UUID Java Object)

```
{
    "destination_column_0": "21c5521c-a6d4-48d4-b2c8-7f6d842f72c3"
}
```

ListType

```
{
    "destination_column_0":
 ["s1FSrgb0lGDxfn2iYT0EtlP47aHSjwmLZgrdr1JqRs0dmbeCcQoaLr4Xhi2KIVvmus9ppFdpWIcOHnJ0omhAPhXH0yns
    "destination_column_1": "[{\"destination_nested_column_0\":\"bb00f8e6-
db82-4241-a5c5-0d9c0d2f71a4\",\"destination_nested_column_1\":907.35345},
{\"destination_nested_column_0\":\"2c77b702-d405-4fe1-beee-fb541d7ab833\",
\"destination_nested_column_1\":544.0026},{\"destination_nested_column_0\":
\"68389200-d6b1-413d-bcd9-fdb931708395\",\"destination_nested_column_1\":153.683},
{\"destination_nested_column_0\":\"bc31cbaa-39cd-4e2f-b357-9ea9ce75532b\",
\"destination_nested_column_1\":977.5165},{\"destination_nested_column_0\":
\"b7d627f9-0d5b-41b7-903a-525488259fba\",\"destination_nested_column_1\":434.17215},
{\"destination_nested_column_0\":\"06b6ec1e-1952-4582-b285-46aaf40064b8\",
\"destination_nested_column_1\":580.33124},{\"destination_nested_column_0\":
\"f04b3bbf-61ad-4c5c-8740-6f666f57c431\",\"destination_nested_column_1\":550.75793}]"
}
```

# リソース

詳細については、以下のリソースを参照してください。

- Stream real-time data into Apache Iceberg tables in Amazon S3 using Amazon Data Firehose
- Apache Iceberg と Amazon Data Firehose による AWS WAF ログ分析の効率化
- Build a data lake for streaming data with Amazon S3 Tables and Amazon Data Firehose

# Firehose ストリームにタグ付けする

Amazon Data Firehose で作成した Firehose ストリームに、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、ストリームに対して定義するキーと値のペアです。タグを使用することは、 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。

CreateDeliveryStream を呼び出して新しい Firehose ストリームを作成するときは、タグを指定できます。既存の Firehose ストリームでは、次の 3 つのオペレーションを使用してタグを追加、リスト、削除できます。

- TagDeliveryStream
- ListTagsForDeliveryStream
- UntagDeliveryStream

## タグの基本を理解する

Amazon Data Firehose API オペレーションを使用して、次のタスクを実行できます。

- Firehose ストリームにタグを追加します。
- Firehose ストリームのタグを一覧表示する
- Firehose ストリームからタグを削除します。

タグを使用すると、Firehose ストリームを分類できます。例えば、目的、所有者、環境などに基づいて Firehose ストリームを分類できます。タグごとにキーと値を定義するため、特定のニーズに合わせてカテゴリのカスタムセットを作成できます。例えば、所有者と、関連するアプリケーションに基づいて Firehose ストリームを追跡するのに役立つタグのセットを定義できます。

次に示すのは、いくつかのタグの例です：

- `Project: Project name`
- `Owner: Name`
- `Purpose: Load testing`
- `Application: Application name`
- `Environment: Production`

CreateDeliveryStream アクションでタグを指定すると、Amazon Data Firehose は firehose:TagDeliveryStream アクションに対して追加の認可を実行し、タグを作成するための許可がユーザーに付与されているかどうかを検証します。この許可を指定しない場合、IAM リソースのタグを使用して新しい Firehose ストリームを作成するリクエストは、次のような AccessDeniedException で失敗します。

```
AccessDeniedException
User: arn:aws:sts::x:assumed-role/x/x is not authorized to perform:
 firehose:TagDeliveryStream on resource: arn:aws:firehose:us-east-1:x:deliverystream/x
 with an explicit deny in an identity-based policy.
```

次の例は、ユーザーが Firehose ストリームを作成してタグを適用できるようにするポリシーを示しています。

# タグ付けでコストを追跡する

タグを使用して、 AWS コストを分類および追跡できます。Firehose ストリームを含むリソースに AWS 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。

# タグの制限を知る

Amazon Data Firehose のタグには次の制限があります。

基本制限

- リソース (ストリーム) あたりのタグの最大数は 50 です。
- タグキーと値は大文字と小文字が区別されます。
- 削除されたストリーム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持つタグを追加すると、新しいタグによって既存のキーと値のペアが上書きされます。

- aws: は AWSが使用するように予約されているため、このプレフィックスを含むタグキーで開始することはできません。 AWS ではユーザーの代わりにこのプレフィックスで始まるタグを作成しますが、ユーザーはこれらのタグを編集または削除することはできません。
- タグキーの長さは 1 ~ 128 文字 (Unicode) にする必要があります。
- タグキーは、次の文字で構成する必要があります。Unicode 文字、数字、空白、特殊文字 (_ . / = + - @)。

タグ値の制限

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

# Amazon Data Firehose のセキュリティ

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

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

- 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。セキュリティの有効性は、AWS コンプライアンスプログラムの一環として、サードパーティーの審査機関によって定期的にテストおよび検証されています。Data Firehose に適用されるコンプライアンスプログラムについては、「コンプライアンスプログラムによるAWS 対象範囲内のサービス」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、お客様のデータの機密性、組織の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

このドキュメントは、Data Firehose の使用時に責任共有モデルがどのように適用されるかを理解するために役立ちます。次のトピックでは、セキュリティおよびコンプライアンスの目的を達成するように Data Firehose を設定する方法を説明します。また、Data Firehose リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

トピック

# Amazon Data Firehose のデータ保護

Amazon Data Firehose は TLS プロトコルを使用して転送中のすべてのデータを暗号化します。さらに、処理中に中間ストレージに保存されたデータについては、Amazon Data Firehose は AWS Key Management Service を使用してデータを暗号化し、チェックサム検証を使用してデータの整合性を検証します。

機密データがある場合、Amazon Data Firehose を使用するときにサーバー側のデータ暗号化を有効にすることができます。これを行う方法は、データソースによって異なります。

> **Note**
>
> コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。

## Kinesis Data Streams を使用したサーバー側の暗号化

データプロデューサーからデータストリームにデータを送信すると、Kinesis Data Streams は保管中のデータを保存する前に AWS Key Management Service (AWS KMS) キーを使用してデータを暗号化します。Firehose ストリームがデータストリームからデータを読み取ると、Amazon Data Streams はまずデータを復号してから Amazon Data Firehose に送信します。Amazon Data Firehose は、指定したバッファリングのヒントに基づいてデータをメモリにバッファリングします。その後、暗号化されていないデータを保存することなく、送信先に配信します。

Kinesis Data Streams でサーバー側の暗号化を有効にする方法については、Amazon Kinesis Data Streams 開発者ガイドの「サーバー側の暗号化の使用」を参照してください。

## Direct PUT または他のデータソースを使用したサーバー側の暗号化

PutRecord または PutRecordBatch を使用して Firehose ストリームにデータを送信する場合、または AWS IoT、Amazon CloudWatch Logs、または CloudWatch Events を使用してデータを送信する場合は、StartDeliveryStreamEncryption オペレーションを使用してサーバー側の暗号化を有効にできます。

サーバー側の暗号化を停止するには、StopDeliveryStreamEncryption オペレーションを使用します。

Firehose ストリームを作成するときに SSE を有効にすることもできます。それを行うには、CreateDeliveryStream を呼び出すときに、DeliveryStreamEncryptionConfigurationInput を指定します。

を正常に使用するにはCUSTOMER_MANAGED_CMK、発信者の IAM ポリシーと KMS キーポリシーの両方が kms:GenerateDataKey および kms:Decryptオペレーションを許可する必要があります。Firehose は、CUSTOMER_MANAGED_CMK暗号化を使用して PutRecord または PutRecordBatch を呼び出すときに、これらのアクセス許可を検証します。さらに、CUSTOMER_MANAGED_CMK暗号化を使用して CreateDeliveryStream または StartDeliveryStreamEncryption を呼び出す場合は、 アクセスkms:CreateGrant許可が必要です。

CMK のタイプが CUSTOMER_MANAGED_CMK のときに、KMSNotFoundException、KMSInvalidStateException、KMSDisabledException、または KMSAccessDeniedException のエラーのために Amazon Data Firehose サービスがレコードを復号できない場合、問題が解決されるまでサービスは最大 24 時間 (保持期間) 待機します。保持期間を超えて問題が解決されない場合、サービスは、保持期間を超えて復号できなかったレコードをスキップし、データを破棄します。Amazon Data Firehose には、4 つの AWS KMS 例外を追跡するために使用できる次の 4 つの CloudWatch メトリクスが用意されています。

- KMSKeyAccessDenied
- KMSKeyDisabled
- KMSKeyInvalidState
- KMSKeyNotFound

これら 4 つのメトリクスの詳細については、「the section called “CloudWatch メトリクスによるモニタリング”」を参照してください。

> **Important**
>
> Firehose ストリームを暗号化するには、対称 CMK を使用します。Amazon Data Firehose は非対称 CMK をサポートしていません。対称 CMKs「対称 CMK と非対称 CMKs」を参照してください。 AWS Key Management Service

> **Note**
>
> カスタマーマネージドキー (CUSTOMER_MANAGED_CMK) を使用して Firehose ストリームのためにサーバー側の暗号化 (SSE) を有効にすると、Firehose サービスは、キーを使用するたびに暗号化コンテキストを設定します。この暗号化コンテキストは、 AWS アカウントが所有するキーが使用された出現を表すため、 AWS アカウントの AWS CloudTrail イベントログの一部として記録されます。この暗号化コンテキストは、Firehose サービスによって生成されたシステムです。アプリケーションは、Firehose サービスによって設定された暗号化コンテキストの形式やコンテンツについて、いかなる想定もすべきではありません。

# Amazon Data Firehose によるアクセスの制御

次のセクションでは、Amazon Data Firehose リソースのアクセスを制御する方法について説明します。例えば、Firehose ストリームにデータを送信できるようにアプリケーションにアクセスを付与する方法などについて説明します。また、Amazon Data Firehose に Amazon Simple Storage Service (Amazon S3) バケット、Amazon Redshift クラスター、または Amazon OpenSearch Service クラスターへのアクセスを付与する方法、および Datadog、Dynatrace、LogicMonitor、MongoDB、New Relic、Splunk、または Sumo Logic を宛先として使用する場合に必要な許可を付与する方法についても説明します。最後に、このトピックガイダンスでは、別の AWS アカウントに属する宛先にデータを配信できるように Amazon Data Firehose を設定する方法について説明します。これらすべての形式のアクセスを管理するテクノロジーは AWS Identity and Access Management (IAM) です。IAM の詳細については、「IAM とは?」を参照してください。

内容

- VPC 内の OpenSearch Service の宛先へのアクセスを Firehose に付与する
- Firehose に公開 OpenSearch Serverless の宛先へのアクセスを付与する
- VPC 内の OpenSearch Serverless の宛先へのアクセスを Firehose に付与する
- Splunk の宛先へのアクセスを Firehose に付与する
- VPC の Splunk へのアクセス
- Amazon Data Firehose を使用して VPC フローログを Splunk に取り込む
- Snowflake または HTTP エンドポイントへのアクセス
- Snowflake の宛先へのアクセスを Firehose に付与する
- VPC での Snowflake へのアクセス
- HTTP エンドポイントの宛先へのアクセスを Firehose に付与する
- Amazon MSK からのクロスアカウント配信
- Amazon S3 の宛先へのクロスアカウント間の配信
- OpenSearch Service の宛先へのクロスアカウント間の配信
- タグを使用したアクセスへのコントロール

## Firehose のリソースへのアクセスを付与する

アプリケーションに Firehose ストリームへのアクセスを付与するには、次の例のようなポリシーを使用します。`Action` セクションを変更するか、`"firehose:*"` を使用してすべてのオペレーションへのアクセス権を付与することで、アクセス権を付与する個別の API オペレーションを調整できます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "firehose:DeleteDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch",
                "firehose:UpdateDestination"
```

```
            ],
            "Resource": [
                "arn:aws:firehose:us-east-1:123456789012:deliverystream/delivery-
stream-name"
            ]
        }
    ]
}
```

## Firehose にプライベートの Amazon MSK Cluster クラスターへのアクセスを付与する

Firehose ストリームのソースがプライベートの Amazon MSK クラスターである場合は、次の例のようなポリシーを使用します。

Amazon MSK `CreateVpcConnection` API オペレーションを呼び出すための許可を Firehose サービスプリンシパルに付与するには、このようなポリシーをクラスターのリソースベースのポリシーに追加する必要があります。

## Firehose が IAM ロールを引き受けることを許可する

このセクションでは、ソースから宛先へのデータの取り込み、処理、配信を行うためのアクセスを Amazon Data Firehose に付与する許可とポリシーについて説明します。

> **Note**
>
> コンソールを使用して Firehose ストリームを作成し、新しいロールを作成するオプションを選択した場合、 は必要な信頼ポリシーをロールに AWS アタッチします。Amazon Data Firehose で既存の IAM ロールを使用する場合、または独自にロールを作成する場合は、Amazon Data Firehose がロールを引き受けることができるように、次の信頼ポリシーをそのロールにアタッチします。account-*id* を AWS アカウント ID に置き換えるには、ポリシーを編集します。ロールの信頼関係を変更する方法については、「ロールの修正」を参照してください。

Amazon Data Firehose は、Firehose ストリームがデータを処理および配信する際に必要になるすべての許可にこの IAM ロールを使用します。Amazon Data Firehose がロールを引き受けることができるように、そのロールに次の信頼ポリシーがアタッチされていることを確認します。

Amazon MSK を Firehose ストリームのソースに選ぶ場合は、指定した Amazon MSK クラスターからソースデータを取り込むための許可を Amazon Data Firehose に付与する別の IAM ロールを指定する必要があります。Amazon Data Firehose がロールを引き受けることができるように、そのロールに次の信頼ポリシーがアタッチされていることを確認します。

JSON

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

指定した Amazon MSK クラスターからソースデータを取り込むための許可を Amazon Data Firehose に付与するこのロールが、次の許可を付与していることを確認します。

## データ形式変換 AWS Glue のために Firehose に へのアクセスを許可する

Firehose ストリームがデータ形式の変換を実行する場合、Amazon Data Firehose は AWS Glueに保存されているテーブル定義を参照します。Amazon Data Firehose に必要なアクセス権を付与するには AWS Glue、次のステートメントをポリシーに追加します。テーブルの ARN を検索する方法については、「 Glue AWS リソース ARNs」を参照してください。

```
{
    "Sid": "",
    "Effect": "Allow",
    "Action": [
        "glue:GetTable",
        "glue:GetTableVersion",
```

```
            "glue:GetTableVersions"
        ],
        "Resource": [
            "arn:aws:glue:us-east-1:123456789012:catalog",
            "arn:aws:glue:us-east-1:123456789012:database/b",
            "arn:aws:glue:us-east-1:123456789012:table/b/easd"
        ]
},
{
      actions: ['glue:GetSchemaVersion'],
      grantee: options.role,
      resourceArns: ['*'],
}
```

スキーマレジストリからスキーマを取得するための推奨ポリシーには、リソースの制限はありません。詳細については、「 AWS Glue デベロッパーガイド」の「デシリアライザーの IAM の例」を参照してください。

## Firehose に Amazon S3 宛先へのアクセスを付与する

Amazon S3 送信先を使用している場合、Amazon Data Firehose はデータを S3 バケットに配信し、オプションでデータ暗号化に所有している AWS KMS キーを使用できます。エラーログ記録が有効になっている場合、Amazon Data Firehose はデータ配信エラーを CloudWatch ロググループとストリームにも送信します。Firehose ストリームを作成するときは、IAM ロールが必要です。Amazon Data Firehose はその IAM ロールを引き受け、指定されたバケット、キー、および CloudWatch ロググループとストリームへのアクセスを取得します。

次のアクセスポリシーを使用して、Amazon Data Firehose が S3 バケットおよび AWS KMS キーにアクセスできるようにします。S3 バケットを所有していない場合、Amazon S3 アクションのリストに `s3:PutObjectAcl` を追加します。これにより、Amazon Data Firehose で配信されるオブジェクトへのフルアクセスが、バケット所有者に付与されます。

JSON

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

```
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
name"
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:us-east-1:123456789012:key/key-id"
           ],
           "Condition": {
               "StringEquals": {
                   "kms:ViaService": "s3.us-east-1.amazonaws.com"
               },
               "StringLike": {
                   "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-
demo-bucket/prefix*"
               }
           }
        },
        {
           "Effect": "Allow",
```

```
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:log-stream-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:function-
name:function-version"
            ]
        }
    ]
}
```

上記のポリシーには、Amazon Kinesis Data Streams へのアクセスを許可するステートメントも含まれています。データソースとして Kinesis Data Streams を使用しない場合は、そのステートメントを削除できます。Amazon MSK をソースとして使用する場合は、このステートメントを次の文に置き換えることができます。

```
{
   "Sid":"",
   "Effect":"Allow",
   "Action":[
      "kafka:GetBootstrapBrokers",
      "kafka:DescribeCluster",
      "kafka:DescribeClusterV2",
      "kafka-cluster:Connect"
   ],
   "Resource":"arn:aws:kafka:{{mskClusterRegion}}:{{mskClusterAccount}}:cluster/
{{mskClusterName}}/{{clusterUUID}}"
},
{
   "Sid":"",
   "Effect":"Allow",
```

```
   "Action":[
      "kafka-cluster:DescribeTopic",
      "kafka-cluster:DescribeTopicDynamicConfiguration",
      "kafka-cluster:ReadData"
   ],
   "Resource":"arn:aws:kafka:{{mskClusterRegion}}:{{mskClusterAccount}}:topic/
{{mskClusterName}}/{{clusterUUID}}/{{mskTopicName}}"
},
{
   "Sid":"",
   "Effect":"Allow",
   "Action":[
      "kafka-cluster:DescribeGroup"
   ],
   "Resource":"arn:aws:kafka:{{mskClusterRegion}}:{{mskClusterAccount}}:group/
{{mskClusterName}}/{{clusterUUID}}/*"
}
```

他の AWS サービスがリソース AWS にアクセスできるようにする方法の詳細については、IAM ユーザーガイドの「 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。

別のアカウントの Amazon S3 の宛先へのアクセスを Amazon Data Firehose に付与する方法については、「the section called “Amazon S3 の宛先へのクロスアカウント間の配信”」を参照してください。

## Firehose に Amazon S3 Tables へのアクセスを許可する

Firehose には、 AWS AWS Glue テーブルにアクセスし、Amazon S3 テーブルバケット内のテーブルにデータを書き込むための特定のアクセス許可を持つ IAM ロールが必要です。Amazon S3 テーブルバケット内のテーブルに書き込むには、IAM ロールに必要なアクセス許可も提供する必要があります。Amazon S3 Tables カタログに必要なアクセス許可は、使用するアクセスコントロールモードによって異なります。

- IAM アクセスコントロール – Firehose 配信ロールには、Amazon S3 Tables リソースに対する IAM アクセス許可が直接必要です。
- Lake Formation アクセスコントロール – Firehose 配信ロールには、テーブルリソースへのアクセスを管理するためのアクセス AWS AWS Lake Formation 許可が必要です。 は、Data Catalog リソースのきめ細かなアクセスコントロールを可能にする独自のアクセス許可モデル AWS Lake Formation を使用します。

Firehose ストリームを作成するときに、この IAM ロールを設定します。アクセスコントロールモードに対応するタブを選択します。

## IAM アクセスコントロール

IAM アクセスコントロールモード ( なし AWS Lake Formation) を使用している場合、Firehose 配信ロールには Amazon S3 Tables リソースと AWS Glue Data Catalog オブジェクトに対する IAM アクセス許可が直接必要です。

にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。

ポリシーを作成し、ポリシーエディタで [JSON] を選択します。必要なアクセス許可を付与する次のインラインポリシーを追加します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3TablesAccessPermission",
            "Effect": "Allow",
            "Action": [
                "s3tables:GetTable",
                "s3tables:GetTableData",
                "s3tables:GetTableMetadataLocation",
                "s3tables:UpdateTableMetadataLocation"
            ],
            "Resource": [
                "arn:aws:s3tables:region:account-id:bucket/*",
                "arn:aws:s3tables:region:account-id:bucket/*/table/*"
            ]
        },
        {
            "Sid": "S3TableBucketAccessPermission",
            "Effect": "Allow",
            "Action": [
                "s3tables:GetTableBucket"
            ],
            "Resource": "arn:aws:s3tables:region:account-id:bucket/*"
        },
        {
            "Sid": "GlueCatalogAccessPermission",
            "Effect": "Allow",
```

```
            "Action": [
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:GetTable",
                "glue:GetTables",
                "glue:UpdateTable"
            ],
            "Resource": [
                "arn:aws:glue:region:account-id:catalog",
                "arn:aws:glue:region:account-id:catalog/s3tablescatalog",
                "arn:aws:glue:region:account-id:catalog/s3tablescatalog/*",
                "arn:aws:glue:region:account-id:database/*",
                "arn:aws:glue:region:account-id:table/*/*"
            ]
        },
        {
            "Sid": "S3DeliveryErrorBucketPermission",
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::error-delivery-bucket",
                "arn:aws:s3:::error-delivery-bucket/*"
            ]
        },
        {
            "Sid": "RequiredWhenUsingKinesisDataStreamsAsSource",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:region:account-id:stream/stream-name"
        },
        {
            "Sid": "KMSPermissionForS3TablesEncryption",
```

```
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:region:account-id:key/key-id"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn":
 "arn:aws:s3tables:region:account-id:bucket/*/table/*"
                }
            }
        },
        {
            "Sid": "RequiredWhenUsingLambdaForDataTransformation",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": "arn:aws:lambda:region:account-id:function:function-
name:function-version"
        },
        {
            "Sid": "CloudWatchLogsPermission",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:region:account-id:log-group:log-group-name:log-
stream:log-stream-name"
        }
    ]
}
```

このポリシーには、Amazon Kinesis Data Streams へのアクセス、Lambda 関数の呼び出し、および AWS KMS キーへのアクセスを許可するステートメントがあります。これらのリソースを使用しない場合は、それぞれのステートメントを削除できます。エラーログ記録が有効になっている場

合、Amazon Data Firehose はデータ配信エラーを CloudWatch ロググループとストリームにも送信します。このオプションを使用するには、ロググループ名とログストリーム名を設定する必要があります。ロググループ名とログストリーム名については、「CloudWatch Logs を使用して Amazon Data Firehose をモニタリングする」を参照してください。

インラインポリシーで、プレースホルダー値を実際のリソース名、 AWS アカウント 番号、リージョンに置き換えます。

ポリシーを作成したら、IAM コンソール (https://console.aws.amazon.com/iam/) を開き、[信頼されたエンティティタイプ] として AWS のサービス を指定して IAM ロールを作成します。

[サービスまたはユースケース] で、[Kinesis] を選択します。[ユースケース]で、[Kinesis Firehose] を選択します。

次のページで、前のステップで作成したポリシーを選択し、このロールにアタッチします。レビューページでは、このロールに既に信頼ポリシーがアタッチされていて、Firehose サービスにこのロールを引き受けるための許可が付与されているのを確認できます。ロールを作成すると、Amazon Data Firehose はロールを引き受けて、 AWS Glue および Amazon S3 Tables で必要なオペレーションを実行できます。Firehose サービスプリンシパルを、作成されたロールの信頼ポリシーに追加します。詳細については、「Firehose が IAM ロールを引き受けることを許可する」を参照してください。

## Lake Formation のアクセスコントロール

AWS Lake Formation アクセスコントロールモードを使用している場合、Firehose 配信ロールには、IAM ポリシーに加えて、認証情報供給のための AWS Lake Formation アクセス許可が必要です。

にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。

ポリシーを作成し、ポリシーエディタで [JSON] を選択します。読み取り/書き込み許可、データカタログのテーブルを更新する許可など、Amazon S3 に許可を付与する次のインラインポリシーを追加します。

JSON

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

```
    {
      "Sid": "S3TableAccessViaGlueFederation",
      "Effect": "Allow",
      "Action": [
        "glue:GetTable",
        "glue:GetDatabase",
        "glue:UpdateTable"
      ],
      "Resource": [
        "arn:aws:glue:us-east-1:123456789012:catalog/s3tablescatalog/*",
        "arn:aws:glue:us-east-1:123456789012:catalog/s3tablescatalog",
        "arn:aws:glue:us-east-1:123456789012:catalog",
        "arn:aws:glue:us-east-1:123456789012:database/*",
        "arn:aws:glue:us-east-1:123456789012:table/*/*"
      ]
    },
    {
      "Sid": "S3DeliveryErrorBucketPermission",
      "Effect": "Allow",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::<error delivery bucket>",
        "arn:aws:s3:::<error delivery bucket>/*"
      ]
    },
    {
      "Sid": "RequiredWhenUsingKinesisDataStreamsAsSource",
      "Effect": "Allow",
      "Action": [
        "kinesis:DescribeStream",
        "kinesis:GetShardIterator",
        "kinesis:GetRecords",
        "kinesis:ListShards"
      ],
      "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/<stream-name>"
    },
    {
```

```
      "Sid":
 "RequiredWhenDoingMetadataReadsANDDataAndMetadataWriteViaLakeformation",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": "*"
    },
    {
      "Sid": "RequiredWhenUsingKMSEncryptionForS3ErrorBucketDelivery",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "arn:aws:kms:us-east-1:123456789012:key/<KMS-key-id>"
      ],
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "s3.us-east-1.amazonaws.com"
        },
        "StringLike": {
          "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::<error delivery
 bucket>/prefix*"
        }
      }
    },
    {
      "Sid": "LoggingInCloudWatch",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name>:log-
stream:<log-stream-name>"
      ]
    },
    {
      "Sid": "RequiredWhenAttachingLambdaToFirehose",
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction",
```

```
        "lambda:GetFunctionConfiguration"
      ],
      "Resource": [
        "arn:aws:lambda:us-east-1:123456789012:function:<function-
name>:<function-version>"
      ]
    }
  ]
}
```

このポリシーには、Amazon Kinesis Data Streams へのアクセス、Lambda 関数の呼び出し、および AWS KMS キーへのアクセスを許可するステートメントがあります。これらのリソースを使用しない場合は、それぞれのステートメントを削除できます。エラーログ記録が有効になっている場合、Amazon Data Firehose はデータ配信エラーを CloudWatch ロググループとストリームにも送信します。このオプションを使用するには、ロググループ名とログストリーム名を設定する必要があります。ロググループ名とログストリーム名については、「CloudWatch Logs を使用して Amazon Data Firehose をモニタリングする」を参照してください。

インラインポリシーで、 を Amazon S3 バケット名<error delivery bucket>に置き換え、aws-account-idリージョンをリソースの有効な AWS アカウント 番号とリージョンに置き換えます。

IAM ポリシーに加えて、Firehose 配信ロールに必要なアクセス許可も付与する必要があります AWS Lake Formation。詳細については、「テーブルに対するアクセス許可の付与」を参照してください。

ポリシーを作成したら、IAM コンソール (https://console.aws.amazon.com/iam/) を開き、[信頼されたエンティティタイプ] として AWS のサービス を指定して IAM ロールを作成します。

[サービスまたはユースケース] で、[Kinesis] を選択します。[ユースケース]で、[Kinesis Firehose] を選択します。

次のページで、前のステップで作成したポリシーを選択し、このロールにアタッチします。レビューページでは、このロールに既に信頼ポリシーがアタッチされていて、Firehose サービスにこのロールを引き受けるための許可が付与されているのを確認できます。ロールを作成すると、Amazon Data Firehose はロールを引き受けて AWS Glue および S3 バケットで必要なオペレーションを実行できます。Firehose サービスプリンシパルを、作成されたロールの信頼ポリシーに追加します。詳細については、「Firehose が IAM ロールを引き受けることを許可する」を参照してください。

# Apache Iceberg テーブルの宛先へのアクセスを Firehose に付与する

AWS Glueを使用して Firehose ストリームと Apache Iceberg テーブルを作成する前に、IAM ロールが必要です。ポリシーと IAM ロールを作成するには、次のステップを実行します。Firehose はこの IAM ロールを引き受け、必要なアクションを実行します。

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. ポリシーを作成し、ポリシーエディタで [JSON] を選択します。
3. 読み取り/書き込み許可、データカタログのテーブルを更新する許可など、Amazon S3 に許可を付与する次のインラインポリシーを追加します。

   このポリシーには、Amazon Kinesis Data Streams へのアクセス、Lambda 関数の呼び出し、KMS キーへのアクセスを許可するステートメントが含まれています。これらのリソースを使用しない場合は、それぞれのステートメントを削除できます。

   エラーログ記録が有効になっている場合、Firehose はデータ配信エラーを CloudWatch ロググループとストリームにも送信します。このためには、ロググループとログストリーム名を設定する必要があります。ロググループ名とログストリーム名については、「CloudWatch Logs を使用して Amazon Data Firehose をモニタリングする」を参照してください。
4. インラインポリシーで、*amzn-s3-demo-bucket* を、Amazon S3 バケット名に置き換え、aws-account-id およびリージョンを、リソースの有効な AWS アカウント 番号とリージョンに置き換えます。

   > **Note**
   >
   > このロールは、データカタログ内のすべてのデータベースとテーブルに許可を付与します。必要に応じて、特定のテーブルとデータベースにのみ許可を付与できます。
5. ポリシーを作成したら、IAM コンソールを開き、[信頼されたエンティティタイプ] として AWS のサービス を使用して IAM ロールを作成します。
6. [サービスまたはユースケース] で、[Kinesis] を選択します。[ユースケース]で、[Kinesis Firehose] を選択します。
7. 次のページで、前のステップで作成したポリシーを選択し、このロールにアタッチします。レビューページでは、このロールを引き受けるための許可を Firehose サービスに付与する信頼ポリシーが既にこのロールにアタッチされているのを確認できます。ロールを作成する

と、Amazon Data Firehose は、 AWS Glue および S3 バケットで必要なオペレーションを実行するために、そのロールを引き受けることができます。

## Amazon Redshift の宛先へのアクセスを Firehose に付与する

Amazon Redshift の宛先を使用して Amazon Data Firehose にアクセスを付与するときは、次を参照してください。

トピック

- IAM ロールとアクセスポリシー
- Amazon Redshift プロビジョンドクラスターまたは Amazon Redshift Serverless サーバーレスワークグループへの VPC アクセス

### IAM ロールとアクセスポリシー

Amazon Redshift の宛先を使用する場合、Amazon Data Firehose は中間の場所として S3 バケットにデータを配信します。必要に応じて、所有している AWS KMS キーをデータ暗号化に使用できます。次に Amazon Data Firehose は、S3 バケットから Amazon Redshift プロビジョンドクラスターまたは Amazon Redshift Serverless ワークグループにデータをロードします。エラーログ記録が有効になっている場合、Amazon Data Firehose はデータ配信エラーを CloudWatch ロググループとストリームにも送信します。Amazon Data Firehose は、指定された Amazon Redshift ユーザー名とパスワードを使用してプロビジョンドクラスターまたは Amazon Redshift Serverless ワークグループにアクセスし、IAM ロールを使って、指定されたバケット、キー、CloudWatch ロググループ、ストリームにアクセスします。Firehose ストリームを作成するときは、IAM ロールが必要です。

次のアクセスポリシーを使用して、Amazon Data Firehose が S3 バケットおよび AWS KMS キーにアクセスできるようにします。S3 バケットを所有していない場合は、Amazon S3 アクションのリストに `s3:PutObjectAcl` を追加します。これにより、Amazon Data Firehose で配信されるオブジェクトへのフルアクセスが、バケット所有者に付与されます。このポリシーには、Amazon Kinesis Data Streams へのアクセスを許可するステートメントも含まれています。データソースとして Kinesis Data Streams を使用しない場合は、そのステートメントを削除できます。

JSON

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

```
    [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:us-east-1:123456789012:key/key-id"
           ],
           "Condition": {
               "StringEquals": {
                   "kms:ViaService": "s3.us-east-1.amazonaws.com"
               },
               "StringLike": {
                   "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-
demo-bucket/prefix*"
               }
           }
        },
        {
           "Effect": "Allow",
           "Action": [
               "kinesis:DescribeStream",
               "kinesis:GetShardIterator",
               "kinesis:GetRecords",
               "kinesis:ListShards"
           ],
```

```
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:log-stream-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:function-
name:function-version"
            ]
        }
    ]
}
```

他の AWS サービスがリソース AWS にアクセスできるようにする方法の詳細については、IAM ユーザーガイドの「 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。

## Amazon Redshift プロビジョンドクラスターまたは Amazon Redshift Serverless サーバーレスワークグループへの VPC アクセス

Amazon Redshift プロビジョンドクラスターまたは Amazon Redshift Serverless ワークグループが仮想プライベートクラウド (VPC) にある場合、それらはパブリック IP アドレスでパブリックにアクセスできる必要があります。また、Amazon Data Firehose IP アドレスのブロックを解除して、Amazon Data Firehose に Amazon Redshift プロビジョンドクラスターまたは Amazon Redshift Serverless ワークグループへのアクセスを付与します。現在、Amazon Data Firehose は利用可能なリージョンごとに 1 つの CIDR ブロックを使用します。

| リージョン | CIDR ブロック |
| --- | --- |
| 米国東部(オハイオ) | 13.58.135.96/27 |
| 米国東部 (バージニア北部) | 52.70.63.192/27 |
| 米国西部 (北カリフォルニア) | 13.57.135.192/27 |
| 米国西部 (オレゴン) | 52.89.255.224/27 |
| AWS GovCloud (米国東部) | 18.253.138.96/27 |
| AWS GovCloud (米国西部) | 52.61.204.160/27 |
| カナダ (中部) | 35.183.92.128/27 |
| カナダ西部 (カルガリー) | 40.176.98.192/27 |
| アジアパシフィック (香港) | 18.162.221.32/27 |
| アジアパシフィック (ムンバイ) | 13.232.67.32/27 |
| アジアパシフィック (ハイデラバード) | 18.60.192.128/27 |
| アジアパシフィック (ソウル) | 13.209.1.64/27 |
| アジアパシフィック (シンガポール) | 13.228.64.192/27 |
| アジアパシフィック (シドニー) | 13.210.67.224/27 |

| リージョン | CIDR ブロック |
| --- | --- |
| アジアパシフィック (ジャカルタ) | 108.136.221.64/27 |
| アジアパシフィック (東京) | 13.113.196.224/27 |
| アジアパシフィック (大阪) | 13.208.177.192/27 |
| アジアパシフィック (タイ) | 43.208.112.96/27 |
| アジアパシフィック (台北) | 43.212.53.160/27 |
| 中国 (北京) | 52.81.151.32/27 |
| 中国 (寧夏) | 161.189.23.64/27 |
| 欧州 (チューリッヒ) | 16.62.183.32/27 |
| 欧州 (フランクフルト) | 35.158.127.160/27 |
| 欧州 (アイルランド) | 52.19.239.192/27 |
| 欧州 (ロンドン) | 18.130.1.96/27 |
| 欧州 (パリ) | 35.180.1.96/27 |
| 欧州 (ストックホルム) | 13.53.63.224/27 |
| 欧州 (スペイン) | 18.100.71.96/27 |
| 中東 (バーレーン) | 15.185.91.0/27 |
| メキシコ (中部) | 78.12.207.32/27 |
| 南米 (サンパウロ) | 18.228.1.128/27 |

| リージョン | CIDR ブロック |
| --- | --- |
| 欧州 (ミラノ) | 15.161.135.128/27 |
| アフリカ (ケープタウン) | 13.244.121.224/27 |
| 中東 (アラブ首長国連邦) | 3.28.159.32/27 |
| イスラエル (テルアビブ) | 51.16.102.0/27 |
| アジアパシフィック (メルボルン) | 16.50.161.128/27 |
| アジアパシフィック (マレーシア) | 43.216.58.0/27 |

IP アドレスのブロック解除方法の詳細については、Amazon Redshift 入門ガイドの「クラスターへのアクセスの許可」を参照してください。

## Firehose に公開 OpenSearch Service の宛先へのアクセスを付与する

OpenSearch Service の宛先を使用している場合、Amazon Data Firehose はデータを OpenSearch Service クラスターに配信し、同時に、失敗したドキュメントまたはすべてのドキュメントを S3 バケットにバックアップします。エラーログ記録が有効になっている場合、Amazon Data Firehose はデータ配信エラーを CloudWatch ロググループとストリームにも送信します。Amazon Data Firehose は、IAM ロールを使用して、指定された OpenSearch Service ドメイン、S3 バケット、AWS KMS キー、CloudWatch ロググループとストリームにアクセスします。Firehose ストリームを作成するときは、IAM ロールが必要です。

次のアクセスポリシーを使用して、Amazon Data Firehose が S3 バケット、OpenSearch Service ドメイン、および AWS KMS キーにアクセスできるようにします。S3 バケットを所有していない場合は、Amazon S3 アクションのリストに `s3:PutObjectAcl` を追加します。これにより、Amazon Data Firehose で配信されるオブジェクトへのフルアクセスが、バケット所有者に付与されます。このポリシーには、Amazon Kinesis Data Streams へのアクセスを許可するステートメントも含まれています。データソースとして Kinesis Data Streams を使用しない場合は、そのステートメントを削除できます。

他の AWS サービスがリソース AWS にアクセスできるようにする方法の詳細については、IAM ユーザーガイドの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。

別のアカウントの OpenSearch Service クラスターへのアクセスを Amazon Data Firehose に付与する方法については、「the section called “OpenSearch Service の宛先へのクロスアカウント間の配信”」を参照してください。

## VPC 内の OpenSearch Service の宛先へのアクセスを Firehose に付与する

OpenSearch Service ドメインが VPC 内にある場合は、前のセクションで説明した許可を Amazon Data Firehose に付与してください。さらに、OpenSearch Service ドメインの VPC にアクセスできるように、次の許可を Amazon Data Firehose に付与する必要があります。

- `ec2:DescribeVpcs`
- `ec2:DescribeVpcAttribute`
- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeNetworkInterfaces`
- `ec2:CreateNetworkInterface`
- `ec2:CreateNetworkInterfacePermission`
- `ec2:DeleteNetworkInterface`

> **Important**
>
> Firehose ストリームの作成後にこれらの許可を取り消さないでください。これらの許可を取り消すと、サービスが ENI のクエリまたは更新を試みるたびに、Firehose ストリームのパフォーマンスが低下するか、または OpenSearch サービスドメインへのデータの配信が停止します。

> **Important**
>
> プライベート VPC の宛先にデータを配信するためのサブネットを指定する場合は、選択したサブネットに十分な数の空き IP アドレスがあることを確認してください。指定されたサブネットに使用可能な空き IP アドレスがない場合、Firehose はプライベート VPC でデータ配信用の ENI を作成または追加できず、配信のパフォーマンスが低下するか、または配信が失敗します。

Firehose ストリームを作成または更新するときに、OpenSearch Service ドメインにデータを送信するときに Firehose で使用するセキュリティグループを指定します。OpenSearch Service ドメインで使用しているのと同じセキュリティグループを使用することも、別のセキュリティグループを使用することもできます。別のセキュリティグループを指定する場合は、そのセキュリティグループで、OpenSearch Service ドメインのセキュリティグループへのアウトバウンド HTTPS トラフィックを必ず許可します。また、OpenSearch Service ドメインのセキュリティグループで、Firehose ストリームの設定時に指定したセキュリティグループからの HTTPS トラフィックを必ず許可します。Firehose ストリームと OpenSearch Service ドメインの両方に同じセキュリティグループを使用する場合は、セキュリティグループのインバウンドルールで HTTPS トラフィックを必ず許可します。セキュリティグループのルールの詳細については、Amazon VPCドキュメントの「セキュリティグループのルール」を参照してください。

## Firehose に公開 OpenSearch Serverless の宛先へのアクセスを付与する

OpenSearch Serverless の宛先を使用している場合、Amazon Data Firehose はデータを OpenSearch Serverless クラスターに配信し、同時に、失敗したドキュメントまたはすべてのドキュメントを S3 バケットにバックアップします。エラーログ記録が有効になっている場合、Amazon Data Firehose はデータ配信エラーを CloudWatch ロググループとストリームにも送信します。Amazon Data Firehose は IAM ロールを使用して、指定された OpenSearch Serverless コレクション、S3 バケット、 AWS KMS キー、CloudWatch ロググループとストリームにアクセスします。Firehose ストリームを作成するときは、IAM ロールが必要です。

次のアクセスポリシーを使用して、Amazon Data Firehose が S3 バケット、OpenSearch Serverless ドメイン、および AWS KMS キーにアクセスできるようにします。S3 バケットを所有していない場合は、Amazon S3 アクションのリストに `s3:PutObjectAcl` を追加します。これにより、Amazon Data Firehose で配信されるオブジェクトへのフルアクセスが、バケット所有者に付与されます。このポリシーには、Amazon Kinesis Data Streams へのアクセスを許可するステートメントも含まれています。データソースとして Kinesis Data Streams を使用しない場合は、そのステートメントを削除できます。

JSON

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

```
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:us-east-1:123456789012:key/key-id"
           ],
           "Condition": {
               "StringEquals": {
                   "kms:ViaService": "s3.us-east-1.amazonaws.com"
               },
               "StringLike": {
                   "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-
demo-bucket/prefix*"
               }
           }
        },
       {
          "Effect": "Allow",
          "Action": [
              "kinesis:DescribeStream",
              "kinesis:GetShardIterator",
              "kinesis:GetRecords",
              "kinesis:ListShards"
          ],
          "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-name"
       },
       {
          "Effect": "Allow",
          "Action": [
```

```
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:log-stream-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:function-
name:function-version"
            ]
        },
        {
         "Effect": "Allow",
         "Action": "aoss:APIAccessAll",
         "Resource": "arn:aws:aoss:us-east-1:123456789012:collection/collection-
id"
       }
     ]
}
```

上記のポリシーに加え、データアクセスポリシーで次の最小限の許可が割り当てられるように Amazon Data Firehose を設定します。

```
[
   {
      "Rules":[
         {
            "ResourceType":"index",
            "Resource":[
               "index/target-collection/target-index"
            ],
            "Permission":[
               "aoss:WriteDocument",
               "aoss:UpdateIndex",
```

```
                "aoss:CreateIndex"
            ]
         }
      ],
      "Principal":[
         "arn:aws:sts::123456789012:assumed-role/firehose-delivery-role-name/*"
      ]
   }
]
```

他の AWS サービスがリソース AWS にアクセスできるようにする方法の詳細については、IAM ユーザーガイドの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。

## VPC 内の OpenSearch Serverless の宛先へのアクセスを Firehose に付与する

OpenSearch Serverless コレクションが VPC 内にある場合は、前のセクションで説明した許可を Amazon Data Firehose に付与してください。さらに、OpenSearch Serverless コレクションの VPC にアクセスできるように、次の許可を Amazon Data Firehose に付与する必要があります。

- `ec2:DescribeVpcs`
- `ec2:DescribeVpcAttribute`
- `ec2:DescribeSubnets`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeNetworkInterfaces`
- `ec2:CreateNetworkInterface`
- `ec2:CreateNetworkInterfacePermission`
- `ec2:DeleteNetworkInterface`

> **Important**
>
> Firehose ストリームの作成後にこれらの許可を取り消さないでください。これらの許可を取り消すと、サービスが ENI のクエリまたは更新を試みるたびに、Firehose ストリームのパフォーマンスが低下するか、または OpenSearch サービスドメインへのデータの配信が停止します。

> **Important**
>
> プライベート VPC の宛先にデータを配信するためのサブネットを指定する場合は、選択したサブネットに十分な数の空き IP アドレスがあることを確認してください。指定されたサブネットに使用可能な空き IP アドレスがない場合、Firehose はプライベート VPC でデータ配信用の ENI を作成または追加できず、配信のパフォーマンスが低下するか、または配信が失敗します。

Firehose ストリームを作成または更新するときは、OpenSearch Serverless コレクションにデータを送信する際に Firehose で使用するセキュリティグループを指定します。OpenSearch Serverless コレクションで使用しているものと同じセキュリティグループを使用できますが、別のセキュリティグループも使用できます。別のセキュリティグループを指定する場合は、そのセキュリティグループが、OpenSearch Serverless コレクションのセキュリティグループへのアウトバウンド HTTPS トラフィックを許可していることを確認します。また、OpenSearch Serverless コレクションのセキュリティグループが、Firehose ストリームを設定したときに指定したセキュリティグループからの HTTPS トラフィックを許可していることも確認します。Firehose ストリームと OpenSearch Serverless コレクションの両方で同じセキュリティグループを使用する場合は、セキュリティグループのインバウンドルールが HTTPS トラフィックを許可していることを確認します。セキュリティグループのルールの詳細については、Amazon VPCドキュメントの「セキュリティグループのルール」を参照してください。

## Splunk の宛先へのアクセスを Firehose に付与する

Splunk の宛先を使用している場合、Amazon Data Firehose はデータを Splunk HTTP Event Collector (HEC) エンドポイントに配信します。また、指定した Amazon S3 バケットにデータをバックアップし、オプションで Amazon S3 サーバー側の暗号化用に所有している AWS KMS キーを使用することもできます。エラーログ記録が有効になっている場合、Firehose はデータ配信エラーを CloudWatch ログストリームに送信します。データ変換 AWS Lambda に を使用することもできます。

AWS ロードバランサーを使用する場合は、それが Classic Load Balancer または Application Load Balancer であることを確認します。また、Classic Load Balancer の Cookie の有効期限を無効にした状態で、期間ベースのスティッキーセッションを有効にし、Application Load Balancer の有効期限を最大値 (7 日間) に設定します。これを行う方法の詳細については、Classic Load Balancer または Application Load Balancer の「Duration-Based Session Stickiness」を参照してください。

Firehose ストリームを作成する際には、IAM ロールが必要です。Firehose はその IAM ロールを引き受け、指定されたバケット、キー、および CloudWatch ロググループとストリームへのアクセスを取得します。

次のアクセスポリシーを使用して、Amazon Data Firehose が S3 バケットにアクセスできるようにします。S3 バケットを所有していない場合は、Amazon S3 アクションのリストに `s3:PutObjectAcl` を追加します。これにより、Amazon Data Firehose で配信されるオブジェクトへのフルアクセスが、バケット所有者に付与されます。このポリシーは、Amazon Data Firehose にエラーログ記録用の CloudWatch へのアクセスと、データ変換 AWS Lambda 用の へのアクセスも許可します。このポリシーには、Amazon Kinesis Data Streams へのアクセスを許可するステートメントも含まれています。データソースとして Kinesis Data Streams を使用しない場合は、そのステートメントを削除できます。Amazon Data Firehose は Splunk へのアクセスに IAM を使用しません。Splunk へのアクセスには、HEC トークンが使用されます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
```

```
            "Resource": [
                "arn:aws:kms:us-east-1:123456789012:key/key-id"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.us-east-1.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-
demo-bucket/prefix*"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:function-
name:function-version"
            ]
```

```
        }
    ]
}
```

他の AWS サービスがリソース AWS にアクセスできるようにする方法の詳細については、IAM ユーザーガイドの「 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。

## VPC の Splunk へのアクセス

Splunk プラットフォームが VPC にある場合、パブリック IP アドレスでパブリックにアクセス可能である必要があります。また、Amazon Data Firehose の IP アドレスをブロック解除して、Splunk プラットフォームへのアクセスを Amazon Data Firehose に付与する必要があります。Amazon Data Firehose は現在、次の CIDR ブロックを使用します。

| リージョン | CIDR ブロック |
| --- | --- |
| 米国東部(オハイオ) | `18.216.68.160/27, 18.216.170.64/27, 18.216.170.96/27 \` |
| 米国東部 (バージニア北部) | `34.238.188.128/26, 34.238.188.192/26, 34.238.195.0/26` |
| 米国西部 (北カリフォルニア) | `13.57.180.0/26` |
| 米国西部 (オレゴン) | `34.216.24.32/27, 34.216.24.192/27, 34.216.24.224/27` |
| AWS GovCloud (米国東部) | `18.253.138.192/26` |
| AWS GovCloud (米国西部) | `52.61.204.192/26` |
| アジアパシフィック (香港) | `18.162.221.64/26` |
| アジアパシフィック (ムンバイ) | `13.232.67.64/26` |

| リージョン | CIDR ブロック |
| --- | --- |
| アジアパシフィック (ソウル) | 13.209.71.0/26 |
| アジアパシフィック (シンガポール) | 13.229.187.128/26 |
| アジアパシフィック (シドニー) | 13.211.12.0/26 |
| アジアパシフィック (タイ) | 43.208.112.128/26 |
| アジアパシフィック (東京) | 13.230.21.0/27, 13.230.21.32/27 |
| カナダ (中部) | 35.183.92.64/26 |
| カナダ西部 (カルガリー) | 40.176.98.128/26 |
| 欧州 (フランクフルト) | 18.194.95.192/27, 18.194.95.224/27, 18.195.48.0/27 |
| 欧州 (アイルランド) | 34.241.197.32/27, 34.241.197.64/27, 34.241.197.96/27 |
| 欧州 (ロンドン) | 18.130.91.0/26 |
| 欧州 (パリ) | 35.180.112.0/26 |
| 欧州 (スペイン) | 18.100.194.0/26 |
| 欧州 (ストックホルム) | 13.53.191.0/26 |
| 中東 (バーレーン) | 15.185.91.64/26 |
| メキシコ (中部) | 78.12.207.64/26 |
| 南米 (サンパウロ) | 18.228.1.192/26 |

| リージョン | CIDR ブロック |
| --- | --- |
| 欧州 (ミラノ) | 15.161.135.192/26 |
| アフリカ (ケープタウン) | 13.244.165.128/26 |
| アジアパシフィック (大阪) | 13.208.217.0/26 |
| 中国 (北京) | 52.81.151.64/26 |
| 中国 (寧夏) | 161.189.23.128/26 |
| アジアパシフィック (ジャカルタ) | 108.136.221.128/26 |
| 中東 (アラブ首長国連邦) | 3.28.159.64/26 |
| イスラエル (テルアビブ) | 51.16.102.64/26 |
| 欧州 (チューリッヒ) | 16.62.183.64/26 |
| アジアパシフィック (ハイデラバード) | 18.60.192.192/26 |
| アジアパシフィック (メルボルン) | 16.50.161.192/26 |
| アジアパシフィック (マレーシア) | 43.216.44.192/26 |

## Amazon Data Firehose を使用して VPC フローログを Splunk に取り込む

VPC フローログサブスクリプションを作成し、Firehose に発行して、サポートされている宛先に VPC フローログを送信する方法の詳細については、「Amazon Data Firehose を使用して VPC フローログを Splunk に取り込む」を参照してください。

## Snowflake または HTTP エンドポイントへのアクセス

宛先が HTTP エンドポイントまたは Snowflake パブリッククラスターである場合、Amazon Data Firehose に固有の AWS IP アドレス範囲のサブセットはありません。

Firehose をパブリック Snowflake クラスターの許可リストまたはパブリック HTTP もしくは HTTPS エンドポイントに追加するには、現在の AWS IP アドレス範囲をすべてイングレスルールに追加します。

> **Note**
>
> 通知は、関連付けられたトピックと同じ AWS リージョンの IP アドレスから取得されるとは限りません。すべてのリージョンの AWS IP アドレス範囲を含める必要があります。

## Snowflake の宛先へのアクセスを Firehose に付与する

Snowflake を宛先として使用する場合、Firehose は Snowflake アカウント URL を使用して Snowflake アカウントにデータを配信します。また、指定した Amazon Simple Storage Service バケットにエラーデータをバックアップし、オプションで Amazon S3 サーバー側の暗号化用に所有している AWS Key Management Service キーを使用することもできます。エラーログ記録が有効になっている場合、Firehose はデータ配信エラーを CloudWatch Logs ストリームに送信します。

Firehose ストリームを作成する前に、IAM ロールが必要です。Firehose はその IAM ロールを引き受け、指定されたバケット、キー、および CloudWatch ロググループとストリームへのアクセスを取得します。次のアクセスポリシーを使用して、Firehose が S3 バケットにアクセスできるようにします。S3 バケットを所有していない場合は、Amazon Simple Storage Service アクションのリストに `s3:PutObjectAcl` を追加します。これにより、Firehose で配信されるオブジェクトへのフルアクセスが、バケット所有者に付与されます。また、このポリシーは、エラーログ記録のために CloudWatch へのアクセスを Firehose に付与します。このポリシーには、Amazon Kinesis Data Streams へのアクセスを許可するステートメントも含まれています。データソースとして Kinesis Data Streams を使用しない場合は、そのステートメントを削除できます。Firehose は Snowflake にアクセスするために IAM を使用しません。Snowflake へのアクセスには、プライベートクラスターの場合は Snowflake アカウントの URL と PrivateLink Vpce Id を使用します。

JSON

```
{
```

```
"Version":"2012-10-17",
    "Statement":
    [
        {
"Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
"Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:123456789012:key/key-id"
            ],
            "Condition": {
"StringEquals": {
"kms:ViaService": "s3.us-east-1.amazonaws.com"
                },
                "StringLike": {
"kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-demo-bucket/prefix*"
                }
            }
        },
        {
"Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
```

```
            "Resource": "arn:aws:kinesis:us-east-1:123456789012:stream/stream-
name"
        },
        {
"Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name:log-
stream:*"
            ]
        }
    ]
}
```

他の AWS サービスが AWS リソースにアクセスできるようにする方法の詳細については、IAM ユーザーガイドの「 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。

## VPC での Snowflake へのアクセス

Snowflake クラスターでプライベートリンクが有効になっている場合、Firehose はプライベートリンクの作成時に次のいずれかの VPC エンドポイントを使用して、パブリックインターネットを経由せずにプライベートクラスターにデータを配信します。そのためには、Snowflake ネットワークルールを作成して、AwsVpceIds AWS リージョン クラスターの に以下からの進入を許可します。詳細については、「Snowflake ユーザーガイド」の「Creating network rule」を参照してください。

クラスターが存在するリージョンに基づいて使用する VPC エンドポイント ID

| AWS リージョン | **VPCE IDs** |
| --- | --- |
| 米国東部 (オハイオ) | vpce-0d96cafcd96a50aeb<br>vpce-0cec34343d48f537b |
| 米国東部 (バージニア北部) | vpce-0b4d7e8478e141ba8<br>vpce-0b75cd681fb507352<br>vpce-01c03e63820ec00d8<br>vpce-0c2cfc51dc2882422 |

| AWS リージョン | VPCE IDs |
| --- | --- |
| | vpce-06ca862f019e4e056 |
| | vpce-020cda0cfa63f8d1c |
| | vpce-0b80504a1a783cd70 |
| | vpce-0289b9ff0b5259a96 |
| | vpce-0d7add8628bd69a12 |
| | vpce-02bfb5966cc59b2af |
| | vpce-09e707674af878bf2 |
| | vpce-049b52e96cc1a2165 |
| | vpce-0bb6c7b7a8a86cdbb |
| | vpce-03b22d599f51e80f3 |
| | vpce-01d60dc60fc106fe1 |
| | vpce-0186d20a4b24ecbef |
| | vpce-0533906401a36e416 |
| | vpce-05111fb13d396710e |
| | vpce-0694613f4fbd6f514 |
| | vpce-09b21cb25fe4cc4f4 |
| | vpce-06029c3550e4d2399 |
| | vpce-00961862a21b033da |
| | vpce-01620b9ae33273587 |
| | vpce-078cf4ec226880ac9 |
| | vpce-0d711bf076ce56381 |

| AWS リージョン | VPCE IDs |
| --- | --- |
| | vpce-066b7e13cbfca6f6e<br>vpce-0674541252d9ccc26<br>vpce-03540b88dedb4b000<br>vpce-0b1828e79ad394b95<br>vpce-0dc0e6f001fb1a60d<br>vpce-0d8f82e71a244098a<br>vpce-00e374d9e3f1af5ce<br>vpce-0c1e3d6631ddb442f |
| 米国西部 (オレゴン) | vpce-0f60f72da4cd1e4e7<br>vpce-0c60d21eb8b1669fd<br>vpce-01c4e3e29afdafbef<br>vpce-0cc6bf2a88da139de<br>vpce-0797e08e169e50662<br>vpce-033cbe480381b5c0e<br>vpce-00debbdd8f9eb10a5<br>vpce-08ec2f386c809e889<br>vpce-0856d14310857b545 |
| 欧州 (フランクフルト) | vpce-068dbb7d71c9460fb<br>vpce-0a7a7f095942d4ec9 |

| AWS リージョン | VPCE IDs |
| --- | --- |
| 欧州 (アイルランド) | vpce-06857e59c005a6276<br>vpce-04390f4f8778b75f2<br>vpce-011fd2b1f0aa172fd |
| アジアパシフィック (東京) | vpce-06369e5258144e68a<br>vpce-0f2363cdb8926fbe8 |
| アジアパシフィック (シンガポール) | vpce-049cd46cce7a12d52<br>vpce-0e8965a1a4bdb8941 |
| アジアパシフィック (ソウル) | vpce-0aa444d9001e1faa1<br>vpce-04a49d4dcfd02b884 |
| アジアパシフィック (シドニー) | vpce-048a60a182c52be63<br>vpce-03c19949787fd1859 |
| アジアパシフィック (ムンバイ) | vpce-0d68cb822f6f0db68<br>vpce-0517d32692ffcbde2 |
| 欧州 (ロンドン) | vpce-0fd1874a0ba3b9374<br>vpce-08091b1a85e206029 |
| 南米 ( サンパウロ ) | vpce-065169b8144e4f12e<br>vpce-0493699f0e5762d63 |
| カナダ (中部) | vpce-07e6ed81689d5271f<br>vpce-0f53239730541394c |

| AWS リージョン | VPCE IDs |
| --- | --- |
| 欧州 (パリ) | vpce-09419680077e6488a<br>vpce-0ea81ba2c08140c14 |
| アジアパシフィック (大阪) | vpce-0a9f003e6a7e38c05<br>vpce-02886510b897b1c5a |
| 欧州 (ストックホルム) | vpce-0d96410833219025a<br>vpce-060a32f9a75ba969f |
| アジアパシフィック (ジャカルタ) | vpce-00add4b9a25e5c649<br>vpce-004ae2de34338a856 |

## HTTP エンドポイントの宛先へのアクセスを Firehose に付与する

Amazon Data Firehose を使用して、任意の HTTP エンドポイントの宛先にデータを配信できます。また、Amazon Data Firehose は指定した Amazon S3 バケットにデータをバックアップします。必要に応じて Amazon S3 のサーバー側の暗号化のために所有している AWS KMS キーを使用することもできます。エラーログ記録が有効になっている場合、Amazon Data Firehose はデータ配信エラーを CloudWatch ログストリームに送信します。データ変換 AWS Lambda に を使用することもできます。

Firehose ストリームを作成するときは、IAM ロールが必要です。Amazon Data Firehose はその IAM ロールを引き受け、指定されたバケット、キー、および CloudWatch ロググループとストリームへのアクセスを取得します。

次のアクセスポリシーを使用して、Amazon Data Firehose がデータバックアップ用に指定した S3 バケットにアクセスできるようにします。S3 バケットを所有していない場合は、Amazon S3 アクションのリストに `s3:PutObjectAcl` を追加します。これにより、Amazon Data Firehose で配信されるオブジェクトへのフルアクセスが、バケット所有者に付与されます。このポリシーは、Amazon Data Firehose にエラーログ記録用の CloudWatch へのアクセスと、データ変換 AWS Lambda 用の へのアクセスも許可します。このポリシーには、Amazon Kinesis Data Streams へのアクセスを許可するステートメントも含まれています。データソースとして Kinesis Data Streams を使用しない場合は、そのステートメントを削除できます。

> **Important**
>
> Amazon Data Firehose は IAM を使用して、サポートされているサードパーティーサービスプロバイダーが所有する HTTP エンドポイントの宛先 (Datadog、Dynatrace、LogicMonitor、MongoDB、New Relic、Splunk、または Sumo Logic など) にアクセスすることはありません。サポートされているサードパーティーサービスプロバイダーが所有する指定された HTTP エンドポイントの宛先にアクセスするには、そのサービスプロバイダーに連絡して、Amazon Data Firehose からそのサービスへのデータ配信を有効にするために必要な API キーまたはアクセスキーを取得します。

他の AWS サービスがリソース AWS にアクセスできるようにする方法の詳細については、IAM ユーザーガイドの「 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。

> **Important**
>
> 現在、Amazon Data Firehose は VPC 内の HTTP エンドポイントへのデータ配信をサポートしていません。

## Amazon MSK からのクロスアカウント配信

Firehose アカウント (アカウント B など) から Firehose ストリームを作成し、ソースが別の AWS アカウント (アカウント A) の MSK クラスターである場合は、次の設定が必要です。

アカウント A:

1. Amazon MSK コンソールで、プロビジョンドクラスターを選択し、[プロパティ] を選択します。
2. [ネットワーク設定] で [編集] を選択し、[マルチ VPC 接続] をオンにします。
3. [セキュリティ設定] で [クラスターポリシーの編集] を選択します。

    a. クラスターにまだポリシーが設定されていない場合は、[Firehose サービスプリンシパルを含める] と [Firehose のクロスアカウント S3 配信を有効にする] にチェックを入れます。AWS マネジメントコンソール は、適切なアクセス許可を持つポリシーを自動的に生成します。

    b. クラスターに既にポリシーが設定されている場合は、既存のポリシーに次のアクセス許可を追加します。

```
{
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::us-east-1:role/mskaasTestDeliveryRole"
      },
      "Action": [
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeCluster",
        "kafka:DescribeClusterV2",
        "kafka-cluster:Connect"
      ],
      "Resource": "arn:aws:kafka:us-east-1:123456789012:cluster/DO-NOT-TOUCH-
mskaas-provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20"  // ARN
 of the cluster
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:::iam::us-east-1:role/mskaasTestDeliveryRole"
      },
      "Action": [
        "kafka-cluster:DescribeTopic",
        "kafka-cluster:DescribeTopicDynamicConfiguration",
        "kafka-cluster:ReadData"
      ],
      "Resource": "arn:aws:kafka:us-east-1:arn:topic/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*"//topic of the
 cluster
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::us-east-1:role/mskaasTestDeliveryRole"
      },
      "Action": "kafka-cluster:DescribeGroup",
      "Resource": "arn:aws:kafka:us-east-1:arn:group/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*" //topic of
 the cluster
    },
 }
```

4. [AWS プリンシパル] にアカウント B のプリンシパル ID を入力します。

5. [トピック] で、Firehose ストリームでデータを取り込む Apache Kafka トピックを指定します。Firehose ストリームを作成した後は、このトピックを更新することはできません。
6. [変更を保存] を選択します。

アカウント B:

1. Firehose コンソールで、アカウント B を使用して [Firehose ストリームを作成] を選択します。
2. [ソース] で、[Amazon Managed Streaming for Apache Kafka] を選択します。
3. [ソース設定] の [Apache Kafka クラスター用 Amazon Managed Streaming] で、アカウント A の Amazon MSK クラスターの ARN を入力します。
4. [トピック] で、Firehose ストリームでデータを取り込む Apache Kafka トピックを指定します。Firehose ストリームを作成した後は、このトピックを更新することはできません。
5. [配信ストリーム名] に Firehose ストリームの名前を入力します。

アカウント B で Firehose ストリームを作成するときは、設定されたトピックのクロスアカウント Amazon MSK クラスターへの「読み取り」アクセスを Firehose ストリームに許可する IAM ロール ( の使用時にデフォルトで作成 AWS マネジメントコンソール) が必要です。

AWS マネジメントコンソールでの設定内容は以下のとおりです。

```
{
    "Sid": "",
    "Effect": "Allow",
    "Action": [
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeCluster",
        "kafka:DescribeClusterV2",
        "kafka-cluster:Connect"
        ],
    "Resource": "arn:aws:kafka:us-east-1:arn:aws:cluster/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*" //topic of the
 cluster
    },
    {
    "Sid": "",
    "Effect": "Allow",
    "Action": [
        "kafka-cluster:DescribeTopic",
        "kafka-cluster:DescribeTopicDynamicConfiguration",
```

```
        "kafka-cluster:ReadData"
    ],
    "Resource": "arn:aws:kafka:us-east-1:arn:aws:topic/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/mskaas_test_topic" //
topic of the cluster
    },
    {
    "Sid": "",
    "Effect": "Allow",
    "Action": [
        "kafka-cluster:DescribeGroup"
    ],
    "Resource": "arn:aws:kafka:us-east-1:arn:aws:group/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20/*" //topic of the
 cluster
    },
 }
```

次に、レコード変換とレコード形式の変換を設定するオプションのステップを実行できます。詳細については、「(オプション) レコード変換と形式転換を設定する」を参照してください。

## Amazon S3 の宛先へのクロスアカウント間の配信

AWS CLI または Amazon Data Firehose APIs を使用して、別のアカウントの Amazon S3 送信先を持つ 1 つの AWS アカウントに Firehose ストリームを作成できます。次の手順では、アカウント A が所有する Firehose ストリームを設定して、アカウント B が所有する Amazon S3 バケットにデータを配信する例を示します。

1. [Amazon S3 宛先へのアクセスを Firehose に付与する] で説明されているステップを使用して、アカウント A で IAM ロールを作成します。

> **Note**
>
> この場合、アクセスポリシーで指定した Amazon S3 バケットはアカウント B が所有しています。Amazon Data Firehose が配信するオブジェクトへのフルアクセスがアカウント B に付与されるよう、アクセスポリシーで Amazon S3 アクションのリストに `s3:PutObjectAcl` を追加したことを確認してください。このアクセス許可は、クロスアカウント配信に必要です。Amazon Data Firehose は、リクエストの「x-amz-acl」ヘッダーを「bucket-owner-full-control」に設定します。

2. 以前に作成した IAM ロールからのアクセスを許可するには、アカウント B で S3 バケットポリシーを作成します。次のコードは、バケットポリシーの例です。詳細については、「バケットポリシーとユーザーポリシーの使用」を参照してください。

   JSON

```
{

    "Version":"2012-10-17",
    "Id": "PolicyID",
    "Statement": [
        {
            "Sid": "StmtID",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/iam-role-name"
            },
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

3. ステップ 1 で作成した IAM ロールを使用して、アカウント A で Firehose ストリームを作成します。

# OpenSearch Service の宛先へのクロスアカウント間の配信

AWS CLI または Amazon Data Firehose APIs を使用して、ある AWS アカウントに Firehose ストリームを作成し、別のアカウントに OpenSearch Service の送信先を設定できます。次の手順では、アカウント A で Firehose ストリームを作成し、アカウント B が所有する OpenSearch Service の宛先にデータを配信するよう設定する方法の例を示しています。

1. the section called “Firehose に公開 OpenSearch Service の宛先へのアクセスを付与する” で示されているステップを使用して、アカウント A に IAM ロールを作成します。
2. 前のステップで作成した IAM ロールからのアクセスを許可するには、アカウント B に OpenSearch Service ポリシーを作成します。例として、JSON を以下に示します。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/firehose_delivery_role "
      },
      "Action": "es:ESHttpGet",
      "Resource": [
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/_all/
_settings",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_cluster/stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
roletest*/_mapping/roletest",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_nodes",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_nodes/stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_nodes/*/stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
_stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/
roletest*/_stats",
        "arn:aws:es:us-east-1:123456789012:domain/cross-account-cluster/"
```

```
      ]
    }
  ]
}
```

3. ステップ 1 で作成した IAM ロールを使用して、アカウント A で Firehose ストリームを作成します。Firehose ストリームを作成するときは、 AWS CLI または Amazon Data Firehose APIs を使用して、DomainARNOpenSearch Service の代わりに ClusterEndpointフィールドを指定します。

> **Note**
>
> 別の AWS アカウントの OpenSearch Service の送信先を持つ 1 つのアカウントに Firehose ストリームを作成するには、 AWS CLI または Amazon Data Firehose APIsを使用する必要があります。を使用して、この種のクロスアカウント設定 AWS マネジメントコンソール を作成することはできません。

## タグを使用したアクセスへのコントロール

IAM ポリシーでオプションの Condition 要素 (または Condition ブロック) を使用して、タグキーと値に基づいて Amazon Data Firehose オペレーションへのアクセスをファインチューニングできます。次のサブセクションでは、さまざまな Amazon Data Firehose オペレーションのためにこれを行う方法について説明します。Condition 要素とその中で使用できる演算子の使用の詳細については、「IAM JSON ポリシー要素: Condition」を参照してください。

### CreateDeliveryStream

CreateDeliveryStream オペレーションでは、aws:RequestTag 条件キーを使用します。次の例では、MyKey と MyValue はキー、およびタグの対応する値を表しています。詳細については、タグの基本を理解するを参照してください。

JSON

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

```
            "firehose:CreateDeliveryStream",
            "firehose:TagDeliveryStream"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/MyKey": "MyValue"
            }
        }
    }]
}
```

## TagDeliveryStream

TagDeliveryStream オペレーションでは、aws:TagKeys 条件キーを使用します。次の例では、MyKey はサンプルタグキーです。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "firehose:TagDeliveryStream",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "MyKey"
                 }
            }
        }
    ]
}
```

## UntagDeliveryStream

UntagDeliveryStream オペレーションでは、aws:TagKeys 条件キーを使用します。次の例では、MyKey はサンプルタグキーです。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "firehose:UntagDeliveryStream",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "MyKey"
                 }
            }
        }
    ]
}
```

## ListDeliveryStreams

`ListDeliveryStreams` でタグベースのアクセスコントロールを使用することはできません。

## その他のオペレーション

`CreateDeliveryStream`、`TagDeliveryStream`、`UntagDeliveryStream`、および `ListDeliveryStreams` 以外のすべての Firehose オペレーションで、`aws:RequestTag` 条件キーを使用します。次の例では、`MyKey` と `MyValue` はキー、およびタグの対応する値を表しています。

`ListDeliveryStreams` は、`firehose:ResourceTag` 条件キーを使用して、その Firehose ストリームのタグに基づいてアクセスをコントロールします。

次の例では、`MyKey` と `MyValue` はキー、およびタグの対応する値を表しています。このポリシーは、値が `MyValue` である `MyKey` というタグを持つ Data Firehose ストリームにのみ適用されます。リソースタグに基づくアクセスの制御の詳細については、IAM ユーザーガイドの「タグを使用した AWS リソースへのアクセスの制御」を参照してください。

# Amazon Data Firehose AWS Secrets Manager で を使用して認証する

Amazon Data Firehose は と統合 AWS Secrets Manager して、シークレットへの安全なアクセスを提供し、認証情報のローテーションを自動化します。この統合により、Firehose は実行時に Secrets Manager からシークレットを取得して、前述のストリーミングの宛先に接続し、データストリームを配信できます。これにより、 AWS マネジメントコンソール または API パラメータのストリーム作成ワークフロー中にシークレットがプレーンテキストで表示されません。シークレットを管理するための安全なプラクティスを提供し、パスワードローテーションを管理するためのカスタム Lambda 関数の設定など、複雑な認証情報管理アクティビティから解放します。

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

トピック

- シークレットを理解する
- シークレットを作成する
- シークレットを使用する
- シークレットをローテーションする

## シークレットを理解する

シークレットは、パスワード、ユーザーネームやパスワードなどの一連の認証情報、OAuth トークン、または、暗号化された形式で Secrets Manager に保存されるその他のシークレット情報にすることができます。

宛先ごとに、次のセクションに示すように、シークレット key-value ペアを正しい JSON 形式で指定する必要があります。Amazon Data Firehose は、シークレットが宛先に基づいて正しい JSON 形式を持っていない場合、宛先への接続に失敗します。

MySQL や PostgreSQL などのデータベースのシークレットの形式

```
{
    "username":  "<username>",
    "password":  "<password>"
}
```

Amazon Redshift プロビジョンドクラスターと Amazon Redshift Serverless ワークグループのシークレットの形式

```
{
    "username":  "<username>",
    "password":  "<password>"
}
```

Splunk のシークレットの形式

```
{
    "hec_token":  "<hec token>"
}
```

Snowflake のシークレットの形式

```
{
    "user":  "<snowflake-username>",
    "private_key":  "<snowflake-private-key>", // without the beginning and ending
 private key, remove all spaces and newlines
    "key_passphrase":  "<snowflake-private-key-passphrase>" // optional
}
```

HTTP エンドポイント、Coralogix、Datadog、Dynatrace、Elastic、Honeycomb、LogicMonitor、Logz.io、MongoDB Cloud、および New Relic のシークレットの形式

```
{
    "api_key":  "<apikey>"
}
```

## シークレットを作成する

シークレットを作成するには、「 AWS Secrets Manager ユーザーガイド」の「 AWS Secrets Manager シークレットを作成する」の手順に従います。

## シークレットを使用する

AWS Secrets Manager を使用して認証情報またはキーを保存し、Amazon Redshift、HTTP エンドポイン

ト、Snowflake、Splunk、Coralogix、Datadog、Dynatrace、Elastic、Honeycomb、LogicMonitor、Logz.io、Cloud、New Relic などのストリーミング送信先に接続することをお勧めします。

Firehose ストリームの作成時に、 AWS マネジメントコンソールを通じて Secrets Manager を使用して、これらの宛先のために認証を設定できます。詳細については、「宛先の設定を構成する」を参照してください。あるいは、CreateDeliveryStream および UpdateDestination API オペレーションを使用して Secrets Manager による認証を設定することもできます。

Firehose は暗号化を使用してシークレットをキャッシュし、宛先への各接続のためにそれらを使用します。最新の認証情報が使用されるように、10 分ごとにキャッシュを更新します。

ストリームのライフサイクル中はいつでも、Secrets Manager からシークレットを取得する機能をオフにすることを選択できます。シークレットを取得するために Secrets Manager を使用しない場合は、代わりにユーザー名/パスワードまたは API キーを使用できます。

> **Note**
>
> Firehose ではこの機能には追加コストはかかりませんが、Secrets Manager へのアクセスとメンテナンスについては課金されます。詳細については、AWS Secrets Manager の料金ページを参照してください。

## シークレットを取得するために Firehose へのアクセスを付与する

Firehose がシークレットを取得するには AWS Secrets Manager、シークレットにアクセスするために必要なアクセス許可と、シークレットを暗号化するキーを Firehose に提供する必要があります。

AWS Secrets Manager を使用してシークレットを保存および取得する場合、シークレットの保存場所と暗号化方法に応じて、いくつかの異なる設定オプションがあります。

- シークレットが IAM ロールと同じ AWS アカウントに保存されていて、デフォルトの AWS マネージドキー (aws/secretsmanager) で暗号化されている場合、Firehose が引き受ける IAM ロールはシークレットに対するsecretsmanager:GetSecretValueアクセス許可のみを必要とします。

```
// secret role policy
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "Secret ARN"
        }
    ]
}
```

IAM ポリシーの詳細については、「Permissions policy examples for AWS Secrets Manager」を参照してください。

- シークレットがロールと同じアカウントに保存されているが、カスタマーマネージドキー (CMK) で暗号化されている場合、ロールには `secretsmanager:GetSecretValue` 許可と `kms:Decrypt` 許可の両方が必要です。また、CMK ポリシーは、IAM ロールが `kms:Decrypt` を実行することを許可する必要があります。
- シークレットがロールとは異なる AWS アカウントに保存されていて、デフォルトの AWS マネージドキーで暗号化されている場合、シークレットが AWS マネージドキーで暗号化されている場合、Secrets Manager はクロスアカウントアクセスを許可しないため、この設定はできません。
- シークレットが別のアカウントに保存され、CMK を使用して暗号化されている場合、IAM ロールにはシークレットに対する `secretsmanager:GetSecretValue` 許可と CMK に対する `kms:Decrypt` 許可が必要です。シークレットのリソースポリシーと他のアカウントの CMK ポリシーも、IAM ロールに必要な許可を付与する必要があります。詳細については、「クロスアカウントアクセス」を参照してください。

### シークレットをローテーションする

ローテーションとは、シークレットを定期的に更新することです。指定したスケジュールでシークレットを自動的にローテーション AWS Secrets Manager するように を設定できます。そうすれば、長期のシークレットを短期のシークレットに置き換えることができます。これは、侵害のリスクを低減するのに役立ちます。詳細については、「 AWS Secrets Manager ユーザーガイド」の「シークレットのローテーション AWS Secrets Manager」を参照してください。

## Amazon Data Firehose コンソールを通じて IAM ロールを管理する

Amazon Data Firehose は、リアルタイムストリーミングデータを宛先に配信するフルマネージドサービスです。配信前にデータの形式を変換するように Firehose を設定することもできます。これらの機能を使用するには、まず Firehose ストリームを作成または編集する際に、Firehose に許可を付与する IAM ロールを指定する必要があります。Firehose は、Firehose ストリームに必要なすべての許可のために IAM ロールを使用します。

例えば、Amazon S3 にデータを配信する Firehose ストリームを作成し、この Firehose ストリームで AWS Lambda 機能を有効にした変換ソースレコードがあるとします。この場合、次に示すように、S3 バケットにアクセスして Lambda 関数を呼び出すための許可を Firehose に付与する IAM ロールを指定する必要があります。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "lambdaProcessing",
        "Effect": "Allow",
        "Action": ["lambda:InvokeFunction", "lambda:GetFunctionConfiguration"],
        "Resource": "arn:aws:lambda:us-east-1:123456789012:function:<lambda
 function name>:<lambda function version>"
    }, {
        "Sid": "s3Permissions",
        "Effect": "Allow",
        "Action": ["s3:AbortMultipartUpload", "s3:GetBucketLocation",
 "s3:GetObject", "s3:ListBucket", "s3:ListBucketMultipartUploads",
 "s3:PutObject"],
        "Resource": ["arn:aws:s3:::<bucket name>", "arn:aws:s3:::<bucket name>/
*"]
    }]
}
```

Firehose コンソールでは、これらのロールの指定方法を選択できます。次のオプションのいずれかから選択できます。

- 既存の IAM ロールを選択する
- コンソールから新しい IAM ロールを作成する

## 既存の IAM ロールを選択する

既存の IAM ロールから選択できます。このオプションでは、選択した IAM ロールに、ソースと宛先で必要な、適切な信頼ポリシーと許可があることを確認します。詳細については、「Amazon Data Firehose によるアクセスの制御」を参照してください。

# コンソールから新しい IAM ロールを作成する

あるいは、Firehose コンソールを使用して、ユーザーに代わって新しいロールを作成させることもできます。

Firehose がユーザーに代わって IAM ロールを作成する場合、そのロールには、Firehose ストリーム設定に基づいて必要な許可を付与するすべての許可と信頼ポリシーが自動的に含まれます。

例えば、[ AWS Lambdaを使用してソースレコードを変換] 機能を有効にしなかった場合、コンソールは許可ポリシーで次のステートメントを生成します。

```
{
  "Sid": "lambdaProcessing",
  "Effect": "Allow",
   "Action": [
     "lambda:InvokeFunction",
     "lambda:GetFunctionConfiguration"
   ],
   "Resource": "arn:aws:lambda:us-east-1123456789012:function:
%FIREHOSE_POLICY_TEMPLATE_PLACEHOLDER%"
}
```

> **Note**
>
> %FIREHOSE_POLICY_TEMPLATE_PLACEHOLDER% を含むポリシーステートメントはリソースに対する許可を付与しないため、無視してもかまいません。

コンソールが Firehose ストリームワークフローを作成および編集すると、信頼ポリシーも作成され、IAM ロールにアタッチされます。信頼ポリシーにより、Firehose は IAM ロールを引き受けることができます。信頼ポリシーの例を次に示します。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "firehoseAssume",
        "Effect": "Allow",
```

```
        "Principal": {
            "Service": "firehose.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
    }]
}
```

**Important**

- 複数の Firehose ストリームのために同じコンソールマネージド IAM ロールを使用しないようにしてください。そうしないと、IAM ロールが過度に許容されたり、エラーが発生したりする可能性があります。
- コンソールマネージド IAM ロールから許可ポリシー内で異なるポリシーステートメントを使用するには、独自の IAM ロールを作成し、新しいロールにアタッチされた許可ポリシーにそのポリシーステートメントをコピーします。ロールを Firehose ストリームにアタッチするには、[サービスアクセス] で [既存の IAM ロールを選択] オプションを選択します。
- コンソールは、ARN に文字列 service-role を含むすべての IAM ロールを管理します。既存の IAM ロールオプションを選択する際には、コンソールが変更を加えないように、ARN に service-role 文字列が含まれていない IAM ロールを必ず選択してください。

コンソールから IAM ロールを作成するステップ

1. Firehose コンソール (https://console.aws.amazon.com/firehose/) を開きます。
2. [Firehose ストリームを作成] を選択します。
3. ソースと宛先を選択します。詳細については、「チュートリアル: コンソールから Firehose ストリームを作成する」を参照してください。
4. 宛先の設定を選択します。詳細については、「宛先の設定を構成する」を参照してください。
5. [高度な設定] の [サービスアクセス] で、[IAM ロールを作成または更新] を選択します。

**Note**

これはデフォルトのオプションです。既存のロールを使用するには、[既存の IAM ロールを選択] オプションを選択します。Firehose コンソールは、独自のロールを変更しません。

6. [Firehose ストリームを作成] を選択します。

## コンソールから IAM ロールを編集する

Firehose ストリームを編集すると、Firehose はそれに応じて対応する許可ポリシーを更新し、設定と許可の変更を反映します。

例えば、Firehose ストリームを編集し、最新バージョンの Lambda 関数を exampleLambdaFunction として使用して [ AWS Lambdaを使用してソースレコードを変換] 機能を有効にすると、次のポリシーステートメントが許可ポリシーに含まれます。

```
{
  "Sid": "lambdaProcessing",
  "Effect": "Allow",
  "Action": [
    "lambda:InvokeFunction",
    "lambda:GetFunctionConfiguration"
  ],
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:exampleLambdaFunction:
$LATEST"
}
```

> **Important**
>
> コンソールマネージド IAM ロールは、自律的であるように設計されています。コンソールの外部で許可ポリシーまたは信頼ポリシーを変更することはお勧めしません。

### コンソールから IAM ロールを編集するステップ

1. Firehose コンソール (https://console.aws.amazon.com/firehose/) を開きます。
2. [Firehose ストリーム] を選択し、更新する Firehose ストリームの名前を選択します。
3. [設定] タブの [サーバーアクセス] セクションで、[編集] を選択します。
4. IAM ロールオプションを更新します。

> **Note**
>
> デフォルトでは、コンソールは常に、ARN に含まれるパターン service-role を使用して IAM ロールを更新します。既存の IAM ロールオプションを選択する際には、コンソール

が変更を加えないように、ARN に service-role 文字列が含まれていない IAM ロールを必ず選択してください。

5. [Save changes] (変更の保存) をクリックします。

# Amazon Data Firehose のコンプライアンスを理解する

サードパーティーの監査者は、複数のコンプライアンスプログラムの一環として Amazon Data Firehose のセキュリティと AWS コンプライアンスを評価します。これらのプログラムには、SOC、PCI、FedRAMP、HIPAA などがあります。

特定のコンプライアンスプログラムの対象となる AWS サービスのリストについては、AWS「コンプライアンスプログラムによる対象範囲内のサービス」を参照してください。一般的な情報については、「AWS コンプライアンスプログラム」を参照してください。

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、AWS「アーティファクトでのレポートのダウンロード」を参照してください。

Data Firehose を使用する際のお客様のコンプライアンス責任は、お客様のデータの機密性や貴社のコンプライアンス目的、適用可能な法律および規制によって決定されます。Data Firehose の使用が HIPAA、PCI、FedRAMP などの標準への準拠の対象である場合、 は以下に役立つリソース AWS を提供します。

- セキュリティとコンプライアンスのクイックスタートガイド – これらのデプロイガイドでは、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に重点を置いたベースライン環境をデプロイする手順について説明します AWS。
- 「Architecting for HIPAA Security and Compliance」ホワイトペーパー – このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。
- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- AWS Config – この AWS 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 Security Hub CSPM – この AWS サービスは、 内のセキュリティ状態を包括的に把握 AWS し、セキュリティ業界標準とベストプラクティスへの準拠を確認するのに役立ちます。

# Amazon Data Firehose の回復力

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。 AWS リージョンは、低レイテンシー、高スループット、高度に冗長なネットワークで接続された複数の物理的に分離されたアベイラビリティーゾーンを提供します。アベイラビリティーゾーンでは、アベイラビリティーゾーン間で中断せずに、自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、およびスケーラビリティが優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

Data Firehose は、 AWS グローバルインフラストラクチャに加えて、データの耐障害性とバックアップのニーズをサポートするのに役立ついくつかの機能を提供しています。

## ディザスタリカバリ

Amazon Data Firehose はサーバーレスモードで実行され、ホストのパフォーマンス低下、アベイラビリティーゾーンの可用性、および自動移行に伴うインフラストラクチャ関連の他の問題に対応します。問題が発生した場合でも、Amazon Data Firehose によって、Firehose ストリームがデータ損失なしで移行されます。

# Amazon Data Firehose のインフラストラクチャセキュリティを理解する

マネージドサービスである Amazon Data Firehose は、 AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して環境を AWS 設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

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

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。

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

> **Note**
>
> 送信 HTTPS リクエストの場合、Amazon Data Firehose は、宛先側でサポートされている最高レベルの TLS プロトコルバージョンを自動選択する HTTP ライブラリを使用します。

## AWS PrivateLink での Amazon Data Firehose の使用

インターフェイス VPC エンドポイント (AWS PrivateLink) を使用して、インターネットゲートウェイや NAT ゲートウェイを必要とせずに VPC から Amazon Data Firehose にアクセスできます。インターフェイス VPC エンドポイントには、インターネットゲートウェイ、NAT デバイス、VPN 接続、または Direct Connect 接続は必要ありません。インターフェイス VPC エンドポイントは、Amazon VPC 内のプライベート IPs 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、「Amazon Virtual Private Cloud」を参照してください。

### Firehose でのインターフェイス VPC エンドポイント (AWS PrivateLink) の使用

開始するには、Amazon VPC リソースからの Amazon Data Firehose のトラフィックがインターフェイス VPC エンドポイントを経由して流れるように、インターフェイス VPC エンドポイントを作成するだけです。エンドポイントを作成するときは、Amazon Data Firehose へのアクセスを制御するエンドポイントポリシーをエンドポイントにアタッチします。ポリシーを使用して VPC エンドポイントから Amazon Data Firehose へのアクセスを制御する方法については、「VPC エンドポイントを用いたサービスへのアクセスの制御」を参照してください。

次の例は、VPC で AWS Lambda 関数をセットアップし、VPC エンドポイントを作成して、関数が Amazon Data Firehose サービスと安全に通信できるようにする方法を示しています。この例では、Lambda 関数に現在のリージョンの Firehose ストリームのリスト表示を許可するが、いずれの Firehose ストリームについても説明は許可しないポリシーを使用します。

VPC エンドポイントの作成

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/vpc/ で Amazon VPC コンソールを開きます。

2. VPC ダッシュボードで、[エンドポイント] を選択します。
3. エンドポイントの作成 を選択します。
4. サービス名のリストで、[com.amazonaws.*your_region*.kinesis-firehose] を選択します。
5. エンドポイントを作成する VPC および 1 つ以上のサブネットを選択します。
6. エンドポイントに関連付ける 1 つ以上のセキュリティグループを選択します。
7. [ポリシー] で [カスタム] を選択し、次のポリシーを貼り付けます。

```
{
    "Statement": [
        {
            "Sid": "Allow-only-specific-PrivateAPIs",
            "Principal": "*",
            "Action": [
                "firehose:ListDeliveryStreams"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "Allow-only-specific-PrivateAPIs",
            "Principal": "*",
            "Action": [
                "firehose:DescribeDeliveryStream"
            ],
            "Effect": "Deny",
            "Resource": [
                "*"
            ]
        }
    ]
}
```

8. エンドポイントの作成 を選択します。

Lambda 関数で使用する IAM ロールを作成します。

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

2. 左側のペインから、[Roles (ロール)] を選択してから、[Create role (ロールの作成)] をクリックします。
3. [Select type of trusted entity (信頼されたエンティティのタイプの選択)] で、デフォルトの選択である [AWS のサービス] をそのままにします。
4. [このロールを使用するサービスを選択] の下で、[Lambda] を選択します。
5. [Next: Permissions] (次のステップ: 許可) を選択します。
6. ポリシーのリストで、`AWS LambdaVPCAccessExecutionRole` および `AmazonDataFirehoseReadOnlyAccess` という名前の 2 つのポリシーを探して追加します。

   > **Important**
   >
   > 以下に例を示します。本番環境では、より厳格なポリシーが必要になる場合があります。

7. [Next: Tags] (次へ: タグ) を選択します。この演習の目的を達成するにはタグは不要です。[次へ: レビュー] を選択します。
8. ロールの名前を入力し、[ロールの作成] を選択します。

VPC 内に Lambda 関数を作成します。

1. https://console.aws.amazon.com/lambda/ で AWS Lambda コンソールを開きます。
2. 関数の作成 を選択します。
3. Author from scratch (製作者を最初から) を選択します。
4. 関数の名前を入力し、[ランタイム]を `Python 3.9` 以上に設定します。
5. [Permissions (アクセス許可)] で、[実行ロールの選択または作成] を選択します。
6. [実行ロール] リストから [既存のロールを使用する] を選択します。
7. [既存のロール] リストで、上記で作成したロールを選択します。
8. [関数の作成] を選択してください。
9. [関数コード] に次のコードを貼り付けます。

```
import json
import boto3
import os
from botocore.exceptions import ClientError
```

```
   def lambda_handler(event, context):
       REGION = os.environ['AWS_REGION']
       client = boto3.client(
           'firehose',
           REGION

       )
       print("Calling list_delivery_streams with ListDeliveryStreams allowed
policy.")
       delivery_stream_request = client.list_delivery_streams()
       print("Successfully returned list_delivery_streams request %s." % (
           delivery_stream_request
       ))
       describe_access_denied = False
       try:
           print("Calling describe_delivery_stream with DescribeDeliveryStream
denied policy.")
           delivery_stream_info =
client.describe_delivery_stream(DeliveryStreamName='test-describe-denied')
       except ClientError as e:
           error_code = e.response['Error']['Code']
           print ("Caught %s." % (error_code))
           if error_code == 'AccessDeniedException':
               describe_access_denied = True

       if not describe_access_denied:
           raise
       else:
           print("Access denied test succeeded.")
```

10. [基本設定] で、タイムアウトを 1 分に設定します。
11. [ネットワーク] で、上記でエンドポイントを作成した VPC を選択し、作成時にエンドポイントに関連付けたサブネットとセキュリティグループを選択します。
12. ページの上部付近にある [Save (保存)] を選択します。
13. [テスト] を選択します。
14. イベント名を入力し、[Create (作成)] を選択します。
15. [テスト] を再度選択します。これにより、関数が実行されます。実行結果が表示されたら [詳細] を展開し、ログ出力を関数コードと比較します。成功した結果では、リージョンの Firehose ストリームのリストと、次のような出力が表示されます。

```
Calling describe_delivery_stream.
```

```
    AccessDeniedException

    Access denied test succeeded.
```

## サポートされている AWS リージョン

現在、インターフェイス VPC エンドポイントは次のリージョン内でサポートされています。

- 米国東部(オハイオ)
- 米国東部 (バージニア北部)
- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (ソウル)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (タイ)
- アジアパシフィック (東京)
- アジアパシフィック (香港)
- カナダ (中部)
- カナダ西部 (カルガリー)
- 中国 (北京)
- 中国 (寧夏)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (パリ)
- メキシコ (中部)
- 南米 (サンパウロ)
- AWS GovCloud (米国東部)
- AWS GovCloud (米国西部)

- 欧州 (スペイン)
- 中東 (アラブ首長国連邦)
- 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)
- イスラエル (テルアビブ)
- アジアパシフィック (マレーシア)

# Amazon Data Firehose のセキュリティのベストプラクティスを実装する

Amazon Data Firehose には、独自のセキュリティポリシーを策定および実装する際に考慮すべきさまざまなセキュリティ機能が用意されています。以下のベストプラクティスは一般的なガイドラインであり、完全なセキュリティソリューションを説明するものではありません。これらのベストプラクティスはお客様の環境に適切ではないか、十分ではない場合があるため、これらは指示ではなく、有用な考慮事項と見なしてください。

## 最小特権アクセスの実装

許可を付与する場合、どのユーザーにどの Amazon Data Firehose リソースに対する許可を付与するかは、お客様が決定します。これらのリソースで許可したい特定のアクションを有効にするのも、お客様になります。このため、タスクの実行に必要なアクセス許可のみを付与する必要があります。最小特権アクセスの実装は、セキュリティリスクと、エラーや悪意によってもたらされる可能性のある影響の低減における基本になります。

## IAM ロールの使用

プロデューサーおよびクライアントアプリケーションには、Firehose ストリームにアクセスするために有効な認証情報が必要です。また、Firehose ストリームには、宛先にアクセスするための有効な認証情報が必要です。 AWS 認証情報は、クライアントアプリケーションや Amazon S3 バケットに直接保存しないでください。これらは自動的にローテーションされない長期的な認証情報であり、漏洩するとビジネスに大きな影響が及ぶ場合があります。

代わりに、IAM ロールを使用して、Firehose ストリームにアクセスするためのプロデューサーおよびクライアントアプリケーションの一時的な認証情報を管理してください。ロールを使用するときは、他のリソースにアクセスするために長期的な認証情報 (ユーザー名とパスワード、またはアクセスキーなど) を使用する必要がありません。

詳細については、「IAM ユーザーガイド」にある下記のトピックを参照してください。

- IAM ロール
- ロールの一般的なシナリオ: ユーザー、アプリケーション、およびサービス

## 依存リソースでのサーバー側の暗号化の実装

保管中のデータと転送中のデータは Amazon Data Firehose で暗号化できます。詳細については、「Amazon Data Firehose のデータ保護」を参照してください。

## CloudTrail を使用して API コールをモニタリングする

Amazon Data Firehose は AWS CloudTrail、Amazon Data Firehose のユーザー、ロール、または のサービスによって実行されたアクションを記録する AWS サービスである と統合されています。

CloudTrail により収集された情報を使用して、Amazon Data Firehose に対して行われたリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエストが行われた日時、および追加の詳細を特定することができます。

詳細については、「the section called “Firehose API コールをログ記録する”」を参照してください。

# Amazon Data Firehose をモニタリングする

次の機能を使用して Amazon Data Firehose をモニタリングできます。

トピック



## CloudWatch アラームを使用したベストプラクティスを実装する

次のメトリクスがバッファリングの制限 (最大 15 分) を超えたときの CloudWatch アラームを追加します。

- `DeliveryToS3.DataFreshness`
- `DeliveryToIceberg.DataFreshness`
- `DeliveryToSplunk.DataFreshness`
- `DeliveryToAmazonOpenSearchService.DataFreshness`
- `DeliveryToAmazonOpenSearchServerless.DataFreshness`
- `DeliveryToHttpEndpoint.DataFreshness`

また、次のメトリクス数式に基づいてアラームを作成します。

- `IncomingBytes (Sum per 5 Minutes) / 300` が `BytesPerSecondLimit` の割合に近づく。
- `IncomingRecords (Sum per 5 Minutes) / 300` が `RecordsPerSecondLimit` の割合に近づく。
- `IncomingPutRequests (Sum per 5 Minutes) / 300` が `PutRequestsPerSecondLimit` の割合に近づく。

アラームを推奨するもう 1 つのメトリクスは ThrottledRecords です。

アラームが ALARM 状態になった場合のトラブルシューティングについては、「エラーのトラブルシューティング」を参照してください。

# CloudWatch メトリクスを使用して Amazon Data Firehose をモニタリングする

> **Important**
>
> エラーをタイミングよく特定できるように、送信先に属するすべての CloudWatch メトリクスでアラームを有効にします。

Amazon Data Firehose は Amazon CloudWatch メトリクスと統合されているため、Firehose ストリームの CloudWatch メトリクスを収集、表示、および分析できます。例えば、IncomingBytes および IncomingRecords メトリクスをモニタリングして、データプロデューサーから Amazon Data Firehose に取り込まれるデータを追跡できます。

Amazon Data Firehose は CloudWatch メトリクスを 1 分ごとに収集して公開します。ただし、受信データのバーストが数秒間しか続かない場合は、1 分ごとの収集ではメトリクスが完全にキャプチャされないか表示されない可能性があります。これは、CloudWatch メトリクスが Amazon Data Firehose から 1 分間隔で集約されるためです。

Firehose ストリーム用に収集されたメトリクスは無料です。Kinesis エージェントのメトリクスの詳細については、「Kinesis エージェントの状態をモニタリングする」を参照してください。

トピック

- 動的パーティショニングの CloudWatch メトリクス
- データ配信の CloudWatch メトリクス
- データ取り込みメトリクス
- API レベルの CloudWatch メトリクス
- データ変換 CloudWatch メトリクス
- CloudWatch Logs の解凍メトリクス
- 形式変換 CloudWatch メトリクス

- サーバー側の暗号化 (SSE) CloudWatch のメトリクス
- Amazon Data Firehose のディメンション
- Amazon Data Firehose の使用状況メトリクス

## 動的パーティショニングの CloudWatch メトリクス

動的パーティショニングが有効になっている場合、 AWS/Firehose 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 |
| --- | --- |
| `ActivePartitionsLimit` | エラーバケットにデータを送信する前に Firehose ストリームが処理するアクティブなパーティションの最大数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| `PartitionCount` | 処理されているパーティションの数、つまりアクティブなパーティション数。この数は、1 からパーティション数の制限である 500 (デフォルト) の間で変化します。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| `PartitionCountExceeded` | このメトリクスは、パーティション数の制限を超えているかどうかを示します。制限に違反したかどうかに基づいて 1 または 0 を出力します。 |
| `JQProcessing.Duration` | JQ Lambda 関数で JQ 式を実行するのにかかった時間を返します。<br><br>単位: ミリ秒 |
| `PerPartitionThroughput` | パーティションごとに処理されているスループットを示します。このメトリクスを使用すると、パーティションごとのスループットをモニタリングできます。 |

| メトリクス | 説明 |
| --- | --- |
| | 単位: StandardUnit.BytesSecond |
| DeliveryToS3.ObjectCount | S3 バケットに配信されるオブジェクトの数を示します。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |

## データ配信の CloudWatch メトリクス

AWS/Firehose 名前空間には、次のサービスレベルメトリクスが含まれます。BackupToS3.Success、DeliveryToS3.Success、DeliveryToSplunk.Success、DeliveryTo または DeliveryToRedshift.Success の平均値がわずかに下がっても、データ損失を示しているわけではありません。Amazon Data Firehose は、配信エラーを再試行し、設定された宛先またはバックアップ S3 バケットのいずれかにレコードが正常に配信されるまで次の処理に移行しません。

トピック

- OpenSearch サービスへの配信
- OpenSearch Serverless への配信
- Amazon Redshift への配信
- Amazon S3 への配信
- Snowflake への配信
- Splunk への配信
- HTTP エンドポイントへの配信

### OpenSearch サービスへの配信

| メトリクス | 説明 |
| --- | --- |
| DeliveryToAmazonOpenSearchService.Bytes | 指定された期間に OpenSearch Service にインデックス作成されたバイト数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples |

| メトリクス | 説明 |
| --- | --- |
| | 単位: バイト |
| DeliveryToAmazonOpenSearchService.DataFreshness | Amazon Data Firehose の最も古いレコードの経過時間 (Amazon Data Firehose に入ってから現在まで)。この経過時間より古いレコードはすべて OpenSearch Service に配信済みです。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: 秒 |
| DeliveryToAmazonOpenSearchService.Records | 指定された期間に OpenSearch Service にインデックス作成されたレコード数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToAmazonOpenSearchService.Success | 正常にインデックスが作成されたレコードの合計。 |
| DeliveryToS3.Bytes | 指定された期間に Amazon S3 に配信されたバイト数。Amazon Data Firehose は、すべてのドキュメントのバックアップを有効にした場合にのみこのメトリクスを出力します。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToS3.DataFreshness | Amazon Data Firehose の最も古いレコードの経過時間 (Amazon Data Firehose に入ってから現在まで)。この経過時間より古いレコードは S3 バケットに配信済みです。Amazon Data Firehose は、すべてのドキュメントのバックアップを有効にした場合にのみこのメトリクスを出力します。<br><br>単位: 秒 |

| メトリクス | 説明 |
| --- | --- |
| DeliveryToS3.Records | 指定された期間に Amazon S3 に配信されたレコード数。Amazon Data Firehose は、すべてのドキュメントのバックアップを有効にした場合にのみこのメトリクスを出力します。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToS3.Success | 成功した Amazon S3 の put コマンドの合計。Amazon Data Firehose は、バックアップが失敗したドキュメントのためにのみ有効か、すべてのドキュメントのために有効かにかかわらず、常にこのメトリクスを出力します。 |
| DeliveryToAmazonOpenSearchService.AuthFailure | 認証/認可のエラー。OS/ES クラスターポリシーとロールのアクセス許可を確認します。<br><br>0 は問題がないことを示します。1 は認証に失敗したことを示します。 |
| DeliveryToAmazonOpenSearchService.DeliveryRejected | 配信拒否エラー。OS/ES クラスターポリシーとロールのアクセス許可を確認します。<br><br>0 は問題がないことを示します。1 は配信に失敗したことを示します。 |

## OpenSearch Serverless への配信

| メトリクス | 説明 |
| --- | --- |
| DeliveryToAmazonOpenSearchServerless.Bytes | 指定された期間に OpenSearch Serverless にインデックス作成されたバイト数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |

| メトリクス | 説明 |
| --- | --- |
| DeliveryToAmazonOpenSearchServerless.DataFreshness | Amazon Data Firehose の最も古いレコードの経過時間 (Amazon Data Firehose に入ってから現在まで)。この経過時間より古いレコードはすべて OpenSearch Serverless に配信済みです。<br><br>単位: 秒 |
| DeliveryToAmazonOpenSearchServerless.Records | 指定された期間に OpenSearch Serverless にインデックス作成されたレコード数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToAmazonOpenSearchServerless.Success | 正常にインデックスが作成されたレコードの合計。 |
| DeliveryToS3.Bytes | 指定された期間に Amazon S3 に配信されたバイト数。Amazon Data Firehose は、すべてのドキュメントのバックアップを有効にした場合にのみこのメトリクスを出力します。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToS3.DataFreshness | Amazon Data Firehose の最も古いレコードの経過時間 (Amazon Data Firehose に入ってから現在まで)。この経過時間より古いレコードは S3 バケットに配信済みです。Amazon Data Firehose は、すべてのドキュメントのバックアップを有効にした場合にのみこのメトリクスを出力します。<br><br>単位: 秒 |

| メトリクス | 説明 |
| --- | --- |
| DeliveryToS3.Records | 指定された期間に Amazon S3 に配信されたレコード数。Amazon Data Firehose は、すべてのドキュメントのバックアップを有効にした場合にのみこのメトリクスを出力します。<br><br>単位: カウント |
| DeliveryToS3.Success | 成功した Amazon S3 の put コマンドの合計。Amazon Data Firehose は、バックアップが失敗したドキュメントのためにのみ有効か、すべてのドキュメントのために有効かにかかわらず、常にこのメトリクスを出力します。 |
| DeliveryToAmazonOpenSearchServerless.AuthFailure | 認証/認可のエラー。OS/ES クラスターポリシーとロールのアクセス許可を確認します。<br><br>0 は問題がないことを示します。1 は認証の失敗があることを示します。 |
| DeliveryToAmazonOpenSearchServerless.DeliveryRejected | 配信拒否エラー。OS/ES クラスターポリシーとロールのアクセス許可を確認します。<br><br>0 は問題がないことを示します。1 は配信の失敗があることを示します。 |

## Amazon Redshift への配信

| メトリクス | 説明 |
| --- | --- |
| DeliveryToRedshift.Bytes | 指定された期間に Amazon Redshift にコピーされたバイト数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToRedshift.Records | 指定された期間に Amazon Redshift にコピーされたレコード数。 |

| メトリクス | 説明 |
| --- | --- |
| | 統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToRedshift.Success | 成功した Amazon Redshift の COPY コマンドの合計。 |
| DeliveryToS3.Bytes | 指定された期間に Amazon S3 に配信されたバイト数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |
| DeliveryToS3.DataFreshness | Amazon Data Firehose の最も古いレコードの経過時間 (Amazon Data Firehose に入ってから現在まで)。この経過時間より古いレコードはすべて S3 バケットに配信されます。<br><br>単位: 秒 |
| DeliveryToS3.Records | 指定された期間に Amazon S3 に配信されたレコード数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToS3.Success | 成功した Amazon S3 の put コマンドの合計。 |
| DeliveryToRedshift.DataFreshness | Amazon Data Firehose の最も古いレコードの経過時間 (Amazon Data Firehose に入ってから現在まで)。この経過時間より古いレコードはすべて Amazon Redshift クラスターに配信されます。 |

| メトリクス | 説明 |
| --- | --- |
| BackupToS3.Bytes | 指定された期間にバックアップのために Amazon S3 に配信されたバイト数。Amazon Data Firehose は、Amazon S3 へのバックアップが有効になっているときに、このメトリクスを出力します。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| BackupToS3.DataFreshness | Amazon Data Firehose の最も古いレコードの経過時間 (Amazon Data Firehose に入ってから現在まで)。この経過時間より古いレコードはバックアップのために Amazon S3 バケットに配信済みです。Amazon Data Firehose は、Amazon S3 へのバックアップが有効になっているときに、このメトリクスを出力します。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: 秒 |
| BackupToS3.Records | 指定された期間にバックアップのために Amazon S3 に配信されたレコード数。Amazon Data Firehose は、Amazon S3 へのバックアップが有効になっているときに、このメトリクスを出力します。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| BackupToS3.Success | 成功した Amazon S3 の put コマンド (バックアップ用) の合計。Amazon Data Firehose は、Amazon S3 へのバックアップが有効になっているときに、このメトリクスを出力します。 |

## Amazon S3 への配信

次の表のメトリクスは、Firehose ストリームのメイン宛先が Amazon S3 の場合に、その配信に関連するものです。

| メトリクス | 説明 |
| --- | --- |
| DeliveryToS3.Bytes | 指定された期間に Amazon S3 に配信されたバイト数。データ変換を有効にすると、このメトリクスは変換前に前処理されたバイトサイズを反映します。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |
| DeliveryToS3.DataFreshness | Amazon Data Firehose の最も古いレコードの経過時間 (Amazon Data Firehose に入ってから現在まで)。この経過時間より古いレコードは S3 バケットに配信済みです。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: 秒 |
| DeliveryToS3.Records | 指定された期間に Amazon S3 に配信されたレコード数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToS3.Success | 成功した Amazon S3 の put コマンドの合計。 |
| BackupToS3.Bytes | 指定された期間にバックアップのために Amazon S3 に配信されたバイト数。Amazon Data Firehose は、バックアップが有効になっているときに、このメトリクスを出力します (データ変換も有効になっている場合にのみ可能)。<br><br>統計: Minimum、Maximum、Average、Sum、Samples |

| メトリクス | 説明 |
| --- | --- |
| | 単位: カウント |
| BackupToS3.DataFreshness | Amazon Data Firehose の最も古いレコードの経過時間 (Amazon Data Firehose に入ってから現在まで)。この経過時間より古いレコードはバックアップのために Amazon S3 バケットに配信済みです。Amazon Data Firehose は、バックアップが有効になっているときに、このメトリクスを出力します (データ変換も有効になっている場合にのみ可能)。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: 秒 |
| BackupToS3.Records | 指定された期間にバックアップのために Amazon S3 に配信されたレコード数。Amazon Data Firehose は、バックアップが有効になっているときに、このメトリクスを出力します (データ変換も有効になっている場合にのみ可能)。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| BackupToS3.Success | 成功した Amazon S3 の put コマンド (バックアップ用) の合計。Amazon Data Firehose は、バックアップが有効になっているときに、このメトリクスを出力します (データ変換も有効になっている場合にのみ可能)。 |

## Snowflake への配信

| メトリクス | 説明 |
| --- | --- |
| DeliveryToSnowflake.Bytes | 指定された期間に Snowflake に配信されたバイト数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples |

| メトリクス | 説明 |
| --- | --- |
| | 単位: バイト |
| DeliveryToSnowflake.DataFreshness | Firehose の最も古いレコードの経過時間 (Firehose に入ってから現在まで)。この経過時間より古いレコードはすべて Snowflake に配信済みです。Firehose 挿入呼び出しが成功した後、Snowflake にデータをコミットするまでに数秒かかる場合があることに留意してください。Snowflake にデータをコミットするのにかかる時間については、DeliveryToSnowflake.DataCommitLatency メトリクスを参照してください。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: 秒 |
| DeliveryToSnowflake.DataCommitLatency | Firehose がレコードを正常に挿入した後、データが Snowflake にコミットされるまでにかかる時間。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: 秒 |
| DeliveryToSnowflake.Records | 指定された期間に Snowflake に配信されたレコード数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToSnowflake.Success | Snowflake に対して行われた正常な挿入呼び出しの合計。 |
| DeliveryToS3.Bytes | 指定された期間に Amazon S3 に配信されたバイト数。このメトリクスは、Snowflake への配信が失敗し、失敗したデータの S3 へのバックアップを Firehose が試行する場合にのみ使用できます。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |

| メトリクス | 説明 |
| --- | --- |
| DeliveryToS3.Records | 指定された期間に Amazon S3 に配信されたレコード数。このメトリクスは、Snowflake への配信が失敗し、失敗したデータの S3 へのバックアップを Firehose が試行する場合にのみ使用できます。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToS3.Success | 成功した Amazon S3 の put コマンドの合計。このメトリクスは、Snowflake への配信が失敗し、失敗したデータの S3 へのバックアップを Firehose が試行する場合にのみ使用できます。 |
| BackupToS3.DataFreshness | Firehose 内の最も古いレコードが作成されてから経過した期間 (Firehose に入ってから現在まで)。作成されてから経過した期間がこれよりも長いレコードはすべて Amazon S3 バケットにバックアップされます。このメトリクスは、すべてのデータをバックアップするように Firehose ストリームが設定されている場合に使用できます。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: 秒 |
| BackupToS3.Records | 指定された期間にバックアップのために Amazon S3 に配信されたレコード数。このメトリクスは、すべてのデータをバックアップするように Firehose ストリームが設定されている場合に使用できます。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |

| メトリクス | 説明 |
| --- | --- |
| BackupToS3.Bytes | 指定された期間にバックアップのために Amazon S3 に配信されたバイト数。このメトリクスは、すべてのデータをバックアップするように Firehose ストリームが設定されている場合に使用できます。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| BackupToS3.Success | 成功した Amazon S3 の put コマンド (バックアップ用) の合計。Firehose は、Firehose ストリームがすべてのデータをバックアップするように設定されている場合に、このメトリクスを出力します。 |

## Splunk への配信

| メトリクス | 説明 |
| --- | --- |
| DeliveryToSplunk.Bytes | 指定された期間に Splunk に配信されたバイト数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |
| DeliveryToSplunk.DataAckLatency | Amazon Data Firehose がデータを送信した後、Splunk から確認応答を受信するまでの概算時間。このメトリクスの増加または減少傾向は、絶対概算値よりも有用です。傾向が増加すると、Splunk インデクサからのインデックス作成および送達確認の速度が遅くなる可能性があります。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: 秒 |

| メトリクス | 説明 |
| --- | --- |
| DeliveryToSplunk.DataFreshness | Amazon Data Firehose の最も古いレコードの経過時間 (Amazon Data Firehose に入ってから現在まで)。この経過時間より古いレコードは Splunk に配信済みです。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: 秒 |
| DeliveryToSplunk.Records | 指定された期間に Splunk に配信されたレコード数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToSplunk.Success | 正常にインデックスが作成されたレコードの合計。 |
| DeliveryToS3.Success | 成功した Amazon S3 の put コマンドの合計。このメトリクスは、Amazon S3 へのバックアップが有効な場合に出力されます。 |
| BackupToS3.Bytes | 指定された期間にバックアップのために Amazon S3 に配信されたバイト数。Amazon Data Firehose は、Firehose ストリームがすべてのドキュメントをバックアップするように設定されている場合に、このメトリクスを出力します。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |

| メトリクス | 説明 |
| --- | --- |
| BackupToS3.DataFreshness | Amazon Data Firehose の最も古いレコードの経過時間 (Amazon Data Firehose に入ってから現在まで)。この経過時間より古いレコードはバックアップのために Amazon S3 バケットに配信済みです。Amazon Data Firehose は、Firehose ストリームがすべてのドキュメントをバックアップするように設定されている場合に、このメトリクスを出力します。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: 秒 |
| BackupToS3.Records | 指定された期間にバックアップのために Amazon S3 に配信されたレコード数。Amazon Data Firehose は、Firehose ストリームがすべてのドキュメントをバックアップするように設定されている場合に、このメトリクスを出力します。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| BackupToS3.Success | 成功した Amazon S3 の put コマンド (バックアップ用) の合計。Amazon Data Firehose は、Firehose ストリームがすべてのドキュメントをバックアップするように設定されている場合に、このメトリクスを出力します。 |

## HTTP エンドポイントへの配信

| メトリクス | 説明 |
| --- | --- |
| DeliveryToHttpEndpoint.Bytes | HTTP エンドポイントに正常に配信されたバイト数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |

| メトリクス | 説明 |
| --- | --- |
| DeliveryToHttpEndpoint.Records | HTTP エンドポイントに正常に配信されたレコード数。このメトリクスは、成功した配信試行に対してのみ出力され、配信試行が失敗したときに出力されません。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToHttpEndpoint.DataFreshness | Amazon Data Firehose で最も古い記録の経過期間。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: 秒 |
| DeliveryToHttpEndpoint.Success | 配信試行ごとに HTTP エンドポイントに正常に配信されたレコードの数。とは異なりDeliveryToHttpEndpoint.Records 、このメトリクスは配信の試行ごとに出力されます。成功すると、 値は配信試行のレコード数と等しくなります。配信試行のすべてのレコードが失敗した場合、値は 0 です。Minimum 統計を使用して、配信の失敗をモニタリングします。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToHttpEndpoint.ProcessedBytes | 再試行を含む、試行された処理済みのバイト数。 |
| DeliveryToHttpEndpoint.ProcessedRecords | 再試行を含む試行されたレコードの数。 |

## データ取り込みメトリクス

トピック

- Kinesis Data Streams によるデータ取り込み
- Direct PUT によるデータ取り込み

- MSK からのデータの取り込み

## Kinesis Data Streams によるデータ取り込み

| メトリクス | 説明 |
| --- | --- |
| DataReadFromKinesisStream.Bytes | データソースが Kinesis データストリームである場合、このメトリクスは、そのデータストリームから読み取られたバイト数を示します。この数には、フェイルオーバーによる再読み取りが含まれます。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |
| DataReadFromKinesisStream.Records | データソースが Kinesis データストリームである場合、このメトリクスは、そのデータストリームから読み取ったレコード数を示します。この数には、フェイルオーバーによる再読み取りが含まれます。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| ThrottledDescribeStream | データソースが Kinesis データストリームである場合に DescribeStream オペレーションが調整される合計回数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| ThrottledGetRecords | データソースが Kinesis データストリームである場合に GetRecords オペレーションが調整される合計回数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |

| メトリクス | 説明 |
| --- | --- |
| ThrottledGetShardIterator | データソースが Kinesis データストリームである場合に GetShardIterator オペレーションが調整される合計回数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| KinesisMillisBehindLatest | データソースが Kinesis データストリームである場合、このメトリクスは、Kinesis データストリームで最後の読み取りレコードが最新のレコードから遅れている時間 (ミリ秒単位) を示します。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: ミリ秒 |

## Direct PUT によるデータ取り込み

| メトリクス | 説明 |
| --- | --- |
| BackupToS3.Bytes | 指定された期間にバックアップのために Amazon S3 に配信されたバイト数。Amazon S3 または Amazon Redshift の宛先のためにデータ変換が有効になっているときに、Amazon Data Firehose は、このメトリクスを出力します。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |
| BackupToS3.DataFreshness | Amazon Data Firehose の最も古いレコードの経過時間 (Amazon Data Firehose に入ってから現在まで)。この経過時間より古いレコードはバックアップのために Amazon S3 バケットに配信済みです。Amazon S3 または Amazon Redshift の宛先のためにデータ変換が有効に |

| メトリクス | 説明 |
| --- | --- |
| | なっているときに、Amazon Data Firehose は、このメトリクスを出力します。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: 秒 |
| BackupToS3.Records | 指定された期間にバックアップのために Amazon S3 に配信されたレコード数。Amazon S3 または Amazon Redshift の宛先のためにデータ変換が有効になっているときに、Amazon Data Firehose は、このメトリクスを出力します。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| BackupToS3.Success | 成功した Amazon S3 の put コマンド (バックアップ用) の合計。Amazon S3 または Amazon Redshift の宛先のためにデータ変換が有効になっているときに、Amazon Data Firehose は、このメトリクスを出力します。 |
| BytesPerSecondLimit | Firehose ストリームがスロットリング前に取り込むことのできる 1 秒あたりの現在の最大バイト数。この制限の引き上げをリクエストするには、AWS サポートセンターに移動し、[ケースの作成]、[サービスの制限緩和] の順に選択します。 |
| DeliveryToAmazonOpenSearchService.Bytes | 指定された期間に OpenSearch Service にインデックス作成されたバイト数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |

| メトリクス | 説明 |
| --- | --- |
| DeliveryToAmazonOpenSearchService.DataFreshness | Amazon Data Firehose の最も古いレコードの経過時間 (Amazon Data Firehose に入ってから現在まで)。この経過時間より古いレコードはすべて OpenSearch Service に配信済みです。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: 秒 |
| DeliveryToAmazonOpenSearchService.Records | 指定された期間に OpenSearch Service にインデックス作成されたレコード数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToAmazonOpenSearchService.Success | 正常にインデックスが作成されたレコードの合計。 |
| DeliveryToRedshift.Bytes | 指定された期間に Amazon Redshift にコピーされたバイト数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |
| DeliveryToRedshift.Records | 指定された期間に Amazon Redshift にコピーされたレコード数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToRedshift.Success | 成功した Amazon Redshift の COPY コマンドの合計。 |

| メトリクス | 説明 |
| --- | --- |
| DeliveryToS3.Bytes | 指定された期間に Amazon S3 に配信されたバイト数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |
| DeliveryToS3.DataFreshness | Amazon Data Firehose の最も古いレコードの経過時間 (Amazon Data Firehose に入ってから現在まで)。この経過時間より古いレコードは S3 バケットに配信済みです。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: 秒 |
| DeliveryToS3.Records | 指定された期間に Amazon S3 に配信されたレコード数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToS3.Success | 成功した Amazon S3 の put コマンドの合計。 |
| DeliveryToSplunk.Bytes | 指定された期間に Splunk に配信されたバイト数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |

| メトリクス | 説明 |
| --- | --- |
| DeliveryToSplunk.DataAckLatency | Amazon Data Firehose がデータを送信した後、Splunk から確認応答を受信するまでの概算時間。このメトリクスの増加または減少傾向は、絶対概算値よりも有用です。傾向が増加すると、Splunk インデクサからのインデックス作成および送達確認の速度が遅くなる可能性があります。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: 秒 |
| DeliveryToSplunk.DataFreshness | Amazon Data Firehose の最も古いレコードの経過時間 (Amazon Data Firehose に入ってから現在まで)。この経過時間より古いレコードは Splunk に配信済みです。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: 秒 |
| DeliveryToSplunk.Records | 指定された期間に Splunk に配信されたレコード数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| DeliveryToSplunk.Success | 正常にインデックスが作成されたレコードの合計。 |
| IncomingBytes | 指定された期間に Firehose ストリームに正常に取り込まれたバイト数。データインジェストは、Firehose ストリーム制限のいずれかを超えるとスロットリングされる可能性があります。スロットリングされたデータは IncomingBytes にカウントされません。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |

| メトリクス | 説明 |
| --- | --- |
| IncomingPutRequests | 指定した期間に成功した PutRecord リクエストと PutRecordBatch リクエストの数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| IncomingRecords | 指定された期間に Firehose ストリームに正常に取り込まれたレコード数。データインジェストは、Firehose ストリーム制限のいずれかを超えるとスロットリングされる可能性があります。スロットリングされたデータは IncomingRecords にカウントされません。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| RecordsPerSecondLimit | Firehose ストリームがスロットリング前に取り込むことのできる 1 秒あたりの現在の最大レコード数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| ThrottledRecords | データ取り込みが Firehose ストリームの制限の 1 つを超えたために調整されたレコードの数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |

## MSK からのデータの取り込み

| メトリクス | 説明 |
| --- | --- |
| DataReadFromSource.Records | ソースの Kafka トピックから読み取ったレコード数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples |

| メトリクス | 説明 |
| --- | --- |
| | 単位: カウント |
| DataReadFromSource.Bytes | ソースの Kafka トピックから読み取ったバイト数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |
| SourceThrottled.Delay | ソースの Kafka クラスターが、ソースの Kafka トピックのレコードを返すときの遅延時間。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: ミリ秒 |
| BytesPerSecondLimit | Firehose がソースの Kafka トピックの各パーティションから読み取るスループットの現在の上限値。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト/秒 |
| KafkaOffsetLag | Firehose がソースの Kafka トピックから読み取ったレコードの最大オフセットと、ソースの Kafka トピックから入手できるレコードの最大オフセットとの差。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| FailedValidation.Records | レコード検証に失敗したレコード数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |

| メトリクス | 説明 |
| --- | --- |
| FailedValidation.Bytes | レコード検証に失敗したバイト数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |
| DataReadFromSource.Backpressured | パーティションごとの BytesPerSecondLimit を超えたか、通常の配信フローが遅れているまたは停止しているために、Firehose ストリームによるソースパーティションからのレコードの読み込みが遅延していることを示します<br><br>単位: ブール |

## API レベルの CloudWatch メトリクス

AWS/Firehose 名前空間には、次の API レベルメトリクスが含まれます。

| メトリクス | 説明 |
| --- | --- |
| DescribeDeliveryStream.Latency | 指定された期間に測定された DescribeDeliveryStream オペレーションごとにかかった時間。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: ミリ秒 |
| DescribeDeliveryStream.Requests | DescribeDeliveryStream リクエストの総数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| ListDeliveryStreams.Latency | 指定された期間に測定された ListDeliveryStream オペレーションごとにかかった時間。<br><br>統計: Minimum、Maximum、Average、Samples |

| メトリクス | 説明 |
| --- | --- |
| | 単位: ミリ秒 |
| ListDeliveryStreams.Requests | ListFirehose リクエストの総数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| PutRecord.Bytes | 指定された期間に PutRecord を使用して Firehose ストリームに送信されたバイト数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |
| PutRecord.Latency | 指定された期間に測定された PutRecord オペレーションごとにかかった時間。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: ミリ秒 |
| PutRecord.Requests | PutRecord オペレーションのレコード総数に等しい、PutRecord リクエストの総数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| PutRecordBatch.Bytes | 指定された期間に PutRecordBatch を使用して Firehose ストリームに送信されたバイト数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: バイト |

| メトリクス | 説明 |
| --- | --- |
| PutRecordBatch.Latency | 指定された期間に測定された PutRecordBatch オペレーションごとにかかった時間。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: ミリ秒 |
| PutRecordBatch.Records | PutRecordBatch オペレーションのレコード総数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| PutRecordBatch.Requests | PutRecordBatch リクエストの総数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| PutRequestsPerSecondLimit | Firehose ストリームがスロットリング前に処理できる 1 秒あたりの最大 put リクエスト数。この数には、PutRecord および PutRecordBatch リクエストが含まれます。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| ThrottledDescribeStream | データソースが Kinesis データストリームである場合に DescribeStream オペレーションが調整される合計回数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |

| メトリクス | 説明 |
| --- | --- |
| ThrottledGetRecords | データソースが Kinesis データストリームである場合に GetRecords オペレーションが調整される合計回数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| ThrottledGetShardIterator | データソースが Kinesis データストリームである場合に GetShardIterator オペレーションが調整される合計回数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| UpdateDeliveryStream.Latency | 指定された期間に測定された UpdateDeliveryStream オペレーションごとにかかった時間。<br><br>統計: Minimum、Maximum、Average、Samples<br><br>単位: ミリ秒 |
| UpdateDeliveryStream.Requests | UpdateDeliveryStream リクエストの総数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |

## データ変換 CloudWatch メトリクス

Lambda でのデータ変換が有効になっている場合、AWS/Firehose 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 |
| --- | --- |
| ExecutePr ocessing. Duration | Firehose によって実行される各 Lambda 関数の呼び出しにかかる時間。<br>単位: ミリ秒 |
| ExecutePr ocessing. Success | Lambda 関数の呼び出しの合計に対する成功した Lambda 関数の呼び出しの合計です。 |
| SucceedPr ocessing. Records | 指定された期間に、正常に処理されたレコードの数。<br>単位: カウント |
| SucceedPr ocessing.Bytes | 指定された期間に、正常に処理されたバイト数。<br>単位: バイト |

## CloudWatch Logs の解凍メトリクス

CloudWatch Logs 配信のために解凍が有効になっている場合、AWS/Firehose 名前空間は次のメトリクスを含みます。

| メトリクス | 説明 |
| --- | --- |
| OutputDecompressed Bytes.Success | 成功した解凍データ (バイト)<br>統計: Minimum、Maximum、Average、Sum、Samples<br>単位: バイト |
| OutputDecompressed Bytes.Failed | 失敗した解凍データ (バイト)<br>統計: Minimum、Maximum、Average、Sum、Samples<br>単位: バイト |
| OutputDecompressed Records.Success | 成功した解凍レコードの数 |

| メトリクス | 説明 |
| --- | --- |
| | 統計: Minimum、Maximum、Average、Sum、Samples<br>単位: カウント |
| OutputDecompressedRecords.Failed | 失敗した解凍レコードの数<br>統計: Minimum、Maximum、Average、Sum、Samples<br>単位: カウント |

## 形式変換 CloudWatch メトリクス

形式の変換が有効な場合、AWS/Firehose 名前空間には以下のメトリクスが含まれます。

| メトリクス | 説明 |
| --- | --- |
| SucceedConversion.Records | 正常に変換されたレコードの数。<br>単位: カウント |
| SucceedConversion.Bytes | 正常に変換されたレコードのサイズ。<br>単位: バイト |
| FailedConversion.Records | 変換できなかったレコードの数。<br>単位: カウント |
| FailedConversion.Bytes | 変換できなかったレコードのサイズ。<br>単位: バイト |

## サーバー側の暗号化 (SSE) CloudWatch のメトリクス

AWS/Firehose 名前空間には、SSE に関連する以下のメトリクスが含まれます。

| メトリクス | 説明 |
| --- | --- |
| KMSKeyAccessDenied | サービスが、Firehose ストリームの KMSAccessDeniedException を検出した回数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| KMSKeyDisabled | サービスが、Firehose ストリームの KMSDisabledException を検出した回数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| KMSKeyInvalidState | サービスが、Firehose ストリームの KMSInvalidStateException を検出した回数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |
| KMSKeyNotFound | サービスが、Firehose ストリームの KMSNotFoundException を検出した回数。<br><br>統計: Minimum、Maximum、Average、Sum、Samples<br><br>単位: カウント |

## Amazon Data Firehose のディメンション

Firehose ストリームでメトリクスをフィルタするには、DeliveryStreamName ディメンションを使用します。

## Amazon Data Firehose の使用状況メトリクス

CloudWatch 使用状況メトリクスを使用して、アカウントのリソースの使用状況を把握できます。これらのメトリクスを使用して、CloudWatch グラフやダッシュボードで現在のサービスの使用状況を可視化できます。

サービスクォータ使用状況メトリクスは、 AWS/Usage 名前空間にあり、3 分ごとに収集されます。

現在、この名前空間で CloudWatch が発行する唯一のメトリクス名は ResourceCount です。このメトリクスは、Service、Class、Type、および Resource のディメンションで発行されます。

| メトリクス | 説明 |
| --- | --- |
| ResourceCount | アカウントで実行されている指定されたリソースの数。リソースはメトリクスに関連付けられたディメンションによって定義されます。<br><br>このメトリクスで最も有用な統計は MAXIMUM です。MAXIMUM は、3 分間に使用されるリソースの最大数を表します。 |

次のディメンションは、Amazon Data Firehose によって発行される使用状況メトリクスを絞り込むために使用されます。

| ディメンション | 説明 |
| --- | --- |
| Service | リソースを含む AWS サービスの名前。Amazon Data Firehose 使用状況メトリクスの場合、このディメンションの値は Firehose です。 |
| Class | 追跡されているリソースのクラス。Amazon Data Firehose 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 Firehose である場合、Type の有効な値は Resource のみです。 |
| Resource | AWS リソースの名前。現在、Service ディメンションが Firehose である場合、Resource の有効な値は DeliveryStreams のみです。 |

# Amazon Data Firehose の CloudWatch メトリクスにアクセスする

CloudWatch コンソール、コマンドライン、または CloudWatch API を使用して、Amazon Data Firehose のメトリクスをモニタリングできます。次の手順は、これらのさまざまなメソッドを使用してメトリクスにアクセス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にアクセスするには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. ナビゲーションバーで、リージョンを選択します。
3. ナビゲーションペインで [Metrics (メトリクス)] を選択してください。
4. Firehose の名前空間を選択します。
5. [Firehose ストリームメトリクス] または [Firehose のメトリクス] を選択します。
6. グラフに追加するメトリクスを選択します。

を使用してメトリクスにアクセスするには AWS CLI

list-metrics および get-metric-statistics コマンドを使用します。

```
aws cloudwatch list-metrics --namespace "AWS/Firehose"
```

```
aws cloudwatch get-metric-statistics --namespace "AWS/Firehose" \
--metric-name DescribeDeliveryStream.Latency --statistics Average --period 3600 \
--start-time 2017-06-01T00:00:00Z --end-time 2017-06-30T00:00:00Z
```

# CloudWatch Logs を使用して Amazon Data Firehose をモニタリングする

Amazon Data Firehose は Amazon CloudWatch Logs と統合されているため、お客様はデータ変換用の Lambda 呼び出しまたはデータ配信が失敗した場合の特定のエラーログを表示できます。Amazon Data Firehose のエラーログ記録は、Firehose ストリームの作成時に有効にすることができます。

Amazon Data Firehose コンソールで Amazon Data Firehose エラーログ記録を有効にすると、ロググループと対応するログストリームが、お客様に代わって Firehose ストリーム用に作成されます。ロググループ名の形式は `/aws/kinesisfirehose/delivery-stream-name` です。*delivery-stream-name* は対応する Firehose ストリームの名前です。`DestinationDelivery` は、最初の

宛先への配信に関連するエラーを記録するために作成され、使用されるログストリームです。もう 1 つの BackupDelivery というログストリームは、S3 バックアップが送信先で有効になっている場合のみ作成されます。BackupDelivery ログストリームは、S3 バックアップへの配信に関連するすべてのエラーを記録するために使用されます。

例えば、Amazon Redshift を宛先とする Firehose ストリーム「MyStream」を作成し、Amazon Data Firehose エラーログ記録を有効にした場合、aws/kinesisfirehose/MyStream という名前のロググループと、DestinationDelivery および BackupDelivery という名前の 2 つのログストリームが自動的に作成されます。こちらの例では、DestinationDelivery を使用して Amazon Redshift の送信先と中間の S3 送信先への配信に関連するすべてのエラーが記録されます。S3 バックアップが有効になっている場合、S3 バックアップバケットへの配信に関連するすべてのエラーの記録には BackupDelivery が使用されます。

Amazon Data Firehose エラーログ記録は AWS CLI、、 API、または CloudWatchLoggingOptions設定 CloudFormation を使用して有効にできます。これを行うには、事前にロググループとログストリームを作成します。そのロググループとログストリームを、Amazon Data Firehose エラーログ記録専用に予約することをお勧めします。また、関連付けられた IAM ポリシーに "logs:putLogEvents" アクセス許可があることを確認します。詳細については、「Amazon Data Firehose によるアクセスの制御」を参照してください。

Amazon Data Firehose では、すべての配信エラーログが CloudWatch Logs に送信されることは保証されません。配信失敗率が高い状況では、Amazon Data Firehose は配信エラーログをサンプリングしてから、CloudWatch Logs に送信します。

CloudWatch Logs に送信されるエラーログには、わずかな金額が課金されます。詳細については、「Amazon CloudWatch 料金表」をご覧ください。

内容



## データ配信エラー

次に示しているのは、Amazon Data Firehose の宛先ごとのデータ配信エラーコードおよびメッセージのリストです。各エラーメッセージには、問題を解決するために実行する適切なアクションも示されます。

エラー

- Apache Iceberg テーブルのデータ配信エラー
- Amazon Redshift データ配信エラー
- Snowflake データ配信エラー
- Splunk データ配信エラー
- ElasticSearch データ配信のエラー
- HTTPS エンドポイントデータ配信エラー
- Amazon OpenSearch Service のデータ配信エラー
- Lambda 呼び出しエラー
- Kinesis 呼び出しのエラー
- Kinesis DirectPut 呼び出しエラー
- AWS Glue 呼び出しエラー
- DataFormatConversion 呼び出しエラー

## Amazon S3 データ配信エラー

Amazon Data Firehose は、次の Amazon S3 関連のエラーを CloudWatch Logs に送信できます。

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| `S3.KMS.NotFoundException` | 「指定された AWS KMS キーが見つかりませんでした。正しいロールを持つ有効な AWS KMS キーであると思われるものを使用している場合は、 AWS KMS キーがアタッチされているアカウントに問題があるかどうかを確認します。」 |
| `S3.KMS.RequestLimitExceeded` | "S3 オブジェクトを暗号化しようとしているときに、KMS リクエスト/秒の制限を超えました。1 秒あたりのリクエストの制限を引き上げてください。"<br><br>詳細については、AWS Key Management Service デベロッパーガイドの「制限」を参照してください。 |
| `S3.AccessDenied` | 「アクセスが拒否されました。指定された IAM ロールの信頼ポリシーで、Amazon Data Firehose によるロールの引き受けが許可されること、およびアクセスポリシーで S3 バケットへのアクセスが許可されることを確認してください」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| S3.AccountProblem | AWS 「アカウントでオペレーションが正常に完了できない問題があります。 AWS サポートにお問い合わせください。」 |
| S3.AllAccessDisabled | 「指定されたアカウントへのアクセスが無効になっています。 AWS サポートにお問い合わせください。」 |
| S3.InvalidPayer | 「指定されたアカウントへのアクセスが無効になっています。 AWS サポートにお問い合わせください。」 |
| S3.NotSignedUp | 「アカウントは Amazon S3 に対してサインアップされていません。アカウントにサインアップするか、別のアカウントを使用します。」 |
| S3.NoSuchBucket | 「指定されたバケットが存在しません。バケットを作成するか、存在する別のバケットを使用します。」 |
| S3.MethodNotAllowed | 「指定されたメソッドは、このリソースに対して許可されていません。Amazon S3 オペレーションの正しいアクセス許可を許可するようバケットのポリシーを変更します。」 |
| InternalError | 「データを配信しようとして内部エラーが発生しました。配信は再試行されます。エラーが解決しない場合は、解決 AWS のために に報告されます。」 |
| S3.KMS.KeyDisabled | 「指定された KMS キーが無効です。キーを有効にするか別のキーを使用します。」 |
| S3.KMS.InvalidStateException | 「指定された KMS キーは無効な状態です。別のキーを使用してください。」 |
| KMS.InvalidStateException | 「指定された KMS キーは無効な状態です。別のキーを使用してください。」 |
| KMS.DisabledException | 「指定された KMS キーが無効です。キーを修正するか別のキーを使用してください。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| S3.SlowDown | 「指定されたバケットへの PUT リクエスト率が高すぎます。Firehose ストリームのバッファリングサイズを増やすか、他のアプリケーションからの PUT リクエストを減らしてください」 |
| S3.SubscriptionRequired | 「S3 を呼び出すときにアクセスが拒否されました。渡された IAM ロールと KMS キー (指定されている場合) に Amazon S3 サブスクリプションがあることを確認してください。」 |
| S3.InvalidToken | 「指定されたトークンは形式に誤りがあるか無効です。入力した認証情報を確認してください。」 |
| S3.KMS.KeyNotConfigured | 「KMS キーが設定されていません。KMSMasterKeyID を設定するか、S3 バケットの暗号化を無効にしてください。」 |
| S3.KMS.AsymmetricCMKNotSupported | 「Amazon S3 がサポートしているのは対称 CMK のみです。非対称 CMK を使用して Amazon S3 にあるデータを暗号化することはできません。CMK のタイプを取得するには、KMS DescribeKey オペレーションを使用します。」 |
| S3.IllegalLocationConstraintException | 「Firehose は現在、設定された S3 バケットへのデータ配信に S3 グローバルエンドポイントを使用しています。設定された S3 バケットのリージョンは、S3 グローバルエンドポイントをサポートしていません。S3 バケットと同じリージョンに Firehose ストリームを作成するか、グローバルエンドポイントをサポートしているリージョンの S3 バケットを使用してください」 |
| S3.InvalidPrefixConfigurationException | 「タイムスタンプ評価に使用したカスタムの S3 プレフィックスが無効です。S3 プレフィックスに現在の日付と時刻を示す有効な表示が含まれていることを確認してください。」 |
| DataFormatConversion.MalformedData | 「トークンの間に不正な文字が見つかりました。」 |

## Apache Iceberg テーブルのデータ配信エラー

Apache Iceberg テーブルのデータ配信エラーについては、「Apache Iceberg テーブルにデータを配信する」を参照してください。

## Amazon Redshift データ配信エラー

Amazon Data Firehose は、次の Amazon RedShift 関連のエラーを CloudWatch Logs に送信できます。

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| `Redshift.TableNotFound` | 「データのロード先となるテーブルが見つかりませんでした。指定されたテーブルが存在することを確認してください。」<br><br>S3 からのデータのコピー先となる Amazon Redshift の送信先テーブルが見つかりませんでした。Amazon Redshift テーブルが存在しない場合、Amazon Data Firehose は Amazon Redshift テーブルを作成しないことに注意してください。 |
| `Redshift.SyntaxError` | 「COPY コマンドには構文エラーが含まれています。コマンドを再試行してください。」 |
| `Redshift.AuthenticationFailed` | 「指定されたユーザー名とパスワードで認証に失敗しました。有効なユーザー名とパスワードを指定してください。」 |
| `Redshift.AccessDenied` | 「アクセスが拒否されました。指定された IAM ロールの信頼ポリシーで、Amazon Data Firehose がロールを引き受けられることを確認します」 |
| `Redshift.S3BucketAccessDenied` | 「COPY コマンドは S3 バケットにアクセスできませんでした。指定した IAM ロールのアクセスポリシーで、S3 バケットへのアクセスが許可されることを確認してください。」 |
| `Redshift.DataLoadFailed` | 「テーブルへのデータのロードに失敗しました。詳細については、STL_LOAD_ERRORS システムテーブルを確認してください。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| `Redshift.ColumnNotFound` | 「COPY コマンドの列がテーブルに存在しません。有効な列名を指定してください。」 |
| `Redshift.DatabaseNotFound` | 「Amazon Redshift の送信先設定または JDBC URL で指定されたデータベースは見つかりませんでした。有効なデータベース名を指定してください。」 |
| `Redshift.IncorrectCopyOptions` | 「競合するか冗長な COPY のオプションが指定されました。一部のオプションは、特定の組み合わせと互換性がありません。詳細については、COPY コマンドのリファレンスを参照してください。」<br><br>詳細については、Amazon Redshift データベース開発者ガイドの「Amazon Redshift COPY コマンド」を参照してください。 |
| `Redshift.MissingColumn` | 「デフォルト値がなく、列リストに含まれていない、NOT NULL としてテーブルスキーマに定義されている列があります。この列を除外し、ロードされたデータが常にこの列の値を提供することを確認するか、このテーブルの Amazon Redshift スキーマにデフォルト値を追加します。」 |
| `Redshift.ConnectionFailed` | 「指定された Amazon Redshift クラスターへの接続に失敗しました。セキュリティ設定により、Amazon Data Firehose 接続が許可され、Amazon Redshift の宛先設定または JDBC URL で指定されたクラスターまたはデータベースが正しく、クラスターが使用可能なことを確認してください」 |
| `Redshift.ColumnMismatch` | 「COPY コマンドの jsonpaths の数および宛先テーブルの列数が一致する必要があります。コマンドを再試行してください。」 |
| `Redshift.IncorrectOrMissingRegion` | 「Amazon Redshift は、S3 バケットにアクセスするために間違ったリージョンのエンドポイントを使用しようとしました。COPY コマンドのオプションで正しいリージョンの値を指定するか、S3 バケットが Amazon Redshift データベースと同じリージョンにあることを確認します。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| Redshift.IncorrectJsonPathsFile | 「指定された jsonpaths ファイルが、サポートされている JSON 形式ではありません。コマンドを再試行してください。」 |
| Redshift.MissingS3File | 「Amazon Redshift で必要な 1 つまたは複数の S3 ファイルが S3 バケットから削除されました。S3 バケットポリシーを確認し、S3 ファイルの自動削除がある場合はそれを除外します。」 |
| Redshift.InsufficientPrivilege | 「データをテーブルにロードするアクセス許可がユーザーにありません。INSERT 権限に対する Amazon Redshift ユーザー権限を確認してください。」 |
| Redshift.ReadOnlyCluster | 「システムがサイズ変更モードであるため、クエリを実行できません。後でもう一度クエリの実行を試みてください。」 |
| Redshift.DiskFull | 「ディスクがいっぱいのため、データをロードできませんでした。Amazon Redshift クラスターの容量を増やすか、使用されていないデータを削除してディスク容量を解放してください。」 |
| InternalError | 「データを配信しようとして内部エラーが発生しました。配信は再試行されます。エラーが解決しない場合は、解決 AWS のために に報告されます。」 |
| Redshift.ArgumentNotSupported | 「COPY コマンドにサポートされていないオプションが含まれています。」 |
| Redshift.AnalyzeTableAccessDenied | 「アクセスが拒否されました。テーブルの分析を実行できるのはテーブルかデータベースの所有者のみであるため、S3 から Redshift へのコピーは失敗します。」 |
| Redshift.SchemaNotFound | 「Amazon Redshift の送信先設定の DataTableName で指定されたスキーマが見つかりませんでした。有効なスキーマ名を指定してください。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| Redshift.ColumnSpecifiedMoreThanOnce | 「列リストに 2 回以上指定されている列があります。重複している列を削除してください。」 |
| Redshift.ColumnNotNullWithoutDefault | 「列リストに含まれていない列に、DEFAULT 値がない NULL 以外の列があります。そのような列が列リストに含まれていることを確認してください。」 |
| Redshift.IncorrectBucketRegion | 「Redshift がクラスターとは異なるリージョンのバケットを使用しようとしました。クラスターと同じリージョンにあるバケットを指定してください。」 |
| Redshift.S3SlowDown | 「S3 へのリクエストレートが高くなっています。スロットリングを避けるためレートを下げてください。」 |
| Redshift.InvalidCopyOptionForJson | 「JSON copyOption には、auto パスか有効な S3 パスのいずれかを使用します。」 |
| Redshift.InvalidCopyOptionJSONPathFormat | 「COPY が次のエラーにより失敗しました: \"無効な JSONPath 形式です。配列インデックスが範囲外です\"。JSONPath 式を修正してください。」 |
| Redshift.InvalidCopyOptionRBACAclNotAllowed | 「COPY が次のエラーにより失敗しました: \"アクセス許可の伝播が有効になっていない間は RBAC ACL フレームワークを使用できません。\"」 |
| Redshift.DiskSpaceQuotaExceeded | 「ディスク容量のクォータを超えたためトランザクションが中止されました。ディスク容量を解放するか、スキーマのクォータを増やすようリクエストしてください。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| Redshift.ConnectionsLimitExceeded | 「ユーザーの接続上限を超えました。」 |
| Redshift.SslNotSupported | 「サーバーが SSL をサポートしていないため、指定された Amazon Redshift クラスターへの接続に失敗しました。クラスターの設定を確認してください。」 |
| Redshift.HoseNotFound | 「Firehose が削除されました。Hose の状態を確認してください。」 |
| Redshift.Delimiter | 「copyCommand の copyOptions 区切り文字が無効です。1 文字になっていることを確認してください。」 |
| Redshift.QueryCancelled | 「ユーザーが COPY オペレーションをキャンセルしました。」 |
| Redshift.CompressionMismatch | 「Hose には UNCOMPRESSED が設定されていますが、copyOption には圧縮形式が含まれています。」 |
| Redshift.EncryptionCredentials | 「ENCRYPTED オプションでは、認証情報は次の形式になっている必要があります。'aws_iam_role=...;master_symmetric_key=...' or 'aws_access_key_id=...;aws_secret_access_key=...[;token=...];master_symmetric_key=...'」 |
| Redshift.InvalidCopyOptions | 「COPY 設定オプションが無効です。」 |
| Redshift.InvalidMessageFormat | 「Copy コマンドに無効な文字が含まれています。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| Redshift.TransactionIdLimitReached | 「トランザクション ID の上限に達しました。」 |
| Redshift.DestinationRemoved | 「Redshift の送信先が存在し、Firehose 設定で正しく設定されていることを確認してください。」 |
| Redshift.OutOfMemory | 「Redshift クラスターのメモリが不足しています。クラスターに十分な容量があることを確認してください。」 |
| Redshift.CannotForkProcess | 「Redshift クラスターのメモリが不足しています。クラスターに十分な容量があることを確認してください。」 |
| Redshift.SslFailure | 「ハンドシェイク中に SSL 接続が終了しました。」 |
| Redshift.Resize | 「Redshift クラスターのサイズを変更しています。Firehose は、クラスターのサイズ変更中はデータを配信できません。」 |
| Redshift.ImproperQualifiedName | 「修飾名が正しくありません (ドット表記名が多すぎる)。」 |
| Redshift.InvalidJsonPathFormat | 「JSONPath 形式が無効です。」 |
| Redshift.TooManyConnectionsException | 「Redshift への接続が多すぎます。」 |
| Redshift.PSQLException | 「Redshift で PSQlException が確認されました。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| Redshift.DuplicateSecondsSpecification | 「日付/時刻形式の秒指定が重複しています。」 |
| Redshift.RelationCouldNotBeOpened | 「Redshift でエラーが発生し、関係を開けませんでした。指定された DB の Redshift ログを確認してください。」 |
| Redshift.TooManyClients | 「Redshift で多数のクライアントの例外が発生しました。複数のプロデューサーが同時にデータベースに書き込んでいる場合は、データベースへの最大接続数を再検討してください」 |

## Snowflake データ配信エラー

Firehose は CloudWatch Logs に次の Snowflake 関連のエラーを送信できます。

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| Snowflake.InvalidUrl | 「Firehose は Snowflake に接続できません。Snowflake の宛先設定で、アカウント URL が正しく指定されていることを確認してください」 |
| Snowflake.InvalidUser | 「Firehose は Snowflake に接続できません。Snowflake の宛先設定で、ユーザーが正しく指定されていることを確認してください」 |
| Snowflake.InvalidRole | 「指定された Snowflake ロールが存在しないか、または認可されていません。指定されたユーザーにロールが付与されていることを確認してください」 |
| Snowflake.InvalidTable | 「指定されたテーブルが存在しないか、または認可されていません」 |
| Snowflake.InvalidSchema | 「指定されたスキーマが存在しないか、または認可されていません」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| `Snowflake.InvalidDatabase` | 「指定されたデータベースが存在しないか、または認可されていません」 |
| `Snowflake.InvalidPrivateKeyOrPassphrase` | 「指定されたプライベートキーまたはパスフレーズが無効です。指定されるプライベートキーは有効な PEM RSA プライベートキーである必要があることに留意してください」 |
| `Snowflake.MissingColumns` | 「入力ペイロードに列がないため、挿入リクエストは拒否されます。nullable 以外のすべての列のために値が指定されていることを確認してください」 |
| `Snowflake.ExtraColumns` | 「余分な列があるため、挿入リクエストは拒否されます。テーブルに存在しない列は指定しないでください」 |
| `Snowflake.InvalidInput` | 「無効な入力形式のため、配信に失敗しました。指定された入力ペイロードが、許容可能な JSON 形式であることを確認してください」 |
| `Snowflake.IncorrectValue` | 「入力ペイロードのデータ型が正しくないため、配信に失敗しました。入力ペイロードで指定された JSON の値が Snowflake テーブル定義で宣言されたデータ型に準拠していることを確認してください」 |

## Splunk データ配信エラー

Amazon Data Firehose は CloudWatch Logs に次の Splunk 関連のエラーを送信できます。

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| `Splunk.ProxyWithoutStickySessions` | 「Amazon Data Firehose と HEC ノードの間にプロキシ (ELB など) がある場合、HEC ACK をサポートするには、スティッキーセッションを有効にする必要があります」 |
| `Splunk.DisabledToken` | "The HEC token is disabled。トークンを有効にして Splunk へのデータ配信を許可します。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| `Splunk.InvalidToken` | "The HEC token is invalid。有効な HEC トークンを使用して Amazon Data Firehose を更新します」 |
| `Splunk.InvalidDataFormat` | "The data is not formatted correctly。Raw または Event HEC のエンドポイントのデータを適切にフォーマットする方法については、「Splunk Event Data」を参照してください。」 |
| `Splunk.InvalidIndex` | "The HEC token or input is configured with an invalid index。インデックスの設定を確認して再試行してください。」 |
| `Splunk.ServerError` | "Data delivery to Splunk failed due to a server error from the HEC node。Amazon Data Firehose の再試行期間が 0 より大きい場合、Amazon Data Firehose はデータ送信を再試行します。すべての再試行が失敗した場合、Amazon Data Firehose はデータを Amazon S3 にバックアップします」 |
| `Splunk.DisabledAck` | "Indexer acknowledgement is disabled for the HEC token。Enable indexer acknowledgement and try again。詳細については、「Enable indexer acknowledgement」を参照してください。」 |
| `Splunk.AckTimeout` | "Did not receive an acknowledgement from HEC before the HEC acknowledgement timeout expired。Despite the acknowledgement timeout, it's possible the data was indexed successfully in Splunk。Amazon Data Firehose は確認応答のタイムアウトが切れたデータを Amazon S3 にバックアップします」 |
| `Splunk.MaxRetriesFailed` | "Failed to deliver data to Splunk or to receive acknowledgment。HEC の状態を確認し、再試行してください。」 |
| `Splunk.ConnectionTimeout` | "The connection to Splunk timed out。This might be a transient error and the request will be retried。すべてのリトライが失敗した場合、Amazon Data Firehose は Amazon S3 にデータをバックアップします」 |
| `Splunk.InvalidEndpoint` | "Could not connect to the HEC endpoint。HEC のエンドポイント URL が有効で、Amazon Data Firehose から到達可能であることを確認してください」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| Splunk.ConnectionClosed | "Unable to send data to Splunk due to a connection failure。This might be a transient error。Amazon Data Firehose の設定で再試行の期間を長くすると、このような一時的な失敗を防ぐことができます」 |
| Splunk.SSLUnverified | "Could not connect to the HEC endpoint。ホストがピアによって提供された証明書と一致しません。証明書とホストが有効であることを確認してください。" |
| Splunk.SSLHandshake | "Could not connect to the HEC endpoint。証明書とホストが有効であることを確認してください。" |
| Splunk.URLNotFound | 「リクエストされた URL が Splunk サーバーで見つかりませんでした。Splunk クラスターをチェックし、正しく設定されていることを確認してください。」 |
| Splunk.ServerError.ContentTooLarge | 「サーバーエラー (statusCode: 413、メッセージ: クライアントが送信したリクエストが大きすぎます) により Splunk へのデータ配信が失敗しました。max_content_length の設定方法については、Splunk のドキュメントを参照してください。」 |
| Splunk.IndexerBusy | "Data delivery to Splunk failed due to a server error from the HEC node。HEC エンドポイントまたは Elastic Load Balancer にアクセスできること、それらが正常であることを確認してください。」 |
| Splunk.ConnectionRecycled | 「Firehose から Splunk への接続はリサイクルされました。配信は再試行されます。」 |
| Splunk.AcknowledgementsDisabled | 「POST での受信確認を取得できませんでした。HEC エンドポイントで確認が有効になっていることを確認してください。」 |
| Splunk.InvalidHecResponseCharacter | 「HEC レスポンスで無効な文字が見つかりました。サービスと HEC の設定を確認してください。」 |

## ElasticSearch データ配信のエラー

Amazon Data Firehose では、次の ElasticSearch のエラーを CloudWatch Logs に送信できます。

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| ES.AccessDenied | 「アクセスが拒否されました。Firehose に関連付けられている指定された IAM ロールが削除されていないことを確認してください。」 |
| ES.ResourceNotFound | 「指定された Elasticsearch AWS ドメインは存在しません。」 |

## HTTPS エンドポイントデータ配信エラー

Amazon Data Firehose は、次の HTTP エンドポイント関連のエラーを CloudWatch Logs に送信できます。これらのエラーのいずれも発生している問題と一致しない場合、デフォルトのエラーは次のとおりです。「データの配信中に内部エラーが発生しました。配信は再試行されます。エラーが解決しない場合は、解決 AWS のために に報告されます。」

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| HttpEndpoint.RequestTimeout | レスポンスを受信する前に配信がタイムアウトし、再試行されます。このエラーが解決しない場合は、 AWS Firehose サービスチームにお問い合わせください。 |
| HttpEndpoint.ResponseTooLarge | 「エンドポイントから受信したレスポンスが大きすぎます。この問題を解決するには、エンドポイントの所有者にお問い合わせください。」 |
| HttpEndpoint.InvalidResponseFromDestination | 「指定されたエンドポイントから受信したレスポンスが無効です。問題を解決するには、エンドポイントの所有者にお問い合わせください。」 |
| HttpEndpoint.Desti | 「エンドポイントの送信先から次のレスポンスが受信されました。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| nationExc eption | |
| HttpEndpo int.Conne ctionFailed | 「送信先エンドポイントに接続できません。この問題を解決するには、エンドポイントの所有者にお問い合わせください。」 |
| HttpEndpo int.Conne ctionReset | 「エンドポイントとの接続を維持できません。この問題を解決するには、エンドポイントの所有者にお問い合わせください。」 |
| HttpEndpo int.Conne ctionReset | 「エンドポイントとの接続の維持に問題があります。エンドポイントの所有者に連絡してください。」 |
| HttpEndpo int.Respo nseReason PhraseExc eededLimit | 「エンドポイントから受信したレスポンスの理由のフレーズが、設定されている上限の 64 文字を超えています。」 |
| HttpEndpo int.Inval idRespons eFromDest ination | 「エンドポイントから受信したレスポンスが無効です。詳細については、Firehose ドキュメントの「Troubleshooting HTTP Endpoints」を参照してください。理由:「 |
| HttpEndpo int.Desti nationExc eption | 「エンドポイントへの配信に失敗しました。詳細については、Firehose ドキュメントの「Troubleshooting HTTP Endpoints」を参照してください。ステータスコード付きのレスポンスを受信しました。」 |
| HttpEndpo int.Inval idStatusCode | 「無効なレスポンスステータスコードを受信しました。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| HttpEndpoint.SSLHandshakeFailure | 「エンドポイントとの SSL ハンドシェイクを完了できません。この問題を解決するには、エンドポイントの所有者にお問い合わせください。」 |
| HttpEndpoint.SSLHandshakeFailure | 「エンドポイントとの SSL ハンドシェイクを完了できません。この問題を解決するには、エンドポイントの所有者にお問い合わせください。」 |
| HttpEndpoint.SSLFailure | 「エンドポイントとの TLS ハンドシェイクを完了できません。この問題を解決するには、エンドポイントの所有者にお問い合わせください。」 |
| HttpEndpoint.SSLHandshakeCertificatePathFailure | 「証明書パスが無効であるため、エンドポイントとの SSL ハンドシェイクを完了できません。この問題を解決するには、エンドポイントの所有者にお問い合わせください。」 |
| HttpEndpoint.SSLHandshakeCertificatePathValidationFailure | 「証明書パスの検証に失敗したため、エンドポイントとの SSL ハンドシェイクを完了できません。この問題を解決するには、エンドポイントの所有者にお問い合わせください。」 |
| HttpEndpoint.MakeRequestFailure.IllegalUriException | 「URI の入力が無効であるため、HttpEndpoint リクエストが失敗しました。入力 URI の文字がすべて有効であることを確認してください。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| HttpEndpoint.MakeRequestFailure.IllegalCharacterInHeaderValue | 「不正なレスポンスエラーにより、HttpEndpoint リクエストが失敗しました。ヘッダー値に不正な文字 '\n' が含まれています。」 |
| HttpEndpoint.IllegalResponseFailure | 「不正なレスポンスエラーにより、HttpEndpoint リクエストが失敗しました。HTTP メッセージに複数の Content-Type ヘッダーを含めることはできません。」 |
| HttpEndpoint.IllegalMessageStart | 「不正なレスポンスエラーにより、HttpEndpoint リクエストが失敗しました。不正な HTTP メッセージが開始されました。詳細については、Firehose ドキュメントの「Troubleshooting HTTP Endpoints」を参照してください。」 |

## Amazon OpenSearch Service のデータ配信エラー

OpenSearch Service の宛先については、Amazon Data Firehose は OpenSearch Service によって返されるエラーを CloudWatch Logs に送信します。

OpenSearch クラスターから返されるエラーのほか、次の 2 つのエラーが発生することがあります。

- データを送信先の OpenSearch Service クラスターに配信しようとしたときに、認証/認可エラーが発生する。これは、許可の問題で発生したり、Amazon Data Firehose のターゲットの OpenSearch Service ドメイン設定が変更されたときに断続的に発生したりすることがあります。クラスターポリシーとロールのアクセス許可を確認してください。
- 認証/認可に失敗したため、データを送信先の OpenSearch Service クラスターに配信できなかった。これは、許可の問題で発生したり、Amazon Data Firehose のターゲットの OpenSearch Service ドメイン設定が変更されたときに断続的に発生したりすることがあります。クラスターポリシーとロールのアクセス許可を確認してください。

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| OS.AccessDenied | 「アクセスが拒否されました。指定された IAM ロールの信頼ポリシーで、Firehose によるロールの引き受けが許可されていること、およびアクセスポリシーで Amazon OpenSearch Service API へのアクセスが許可されていることを確認してください。」 |
| OS.AccessDenied | 「アクセスが拒否されました。指定された IAM ロールの信頼ポリシーで、Firehose によるロールの引き受けが許可されていること、およびアクセスポリシーで Amazon OpenSearch Service API へのアクセスが許可されていることを確認してください。」 |
| OS.AccessDenied | 「アクセスが拒否されました。Firehose に関連付けられている指定された IAM ロールが削除されていないことを確認してください。」 |
| OS.AccessDenied | 「アクセスが拒否されました。Firehose に関連付けられている指定された IAM ロールが削除されていないことを確認してください。」 |
| OS.ResourceNotFound | 「指定された Amazon OpenSearch Service ドメインは存在しません。」 |
| OS.ResourceNotFound | 「指定された Amazon OpenSearch Service ドメインは存在しません。」 |
| OS.AccessDenied | 「アクセスが拒否されました。指定された IAM ロールの信頼ポリシーで、Firehose によるロールの引き受けが許可されていること、およびアクセスポリシーで Amazon OpenSearch Service API へのアクセスが許可されていることを確認してください。」 |
| OS.RequestTimeout | 「Amazon OpenSearch Service クラスターまたは OpenSearch Serverless コレクションへのリクエストがタイムアウトになりました。クラスターまたはコレクションに、現在のワークロードに十分対応できる容量があることを確認してください。」 |
| OS.ClusterError | 「Amazon OpenSearch Service クラスターが詳細不明のエラーを返しました。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| OS.RequestTimeout | 「Amazon OpenSearch Service クラスターへのリクエストがタイムアウトになりました。クラスターに、現在のワークロードに十分対応できる容量があることを確認してください。」 |
| OS.ConnectionFailed | 「Amazon OpenSearch Service クラスターまたは OpenSearch Serverless コレクションへの接続で問題が発生しました。クラスターまたはコレクションが正常であり、アクセス可能であることを確認してください。」 |
| OS.ConnectionReset | 「Amazon OpenSearch Service クラスターまたは OpenSearch Serverless コレクションへの接続を維持できません。この問題を解決するには、クラスターまたはコレクションの所有者に問い合わせてください。」 |
| OS.ConnectionReset | 「Amazon OpenSearch Service クラスターまたは OpenSearch Serverless コレクションへの接続の維持に関する問題が発生しました。クラスターまたはコレクションが正常であり、現在のワークロードに十分対応できる容量があることを確認してください。」 |
| OS.ConnectionReset | 「Amazon OpenSearch Service クラスターまたは OpenSearch Serverless コレクションへの接続の維持に関する問題が発生しました。クラスターまたはコレクションが正常であり、現在のワークロードに十分対応できる容量があることを確認してください。」 |
| OS.AccessDenied | 「アクセスが拒否されました。Amazon OpenSearch Service クラスターのアクセスポリシーで、設定された IAM ロールへのアクセスが許可されていることを確認してください。」 |
| OS.ValidationException | 「OpenSearch クラスターが ESServiceException を返しました。原因の 1 つとして、クラスターが OS 2.x 以上にアップグレードされているが、Firehose では TypeName パラメータが設定されたままになっていることが挙げられます。TypeName を空の文字列に設定して Firehose の設定を更新するか、エンドポイントを Type パラメータをサポートしているクラスターに変更してください。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| OS.ValidationException | 「メンバーは次の正規表現のパターンを満たす必要があります: [a-z][a-z0-9\\-]+。 |
| OS.JsonParseException | 「Amazon OpenSearch Service クラスターが JsonParseException を返しました。入力されたデータが有効であることを確認してください。」 |
| OS.AmazonOpenSearchServiceParseException | 「Amazon OpenSearch Service クラスターが AmazonOpenSearchServiceParseException を返しました。入力されたデータが有効であることを確認してください。」 |
| OS.ExplicitIndexInBulkNotAllowed | 「Amazon OpenSearch Service クラスターで rest.action.multi.allow_explicit_index が true に設定されていることを確認してください。」 |
| OS.ClusterError | 「Amazon OpenSearch Service クラスターまたは OpenSearch Serverless コレクションから詳細不明のエラーが返されました」 |
| OS.ClusterBlockException | 「クラスターが ClusterBlockException を返しました。オーバーロードが発生している可能性があります。」 |
| OS.InvalidARN | 「指定された Amazon OpenSearch Service ARN が無効です。DeliveryStream 設定を確認してください。」 |
| OS.MalformedData | 「1 つまたは複数のレコード形式が不正です。各レコードには有効な JSON オブジェクトが 1 つしか含まれていないこと、および改行が含まれていないことを確認してください。」 |
| OS.InternalError | 「データを配信する際に内部エラーが発生しました。配信は再試行されます。エラーが解決しない場合は、解決 AWS のために に報告されます。」 |
| OS.AliasWithMultipleIndicesNotAllowed | 「エイリアスに複数のインデックスが関連付けられています。エイリアスに 1 つのインデックスのみが関連付けられていることを確認してください。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| OS.UnsupportedVersion | 「Amazon OpenSearch Service 6.0 は、現在 Amazon Data Firehose でサポートされていません。詳細については、 AWS サポートにお問い合わせください。」 |
| OS.CharConversionException | 「1 つ以上のレコードに無効な文字が含まれています。」 |
| OS.InvalidDomainNameLength | 「ドメイン名の長さが OS の有効な上限を超えています。」 |
| OS.VPCDomainNotSupported | 「VPC 内の Amazon OpenSearch Service のドメインは、現在サポートされていません。」 |
| OS.ConnectionError | 「HTTP サーバーが予期せず接続を終了しました。Amazon OpenSearch Service クラスターまたは OpenSearch Serverless コレクションの状態を確認してください。」 |
| OS.LargeFieldData | 「Amazon OpenSearch Service クラスターは、許容範囲を超えるフィールドデータが含まれていたためリクエストを中断しました。」 |
| OS.BadGateway | 「Amazon OpenSearch Service クラスターまたは OpenSearch Serverless コレクションは、502 Bad Gateway というレスポンスを受け取り、リクエストを中断しました。」 |
| OS.ServiceException | 「Amazon OpenSearch Service クラスターまたは OpenSearch Serverless コレクショからエラーを受信しました。クラスターまたはコレクションが VPC の背後にある場合は、ネットワーク構成で接続が可能になっていることを確認してください。」 |
| OS.GatewayTimeout | 「Firehose で、Amazon OpenSearch Service クラスターまたは OpenSearch Serverless コレクションへの接続時にタイムアウトのエラーが発生しました。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| OS.MalformedData | 「Amazon Data Firehose は、Firehose レコード内の Amazon OpenSearch Service Bulk API コマンドをサポートしていません」 |
| OS.ResponseEntryCountMismatch | 「Bulk API からのレスポンスには、送信されたレコードの数よりも多いエントリが含まれていました。各レコードに有効な JSON オブジェクトが 1 つしか含まれていないこと、および改行が含まれていないことを確認してください。」 |

## Lambda 呼び出しエラー

Amazon Data Firehose は、次の Lambda 呼び出しエラーを CloudWatch Logs に送信できます。

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| Lambda.AssumeRoleAccessDenied | 「アクセスが拒否されました。指定された IAM ロールの信頼ポリシーで、Amazon Data Firehose がロールを引き受けられることを確認します」 |
| Lambda.InvokeAccessDenied | 「アクセスが拒否されました。アクセスポリシーで、Lambda 関数へのアクセスが許可されていることを確認してください。」 |
| Lambda.JsonProcessingException | 「Lambda 関数から返されたレコードの解析時にエラーが発生しました。返されたレコードが Amazon Data Firehose からリクエストされたステータスモデルに従っていることを確認してください」<br><br>詳細については、「データ変換に必要なパラメータ」を参照してください。 |
| Lambda.InvokeLimitExceeded | 「Lambda の同時実行の制限を超えています。同時実行の制限を引き上げてください。」<br><br>詳細については、AWS Lambda 開発者ガイドの「AWS Lambda の制限」を参照してください。 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| Lambda.DuplicatedRecordId | 「複数のレコードに対して同じレコード ID が返されました。Lambda 関数がレコードごとに一意のレコード ID を返すことを確認してください。」<br><br>詳細については、「データ変換に必要なパラメータ」を参照してください。 |
| Lambda.MissingRecordId | 「1 つ以上のレコード ID が返されませんでした。Lambda 関数がすべての渡されたレコード ID を返すことを確認してください。」<br><br>詳細については、「データ変換に必要なパラメータ」を参照してください。 |
| Lambda.ResourceNotFound | 「指定した Lambda 関数は存在しません。存在する別の関数を使用してください。」 |
| Lambda.InvalidSubnetIDException | 「Lambda 関数の VPC 設定で指定したサブネット ID が無効です。サブネット ID が有効であることを確認してください。」 |
| Lambda.InvalidSecurityGroupIDException | 「Lambda 関数の VPC 設定で指定したセキュリティグループ ID が無効です。セキュリティグループ ID が有効であることを確認してください。」 |
| Lambda.SubnetIPAddressLimitReachedException | AWS Lambda 「設定された 1 つ以上のサブネットに使用可能な IP アドレスがないため、Lambda 関数の VPC アクセスを設定できませんでした。IP アドレスの制限を引き上げてください。」<br><br>詳細については、Amazon VPC ユーザーガイドの「Amazon VPC の制限 - VPC とサブネット」を参照してください。 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| `Lambda.ENILimitReachedException` | AWS Lambda 「ネットワークインターフェイスの制限に達したため、Lambda 関数設定の一部として指定された VPC に Elastic Network Interface (ENI) を作成できませんでした。ネットワークインターフェイスの制限を引き上げてください。」<br><br>詳細については、Amazon VPC ユーザーガイドの「Amazon VPC の制限 - ネットワークインターフェイス」を参照してください。 |
| `Lambda.FunctionTimedOut` | Lambda 関数の呼び出しがタイムアウトしました。Lambda 関数の Timeout 設定を増やします。詳細については、「関数のタイムアウトの設定」を参照してください。 |
| `Lambda.FunctionError` | これは次のいずれかのエラーが原因で発生します。<br><br>• 出力構造が無効です。関数をチェックし、出力が必要な形式になっていることを確認してください。また、処理されたレコードに有効な結果ステータス (`Dropped`、`Ok`、`ProcessingFailed` のいずれか) が含まれていることを確認してください。<br>• Lambda 関数は正常に呼び出されたが、エラー結果が返されました。<br>• KMS アクセスが拒否されたため、Lambda は環境変数を復号できませんでした。関数の KMS キー設定とキーポリシーをチェックしてください。詳細については「キーアクセスのトラブルシューティング」を参照してください。 |
| `Lambda.FunctionRequestTimedOut` | Amazon Data Firehose で、Lambda の呼び出し時にリクエストが完了せず、リクエストのタイムアウト設定のエラーが発生しました。Lambda コードを見直し、Lambda コードが、設定されたタイムアウトよりも後に実行するように設定されていないかを確認してください。設定されている場合は、メモリ、タイムアウトなど Lambda 設定の調整を検討してください。詳細については「Lambda 関数オプションの設定」を参照してください。 |
| `Lambda.TargetServerFailedToRespond` | Amazon Data Firehose にエラーが発生した。 AWS Lambda サービスを呼び出すときにターゲットサーバーがエラーの応答に失敗しました。 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| Lambda.InvalidZipFileException | Amazon Data Firehose で、Lambda 関数を呼び出すときに InvalidZipFileException が発生しました。Lambda 関数の構成設定と Lambda コードの zip ファイルを確認してください。 |
| Lambda.InternalServerError | 「Amazon Data Firehose が AWS Lambda サービスを呼び出すときに InternalServerError を検出しました。Amazon Data Firehose はデータの送信を一定の回数で再試行します。CreateDeliveryStream または UpdateDestination API を使用して、再試行のオプションを指定または上書きできます。エラーが解決しない場合は、 AWS Lambda サポートチームにお問い合わせください。 |
| Lambda.ServiceUnavailable | Amazon Data Firehose が AWS Lambda サービスを呼び出すときに ServiceUnavailableException を検出しました。Amazon Data Firehose はデータの送信を一定の回数で再試行します。CreateDeliveryStream または UpdateDestination API を使用して、再試行のオプションを指定または上書きできます。エラーが解決しない場合は、 AWS Lambda サポートにお問い合わせください。 |
| Lambda.InvalidSecurityToken | セキュリティトークンが無効であるため Lambda 関数を呼び出すことができません。クロスパーティションの Lambda 呼び出しはサポートされていません。 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| `Lambda.InvocationFailure` | これは次のいずれかのエラーが原因で発生します。<br>• Amazon Data Firehose が AWS Lambda を呼び出すときにエラーが発生しました。オペレーションは再試行されます。それでもエラーが解消しない場合、問題解決は AWS に委ねられます。」<br>• Amazon Data Firehose で Lambda の KMSInvalidStateException が発生しました。使用されている KMS キーが複合無効状態のため、Lambda は環境変数を復号できませんでした。Lambda 関数の KMS キー設定をチェックしてください。<br>• Amazon Data Firehose で Lambda の AWS LambdaException が発生しました。Lambda が指定されたコンテナイメージを初期化できませんでした。イメージを確認してください。<br>• Amazon Data Firehose が AWS Lambda を呼び出すときにタイムアウトエラーが発生しました。サポートされている関数のタイムアウトは最大 5 分です。詳細については、「Data Transformation Execution Duration」を参照してください。 |
| `Lambda.JsonMappingException` | Lambda 関数から返されたレコードを解析しているときエラーが発生しました。データフィールドが Base64 エンコードであることを確認してください。 |

## Kinesis 呼び出しのエラー

Amazon Data Firehose は、次の Kinesis 呼び出しエラーを CloudWatch Logs に送信できます。

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| `Kinesis.AccessDenied` | 「Kinesis を呼び出すときにアクセスが拒否されました。使用している IAM ロールのアクセスポリシーが、適切な Kinesis API へのアクセスを許可していることを確認してください。」 |
| `Kinesis.ResourceNotFound` | 「Firehose がストリームからの読み取りに失敗しました。Firehose が Kinesis ストリームに接続されている場合、ストリームが存在しないか、シャードが結合されているか、あるいは分割されている可能性があ |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| | ります。Firehose が DirectPut タイプの場合、その Firehose はもう存在していない可能性があります。」 |
| Kinesis.Subscriptionrequired | 「Kinesis を呼び出すときにアクセスが拒否されました。Kinesis ストリームアクセスに渡された IAM ロールに Kinesis AWS サブスクリプションがあることを確認します。」 |
| Kinesis.Throttling | 「Kinesis を呼び出すときにスロットリングエラーが発生しました。これは、他のアプリケーションが Firehose ストリームと同じ API を呼び出しているか、ソースと同じ Kinesis ストリームを使用して作成した Firehose ストリームの数が多すぎたことが原因である可能性があります」 |
| Kinesis.Throttling | 「Kinesis を呼び出すときにスロットリングエラーが発生しました。これは、他のアプリケーションが Firehose ストリームと同じ API を呼び出しているか、ソースと同じ Kinesis ストリームを使用して作成した Firehose ストリームの数が多すぎたことが原因である可能性があります」 |
| Kinesis.AccessDenied | 「Kinesis を呼び出すときにアクセスが拒否されました。使用している IAM ロールのアクセスポリシーが、適切な Kinesis API へのアクセスを許可していることを確認してください。」 |
| Kinesis.AccessDenied | 「基盤となる Kinesis ストリームで API オペレーションを呼び出すときにアクセスが拒否されました。IAM ロールが伝播され、有効であることを確認してください」 |
| Kinesis.KMS.AccessDeniedException | 「Firehose は、Kinesis ストリームの暗号化/復号に使用される KMS キーにアクセスできません。Firehose 配信ロールに KMS キーへのアクセス権を付与してください。」 |
| Kinesis.KMS.KeyDisabled | 「暗号化/復号に使用される KMS キーが無効になっているため、Firehose はソース Kinesis ストリームから読み取りすることができません。キーを有効にして読み取りできるようにしてください。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| `Kinesis.KMS.InvalidStateException` | 「Firehose は、暗号化に使用される KMS キーが無効状態であるため、ソース Kinesis ストリームから読み取りすることができません。 |
| `Kinesis.KMS.NotFoundException` | 「Firehose は、暗号化に使用される KMS キーが見つからなかったため、ソース Kinesis ストリームから読み取りすることができません。 |

## Kinesis DirectPut 呼び出しエラー

Amazon Data Firehose は、次の Kinesis DirectPut の呼び出しエラーを CloudWatch Logs に送信できます。

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| `Firehose.KMS.AccessDeniedException` | 「Firehose は KMS キーにアクセスできません。キーポリシーを確認してください。」 |
| `Firehose.KMS.InvalidStateException` | 「Firehose は、暗号化に使用される KMS キーが無効状態であるため、データを復号することができません。 |
| `Firehose.KMS.NotFoundException` | 「Firehose は、暗号化に使用される KMS キーが見つからないため、データを復号することができません。 |
| `Firehose.KMS.KeyDisabled` | 「Firehose は、暗号化に使用される KMS キーが無効になっているため、データを復号することができません。データ配信を継続できるようにキーを有効にしてください。」 |

## AWS Glue 呼び出しエラー

Amazon Data Firehose は、以下の AWS Glue 呼び出しエラーを CloudWatch Logs に送信できます。

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| DataFormatConversion.InvalidSchema | 「スキーマが無効です。」 |
| DataFormatConversion.EntityNotFound | 「指定したテーブル/データベースは見つかりませんでした。こちらのテーブル/データベースが存在し、スキーマ設定で指定された値が正しいこと、特に大文字と小文字が正しく区別されていることを確認してください。」 |
| DataFormatConversion.InvalidInput | 「Glue から一致するスキーマが見つかりませんでした。入力したカタログ ID を持つ、指定したデータベースが存在することを確認してください。」 |
| DataFormatConversion.InvalidInput | 「Glue から一致するスキーマが見つかりませんでした。渡された ARN が正しい形式であることを確認してください。」 |
| DataFormatConversion.InvalidInput | 「Glue から一致するスキーマが見つかりませんでした。入力した catalogId が有効であることを確認してください。」 |
| DataFormatConversion.InvalidVersionId | 「Glue から一致するスキーマが見つかりませんでした。指定したバージョンのテーブルが存在することを確認してください。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| DataFormatConversion.NonExistentColumns | 「Glue から一致するスキーマが見つかりませんでした。テーブルで、ターゲット列を含む NULL 以外のストレージ記述子が設定されていることを確認してください。」 |
| DataFormatConversion.AccessDenied | 「ロールの引き受け中にアクセスが拒否されました。データ形式の変換設定で指定したロールが、Firehose サービスに対してロールを引き受けるアクセス許可を付与していることを確認してください。」 |
| DataFormatConversion.ThrottledByGlue | 「Glue を呼び出すときにスロットリングエラーが発生しました。リクエスト率の上限を引き上げるか、他のアプリケーションから Glue を呼び出す現在のリクエスト率を減らしてください。」 |
| DataFormatConversion.AccessDenied | 「Glue を呼び出すときにアクセスが拒否されました。データ形式の変換設定で指定したロールが、必要なアクセス許可を持っていることを確認してください。」 |
| DataFormatConversion.InvalidGlueRole | 「ロールが無効です。データ形式の変換設定で指定したロールが存在することを確認してください。」 |
| DataFormatConversion.InvalidGlueRole | リクエストに含まれているセキュリティトークンが無効です。Firehose に関連付けられている指定された IAM ロールが削除されていないことを確認してください。」 |
| DataFormatConversion.GlueNotAvailableInRegion | AWS 「Glue は、指定したリージョンではまだ使用できません。別のリージョンを指定してください。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| DataFormatConversion.GlueEncryptionException | 「マスターキーの取得中にエラーが発生しました。そのキーが存在し、正しいアクセス許可が付与されていることを確認してください。」 |
| DataFormatConversion.SchemaValidationTimeout | 「Glue からテーブルを取得しているときにタイムアウトしました。Glue テーブルのバージョンが多数ある場合は、glue: GetTableVersion のアクセス許可を追加する (推奨) か、未使用のテーブルバージョンを削除します。Glue に多数のテーブルがない場合は、 AWS サポートにお問い合わせください。」 |
| DataFirehose.InternalError | 「Glue からテーブルを取得しているときにタイムアウトしました。Glue テーブルのバージョンが多数ある場合は、glue: GetTableVersion のアクセス許可を追加する (推奨) か、未使用のテーブルバージョンを削除します。Glue に多数のテーブルがない場合は、 AWS サポートにお問い合わせください。」 |
| DataFormatConversion.GlueEncryptionException | 「マスターキーの取得中にエラーが発生しました。そのキーが存在し、状態が正常であることを確認してください。」 |

## DataFormatConversion 呼び出しエラー

Amazon Data Firehose は、次の DataFormatConversion の呼び出しエラーを CloudWatch Logs に送信できます。

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| DataFormatConversion.InvalidSchema | 「スキーマが無効です。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| DataFormatConversion.ValidationException | 「列名と型は空でない文字列にする必要があります。」 |
| DataFormatConversion.ParseError | 「不正な形式の JSON が見つかりました。」 |
| DataFormatConversion.MalformedData | 「データがスキーマと一致しません。」 |
| DataFormatConversion.MalformedData | 「JSON キーの長さは 262144 を超えることはできません。」 |
| DataFormatConversion.MalformedData | 「データを UTF-8 としてデコードすることはできません。」 |
| DataFormatConversion.MalformedData | 「トークンの間に不正な文字が見つかりました。」 |
| DataFormatConversion.InvalidTypeFormat | 「タイプの形式が無効です。タイプ構文をチェックしてください。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| DataFormatConversion.InvalidSchema | 「無効なスキーマです。列名に特殊文字や空白が含まれていないことを確認してください」 |
| DataFormatConversion.InvalidRecord | 「レコードがスキーマに従っていません。map<string,string> に対して 1 つまたは複数のマップキーが無効です。」 |
| DataFormatConversion.MalformedData | 「入力 JSON で最上位にプリミティブが含まれていました。最上位はオブジェクトか配列にする必要があります。」 |
| DataFormatConversion.MalformedData | 「入力 JSON で最上位にプリミティブが含まれていました。最上位はオブジェクトか配列にする必要があります。」 |
| DataFormatConversion.MalformedData | 「レコードが空であるか、空白しか含まれていませんでした。」 |
| DataFormatConversion.MalformedData | 「無効な文字が見つかりました。」 |
| DataFormatConversion.MalformedData | 「無効またはサポートされていないタイムスタンプ形式が見つかりました。サポートされているタイムスタンプの形式については、Firehose のデベロッパーガイドを参照してください。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| DataFormatConversion.MalformedData | 「データにはスカラー型が使用されていましたが、スキーマには複合型が指定されていました。」 |
| DataFormatConversion.MalformedData | 「データがスキーマと一致しません。」 |
| DataFormatConversion.MalformedData | 「データにはスカラー型が使用されていましたが、スキーマには複合型が指定されていました。」 |
| DataFormatConversion.ConversionFailureException | 「ConversionFailureException」 |
| DataFormatConversion.DataFormatConversionCustomerErrorException | 「DataFormatConversionCustomerErrorException」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| DataFormatConversion.DataFormatConversionCustomerErrorException | 「DataFormatConversionCustomerErrorException」 |
| DataFormatConversion.MalformedData | 「データがスキーマと一致しません。」 |
| DataFormatConversion.InvalidSchema | 「スキーマが無効です。」 |
| DataFormatConversion.MalformedData | 「データがスキーマと一致しません。1 つまたは複数の日付の形式が無効です。」 |
| DataFormatConversion.MalformedData | 「データに、サポートされていない高度にネストされた JSON 構造が含まれています。」 |
| DataFormatConversion.EntityNotFound | 「指定したテーブル/データベースは見つかりませんでした。こちらのテーブル/データベースが存在し、スキーマ設定で指定された値が正しいこと、特に大文字と小文字が正しく区別されていることを確認してください。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| DataFormatConversion.InvalidInput | 「Glue から一致するスキーマが見つかりませんでした。入力したカタログ ID を持つ、指定したデータベースが存在することを確認してください。」 |
| DataFormatConversion.InvalidInput | 「Glue から一致するスキーマが見つかりませんでした。渡された ARN が正しい形式であることを確認してください。」 |
| DataFormatConversion.InvalidInput | 「Glue から一致するスキーマが見つかりませんでした。入力した catalogId が有効であることを確認してください。」 |
| DataFormatConversion.InvalidVersionId | 「Glue から一致するスキーマが見つかりませんでした。指定したバージョンのテーブルが存在することを確認してください。」 |
| DataFormatConversion.NonExistentColumns | 「Glue から一致するスキーマが見つかりませんでした。テーブルで、ターゲット列を含む NULL 以外のストレージ記述子が設定されていることを確認してください。」 |
| DataFormatConversion.AccessDenied | 「ロールの引き受け中にアクセスが拒否されました。データ形式の変換設定で指定したロールが、Firehose サービスに対してロールを引き受けるアクセス許可を付与していることを確認してください。」 |
| DataFormatConversion.ThrottledByGlue | 「Glue を呼び出すときにスロットリングエラーが発生しました。リクエスト率の上限を引き上げるか、他のアプリケーションから Glue を呼び出す現在のリクエスト率を減らしてください。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| DataFormatConversion.AccessDenied | 「Glue を呼び出すときにアクセスが拒否されました。データ形式の変換設定で指定したロールが、必要なアクセス許可を持っていることを確認してください。」 |
| DataFormatConversion.InvalidGlueRole | 「ロールが無効です。データ形式の変換設定で指定したロールが存在することを確認してください。」 |
| DataFormatConversion.GlueNotAvailableInRegion | AWS 「Glue は、指定したリージョンではまだ使用できません。別のリージョンを指定してください。」 |
| DataFormatConversion.GlueEncryptionException | 「マスターキーの取得中にエラーが発生しました。そのキーが存在し、正しいアクセス許可が付与されていることを確認してください。」 |
| DataFormatConversion.SchemaValidationTimeout | 「Glue からテーブルを取得しているときにタイムアウトしました。Glue テーブルのバージョンが多数ある場合は、glue: GetTableVersion のアクセス許可を追加する (推奨) か、未使用のテーブルバージョンを削除します。Glue に多数のテーブルがない場合は、 AWS サポートにお問い合わせください。」 |
| DataFirehose.InternalError | 「Glue からテーブルを取得しているときにタイムアウトしました。Glue テーブルのバージョンが多数ある場合は、glue: GetTableVersion のアクセス許可を追加する (推奨) か、未使用のテーブルバージョンを削除します。Glue に多数のテーブルがない場合は、 AWS サポートにお問い合わせください。」 |

| エラーコード | エラーメッセージおよび情報 |
| --- | --- |
| `DataFormatConversion.MalformedData` | 「形式が正しくないフィールドが 1 つ以上あります。」 |

# Amazon Data Firehose の CloudWatch ログにアクセスする

Amazon Data Firehose コンソールまたは CloudWatch コンソールを使用して、Amazon Data Firehose のデータ配信失敗に関連するエラーログを表示できます。次の手順は、これらの 2 つの方法を使用してエラーログにアクセスする方法を示しています。

Amazon Data Firehose コンソールを使用してエラーログにアクセスするには

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/firehose で Firehose コンソールを開きます。
2. ナビゲーションバーで、 AWS リージョンを選択します。
3. Firehose ストリームの詳細ページに移動するには、Firehose ストリーム名を選択します。
4. データ配信の失敗に関連するエラーログのリストを表示するには、[Error Log] を選択します。

CloudWatch コンソールを使用してエラーログにアクセスするには

1. CloudWatch コンソールの https://console.aws.amazon.com/cloudwatch/ を開いてください。
2. ナビゲーションバーで、リージョンを選択します。
3. ナビゲーションペインで [ログ] を選択します。
4. データ配信の失敗に関連するエラーログのリストを表示するには、ロググループとログストリームを選択します。

# Kinesis エージェントの状態をモニタリングする

Kinesis エージェントは、AWS KinesisAgent の名前空間でカスタム CloudWatch メトリクスを発行します。エージェントの状態が正常で、指定されたとおりにデータを Amazon Data Firehose に送信しており、データプロデューサーで適切な量の CPU リソースとメモリリソースを消費しているかを評価するのに役立ちます。

送信されたレコード数やバイト数などのメトリクスは、エージェントが Firehose ストリームにデータを送信する速度を知るのに便利です。これらのメトリクスが、ある程度の割合低下するかゼロになることで期待されるしきい値を下回っている場合は、設定の問題、ネットワークエラー、エージェントの状態の問題を示している場合があります。オンホスト CPU やメモリなどの消費量とエージェントエラーカウンターなどのメトリクスは、プロデューサーのリソース使用率を示し、潜在的な構成またはホストのエラーに対する洞察を提供します。最後に、エージェントの問題を調査するのに役立つサービス例外を記録します。

エージェントメトリクスは、エージェント設定 `cloudwatch.endpoint` で指定されたリージョンで報告されます。詳細については、「エージェント構成設定を指定する」を参照してください。

複数の Kinesis エージェントから発行された CloudWatch メトリクスは、集約または結合されます。

Kinesis エージェントから出力されるメトリクスには、わずかな料金がかかります。これはデフォルトで有効になります。詳細については、「Amazon CloudWatch 料金表」をご覧ください。

## CloudWatch を使用して監視する

Kinesis エージェントは、次のメトリクスを CloudWatch に送信します。

| メトリクス | 説明 |
| --- | --- |
| `BytesSent` | 指定された期間に Firehose ストリームに送信されたバイト数。<br><br>単位: バイト |
| `RecordSendAttempts` | 指定した期間内の `PutRecordBatch` 呼び出しのレコード数 (初回または再試行の)。<br><br>単位: カウント |
| `RecordSendErrors` | 指定した期間内の、`PutRecordBatch` への呼び出しの失敗ステータス (再試行など) のレコード数。<br><br>単位: カウント |
| `ServiceErrors` | 指定した期間内の、サービスエラー (スロットリングエラーを除く) となった `PutRecordBatch` への呼び出し数。<br><br>単位: カウント |

# を使用した Amazon Data Firehose API コールのログ記録 AWS CloudTrail

Amazon Data Firehose は AWS CloudTrail、Amazon Data Firehose のユーザー、ロール、または のサービスによって実行されたアクションを記録する AWS サービスである と統合されています。CloudTrail は、Amazon Data Firehose に対するすべての API コールをイベントとしてキャプチャします。キャプチャされたコールには、Amazon Data Firehose コンソールからのコールと、Amazon Data Firehose API オペレーションへのコードコールが含まれます。証跡を作成する場合は、Amazon Data Firehose のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールの [イベント履歴] で最新のイベントを表示できます。CloudTrail により収集された情報を使用して、Amazon Data Firehose に対して行われたリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエストが行われた日時、および追加の詳細を特定することができます。

設定や有効化の方法など、CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

## CloudTrail の Firehose の情報

CloudTrail は、 AWS アカウントの作成時にアカウントで有効になります。Amazon Data Firehose でサポートされているイベントアクティビティが発生すると、そのアクティビティはイベント履歴の他の AWS サービスイベントとともに CloudTrail イベントに記録されます。最近のイベントは、AWS アカウントで表示、検索、ダウンロードできます。詳細については、CloudTrail イベント履歴でのイベントの表示を参照してください。

Amazon Data Firehose のイベントなど、 AWS アカウント内のイベントの継続的な記録については、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成すると、証跡はすべての AWS リージョンに適用されます。証跡は、 AWS パーティション内のすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析して処理するように他の AWS サービスを設定できます。詳細については、次を参照してください:

- 証跡の作成のための概要
- CloudTrail がサポートするサービスと統合
- CloudTrail 用 Amazon SNS 通知の構成

- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

Amazon Data Firehose は、CloudTrail ログファイルのイベントとして次のアクションのログ記録をサポートします。

- CreateDeliveryStream
- DeleteDeliveryStream
- DescribeDeliveryStream
- ListDeliveryStreams
- ListTagsForDeliveryStream
- TagDeliveryStream
- StartDeliveryStreamEncryption
- StopDeliveryStreamEncryption
- UntagDeliveryStream
- UpdateDestination

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストがルートまたは AWS Identity and Access Management (IAM) ユーザー認証情報を使用し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 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。

## 例: Firehose ログファイルエントリ

「トレイル」は、指定した Amazon S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail のログファイルは、単一か複数のログエントリを含みます。イベントは、任意の出典からの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。CloudTrail ログファイルは、パブリック API 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。

次は
CreateDeliveryStream、DescribeDeliveryStream、ListDeliveryStreams、UpdateDestinat
および DeleteDeliveryStream のアクションを示す CloudTrail ログエントリの例です。


```
{
  "Records":[
        {
            "eventVersion":"1.02",
            "userIdentity":{
                "type":"IAMUser",
                "principalId":"AKIAIOSFODNN7EXAMPLE",
                "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
                "accountId":"111122223333",
                "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
                "userName":"CloudTrail_Test_User"
            },
            "eventTime":"2016-02-24T18:08:22Z",
            "eventSource":"firehose.amazonaws.com",
            "eventName":"CreateDeliveryStream",
            "awsRegion":"us-east-1",
            "sourceIPAddress":"127.0.0.1",
            "userAgent":"aws-internal/3",
            "requestParameters":{
                "deliveryStreamName":"TestRedshiftStream",
                "redshiftDestinationConfiguration":{
                "s3Configuration":{
                    "compressionFormat":"GZIP",
                    "prefix":"prefix",
                    "bucketARN":"arn:aws:s3:::amzn-s3-demo-bucket",
                    "roleARN":"arn:aws:iam::111122223333:role/Firehose",
                    "bufferingHints":{
                        "sizeInMBs":3,
                        "intervalInSeconds":900
                    },
                    "encryptionConfiguration":{
                        "kMSEncryptionConfig":{
                            "aWSKMSKeyARN":"arn:aws:kms:us-east-1:key"
                        }
                    }
                },
                "clusterJDBCURL":"jdbc:redshift://example.abc123.us-
west-2.redshift.amazonaws.com:5439/dev",
                "copyCommand":{
```

```
                    "copyOptions":"copyOptions",
                    "dataTableName":"dataTable"
                },
                "password":"",
                "username":"",
                "roleARN":"arn:aws:iam::111122223333:role/Firehose"
            }
        },
        "responseElements":{
            "deliveryStreamARN":"arn:aws:firehose:us-
east-1:111122223333:deliverystream/TestRedshiftStream"
        },
        "requestID":"958abf6a-db21-11e5-bb88-91ae9617edf5",
        "eventID":"875d2d68-476c-4ad5-bbc6-d02872cfc884",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:08:54Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"DescribeDeliveryStream",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "deliveryStreamName":"TestRedshiftStream"
        },
        "responseElements":null,
        "requestID":"aa6ea5ed-db21-11e5-bb88-91ae9617edf5",
        "eventID":"d9b285d8-d690-4d5c-b9fe-d1ad5ab03f14",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
```

```
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:10:00Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"ListDeliveryStreams",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "limit":10
        },
        "responseElements":null,
        "requestID":"d1bf7f86-db21-11e5-bb88-91ae9617edf5",
        "eventID":"67f63c74-4335-48c0-9004-4ba35ce00128",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:10:09Z",
        "eventSource":"firehose.amazonaws.com",
        "eventName":"UpdateDestination",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "destinationId":"destinationId-000000000001",
            "deliveryStreamName":"TestRedshiftStream",
            "currentDeliveryStreamVersionId":"1",
            "redshiftDestinationUpdate":{
```

```
                "roleARN":"arn:aws:iam::111122223333:role/Firehose",
                "clusterJDBCURL":"jdbc:redshift://example.abc123.us-
west-2.redshift.amazonaws.com:5439/dev",
                "password":"",
                "username":"",
                "copyCommand":{
                    "copyOptions":"copyOptions",
                    "dataTableName":"dataTable"
                },
                "s3Update":{
                    "bucketARN":"arn:aws:s3:::amzn-s3-demo-bucket-update",
                    "roleARN":"arn:aws:iam::111122223333:role/Firehose",
                    "compressionFormat":"GZIP",
                    "bufferingHints":{
                        "sizeInMBs":3,
                        "intervalInSeconds":900
                    },
                    "encryptionConfiguration":{
                        "kMSEncryptionConfig":{
                            "aWSKMSKeyARN":"arn:aws:kms:us-east-1:key"
                        }
                    },
                    "prefix":"arn:aws:s3:::amzn-s3-demo-bucket"
                }
            }
        },
        "responseElements":null,
        "requestID":"d549428d-db21-11e5-bb88-91ae9617edf5",
        "eventID":"1cb21e0b-416a-415d-bbf9-769b152a6585",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    },
    {
        "eventVersion":"1.02",
        "userIdentity":{
            "type":"IAMUser",
            "principalId":"AKIAIOSFODNN7EXAMPLE",
            "arn":"arn:aws:iam::111122223333:user/CloudTrail_Test_User",
            "accountId":"111122223333",
            "accessKeyId":"AKIAI44QH8DHBEXAMPLE",
            "userName":"CloudTrail_Test_User"
        },
        "eventTime":"2016-02-24T18:10:12Z",
        "eventSource":"firehose.amazonaws.com",
```

```
        "eventName":"DeleteDeliveryStream",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"127.0.0.1",
        "userAgent":"aws-internal/3",
        "requestParameters":{
            "deliveryStreamName":"TestRedshiftStream"
        },
        "responseElements":null,
        "requestID":"d85968c1-db21-11e5-bb88-91ae9617edf5",
        "eventID":"dd46bb98-b4e9-42ff-a6af-32d57e636ad1",
        "eventType":"AwsApiCall",
        "recipientAccountId":"111122223333"
    }
  ]
}
```

# AWS SDKsコード例

次のコード例は、 AWS Software Development Kit (SDK) で Firehose 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 Firehose の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

コードの例

- AWS SDKs基本的な例
  - AWS SDKsアクション
    - AWS SDK または CLI PutRecordで を使用する
    - AWS SDK または CLI PutRecordBatchで を使用する
- AWS SDKsシナリオ
  - Amazon Data Firehose を使用して個別レコードとバッチレコードを処理する

# AWS SDKs基本的な例

次のコード例は、 AWS SDK を使用した Amazon Data Firehose の基本的な使用方法を説明しています。

例

- AWS SDKsアクション
  - AWS SDK または CLI PutRecordで を使用する
  - AWS SDK または CLI PutRecordBatchで を使用する

# AWS SDKsアクション

次のコード例は、 AWS SDKs を使用して個々の Firehose 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 Firehose 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 AWS SDKsシナリオ 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 Data Firehose API リファレンス」を参照してください。

例

- AWS SDK または CLI PutRecordで を使用する
- AWS SDK または CLI PutRecordBatchで を使用する

## AWS SDK または CLI **PutRecord**で を使用する

次のサンプルコードは、PutRecord を使用する方法を説明しています。

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。

- Firehose にレコードを配置する

CLI

AWS CLI

ストリームにレコードを書き込むには

次の put-record の例では、データをストリームに書き込みます。データは Base64 形式でエンコードされます。

```
aws firehose put-record \
    --delivery-stream-name my-stream \
    --record '{"Data":"SGVsbG8gd29ybGQ="}'
```

出力:

```
{
    "RecordId": "RjB5K/nnoGFHqwTsZlNd/
TTqvjE8V5dsyXZTQn2JXrdpMTOwssyEb6nfC8fwf1whhwnItt4mvrn+gsqeK5jB7QjuLg283+Ps4Sz/
j1Xujv31iDhnPdaLw4BOyM9Amv7PcCuB2079RuM0NhoakbyUymlwY8yt20G8X2420wu1jlFafhci4erAt7QhDEvpw
    "Encrypted": false
}
```

詳細については、「Amazon Kinesis Data Firehose ディベロッパーガイド」の「Amazon Kinesis Data Firehose 配信ストリームへのデータの送信」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「PutRecord」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。

```
   /**
    * Puts a record to the specified Amazon Kinesis Data Firehose delivery
stream.
    *
    * @param record The record to be put to the delivery stream. The record must
be a {@link Map} of String keys and Object values.
    * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
    * @throws IllegalArgumentException if the input record or delivery stream
name is null or empty.
    * @throws RuntimeException if there is an error putting the record to the
delivery stream.
    */
   public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
       if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
```

```
            throw new IllegalArgumentException("Invalid input: record or delivery
 stream name cannot be null/empty");
        }
        try {
            String jsonRecord = new ObjectMapper().writeValueAsString(record);
            Record firehoseRecord = Record.builder()

 .data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                .build();

            PutRecordRequest putRecordRequest = PutRecordRequest.builder()
                .deliveryStreamName(deliveryStreamName)
                .record(firehoseRecord)
                .build();

            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
 e);
        }
    }
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「PutRecord」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。

```
class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.
```

```
    Attributes:
        config (object): Configuration object with delivery stream name and
region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record(self, record: dict):
        """
        Put individual records to Firehose with backoff and retry.

        Args:
            record (dict): The data record to be sent to Firehose.

        This method attempts to send an individual record to the Firehose
delivery stream.
        It retries with exponential backoff in case of exceptions.
        """
        try:
            entry = self._create_record_entry(record)
            response = self.firehose.put_record(
                DeliveryStreamName=self.delivery_stream_name, Record=entry
            )
            self._log_response(response, entry)
```

```
        except Exception:
            logger.info(f"Fail record: {record}.")
            raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「PutRecord」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **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。

```
TRY.
    DATA(lo_record) = NEW /aws1/cl_frhrecord( iv_data = iv_data ).

    DATA(lo_result) = lo_frh->putrecord(
      iv_deliverystreamname = iv_deliv_stream_name
      io_record             = lo_record ).

    MESSAGE 'Record sent to Firehose delivery stream.' TYPE 'I'.
  CATCH /aws1/cx_frhresourcenotfoundex.
    MESSAGE 'Delivery stream not found.' TYPE 'E'.
  CATCH /aws1/cx_frhinvalidargumentex.
    MESSAGE 'Invalid argument provided.' TYPE 'E'.
  CATCH /aws1/cx_frhserviceunavailex.
    MESSAGE 'Service temporarily unavailable.' TYPE 'E'.
ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「PutRecord」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 Firehose 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 **PutRecordBatch**で を使用する

次のサンプルコードは、PutRecordBatch を使用する方法を説明しています。

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。

- Firehose にレコードを配置する

CLI

AWS CLI

複数のレコードをストリームに書き込むには

この put-record-batch の例は、3 つのレコードをストリームに書き込みます。データは Base64 形式でエンコードされます。

```
aws firehose put-record-batch \
    --delivery-stream-name my-stream \
    --records file://records.json
```

myfile.json の内容:

```
[
    {"Data": "Rmlyc3QgdGhpbmc="},
    {"Data": "U2Vjb25kIHRoaW5n"},
    {"Data": "VGhpcmQgdGhpbmc="}
]
```

出力:

```
{
    "FailedPutCount": 0,
    "Encrypted": false,
    "RequestResponses": [
```

```
        {
            "RecordId": "9D2OJ6t2EqCTZTXwGzeSv/EVHxRoRCw89xd+o3+sXg8DhYOaWKPSmZy/
CGlRVEys1u1xbeKh6VofEYKkoeiDrcjrxhQp9iF7sUW7pujiMEQ5LzlrzCkGosxQn
+3boDnURDEaD42V7GiixpOyLJkYZcae1i7HzlCEoy9LJhMr8EjDSi4Om/9Vc2uhwwuAtGE0XKpxJ2WD7ZRWtAnYlK
        },
        {
            "RecordId": "jFirejqxCLlK5xjH/UNmlMVcjktEN76I7916X9PaZ
+PVaOSXDfU1WGOqEZhxq2js7xcZ552eoeDxsuTU1MSq9nZTbVfb6cQTIXnm/
GsuF37Uhg67GKmR5z90l6XKJ+/
+pDloFv7Hh9a3oUS6wYm3DcNRLTHHAimANp1PhkQvWpvLRfzbuCUkBphR2QVzhP9OiHLbzGwy8/
DfH8sqWEUYASNJKS8GXP5s"
        },
        {
            "RecordId":
 "oy0amQ40o5Y2YV4vxzufdcMOOw6n3EPr3tpPJGoYVNKH4APPVqNcbUgefo1stEFRg4hTLrf2k6eliHu/9+YJ5R3
DTBt3qBlmTj7Xq8SKVbO1S7YvMTpWkMKA86f8JfmT8BMKoMb4XZS/sOkQLe+qh0sYKXWl"
        }
    ]
}
```

詳細については、「Amazon Kinesis Data Firehose ディベロッパーガイド」の「Amazon Kinesis Data Firehose 配信ストリームへのデータの送信」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「PutRecordBatch」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。

```
    /**
     * Puts a batch of records to an Amazon Kinesis Data Firehose delivery
stream.
     *
     * @param records           a list of maps representing the records to be
sent
```

```
    * @param batchSize         the maximum number of records to include in each
batch
    * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
    * @throws IllegalArgumentException if the input parameters are invalid (null
or empty)
    * @throws RuntimeException         if there is an error putting the record
batch
    */
   public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
       if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: records or
delivery stream name cannot be null/empty");
       }
       ObjectMapper objectMapper = new ObjectMapper();

       try {
           for (int i = 0; i < records.size(); i += batchSize) {
               List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

               List<Record> batchRecords = batch.stream().map(record -> {
                   try {
                       String jsonRecord =
objectMapper.writeValueAsString(record);
                       return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                           .build();
                   } catch (Exception e) {
                       throw new RuntimeException("Error creating Firehose
record", e);
                   }
               }).collect(Collectors.toList());

               PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                   .deliveryStreamName(deliveryStreamName)
                   .records(batchRecords)
                   .build();

               PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);
```

```
                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " +
batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「PutRecordBatch」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。

```
class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
```

```
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record_batch(self, data: list, batch_size: int = 500):
        """
        Put records in batches to Firehose with backoff and retry.

        Args:
            data (list): List of data records to be sent to Firehose.
            batch_size (int): Number of records to send in each batch. Default is
500.

        This method attempts to send records in batches to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        for i in range(0, len(data), batch_size):
            batch = data[i : i + batch_size]
            record_dicts = [{"Data": json.dumps(record)} for record in batch]
            try:
                response = self.firehose.put_record_batch(
                    DeliveryStreamName=self.delivery_stream_name,
Records=record_dicts
                )
                self._log_batch_response(response, len(batch))
            except Exception as e:
```

```
                logger.info(f"Failed to send batch of {len(batch)} records.
Error: {e}")
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「PutRecordBatch」を参照してください。

Rust

SDK for Rust

> **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。

```
async fn put_record_batch(
    client: &Client,
    stream: &str,
    data: Vec<Record>,
) -> Result<PutRecordBatchOutput, SdkError<PutRecordBatchError>> {
    client
        .put_record_batch()
        .delivery_stream_name(stream)
        .set_records(Some(data))
        .send()
        .await
}
```

- API の詳細については、AWS SDK for Rust API リファレンスの「PutRecordBatch」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **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。

```
    TRY.
        DATA(lo_result) = lo_frh->putrecordbatch(
          iv_deliverystreamname = iv_deliv_stream_name
          it_records            = it_records ).

        DATA(lv_failed_count) = lo_result->get_failedputcount( ).

        IF lv_failed_count > 0.
          MESSAGE |{ lv_failed_count } records failed to send.| TYPE 'I'.
        ELSE.
          MESSAGE 'All records sent successfully to Firehose delivery stream.'
TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_frhresourcenotfoundex.
        MESSAGE 'Delivery stream not found.' TYPE 'E'.
      CATCH /aws1/cx_frhinvalidargumentex.
        MESSAGE 'Invalid argument provided.' TYPE 'E'.
      CATCH /aws1/cx_frhserviceunavailex.
        MESSAGE 'Service temporarily unavailable.' TYPE 'E'.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスの「PutRecordBatch」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 Firehose の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

# AWS SDKsシナリオ

次のコード例は、 AWS SDKs を使用して Firehose で一般的なシナリオを実装する方法を示しています。これらのシナリオは、Firehose 内で複数の関数を呼び出すか、他の AWS のサービスと組み合わせて、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

- Amazon Data Firehose を使用して個別レコードとバッチレコードを処理する

## Amazon Data Firehose を使用して個別レコードとバッチレコードを処理する

以下のコード例は、Firehose を使用して個別レコードとバッチレコードを処理する方法を示しています。

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。

この例では、個別レコードとバッチレコードを Firehose に配置します。

```
/**
 * Amazon Firehose Scenario example using Java V2 SDK.
 *
 * Demonstrates individual and batch record processing,
 * and monitoring Firehose delivery stream metrics.
 */
public class FirehoseScenario {
```

```
    private static FirehoseClient firehoseClient;
    private static CloudWatchClient cloudWatchClient;

    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <deliveryStreamName>
                Where:
                    deliveryStreamName - The Firehose delivery stream name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String deliveryStreamName = args[0];

        try {
            // Read and parse sample data.
            String jsonContent = readJsonFile("sample_records.json");
            ObjectMapper objectMapper = new ObjectMapper();
            List<Map<String, Object>> sampleData =
objectMapper.readValue(jsonContent, new TypeReference<>() {});

            // Process individual records.
            System.out.println("Processing individual records...");
            sampleData.subList(0, 100).forEach(record -> {
                try {
                    putRecord(record, deliveryStreamName);
                } catch (Exception e) {
                    System.err.println("Error processing record: " +
e.getMessage());
                }
            });

            // Monitor metrics.
            monitorMetrics(deliveryStreamName);

            // Process batch records.
            System.out.println("Processing batch records...");
            putRecordBatch(sampleData.subList(100, sampleData.size()), 500,
deliveryStreamName);
            monitorMetrics(deliveryStreamName);
```

```
        } catch (Exception e) {
            System.err.println("Scenario failed: " + e.getMessage());
        } finally {
            closeClients();
        }
    }

    private static FirehoseClient getFirehoseClient() {
        if (firehoseClient == null) {
            firehoseClient = FirehoseClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return firehoseClient;
    }

    private static CloudWatchClient getCloudWatchClient() {
        if (cloudWatchClient == null) {
            cloudWatchClient = CloudWatchClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return cloudWatchClient;
    }

    /**
     * Puts a record to the specified Amazon Kinesis Data Firehose delivery
stream.
     *
     * @param record The record to be put to the delivery stream. The record must
be a {@link Map} of String keys and Object values.
     * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
     * @throws IllegalArgumentException if the input record or delivery stream
name is null or empty.
     * @throws RuntimeException if there is an error putting the record to the
delivery stream.
     */
    public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
        if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
```

```
            throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
        }
        try {
            String jsonRecord = new ObjectMapper().writeValueAsString(record);
            Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                .build();

            PutRecordRequest putRecordRequest = PutRecordRequest.builder()
                .deliveryStreamName(deliveryStreamName)
                .record(firehoseRecord)
                .build();

            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
        }
    }


    /**
     * Puts a batch of records to an Amazon Kinesis Data Firehose delivery
stream.
     *
     * @param records           a list of maps representing the records to be
sent
     * @param batchSize         the maximum number of records to include in each
batch
     * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
     * @throws IllegalArgumentException if the input parameters are invalid (null
or empty)
     * @throws RuntimeException         if there is an error putting the record
batch
     */
    public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
        if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
```

```
            throw new IllegalArgumentException("Invalid input: records or
delivery stream name cannot be null/empty");
        }
        ObjectMapper objectMapper = new ObjectMapper();

        try {
            for (int i = 0; i < records.size(); i += batchSize) {
                List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

                List<Record> batchRecords = batch.stream().map(record -> {
                    try {
                        String jsonRecord =
objectMapper.writeValueAsString(record);
                        return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                                .build();
                    } catch (Exception e) {
                        throw new RuntimeException("Error creating Firehose
record", e);
                    }
                }).collect(Collectors.toList());

                PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                    .deliveryStreamName(deliveryStreamName)
                    .records(batchRecords)
                    .build();

                PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " +
batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
```

```
        }
    }

    public static void monitorMetrics(String deliveryStreamName) {
        Instant endTime = Instant.now();
        Instant startTime = endTime.minusSeconds(600);

        List<String> metrics = List.of("IncomingBytes", "IncomingRecords",
"FailedPutCount");
        metrics.forEach(metric -> monitorMetric(metric, startTime, endTime,
deliveryStreamName));
    }

    private static void monitorMetric(String metricName, Instant startTime,
Instant endTime, String deliveryStreamName) {
        try {
            GetMetricStatisticsRequest request =
GetMetricStatisticsRequest.builder()
                .namespace("AWS/Firehose")
                .metricName(metricName)

.dimensions(Dimension.builder().name("DeliveryStreamName").value(deliveryStreamName).bui
                .startTime(startTime)
                .endTime(endTime)
                .period(60)
                .statistics(Statistic.SUM)
                .build();

            GetMetricStatisticsResponse response =
getCloudWatchClient().getMetricStatistics(request);
            double totalSum =
response.datapoints().stream().mapToDouble(Datapoint::sum).sum();
            System.out.println(metricName + ": " + totalSum);
        } catch (Exception e) {
            System.err.println("Failed to monitor metric " + metricName + ": " +
e.getMessage());
        }
    }

    public static String readJsonFile(String fileName) throws IOException {
        try (InputStream inputStream =
FirehoseScenario.class.getResourceAsStream("/" + fileName);
             Scanner scanner = new Scanner(inputStream, StandardCharsets.UTF_8))
{
```

```
            return scanner.useDelimiter("\\\\A").next();
        } catch (Exception e) {
            throw new RuntimeException("Error reading file: " + fileName, e);
        }
    }

    private static void closeClients() {
        try {
            if (firehoseClient != null) firehoseClient.close();
            if (cloudWatchClient != null) cloudWatchClient.close();
        } catch (Exception e) {
            System.err.println("Error closing clients: " + e.getMessage());
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - PutRecord
  - PutRecordBatch

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。

このスクリプトは、個別レコードとバッチレコードを Firehose に配置します。

```
import json
import logging
import random
from datetime import datetime, timedelta

import backoff
import boto3
```

```
from config import get_config


def load_sample_data(path: str) -> dict:
    """
    Load sample data from a JSON file.

    Args:
        path (str): The file path to the JSON file containing sample data.

    Returns:
        dict: The loaded sample data as a dictionary.
    """
    with open(path, "r") as f:
        return json.load(f)


# Configure logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
 region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
 region.
        """
```

```
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record(self, record: dict):
        """
        Put individual records to Firehose with backoff and retry.

        Args:
            record (dict): The data record to be sent to Firehose.

        This method attempts to send an individual record to the Firehose
delivery stream.
        It retries with exponential backoff in case of exceptions.
        """
        try:
            entry = self._create_record_entry(record)
            response = self.firehose.put_record(
                DeliveryStreamName=self.delivery_stream_name, Record=entry
            )
            self._log_response(response, entry)
        except Exception:
            logger.info(f"Fail record: {record}.")
            raise


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record_batch(self, data: list, batch_size: int = 500):
        """
        Put records in batches to Firehose with backoff and retry.

        Args:
            data (list): List of data records to be sent to Firehose.
            batch_size (int): Number of records to send in each batch. Default is
500.
```

```
        This method attempts to send records in batches to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        for i in range(0, len(data), batch_size):
            batch = data[i : i + batch_size]
            record_dicts = [{"Data": json.dumps(record)} for record in batch]
            try:
                response = self.firehose.put_record_batch(
                    DeliveryStreamName=self.delivery_stream_name,
Records=record_dicts
                )
                self._log_batch_response(response, len(batch))
            except Exception as e:
                logger.info(f"Failed to send batch of {len(batch)} records.
Error: {e}")


   def get_metric_statistics(
        self,
        metric_name: str,
        start_time: datetime,
        end_time: datetime,
        period: int,
        statistics: list = ["Sum"],
   ) -> list:
        """
        Retrieve metric statistics from CloudWatch.

        Args:
            metric_name (str): The name of the metric.
            start_time (datetime): The start time for the metric statistics.
            end_time (datetime): The end time for the metric statistics.
            period (int): The granularity, in seconds, of the returned data
points.
            statistics (list): A list of statistics to retrieve. Default is
['Sum'].

        Returns:
            list: List of datapoints containing the metric statistics.
        """
        response = self.cloudwatch.get_metric_statistics(
            Namespace="AWS/Firehose",
            MetricName=metric_name,
```

```
            Dimensions=[
                {"Name": "DeliveryStreamName", "Value":
self.delivery_stream_name},
            ],
            StartTime=start_time,
            EndTime=end_time,
            Period=period,
            Statistics=statistics,
        )
        return response["Datapoints"]

    def monitor_metrics(self):
        """
        Monitor Firehose metrics for the last 5 minutes.

        This method retrieves and logs the 'IncomingBytes', 'IncomingRecords',
and 'FailedPutCount' metrics
        from CloudWatch for the last 5 minutes.
        """
        end_time = datetime.utcnow()
        start_time = end_time - timedelta(minutes=10)
        period = int((end_time - start_time).total_seconds())

        metrics = {
            "IncomingBytes": self.get_metric_statistics(
                "IncomingBytes", start_time, end_time, period
            ),
            "IncomingRecords": self.get_metric_statistics(
                "IncomingRecords", start_time, end_time, period
            ),
            "FailedPutCount": self.get_metric_statistics(
                "FailedPutCount", start_time, end_time, period
            ),
        }

        for metric, datapoints in metrics.items():
            if datapoints:
                total_sum = sum(datapoint["Sum"] for datapoint in datapoints)
                if metric == "IncomingBytes":
                    logger.info(
                        f"{metric}: {round(total_sum)} ({total_sum / (1024 *
1024):.2f} MB)"
                    )
                else:
```

```
                    logger.info(f"{metric}: {round(total_sum)}")
            else:
                logger.info(f"No data found for {metric} over the last 5
minutes")


    def _create_record_entry(self, record: dict) -> dict:
        """
        Create a record entry for Firehose.

        Args:
            record (dict): The data record to be sent.

        Returns:
            dict: The record entry formatted for Firehose.

        Raises:
            Exception: If a simulated network error occurs.
        """
        if random.random() < 0.2:
            raise Exception("Simulated network error")
        elif random.random() < 0.1:
            return {"Data": '{"malformed": "data"'}
        else:
            return {"Data": json.dumps(record)}

    def _log_response(self, response: dict, entry: dict):
        """
        Log the response from Firehose.

        Args:
            response (dict): The response from the Firehose put_record API call.
            entry (dict): The record entry that was sent.
        """
        if response["ResponseMetadata"]["HTTPStatusCode"] == 200:
            logger.info(f"Sent record: {entry}")
        else:
            logger.info(f"Fail record: {entry}")

    def _log_batch_response(self, response: dict, batch_size: int):
        """
        Log the batch response from Firehose.

        Args:
```

```
            response (dict): The response from the Firehose put_record_batch API
 call.
            batch_size (int): The number of records in the batch.
        """
        if response.get("FailedPutCount", 0) > 0:
            logger.info(
                f'Failed to send {response["FailedPutCount"]} records in batch of
 {batch_size}'
            )
        else:
            logger.info(f"Successfully sent batch of {batch_size} records")


if __name__ == "__main__":
    config = get_config()
    data = load_sample_data(config.sample_data_file)
    client = FirehoseClient(config)

    # Process the first 100 sample network records
    for record in data[:100]:
        try:
            client.put_record(record)
        except Exception as e:
            logger.info(f"Put record failed after retries and backoff: {e}")
    client.monitor_metrics()

    # Process remaining records using the batch method
    try:
        client.put_record_batch(data[100:])
    except Exception as e:
        logger.info(f"Put record batch failed after retries and backoff: {e}")
    client.monitor_metrics()
```

このファイルには、上記のスクリプトの設定が含まれています。

```
class Config:
    def __init__(self):
        self.delivery_stream_name = "ENTER YOUR DELIVERY STREAM NAME HERE"
        self.region = "us-east-1"
        self.sample_data_file = (
            "../../../../../scenarios/features/firehose/resources/
sample_records.json"
```

```
        )


def get_config():
    return Config()
```

- 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。
  - PutRecord
  - PutRecordBatch

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 Firehose 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 Data Firehose でのエラーをトラブルシューティングする

Firehose は、データの配信中や処理中にエラーが発生すると、設定された再試行期間が終了するまで再試行を続けます。データが正常に配信される前に再試行期間が終了すると、Firehose は設定された S3 バックアップバケットにデータをバックアップします。配信先が Amazon S3 である場合に配信が失敗するか、バックアップ S3 バケットへの配信が失敗すると、Firehose は保持期間が終了するまで再試行を続けます。

CloudWatch を使用した配信エラーの追跡については、「the section called “CloudWatch Logs でモニタリングする”」を参照してください。

Direct PUT

DirectPut Firehose ストリームの場合、Firehose はレコードを 24 時間保持します。データソースが Kinesis データストリームである Firehose ストリームの場合、「データ保持期間の変更」の説明に従って、保持期間を変更できます。この場合、Firehose は、DescribeStream、GetRecords、および GetShardIterator のオペレーションを無期限に再試行します。

Firehose ストリームで DirectPut が使用されている場合は、IncomingBytes メトリクスと IncomingRecords メトリクスをチェックして、着信トラフィックの有無を確認します。PutRecord または PutRecordBatch を使用している場合は、例外をキャッチして再試行してください。ジッターと複数の再試行を使用するエクスポネンシャルバックオフによる再試行ポリシーをお勧めします。また、PutRecordBatch API を使用する場合は、API コールが成功した場合でも、レスポンスの FailedPutCount の値をコードによって確認してください。

Kinesis Data Stream

Firehose ストリームがそのソースとして Kinesis データストリームを使用する場合は、ソースデータストリームの IncomingBytes メトリクスと IncomingRecords メトリクスを確認します。さらに、Firehose ストリームに関して DataReadFromKinesisStream.Bytes メトリクスと DataReadFromKinesisStream.Records メトリクスが生成されていることを確認します。

## 一般的な問題

Firehose ストリームの使用中に一般的な問題を解決するのに役立つトラブルシューティングのヒントを次に示します。

## Firehose ストリームを使用できません

一部のサービスは同じ にある Firehose ストリームにのみメッセージとイベントを送信できるため、Firehose ストリームは CloudWatch Logs、CloudWatch Events、または AWS IoT アクション AWS のターゲットとして使用できません AWS リージョン。Firehose ストリームが他のサービスと同じリージョンにあることを確認します。

## 宛先にデータがありません

データ取り込みの問題がなく、Firehose ストリームに関して生成されるメトリクスが正常と思われる場合でも、宛先にデータが表示されないときは、リーダーロジックを確認します。リーダーがすべてのデータを正しく解析していることを確認します。

## データの鮮度メトリクスの増加または未発行

データの鮮度は、Firehose ストリーム内のデータがどの程度最新であるかを示す指標です。これは、Firehose ストリーム内の最も古いデータレコードの経過時間であり、Firehose がそのデータを取り込んだ時点から現在の時刻までの時間で測定されます。Firehose は、データの鮮度をモニタリングするために使用できるメトリクスを提供します。特定の配信先に関するデータの鮮度メトリクスを確認するには、「the section called “CloudWatch メトリクスによるモニタリング”」を参照してください。

すべてのイベントまたはすべてのドキュメントのバックアップを有効にする場合は、メイン配信先用とバックアップ用に 2 つのデータの鮮度メトリクスを別個にモニタリングします。

データの鮮度メトリクスが生成されていない場合は、Firehose ストリームにアクティブな配信がないことを意味します。これは、データ配信が完全にブロックされているか、着信データがない場合に発生します。

データの鮮度メトリクスが増え続けている場合は、データ配信が遅れていることを意味します。これは、次のいずれかの理由で発生します。

- 配信先が配信率に対応できない。トラフィックが多いために Firehose で一時的なエラーが発生すると、配信が遅れる場合があります。これは、Amazon S3 以外の配信先で発生しがちです (OpenSearch Service、Amazon Redshift、または Splunk で発生することがあります)。配信先に、着信トラフィックを処理するための十分な容量があることを確認します。
- 配信先が遅い。Firehose で高レイテンシーが発生すると、データ配信が遅れることがあります。配信先のレイテンシーメトリクスをモニタリングします。

- Lambda 関数が遅い。これにより、データ配信レートが Firehose ストリームのデータ取り込みレートよりも低くなる場合があります。可能であれば、Lambda 関数の効率を向上させます。たとえば、関数がネットワーク IO を実行する場合は、複数のスレッドまたは非同期 IO を使用して並列処理を増やします。また、Lambda 関数のメモリサイズを大きくすることで CPU 割り当てを相応に増やします。これにより、Lambda 呼び出しが高速化される場合があります。Lambda 関数の設定については、AWS 「Lambda 関数の設定」を参照してください。
- データ配信中に障害が発生した。Amazon CloudWatch Logs を使用してエラーをモニタリングする方法については、「the section called “CloudWatch Logs でモニタリングする”」を参照してください。
- Firehose ストリームのデータソースが Kinesis データストリームである場合、スロットリングが発生している可能性があります。`ThrottledGetRecords` メトリクス、`ThrottledGetShardIterator` メトリクス、および `ThrottledDescribeStream` メトリクスを確認します。Kinesis データストリームに複数のコンシューマがアタッチされている場合は、以下を考慮してください。
  - `ThrottledGetRecords` メトリクスと `ThrottledGetShardIterator` メトリクスが高い場合は、データストリーム用にプロビジョニングするシャードの数を増やすことをお勧めします。
  - `ThrottledDescribeStream` が高い場合、KinesisStreamSourceConfiguration で設定されたロールに対する `kinesis:listshards` アクセス許可を追加することをお勧めします。
- 配信先のバッファリングヒントが低い。これにより、宛先に対して Firehose が行う必要があるラウンドトリップの数が増える場合があります。バッファリングヒントの値を大きくすることを検討します。詳細については、「BufferingHints」を参照してください。
- 再試行期間が長くなると、エラーの発生数が増えたときに配信が遅れる場合があります。再試行期間を短縮することを検討してください。また、エラーをモニタリングし、エラーを減らしてください。Amazon CloudWatch Logs を使用してエラーをモニタリングする方法については、「the section called “CloudWatch Logs でモニタリングする”」を参照してください。
- 配信先が Splunk である場合、`DeliveryToSplunk.DataFreshness` が高くても `DeliveryToSplunk.Success` が適切であると思われるときは、Splunk クラスターがビジー状態である可能性があります。可能であれば Splunk クラスターを解放します。または、 AWS サポートに連絡して、Splunk クラスターとの通信に Firehose が使用するチャネルの数の増加をリクエストしてください。

## Apache Parquet へのレコード形式変換が失敗する

これは、 Setタイプを含む DynamoDB データを取得し、Lambda を介して Firehose ストリームにストリーミングし、 AWS Glue Data Catalog を使用してレコード形式を Apache Parquet に変換した場合に発生します。

AWS Glue クローラが DynamoDB セットのデータ型 (StringSet、NumberSet、および BinarySet) にインデックスを付けるとSET<BINARY>、それぞれ SET<STRING>、SET<BIGINT>、および としてデータカタログに保存されます。ただし、Firehose がデータレコードを Apache Parquet 形式に変換するには、Apache Hive データ型が必要です。セット型は有効な Apache Hive データ型ではないため、変換は失敗します。変換を機能させるには、Apache Hive データ型でデータカタログを更新します。そのためには、データカタログの set を array に変更します。

AWS Glue データカタログ**array**で 1 つ以上のデータ型**set**を から に変更するには

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/glue/ で AWS Glue コンソールを開きます。
2. 左側のペインの [データカタログ] 見出しにある [テーブル] を選択します。
3. テーブルのリストで、1 つ以上のデータ型を変更する必要があるテーブルの名前を選択します。そのテーブルの詳細ページが表示されます。
4. 詳細ページの右上隅にある [Edit schema] ボタンを選択します。
5. [データ型] 列で、最初のデータ型 set を選択します。
6. [Column type] ドロップダウンリストで、set から array に型を変更します。
7. [ArraySchema] に、array<string>、 array<int>、または array<binary> と入力します。これは、シナリオの適切なデータ型によって異なります。
8. [更新] を選択します。
9. 他の set 型を array 型に変換するには、前のステップを繰り返します。
10. [保存] を選択します。

## Lambda のために変換されたオブジェクトのフィールドが欠落しています

Lambda データ変換を使用して JSON データを Parquet オブジェクトに変換すると、変換後に一部のフィールドが欠落している可能性があります。これは、JSON オブジェクトに大文字があり、大文字と小文字の区別が false に設定されている場合に発生します。これにより、データ変換後に

JSON キーに不一致が発生し、S3 バケット内の、結果として得られる Parquet オブジェクトでデータが欠落する可能性があります。

これを修復するには、変換後に JSON キーが一致するように、ホース設定で `deserializationOption: case.insensitive` が `true` に設定されていることを確認してください。

## Amazon S3 トラブルシューティング

Amazon Simple Storage Service (Amazon S3) バケットにデータが配信されない場合は、次の点を確認してください。

- Firehose の `IncomingBytes` および `IncomingRecords` メトリクスで、データが Firehose ストリームに正常に送信されていることを確認します。詳細については、「CloudWatch メトリクスを使用して Amazon Data Firehose をモニタリングする」を参照してください。
- Lambda によるデータ変換が有効な場合は、Firehose の `ExecuteProcessingSuccess` メトリクスで、Firehose が Lambda 関数を呼び出そうとしたことを確認します。詳細については、「CloudWatch メトリクスを使用して Amazon Data Firehose をモニタリングする」を参照してください。
- Firehose の `DeliveryToS3.Success` メトリクスで、Firehose が Amazon S3 バケットにデータを配置しようとしたことを確認します。詳細については、「CloudWatch メトリクスを使用して Amazon Data Firehose をモニタリングする」を参照してください。
- すでに有効になっていない場合はエラーログ記録を有効にし、配信の失敗のエラーログを確認します。詳細については、「CloudWatch Logs を使用して Amazon Data Firehose をモニタリングする」を参照してください。
- 「Firehose が Amazon S3 サービスを呼び出す際に InternalServerError が発生しました。オペレーションは再試行されます。エラーが解決しない場合は、解決方法について S3 にお問い合わせください」というエラーメッセージがログに表示された場合、S3 の 1 つのパーティションでリクエストレートが大幅に増加したことが理由である可能性があります。S3 プレフィックスの設計パターンを最適化して、問題を軽減できます。詳細については、設計パターンのベストプラクティス: Simple Storage Service (Amazon S3) のパフォーマンスの最適化 を参照してください。それでも問題が解決しない場合は、 AWS サポートにお問い合わせください。
- Firehose ストリームで指定した Amazon S3 バケットがまだ存在することを確認します。
- Lambda によるデータ変換が有効な場合は、Firehose ストリームで指定した Lambda 関数がまだ存在することを確認します。

- Firehose ストリームで指定した IAM ロールに、S3 バケットに対するアクセスと、Lambda 関数に対するアクセス (データ変換が有効な場合) があることを確認します。また、IAM ロールがエラーログをチェックするために、CloudWatch ロググループとログストリームにアクセスできることを確認します。詳細については、「Firehose に Amazon S3 宛先へのアクセスを付与する」を参照してください。
- データ変換を使用している場合は、Lambda 関数が、6 MB を超えるペイロードサイズのレスポンスを返さないようにします。詳細については、「Amazon Data Firehose のデータ変換」を参照してください。

## Amazon Redshift のトラブルシューティング

Amazon Redshift プロビジョンドクラスターまたは Amazon Redshift Serverless ワークグループにデータが配信されない場合は、以下を確認してください。

データは、Amazon Redshift にロードされる前に S3 バケットに配信されます。データが S3 バケットに配信されなかった場合は、「Amazon S3 トラブルシューティング」を参照してください。

- Firehose の `DeliveryToRedshift.Success` メトリクスで、Firehose が S3 バケットから Amazon Redshift プロビジョンドクラスターまたは Amazon Redshift Serverless ワークグループにデータをコピーしようとしていたことを確認します。詳細については、「CloudWatch メトリクスを使用して Amazon Data Firehose をモニタリングする」を参照してください。
- すでに有効になっていない場合はエラーログ記録を有効にし、配信の失敗のエラーログを確認します。詳細については、「CloudWatch Logs を使用して Amazon Data Firehose をモニタリングする」を参照してください。
- Amazon Redshift の `STL_CONNECTION_LOG` テーブルで、Firehose が正常な接続を確立できるかどうかを確認します。このテーブルには、ユーザー名に基づく接続と接続ステータスが表示されます。詳細については、Amazon Redshift データベース開発者ガイドの「`STL_CONNECTION_LOG`」を参照してください。
- 前のチェックで、接続が確立されていることを確認したら、Amazon Redshift の `STL_LOAD_ERRORS` テーブルで、COPY の失敗の理由を確認します。詳細については、Amazon Redshift データベース開発者ガイドの「`STL_LOAD_ERRORS`」を参照してください。
- Firehose ストリームの Amazon Redshift 設定が正確で有効であることを確認します。
- Firehose ストリームで指定した IAM ロールに、Amazon Redshift によるデータのコピー元となる S3 バケットに対するアクセスと、データ変換用の Lambda 関数に対するアクセス (データ変換が有効な場合) もあることを確認します。また、IAM ロールがエラーログをチェックするため

に、CloudWatch ロググループとログストリームにアクセスできることを確認します。詳細については、「Amazon Redshift の宛先へのアクセスを Firehose に付与する」を参照してください。

- Amazon Redshift プロビジョンドクラスターまたは Amazon Redshift Serverless ワークグループが仮想プライベートクラウド (VPC) 内にある場合は、クラスターで Firehose の IP アドレスからのアクセスが許可されていることを確認します。詳細については、「Amazon Redshift の宛先へのアクセスを Firehose に付与する」を参照してください。
- Amazon Redshift プロビジョンドクラスターまたは Amazon Redshift Serverless ワークグループがパブリックにアクセス可能であることを確認します。
- データ変換を使用している場合は、Lambda 関数が、6 MB を超えるペイロードサイズのレスポンスを返さないようにします。詳細については、「Amazon Data Firehose のデータ変換」を参照してください。

## Amazon OpenSearch Service のトラブルシューティング

OpenSearch Service ドメインにデータが配信されない場合は、以下の点を確認してください。

データは配信と同時に Amazon S3 バケットにバックアップできます。S3 バケットにデータが配信されなかった場合は、「Amazon S3 トラブルシューティング」を参照してください。

- Firehose の `IncomingBytes` および `IncomingRecords` メトリクスで、データが Firehose ストリームに正常に送信されていることを確認します。詳細については、「CloudWatch メトリクスを使用して Amazon Data Firehose をモニタリングする」を参照してください。
- Lambda によるデータ変換が有効な場合は、Firehose の `ExecuteProcessingSuccess` メトリクスで、Firehose が Lambda 関数を呼び出そうとしたことを確認します。詳細については、「CloudWatch メトリクスを使用して Amazon Data Firehose をモニタリングする」を参照してください。
- Firehose の `DeliveryToAmazonOpenSearchService.Success` メトリクスで、Firehose が OpenSearch Service クラスターにデータのインデックスを作成しようとしたことを確認します。詳細については、「CloudWatch メトリクスを使用して Amazon Data Firehose をモニタリングする」を参照してください。
- すでに有効になっていない場合はエラーログ記録を有効にし、配信の失敗のエラーログを確認します。詳細については、「CloudWatch Logs を使用して Amazon Data Firehose をモニタリングする」を参照してください。
- Firehose ストリームの OpenSearch Service 設定が正確で有効であることを確認します。

- Lambda によるデータ変換が有効な場合は、Firehose ストリームで指定した Lambda 関数がまだ存在することを確認します。また、IAM ロールがエラーログをチェックするために、CloudWatch ロググループとログストリームにアクセスできることを確認します。詳細については、「Grant FirehoseAccess to a Public OpenSearch Service Destination」を参照してください。
- Firehose ストリームで指定した IAM ロールに、OpenSearch Service クラスター、S3 バックアップバケット、Lambda 関数に対するアクセス (データ変換が有効な場合) があることを確認します。また、IAM ロールがエラーログをチェックするために、CloudWatch ロググループとログストリームにアクセスできることを確認します。詳細については、「Firehose に公開 OpenSearch Service の宛先へのアクセスを付与する」を参照してください。
- データ変換を使用している場合は、Lambda 関数が、6 MB を超えるペイロードサイズのレスポンスを返さないようにします。詳細については、「Amazon Data Firehose のデータ変換」を参照してください。
- Amazon Data Firehose は現在、Amazon OpenSearch Service の宛先への CloudWatch Logs の配信をサポートしていません。Amazon CloudWatch が複数のログイベントを 1 つの Firehose レコードにまとめても、Amazon OpenSearch Service は 1 つのレコードで複数のログイベントを受け入れることができないためです。代わりの方法として、CloudWatch Logs で Amazon OpenSearch Service のサブスクリプションフィルターを使用することもできます。

## Splunk のトラブルシューティング

Splunk エンドポイントにデータが配信されない場合は、以下の点を確認してください。

- Splunk プラットフォームが VPC にある場合は、必ず Firehose がアクセスできることを確認します。詳細については、「VPC の Splunk へのアクセス」を参照してください。
- AWS ロードバランサーを使用する場合は、それが Classic Load Balancer または Application Load Balancer であることを確認します。また、Classic Load Balancer の Cookie の有効期限を無効にした状態で、期間ベースのスティッキーセッションを有効にし、Application Load Balancer の有効期限を最大値 (7 日間) に設定します。これを行う方法の詳細については、Classic Load Balancer または Application Load Balancer の「Duration-Based Session Stickiness」を参照してください。
- Splunk プラットフォームの要件を確認します。Firehose 用の Splunk アドオンには、Splunk プラットフォームバージョン 6.6.X 以降が必要です。詳細については、「Splunk Add-on for Amazon Kinesis Firehose」を参照してください。
- Firehose と HTTP Event Collector (HEC) ノードの間にプロキシ (Elastic Load Balancing など) がある場合は、スティッキーセッションを有効にして HEC 確認応答 (ACK) をサポートします。
- 有効な HEC トークンを使用していることを確認します。

- HEC トークンが有効であることを確認します。
- Splunk に送信しているデータの形式が正しいかどうかを確認します。詳細については、「Format events for HTTP Event Collector」を参照してください。
- 有効なインデックスを使用して HEC トークンと入力イベントが設定されていることを確認します。
- HEC ノードのサーバーエラーのため Splunk へのアップロードが失敗すると、リクエストは自動的に再試行されます。すべての再試行が失敗すると、データは Amazon S3 にバックアップされます。データが Amazon S3 にあるかどうかを確認します。ある場合、そのような障害が発生したことを示します。
- HEC トークンでインデクサの送達確認が有効であることを確認します。
- Firehose ストリームの Splunk 宛先設定で、`HECAcknowledgmentTimeoutInSeconds` の値を増やします。
- Firehose ストリームの Splunk 宛先設定で、`RetryOptions` の `DurationInSeconds` の値を増やします。
- HEC のヘルスを確認します。ヘルスチェックを有効にすることは、Splunk へのデータ転送の前提条件です。
- データ変換を使用している場合は、Lambda 関数が、6 MB を超えるペイロードサイズのレスポンスを返さないようにします。詳細については、「Amazon Data Firehose Data Transformation」を参照してください。
- `ackIdleCleanup` と言う名前の Splunk パラメータが、`true` に設定されていることを確認します。これはデフォルトでは false です。このパラメータを `true` に設定するには、次のことを行います。
  - マネージド型の Splunk Cloud デプロイの場合は、Splunk サポートポータルを使用してケースを送信します。その場合は、HTTP イベントコレクターを有効にし、`ackIdleCleanup` で `true` を `inputs.conf` に設定して、このアドオンを使用するようにロードバランサーを作成または変更することを Splunk サポートに依頼します。
  - 分散された Splunk Enterprise デプロイの場合は、`inputs.conf` ファイルで `ackIdleCleanup` パラメータを true に設定します。*nix ユーザーの場合、このファイルは `$SPLUNK_HOME/etc/apps/splunk_httpinput/local/` にあります。 Windows ユーザーの場合、`%SPLUNK_HOME%\etc\apps\splunk_httpinput\local\` にあります。
  - シングルインスタンスの Splunk Enterprise デプロイの場合は、`inputs.conf` ファイルで `ackIdleCleanup` パラメータを `true` に設定します。*nix ユーザーの場合、このファイルは `$SPLUNK_HOME/etc/apps/splunk_httpinput/local/` にあります。 Windows ユーザーの場合、`%SPLUNK_HOME%\etc\apps\splunk_httpinput\local\` にあります。

- Firehose ストリームで指定されている IAM ロールが、S3 バックアップバケットとデータ変換用の Lambda 関数 (データ変換が有効になっている場合) にアクセスできることを確認します。また、IAM ロールがエラーログをチェックするために、CloudWatch Logs グループとログストリームにアクセスできることを確認します。詳細については、「Grant Firehose Access to a Splunk Destination」を参照してください。
- S3 エラーバケット (S3 バックアップ) に配信されたデータを Splunk にリドライブするには、Splunk ドキュメントに記載されているステップに従います。
- 詳細については、Troubleshoot the Splunk Add-on for Amazon Kinesis Firehose を参照してください。

# Snowflake のトラブルシューティング

このセクションでは、Snowflake を宛先として使用する際の一般的なトラブルシューティングステップについて説明します

## Firehose ストリームの作成が失敗する

PrivateLink 対応の Snowflake クラスターにデータを配信するストリームのための Firehose ストリームの作成が失敗した場合、これは、Firehose が VPCE-ID に到達できないことを示唆しています。これは、次のいずれかの理由で発生する場合があります:

- VPCE-ID が正しくありません。タイプミスがないことを確認してください。
- Firehose は、リージョンレス Snowflake URL (プレビュー) をサポートしていません。Snowflake Account Locator を使用して URL を指定してください。詳細については、Snowflake ドキュメントを参照してください。
- Firehose ストリームが Snowflake AWS リージョンと同じリージョンに作成されていることを確認します。
- 問題が解決しない場合は、 AWS サポートにお問い合わせください。

### 配信失敗

データが Snowflake テーブルに配信されない場合は、次を確認してください。Snowflake 配信に失敗したデータは、エラーコードと、ペイロードに対応するエラーメッセージとともに、S3 エラーバケットに配信されます。一般的なエラーシナリオをいくつか次に示します。エラーコードのリスト全体については、「Snowflake データ配信エラー」を参照してください。

- エラーコード: Snowflake.DefaultRoleMissing: Firehose ストリームの作成中に Snowflake ロールが設定されていないことを示します。Snowflake ロールが設定されていない場合は、指定された Snowflake ユーザーにデフォルトのロールが設定されていることを確認してください。
- エラーコード: Snowflake.ExtraColumns: Snowflake への挿入が、入力ペイロード内の余分な列を理由として拒否されることを示します。テーブルに存在しない列は指定しないでください。Snowflake 列名では大文字と小文字が区別されることに留意してください。テーブルに列が存在するにもかかわらず、このエラーで配信が失敗する場合は、入力ペイロードの列名の大文字/小文字の区別と、テーブル定義で宣言された列名が一致していることを確認してください。
- エラーコード: Snowflake.MissingColumns: Snowflake への挿入が、入力ペイロード内で欠落している列を理由として拒否されることを示します。nullable 以外のすべての列のために値が指定されていることを確認してください。
- エラーコード: Snowflake.InvalidInput: これは、Firehose が提供された入力ペイロードを解析して、有効な JSON 形式にできなかった場合に発生する可能性があります。JSON ペイロードが適切に形成されており、余分な二重引用符、引用符、エスケープ文字などがないことを確認してください。現在、Firehose はレコードペイロードとして単一の JSON 項目のみをサポートしていますが、JSON 配列はサポートされていません。
- エラーコード: Snowflake.InvalidValue: 入力ペイロードのデータ型が正しくないことを理由として配信が失敗したことを示します。入力ペイロードで指定された JSON 値が Snowflake テーブル定義で宣言されたデータ型に準拠していることを確認してください。
- エラーコード: Snowflake.InvalidTableType: Firehose ストリームで設定されたテーブルタイプがサポートされていないことを示します。サポートされているテーブル、列、データ型については、Snowpipe Streaming の制限の「Limitations」を参照してください。

> **Note**
>
> 何らかの理由で、Firehose ストリームの作成後に Snowflake の宛先でテーブル定義またはロールの許可が変更された場合、Firehose がこれらの変更を検出するまでに数分かかることがあります。これを理由として配信エラーが表示されている場合は、Firehose ストリームを削除して再作成してみてください。

## Firehose エンドポイントの到達可能性のトラブルシューティング

Firehose API でタイムアウトが発生した場合は、次のステップを実行してエンドポイントの到達可能性をテストします。

- API リクエストが VPC のホストから実行されたかどうかを確認します。VPC からのすべてのトラフィックでは、Firehose VPC エンドポイントを設定する必要があります。詳細については、AWS PrivateLink での Firehose の使用」を参照してください。
- 特定のサブネットに Firehose VPC エンドポイントが設定されたパブリックネットワークまたは VPC からトラフィックが送信されている場合は、ホストから次のコマンドを実行してネットワーク接続を確認します。Firehose のエンドポイントは、「Firehose endpoints and quotas」で確認できます。
  - traceroute や tcping などのツールを使用して、ネットワーク設定が正しいかどうかを確認します。これが失敗した場合は、ネットワーク設定を確認してください。

    例えば、次のようになります。

    ```
    traceroute firehose.us-east-2.amazonaws.com
    ```

    または

    ```
    tcping firehose.us-east-2.amazonaws.com 443
    ```

  - ネットワーク設定が正しく、次のコマンドが失敗した場合、Amazon CA (Certficate Authority) が信頼チェーンにあるかどうかを確認します。

    例えば、次のようになります。

    ```
    curl firehose.us-east-2.amazonaws.com
    ```

  上記のコマンドが成功した場合は、API を再試行して、API から返されたレスポンスがあるかどうかを確認してください。

## HTTP エンドポイントのトラブルシューティング

このセクションでは、Datadog、Dynatrace、LogicMonitor、MongoDB、New Relic、Splunk、Sumo Logic など、汎用 HTTP エンドポイントの宛先およびパートナーの宛先にデータを配信する Amazon Data Firehose を扱う場合の一般的なトラブルシューティングのステップについて説明します。このセクションの目的上、該当するすべての送信先を HTTP エンドポイントと呼びます。Firehose ストリームで指定されている IAM ロールが、S3 バックアップバケットとデータ変換用の Lambda 関数 (データ変換が有効になっている場合) にアクセスできることを確認します。また、IAM ロールがエラーログをチェックするために、CloudWatch ロググループとログストリームにアクセスできること

を確認します。詳細については、「Grant Firehose Access to an HTTP Endpoint Destination」を参照してください。

> **Note**
>
> このセクションの情報は、次の送信先には適用されません: Splunk、OpenSearch Service、S3、Redshift。

## CloudWatch Logs

CloudWatch Logging for Amazon Data Firehose を有効にすることを強くお勧めします。ログは、送信先への配信中にエラーが発生した場合にのみ発行されます。

### 送信先の例外

ErrorCode: HttpEndpoint.DestinationException

```
{
    "deliveryStreamARN": "arn:aws:firehose:us-east-1:123456789012:deliverystream/
ronald-test",
    "destination": "custom.firehose.endpoint.com...",
    "deliveryStreamVersionId": 1,
    "message": "The following response was received from the endpoint destination.
 413: {\"requestId\": \"43b8e724-dbac-4510-adb7-ef211c6044b9\", \"timestamp\":
 1598556019164, \"errorMessage\": \"Payload too large\"}",
    "errorCode": "HttpEndpoint.DestinationException",
    "processor": "arn:aws:lambda:us-east-1:379522611494:function:httpLambdaProcessing"
}
```

送信先の例外は、Firehose がエンドポイントへの接続を確立し、HTTP リクエストを行うことはできますが、200 のレスポンスコードを受信しなかったことを示します。200 ではない 2xx のレスポンスも送信先の例外になります。Amazon Data Firehose は、設定されたエンドポイントから受信したレスポンスコードと切り捨てられたレスポンスペイロードを CloudWatch Logs に記録します。Amazon Data Firehose は、変更や解釈なしでレスポンスコードとペイロードをログに記録するため、Amazon Data Firehose の HTTP 配信リクエストを拒否した正確な理由を提供するのはエンドポイント次第です。これらの例外に関する一般的なトラブルシューティングのレコメンデーションを次に示します。

- 400: Amazon Data Firehose の設定ミスにより、不正なリクエストを送信していることを示します。送信先の url、共通の属性、コンテンツのエンコード、アクセスキー、およびバッファリングヒントが正しいことを確認します。必要な設定については、送信先固有のドキュメントを参照してください。
- 401: Firehose ストリームのために設定したアクセスキーが正しくないか、見つからないことを示します。
- 403: Firehose ストリームのために設定したアクセスキーに、設定済みのエンドポイントにデータを配信する許可がないことを示します。
- 413: Amazon Data Firehose がエンドポイントに送信するリクエストペイロードが大きすぎてエンドポイントで処理できないことを示します。送信先の推奨サイズへバッファリングヒントを下げることを試行します。
- 429: Amazon Data Firehose が、宛先が処理できる速度よりも高い速度でリクエストを送信していることを示します。バッファリング時間を増やしたり、バッファリングサイズを増やしたりして、バッファリングヒントを微調整します (ただし、送信先の制限内にとどまります)。
- 5xx: 送信先に問題があることを示します。Amazon Data Firehose サービスはまだ正常に動作しています。

> **Important**
>
> 重要: これらは一般的なトラブルシューティングのレコメンデーションですが、特定のエンドポイントではレスポンスコードを提供する理由が異なる場合があり、最初はエンドポイント固有のレコメンデーションに従う必要があります。

## 無効なレスポンス

ErrorCode: HttpEndpoint.InvalidResponseFromDestination

```
{
    "deliveryStreamARN": "arn:aws:firehose:us-east-1:123456789012:deliverystream/
ronald-test",
    "destination": "custom.firehose.endpoint.com...",
    "deliveryStreamVersionId": 1,
    "message": "The response received from the specified endpoint is invalid.
 Contact the owner of the endpoint to resolve the issue. Response for request
```

```
 2de9e8e9-7296-47b0-bea6-9f17b133d847 is not recognized as valid JSON or has unexpected
 fields. Raw response received: 200 {\"requestId\": null}",
    "errorCode": "HttpEndpoint.InvalidResponseFromDestination",
    "processor": "arn:aws:lambda:us-east-1:379522611494:function:httpLambdaProcessing"
}
```

無効なレスポンスの例外は、Amazon Data Firehose がエンドポイントの宛先から無効なレスポンスを受信したことを示します。レスポンスがレスポンスの仕様に従う必要があるか、Amazon Data Firehose が配信の試行を失敗とみなし、設定された再試行期間を超えるまで同じデータを再配信します。Amazon Data Firehose は、レスポンスには 200 のステータスがあっても、レスポンスの仕様に従わないレスポンスを失敗として扱います。Amazon Data Firehose 互換エンドポイントを開発している場合は、レスポンスの仕様に従って、データが正常に配信されることを確認します。

以下に、一般的な無効なレスポンスの種類とその修正方法を示します。

- 無効な JSON または予期しないフィールド: レスポンスが JSON として正しく逆シリアル化できないか、予期しないフィールドがあることを示します。レスポンスがコンテンツエンコードされていないことを確認します。
- requestId が見つからない: レスポンスに requestId が含まれていないことを示します。
- requestId が一致しない: レスポンス内の requestId が発信 requestId と一致しないことを示します。
- タイムスタンプが見つからない: レスポンスにタイムスタンプフィールドが含まれていないことを示します。タイムスタンプフィールドは、文字列ではなく数値でなければなりません。
- コンテンツタイプが見つからない: レスポンスに「content-type: application/json」ヘッダーが含まれていないことを示します。その他の content-type は受け入れられません。

> **Important**
>
> 重要: Amazon Data Firehose は、Firehose リクエストとレスポンスの仕様に従うエンドポイントにのみデータを配信できます。サードパーティーサービスへの宛先を設定する場合は、パブリック取り込みエンドポイントとは異なる可能性がある正しい Amazon Data Firehose 互換エンドポイントを使用していることを確認してください。例えば、Datadog の Amazon Data Firehose エンドポイントは `https://aws-kinesis-http-intake.logs.datadoghq.com/`ですが、パブリックエンドポイントは `https://api.datadoghq.com/` です。

### その他の一般的なエラー

追加のエラーコードと定義を以下に示します。

- エラーコード: HttpEndpoint.RequestTimeout - エンドポイントが応答に 3 分以上かかったことを示します。送信先の所有者である場合は、送信先エンドポイントの応答時間を短くします。送信先の所有者でない場合は、所有者に連絡して、応答時間を短縮するために何かできるかどうかを尋ねます (つまり、バッファリングヒントを減らして、リクエストごとに処理されるデータが少なくなるようにします)。
- エラーコード: HttpEndpoint.ResponseTooLarge - レスポンスが大きすぎることを示します。レスポンスは、ヘッダーを含め 1 MiB 未満にする必要があります。
- エラーコード: HttpEndpoint.ConnectionFailed - 設定されたエンドポイントとの接続を確立できなかったことを示します。これは、設定された URL の入力ミス、Amazon Data Firehose からエンドポイントにアクセスできない、またはエンドポイントが接続リクエストに応答するのに時間がかかりすぎていることが原因である可能性があります。
- エラーコード: HttpEndpoint.ConnectionReset - 接続が確立されたけれども、エンドポイントによってリセットされたか、または途中で閉じられたことを示します。
- エラーコード: HttpEndpoint.SSLHandshakeFailure - 設定されたエンドポイントで SSL ハンドシェイクが正常に完了できなかったことを示します。

## MSK As Source のトラブルシューティング

このセクションでは、MSK As Source を使用する際の一般的なトラブルシューティングのステップについて解説します。

> **Note**
>
> 処理、変換、S3 配信関連の問題に対するトラブルシューティングは、前半のセクションを参照してください。

### hose の作成に失敗した

MSK As Source を使用した hose の作成に失敗した場合は、次の点を確認してください。

- ソース MSK クラスターがアクティブな状態であることをチェックします。

- プライベート接続を使用している場合は、クラスターのプライベートリンクがオンになっていることを確認します。

  パブリック接続を使用している場合は、クラスターのパブリックアクセスがオンになっていることを確認します。
- プライベート接続を使用する場合は、Firehose にプライベートリンクの作成を許可するリソースベースのポリシーを必ず追加してください。「MSK cross account permissions」も参照してください。
- クラスターのトピックからデータを取り込むための許可がソース設定のロールに付与されていることを確認します。
- VPC セキュリティグループが、クラスターのブートストラップサーバーで使用されるポートでの受信トラフィックを許可していることを確認します。

## hose が一時停止している

hose の状態が一時停止になっている場合は次の点を確認してください。

- ソース MSK クラスターがアクティブな状態であることをチェックします。
- ソーストピックが存在することをチェックします。トピックを削除して再作成した場合は、Firehose ストリームも削除して再作成する必要があります。

## hose がバックプレシャーされている

DataReadFromSource.Backpressured の値は、パーティションごとの BytesPerSecondLimit を超過したり、通常の配信フローが遅いか停止したりすると、1 になります。

- BytesPerSecondLimit の上限に達した場合は、DataReadFromSource.Bytes メトリクスを確認して、上限の引き上げをリクエストします。
- CloudWatch ログ、送信先メトリクス、データ変換メトリクス、形式の変換メトリクスを確認し、ボトルネックを特定します。

## データの鮮度が正しくない

データの鮮度が正しくない可能性がある

- Firehose は、使用されたレコードのタイムスタンプに基づいてデータの鮮度を計算します。プロデューサーレコードが Kafka のブローカーログで維持されている間、このタイムスタンプが正しく記録されるようにするには、Kafka トピックのタイムスタンプのタイプ設定を `message.timestamp.type=LogAppendTime` に設定します。

## MSK クラスター接続の問題

次の手順では、MSK クラスターへの接続を検証する方法について説明します。Amazon MSK クライアントの設定の詳細については、「Amazon Managed Streaming for Apache Kafka デベロッパーガイド」の「Getting started using Amazon MSK」を参照してください。

MSK クラスターへの接続を検証するには

1. Unix ベースの (AL2 が望ましい) Amazon EC2 インスタンスを作成します。クラスターで VPC 接続のみが有効になっている場合は、EC2 インスタンスが同じ VPC で実行されていることを確認します。インスタンスが使用可能になったら、インスタンスに SSH 接続します。詳細については、「Amazon EC2 ユーザーガイド」のこのチュートリアルを参照してください。
2. 次のコマンドを実行して、Yum パッケージマネージャーを使用して Java をインストールします。詳細については、「Amazon Corretto 8 ユーザーガイド」のインストール手順を参照してください。

```
sudo yum install java-1.8.0
```

3. 次のコマンドを実行して AWS クライアントをインストールします。

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

4. 次のコマンドを実行して、Apache Kafka クライアント 2.6* バージョンをダウンロードします。

```
wget https://archive.apache.org/dist/kafka/2.6.2/kafka_2.12-2.6.2.tgz
tar -xzf kafka_2.12-2.6.2.tgz
```

5. `kafka_2.12-2.6.2/libs` ディレクトリに移動し、次のコマンドを実行して Amazon MSK IAM JAR ファイルをダウンロードします。

```
wget https://github.com/aws/aws-msk-iam-auth/releases/download/v1.1.3/aws-msk-iam-auth-1.1.3-all.jar
```

6. Kafka bin フォルダに client.properties ファイルを作成します。

7. awsRoleArn を Firehose SourceConfiguration で使用したロール ARN に置き換え、証明書の場所を検証します。 AWS クライアントユーザーにロールの引き受けを許可しますawsRoleArn。 AWS クライアントユーザーはここで指定したロールの引き受けを試みます。

```
[ec2-user@ip-xx-xx-xx-xx bin]$ cat client.properties
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required
 awsRoleArn="<role arn>" awsStsRegion="<region name>";
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
awsDebugCreds=true
ssl.truststore.location=/usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.342.b07-1.amzn2.0.1.x86_64/jre/lib/security/cacerts
ssl.truststore.password=changeit
```

8. トピックを一覧表示するには、次の Kafka コマンドを実行します。接続がパブリックの場合は、パブリックエンドポイントのブートストラップサーバーを使用します。接続がプライベートの場合は、プライベートエンドポイントのブートストラップサーバーを使用します。

```
bin/kafka-topics.sh --list --bootstrap-server <bootstrap servers> --command-config
 bin/client.properties
```

リクエストが正常に実行されると、次の例のような出力が表示されます。

```
[ec2-user@ip-xx-xx-xx-xx kafka_2.12-2.6.2]$ bin/kafka-topics.sh --list --bootstrap-
server <bootstrap servers> --command-config bin/client.properties

[xxxx-xx-xx 05:49:50,877] WARN The configuration 'awsDebugCreds' was supplied but
 isn't a known config. (org.apache.kafka.clients.admin.AdminClientConfig)
[xxxx-xx-xx 05:49:50,878] WARN The configuration 'ssl.truststore.location' was
 supplied but isn't a known config.
 (org.apache.kafka.clients.admin.AdminClientConfig)
[xxxx-xx-xx 05:49:50,878] WARN The configuration 'sasl.jaas.config' was supplied
 but isn't a known config. (org.apache.kafka.clients.admin.AdminClientConfig)
[xxxx-xx-xx 05:49:50,878] WARN The configuration
 'sasl.client.callback.handler.class' was supplied but isn't a known config.
 (org.apache.kafka.clients.admin.AdminClientConfig)
```

```
[xxxx-xx-xx 05:49:50,878] WARN The configuration 'ssl.truststore.password' was
 supplied but isn't a known config.
 (org.apache.kafka.clients.admin.AdminClientConfig)
[xxxx-xx-xx 05:50:21,629] WARN [AdminClient clientId=adminclient-1] Connection to
 node...
__amazon_msk_canary
__consumer_offsets
```

9. 前のスクリプトの実行に問題がある場合は、指定したブートストラップサーバーが、指定したポートで到達可能であることを検証します。これを実行するには、次のコマンドに示すように、telnet または同様のユーティリティをダウンロードして使用できます。

```
sudo yum install telnet
telnet <bootstrap servers><port>
```

リクエストが成功すると、次の出力が表示されます。これは、ローカル VPC 内の MSK クラスターに接続でき、指定されたポートでブートストラップサーバーが正常であることを意味します。

```
Connected to ..
```

10. リクエストが失敗した場合は、VPC セキュリティグループのインバウンドルールを確認してください。例えば、インバウンドルールで次のプロパティを使用できます。

```
Type: All traffic
Port: Port used by the bootstrap server (e.g. 14001)
Source: 0.0.0.0/0
```

前のステップで示したように、telnet 接続を再試行します。それでも接続できない場合、または Firehose 接続が引き続き失敗する場合は、AWS サポートにお問い合わせください。

# Amazon Data Firehose のクォータ

このセクションでは、Amazon Data Firehose 内の現在のクォータ (以前は制限と呼ばれていました) について説明します。各クォータは、指定がない限り、リージョン単位で適用されます。

Service Quotas コンソールは、 AWS サービスのクォータを表示および管理し、使用する多くのリソースのクォータ引き上げをリクエストできる中心的な場所です。が提供するクォータ情報を使用して AWS インフラストラクチャを管理します。クォータの引き上げに対するリクエストは、クォータの引き上げが必要となる前に計画してください。

詳細については、「 Amazon Web Services 全般のリファレンス」の「Amazon Data Firehose エンドポイントとクォータ」を参照してください。

次のセクションは、Amazon Data Firehose に次のクォータがあることを示しています。

- Amazon MSK を Firehose ストリームのソースとして使用する場合、各 Firehose ストリームのデフォルトクォータは、パーティションあたりの読み取りスループットが 10 MB/秒で、最大レコードサイズは 10 MBです。
- Amazon MSK を Firehose ストリームのソースとして使用すると、 AWS Lambda が有効になっている場合は最大レコードサイズが 6 MB、Lambda が無効になっている場合は最大レコードサイズが 10 MB になります。 AWS Lambda は受信レコードを 6 MB に制限し、Amazon Data Firehose は 6Mb を超えるレコードをエラー S3 バケットに転送します。Lambda が無効になっている場合、Firehose の着信レコードの制限は 10 MB です。Amazon Data Firehose が Amazon MSK から 10 MB を超えるレコードサイズを受信した場合、Amazon Data Firehose はこのレコードを S3 のエラーバケットに配信し、ユーザーのアカウントに Cloudwatch メトリクスを送信します。 AWS Lambda の制限の詳細については、「Lambda クォータ」を参照してください。
- Firehose ストリームで動的パーティショニングが有効になっている場合、その Firehose ストリームのために作成できるアクティブパーティションのデフォルトクォータは 500 です。アクティブパーティション数は、配信バッファ内のアクティブパーティションの総数です。例えば、動的パーティショニングクエリが 1 秒あたり 3 つのパーティションを構築し、60 秒ごとに配信をトリガーするバッファのヒント設定がある場合、平均して 180 のアクティブパーティションが作成されます。データがパーティションに配信されると、そのパーティションはそれ以降はアクティブではなくなります。さらにパーティションが必要な場合は、Firehose ストリームをさらに作成することでアクティブパーティションをそれらに分散させることができます。
- Firehose ストリームで動的パーティショニングが有効になっている場合、アクティブパーティションごとに 1 GB/秒の最大スループットがサポートされます。

- 各アカウントで維持できる、リージョンごとの Firehose ストリーム数のクォータは次のとおりです。
  - 米国東部 (バージニア北部)、米国東部 (オハイオ)、米国西部 (オレゴン)、欧州 (アイルランド)、アジアパシフィック (東京): 5,000 Firehose ストリーム
  - 欧州 (フランクフルト )、欧州 (ロンドン )、アジアパシフィック (シンガポール )、アジアパシフィック (シドニー )、アジアパシフィック (ソウル )、アジアパシフィック (ムンバイ )、AWS GovCloud (米国西部 )、カナダ (西部 )、カナダ (中部): 2,000 Firehose ストリーム
  - 欧州 (パリ)、欧州 (ミラノ)、欧州 (ストックホルム)、アジアパシフィック (香港)、アジアパシフィック (大阪)、南米 (サンパウロ)、中国 (寧夏)、中国 (北京)、中東 (バーレーン)、 AWS GovCloud (米国東部)、アフリカ (ケープタウン): 500 Firehose ストリーム
  - 欧州 (チューリッヒ)、欧州 (スペイン)、アジアパシフィック (ハイデラバード)、アジアパシフィック (ジャカルタ)、アジアパシフィック (メルボルン)、中東 (UAE)、イスラエル (テルアビブ)、カナダ西部 (カルガリー)、カナダ (中部)、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): 100 Firehose ストリーム
  - この制限を超えた場合、CreateDeliveryStream を呼び出すと `LimitExceededException` 例外が発生します。このクォータを引き上げるには、リージョンで利用可能であればService Quotas を使用します。Service Quotas の使用の詳細については、「クォータ引き上げのリクエスト」を参照してください。
- [Direct PUT] がデータソースとして設定されている場合、各 Firehose ストリームは PutRecord リクエストおよび PutRecordBatch リクエストのために次の結合クォータを提供します。
  - 米国東部 (バージニア北部)、米国西部 (オレゴン)、欧州 (アイルランド) の場合: 50 万レコード/秒、2,000 リクエスト/秒、5 MiB/秒になります。
  - その他 AWS リージョン: 100,000 レコード/秒、1,000 リクエスト/秒、1 MiB/秒。

  データ取り込み量が多くなり、Firehose ストリームのスループットキャパシティを超えたために Direct PUT ストリームでスロットリングが発生した場合、Amazon Data Firehose はスロットリングが抑制されるまでストリームのスループットの上限を自動的に引き上げます。スループットの増加とスロットリングの状況によっては、Firehose がストリームのスループットを所望のレベルに引き上げるまでに時間がかかる場合があります。このため、失敗したデータ取り込みレコードの再試行を継続します。データ量が突発的に増加することが予想される場合、または新しいストリームでデフォルトのスループット制限よりも高いスループットが必要な場合は、スループットの上限の引き上げをリクエストしてください。

  クォータには比例する 3 つのクォータスケールが存在します。たとえば米国東部 (バージニア北部)、米国西部 (オレゴン)、または欧州 (アイルランド) のスループットクォータを 10 MiB/秒に引

き上げると、その他 2 つのクォータは 4,000 リクエスト/秒と 1000,000 レコード/秒に引き上がります。

> **Note**
>
> - リソースレベルの制限とクォータを、サービスの使用を制御する手段として使用しないでください。
> - データソースとして Kinesis Data Streams が設定されている場合、このクォータは適用されず、Amazon Data Firehose のスケールアップとスケールダウンは制限なしで行われます。
> - 引き上げたクォータが実行中のトラフィックよりもはるかに高い場合、送信先への配信バッチは小さくなります。そのため、非効率になり、結果として配信サービスのコストが高くなる場合があります。現在の実行中のトラフィックと一致するようにクォータを引き上げてください。トラフィックが増加した場合は、さらにクォータを引き上げてください。
> - データレコードが小さくなると、コストが高くなる場合があります。Firehose の取り込み料金は、サービスに送信するデータレコードの数、すなわち各レコードのサイズが最も近い 5 KB ( 5,120 バイト ) に切り上げられた回数に基づきます。したがって、同じ量の受信データ (バイト) では、受信レコードの数が多い場合、発生するコストが高くなります。たとえば、受信データ量の合計が 5MiB の場合、5,000 レコードを超えるデータの送信は、1,000 レコードを使用して同じ量のデータを送信する場合と比べて、コストが高くなります。詳細については、AWS 見積りツールの「Amazon Data Firehose」を参照してください。

- 各 Firehose ストリームは、配信先が使用できない場合や、送信元が DirectPut である場合に、最大 24 時間データレコードを保存します。ソースが Kinesis Data Streams (KDS) で、送信先が使用できない場合、データは KDS の設定に基づいて保持されます。
- base64-encoding の前に Amazon Data Firehose に送信されるレコードの最大サイズは、1,000 KiB です。
- PutRecordBatch オペレーションは、コールごとに最大 500 レコードまたは 4 MiB のどちらか小さい方を受け取ることができます。このクォータは変更できません。
- 次の各オペレーションでは、1 秒あたり最大 5 回の呼び出しを提供できます。これはハード制限です。
  - `CreateDeliveryStream`
  - `DeleteDeliveryStream`

  - DescribeDeliveryStream
  - ListDeliveryStreams
  - UpdateDestination
  - TagDeliveryStream
  - UntagDeliveryStream
  - ListTagsForDeliveryStream
  - StartDeliveryStreamEncryption
  - StopDeliveryStreamEncryption
- バッファの間隔のヒントの範囲は、60～900 秒です。
- Amazon Data Firehose から Amazon Redshift への配信では、パブリックアクセス可能な Amazon Redshift クラスターのみがサポートされます。
- 再試行の期間の範囲は、Amazon Redshift および OpenSearch Service への配信で 0～7,200 秒です。
- 宛先が Amazon S3、Amazon Redshift、または OpenSearch Service の場合、Amazon Data Firehose では、シャードごとに最大 5 回の未処理の Lambda 呼び出しが許可されます。Splunk の場合、このクォータはシャードあたり 10 回の未完了の Lambda 呼び出しとなります。
- タイプ CUSTOMER_MANAGED_CMK の CMK を使用して、最大 500 の Firehose ストリームを暗号化できます。

# ドキュメント履歴

次のテーブルに、Amazon Data Firehose ドキュメントの重要な変更を示します。

| 変更 | 説明 | 変更日 |
| --- | --- | --- |
| ソースとしてのデータベース (パブリックプレビュー) の削除 | ソースとしてのデータベース (パブリックプレビュー) は削除されました。 | 2025 年 9 月 24 日 |
| Glue マルチカタログ階層のサポートを追加 | これにより、Firehose と Amazon S3 Tables の統合が簡素化されます。デフォルトのデータカタログと `S3TablesCatalog` との間にリソースリンクは必要ありません。「Setting up a Firehose stream to Amazon S3 tables」を参照してください。 | 2025 年 5 月 14 日 |
| ソースとしてのデータベース (パブリックプレビュー) を追加 | Amazon S3 の Apache Iceberg テーブルにデータベースの変更をレプリケートできるようになりました。 | 2024 年 11 月 15 日 |
| 宛先として追加された Apache Iceberg テーブルの一般提供 (GA) リリース | Apache Iceberg テーブルを宛先として Firehose ストリームを作成できます。「Apache Iceberg テーブルにデータを配信する」を参照してください。 | 2024 年 9 月 30 日 |
| データ型の例を追加 | Apache Iceberg テーブルのためにサポートされているデータ型の例を追加しました。「サポートされているデータ型を理解する」を参照してください。 | 2024 年 8 月 22 日 |
| 新しいリージョンへの対応 | Amazon Data Firehose がアジアパシフィック (マレーシア) で利用できるようになりました。「Amazon Data Firehose のクォータ」を参照してください。 | 2024 年 8 月 22 日 |

| 変更 | 説明 | 変更日 |
| --- | --- | --- |
| 宛先としての Apache Iceberg テーブルを追加 (パブリックプレビュー) | Apache Iceberg テーブルを宛先として Firehose ストリームを作成できます。「Apache Iceberg テーブルにデータを配信する」を参照してください。 | 2024 年 7 月 25 日 |
| Snowflake のバッファリングのヒント | Snowflake がバッファリングのヒントをサポートするようになりました。「the section called “Snowflake の宛先の設定を構成する”」を参照してください。 | 2024 年 7 月 25 日 |
| 新しいリージョンにおける宛先としての Snowflake | Snowflake がアジアパシフィック (シンガポール)、アジアパシフィック (ソウル)、アジアパシフィック (シドニー) の宛先として利用可能になりました。「the section called “Snowflake の宛先の設定を構成する”」を参照してください。 | 2024 年 7 月 25 日 |
| ユーザーガイドのセクションを再構築 | ユーザーガイドのセクションのナビゲーションを簡素化しました。「Firehose ストリームにデータを送信する」および「エラーのトラブルシューティング」を参照してください。 | 2024 年 7 月 5 日 |
| Amazon Data Firehose は と統合されます AWS Secrets Manager | Secrets Manager を使用して、シークレットにアクセスし、認証情報のローテーションを安全に自動化できるようになりました。「the section called “ AWS Secrets Managerを使用して認証する”」を参照してください。 | 2024 年 6 月 6 日 |
| Dynatrace のログの取り込みのサポートを追加 | さらなる分析のために、Dynatrace にログとイベントを送信できるようになりました。「the section called “Dynatrace の宛先の設定を構成する”」を参照してください。 | 2024 年 4 月 18 日 |
| Snowflake の宛先としての一般提供 (GA) リリース | 宛先としての Snowflake の一般提供を開始しました。「the section called “Snowflake の宛先の設定を構成する”」を参照してください。 | 2024 年 4 月 17 日 |

| 変更 | 説明 | 変更日 |
| --- | --- | --- |
| Amazon Kinesis Data Firehose が Amazon Data Firehose に | Amazon Kinesis Data Firehose が Amazon Data Firehose にブランド変更されました。「Amazon Data Firehose とは」を参照してください。 | 2024 年 2 月 9 日 |
| 宛先としての Snowflake を追加 (パブリックプレビュー) | Snowflake を宛先として Firehose ストリームを作成できます。「the section called “Snowflake の宛先の設定を構成する”」を参照してください。 | 2024 年 1 月 19 日 |
| CloudWatch Logs の自動解凍を追加 | 新規または既存のストリームでの解凍を有効にして、解凍された CloudWatch Logs データを Firehose の宛先に送信できます。「the section called “CloudWatch Logs を Firehose に送信する”」を参照してください。 | 2023 年 12 月 15 日 |
| Splunk Observability Cloud が送信先に追加されました | Splunk Observability Cloud を宛先にして Firehose ストリームを作成できるようになりました。「the section called “Splunk Observability Cloud の宛先の設定を構成する”」を参照してください。 | 2023 年 10 月 3 日 |
| Amazon Managed Streaming for Apache Kafka がデータソースとして追加がされました | Firehose ストリームに情報を送信するように、Amazon MSK を設定できるようになりました。「the section called “Amazon MSK のソース設定を構成する”」を参照してください。 | 2023 年 9 月 26 日 |
| OpenSearch サービスの送信先に DocumentID タイプのサポートが追加されました。 | OpenSearch Service が Firehose ストリームの宛先になっている場合、DocumentID タイプはドキュメント ID の設定方法を表します。サポートされている方法は、Firehose が生成したドキュメント ID と OpenSearch Service が生成したドキュメント ID です。「the section called “宛先の設定を構成する”」を参照してください。 | 2023 年 5 月 10 日 |

| 変更 | 説明 | 変更日 |
| --- | --- | --- |
| 動的パーティショニングのサポートが追加されました | Amazon Data Firehose でストリーミングデータの継続的な動的パーティショニングのサポートが追加されました。「ストリーミングデータのパーティショニング」を参照してください。 | 2021 年 8 月 31 日 |
| カスタムプレフィックスについてのトピックを追加しました。 | Amazon S3 に配信されるデータのカスタムプレフィックスを構築する際に使用できる式についてのトピックを追加しました。「the section called “Amazon S3 オブジェクトのカスタムプレフィックスを理解する”」を参照してください。 | 2018 年 12 月 20 日 |
| 新しい Amazon Data Firehose チュートリアルが追加されました | Amazon Data Firehose を介して Amazon VPC フローログを Splunk に送信する方法を示すチュートリアルを追加しました。「Amazon Data Firehose を使用して VPC フローログを Splunk に取り込む」を参照してください。 | 2018 年 10 月 30 日 |
| 4 つの新しい Amazon Data Firehose リージョンが追加されました | パリ、ムンバイ、サンパウロ、ロンドンが追加されました。詳細については、「Amazon Data Firehose のクォータ」を参照してください。 | 2018 年 6 月 27 日 |
| 2 つの新しい Amazon Data Firehose リージョンが追加されました | ソウルおよびモントリオールが追加されました。詳細については、「Amazon Data Firehose のクォータ」を参照してください。 | 2018 年 13 月 6 日 |
| ソース機能としての新しい Kinesis Streams | Firehose ストリーム用レコードのための潜在的ソースとして追加された Kinesis Streams。詳細については、「Firehose ストリームのソースと宛先を選択する」を参照してください。 | 2017 年 8 月 18 日 |

| 変更 | 説明 | 変更日 |
| --- | --- | --- |
| コンソールドキュメントの更新 | Firehose ストリーム作成ウィザードが更新されました。詳細については、「チュートリアル: コンソールから Firehose ストリームを作成する」を参照してください。 | 2017 年 7 月 19 日 |
| 新しいデータ変換 | データを配信前に変換するように、Amazon Data Firehose を設定できます。詳細については、「Amazon Data Firehose でソースデータを変換する」を参照してください。 | 2016 年 19 月 12 日 |
| 新しい Amazon Redshift COPY の再試行 | 失敗した場合に、Amazon Redshift クラスターに対して COPY コマンドを再試行するよう Amazon Data Firehose を設定できます。詳細についてはチュートリアル: コンソールから Firehose ストリームを作成する、Amazon Data Firehose でのデータ配信を理解する、およびAmazon Data Firehose のクォータを参照してください。 | 2016 年 5 月 18 日 |
| 新しい Amazon Data Firehose 宛先、Amazon OpenSearch Service | Amazon OpenSearch Service を宛先として Firehose ストリームを作成できます。詳細についてはチュートリアル: コンソールから Firehose ストリームを作成する、Amazon Data Firehose でのデータ配信を理解する、およびFirehose に公開 OpenSearch Service の宛先へのアクセスを付与するを参照してください。 | 2016 年 4 月 19 日 |
| 新しい強化された CloudWatch メトリクスとトラブルシューティング機能 | 「Amazon Data Firehose をモニタリングする」および「Amazon Data Firehose でのエラーをトラブルシューティングする」を更新しました。 | 2016 年 4 月 19 日 |
| 新しい強化された Kinesis エージェント | データを送信するように Kinesis エージェントを設定する を更新しました。 | 2016 年 4 月 11 日 |

| 変更 | 説明 | 変更日 |
| --- | --- | --- |
| 新しい Kinesis エージェント | データを送信するように Kinesis エージェントを設定するが追加されました。 | 2015 年 10 月 2 日 |
| 初回リリース | Amazon Data Firehose 開発者ガイドの最初のリリース。 | 2015 年 10 月 4 日 |