aws

開発者ガイド

Amazon Data Firehose

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Data Firehose: 開発者ガイド

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標とトレードドレスは、Amazon 以外の製品またはサービスとの関連において、顧客に混乱を招いたり、Amazon の名誉または信用を毀損するような方法で使用することはできません。Amazon が所有しない他の商標はすべてそれぞれの所有者に帰属します。所有者は必ずしも Amazon との提携や関連があるわけではありません。また、Amazon の支援を受けているとはかぎりません。

Table of Contents

..... x
Amazon Data Firehose とは 1
主要な概念を学ぶ 1
Amazon Data Firehose のデータフローを理解する 2
セットアップ 4
にサインアップする AWS 4
(オプション) ライブラリとツールをダウンロードする 4
Firehose ストリームの作成 6
送信元と送信先を設定する 6
レコード変換と形式変換を設定する 8
送信先設定を構成する 10
Amazon S3 の送信先設定を構成する 11
Amazon Redshift の送信先設定を構成する 15
OpenSearch サービスの送信先設定を構成する 21
OpenSearch Serverless の送信先設定を構成する 23
HTTP エンドポイントの送信先設定を構成する 25
Datadog の送信先設定を構成する 27
Honeycomb の送信先設定を構成する 29
Coralogix の送信先設定を構成する 31
Dynatrace の送信先設定を構成する 33
の送信先設定を構成する LogicMonitor 34
Logz.io の送信先設定を構成する 36
MongoDB クラウドの送信先設定を構成する 38
New Relic の送信先設定を構成する 40
Snowflake の送信先設定を構成する 42
Splunk の送信先設定を構成する 44
Splunk Observability Cloud の送信先設定を構成する 46
Sumo Logic の送信先設定を構成する 48
Elastic の送信先設定を構成する 49
バックアップと詳細設定を構成する 51
バックアップ設定を構成する 51
詳細設定の設定 53
バッファリングヒントを理解する 55
Firehose ストリームのテスト 58

前提条件 58
送信先として Amazon S3 を使用してテストする 58
送信先として Amazon Redshift を使用してテストする 59
OpenSearch サービスを送信先として使用してテストする 60
送信先として Splunk を使用してテストする 60
Firehose ストリームへのデータの送信 62
Kinesis Data Streams を使用した書き込み 62
Amazon MSK を使用した書き込み 64
Amazon Data Firehose エージェントを使用した書き込み 66
前提条件 67
認証情報 67
カスタム認証情報プロバイダー 68
エージェントのダウンロードとインストール 69
エージェントの設定と開始 71
エージェントの設定 72
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書き込み 75
エージェントを使用してデータを事前処理する 76
エージェント CLI コマンド 81
よくある質問 81
AWS SDK を使用してデータを送信する 83
を使用した単一書き込みオペレーション PutRecord 83
を使用したバッチ書き込みオペレーション PutRecordBatch 84
CloudWatch ログを使用した書き込み 84
CloudWatch ログの解凍 85
CloudWatch ログの解凍後のメッセージ抽出 85
解凍の有効化と無効化 86
よくある質問 81
CloudWatch イベントを使用した書き込み 90
AWS IoT を使用した書き込み 90
セキュリティ 92
データ保護 93
データソースとして Kinesis Data Streams を使用したサーバー側の暗号化 93
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 93
アクセス権限の制御 95
アプリケーションに Amazon Data Firehose リソースへのアクセスを許可する 96

Amazon Data Firehose にプライベート Amazon MSK クラスターへのアクセスを許可する 96
Amazon Data Firehose に IAM ロールの引き受けを許可する 97
Amazon Data Firehose にデータ形式変換 AWS Glue 用の へのアクセス権を付与する 99
Amazon Data Firehose に Amazon S3 送信先へのアクセス権を付与する 100
Amazon Redshift 送信先へのアクセス権を Amazon Data Firehose に付与する 103
Amazon Data Firehose にパブリック OpenSearch サービスの送信先へのアクセス権を付与する 107
Amazon Data Firehose に VPC のサービス OpenSearch 送信先へのアクセスを許可する ... 110
Amazon Data Firehose にパブリック OpenSearchサーバーレス送信先へのアクセスを許可する 112
Amazon Data Firehose に VPC OpenSearch内のサーバーレス送信先へのアクセスを許可する 115
Amazon Data Firehose に Splunk 送信先へのアクセス権を付与する 116
VPC の Splunk へのアクセス 118
Snowflake または HTTP エンドポイントへのアクセス 120
Amazon Data Firehose に Snowflake の送信先へのアクセス権を付与する 120
VPC の Snowflake へのアクセス 122
Amazon Data Firehose に HTTP エンドポイントの送信先へのアクセス権を付与する 125
Amazon MSK からのクロスアカウント配信 128
Amazon S3 の送信先へのクロスアカウント間の配信 131
サービス宛先への OpenSearchクロスアカウント配信 132
タグを使用したアクセスの制御 134
で認証する AWS Secrets Manager 136
シークレットを理解する 137
シークレットを作成する 138
シークレットを使用する 138
シークレットのローテーション 140
コンソールを使用して IAM ロールを管理する 140
既存の IAM ロールを選択する 141
コンソールから新しい IAM ロールを作成する 141
コンソールから IAM ロールを編集する 143
モニタリング 144
コンプライアンス検証 144
耐障害性 145
災害対策 146

インフラストラクチャセキュリティ 146
VPC エンドポイント (PrivateLink) 147
セキュリティのベストプラクティス 147
最小特権アクセスの実装 147
IAM ロールの使用 147
依存リソースでのサーバー側の暗号化の実装 148
CloudTrail を使用して API コールをモニタリングする 148
データ変換 149
データ変換フロー 149
データ変換とステータスモデル 149
Lambda の設計図 151
データ変換失敗の処理 152
Lambda の呼び出し時間 153
ソースレコードのバックアップ 154
動的パーティショニング 155
パーティショニングキー 156
インライン解析によるパーティショニングキーの作成 156
AWS Lambda 関数を用いてパーティショニングキーを作成する 157
動的パーティショニングの Amazon S3 バケットプレフィックス 160
集約データの動的パーティショニング 162
S3 にデータを配信するときに新しい行区切り文字を追加する 163
動的パーティショニングを有効にする方法 163
動的パーティショニングのエラー処理 164
データバッファリングと動的パーティショニング 165
レコード形式の変換 166
レコード形式の変換の要件 166
JSON デシリアライザーの選択 167
シリアライザーの選択 168
入力レコードの形式の変換 (コンソール) 168
入力レコードの形式の変換 (API) 169
レコード形式変換のエラー処理 170
レコード形式の変換例 170
Managed Service for Apache Flink との統合 171
データ配信 172
データ配信形式を設定する 172
データ配信の頻度を理解する 174

データ配信の失敗を処理する 174
Amazon S3 オブジェクト名の形式を設定する 178
サービスのインデックスロー OpenSearchテーションを設定する 187
AWS アカウントとリージョン間の配信を理解する 188
重複レコード 188
Firehose ストリームの一時停止と再開 188
Firehose が配信失敗を処理する方法を理解する 189
Firehose ストリームの一時停止 189
Firehose ストリームの再開 190
モニタリング 191
CloudWatch アラームに関するベストプラクティス 191
CloudWatch メトリクスによるモニタリング 192
動的パーティショニング CloudWatch メトリクス 193
データ配信 CloudWatch メトリクス 194
データ取り込みメトリクス 206
API レベルの CloudWatch メトリクス 213
データ変換 CloudWatch メトリクス 216
CloudWatch ログの解凍メトリクス 216
変換 CloudWatch メトリクスのフォーマット 217
サーバー側の暗号化 (SSE) CloudWatch メトリクス 217
Amazon Data Firehose のディメンション 218
Amazon Data Firehose 使用状況メトリクス 218
Amazon Data Firehose の CloudWatch メトリクスへのアクセス 219
CloudWatch ログによるモニタリング 220
データ配信エラー 221
Amazon Data Firehose の CloudWatch ログへのアクセス 258
エージェントの状態のモニタリング 259
によるモニタリング CloudWatch 259
を使用した Amazon Data Firehose API コールのログ記録 AWS CloudTrail 260
の Amazon Data Firehose 情報 CloudTrail 260
例: Amazon Data Firehose ログファイルエントリ 262
カスタム Amazon S3 プレフィックス 267
timestamp 名前空間 267
firehose 名前空間 267
partitionKeyFromLambda および partitionKeyFromQuery 名前空間 269
セマンティックルール 269

プレフィックスの例 270
での Amazon Data Firehose の使用 AWS PrivateLink 272
Amazon Data Firehose のインターフェイス VPC エンドポイント (AWS PrivateLink) 272
Amazon Data Firehose でのインターフェイス VPC エンドポイント (AWS PrivateLink) の使用 272
可用性 276
Firehose ストリームのタグ付け 278
タグの基本 278
タグ付けを使用したコストの追跡 279
タグの制限 280
Amazon Data Firehose API を使用した Firehose ストリームのタグ付け 280
チュートリアル: Amazon Data Firehose を使用して VPC フローログを Splunk に取り込む 282
トラブルシューティング 283
一般的な問題 283
Amazon S3 トラブルシューティング 284
Amazon Redshift のトラブルシューティング 285
Amazon OpenSearch Service のトラブルシューティング 286
Splunk のトラブルシューティング 287
Snowflake のトラブルシューティング 289
Firehose ストリームの作成が失敗する 289
Firehose エンドポイント到達可能性のトラブルシューティング 290
HTTP エンドポイントのトラブルシューティング 291
CloudWatch ログ 292
MSK As Source のトラブルシューティング 295
hose の作成に失敗した 295
hose が一時停止している 296
hose がバックプレシャーされている 296
データの鮮度が正しくない 296
MSK クラスター接続の問題 297
データ鮮度メトリクスの増加または出力なし 299
レコード形式の Apache Parquet への変換が失敗する 301
クォータ 303
付録 - HTTP エンドポイント配信リクエストとレスポンスの仕様 307
リクエストの形式 307
レスポンスの形式 311
例 313

ドキュメント履歴 315
AWS 用語集 319

Amazon Data Firehose は、以前は Amazon Kinesis Data Firehose と呼ばれていました。

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

Amazon Data Firehose は、Amazon Simple Storage Service (Amazon S3)、Amazon Redshift、Amazon OpenSearch Service、Amazon OpenSearch Serverless、Splunk、および Datadog、Dynatrace、 LogicMonitorMongoDB、New Relic、Coralogix、Elastic など、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カスタム HTTP エンドポイントまたは HTTP エンドポイントなどの宛先にリアルタイムのストリーミングデータを配信するためのフルマネージドサービスです。Amazon Data Firehose では、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ようにデータプロデューサーを設定すると、指定した送信先にデータが自動的に配信されます。データを配信する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、「 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。

> **Note**
>
> プロデューサー、AWS ストリーミングストレージ、コンシューマー、送信先を流れるデータフローテンプレートを提供する Amazon MSK 用の最新のストリーミングデータソリューションに注意してください。 AWS CloudFormation

主要な概念を学ぶ

Amazon Data Firehose の使用を開始すると、以下の概念を理解しておくとメリットが得られます。

Firehose ストリーム

Amazon Data Firehose の基盤となるエンティティ。Amazon Data Firehose を使用するには、Firehose ストリームを作成し、そのストリームにデータを送信します。詳細については、「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。

record

データプロデューサーが Firehose ストリームに送信する対象のデータ。レコードのサイズは最大 1000 KB です。

データプロデューサー

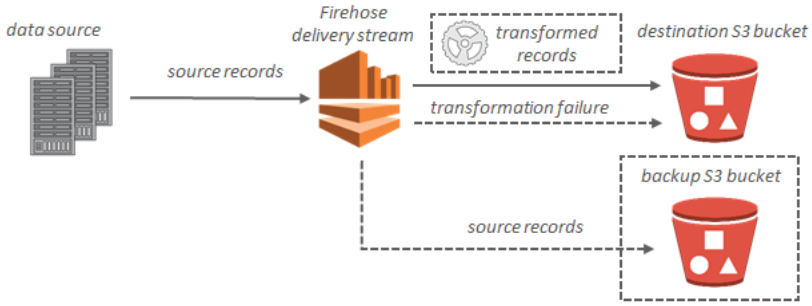
プロデューサーはレコードを Firehose ストリームに送信します。例えば、Firehose ストリームにログデータを送信するウェブサーバーはデータプロデューサーです。既存の Kinesis データストリームからデータを自動的に読み取り、送信先にロードするように Firehose ストリームを設定することもできます。詳細については、「Firehose ストリームにデータを送信する」を参照してください。

バッファサイズおよびバッファの間隔

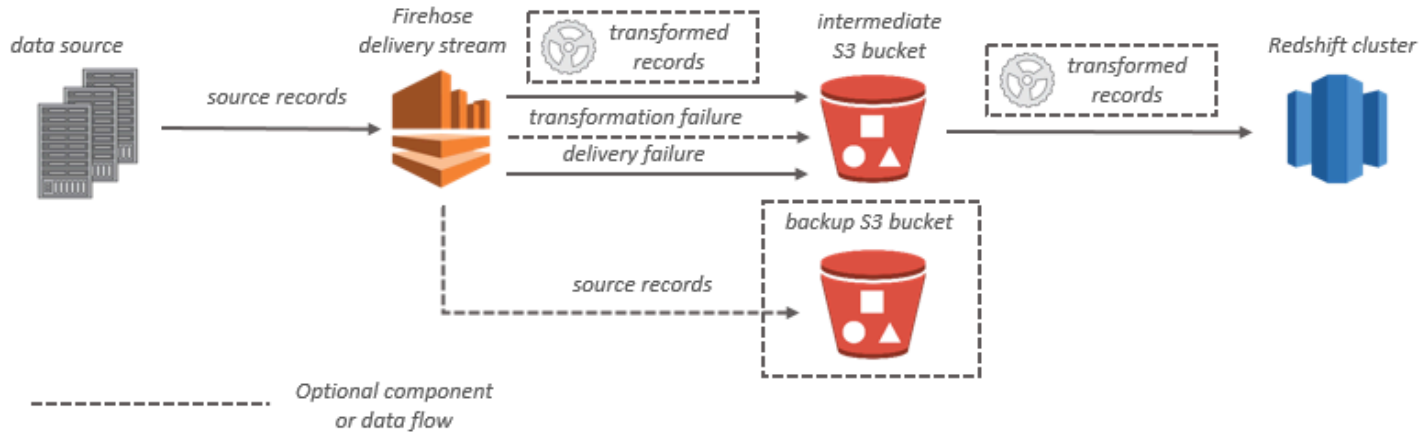
Amazon Data Firehose は、受信ストリーミングデータを特定のサイズまたは一定期間バッファしてから送信先に配信します。 Buffer Sizeは MBs、 Buffer Interval は秒単位です。

Amazon Data Firehose のデータフローを理解する

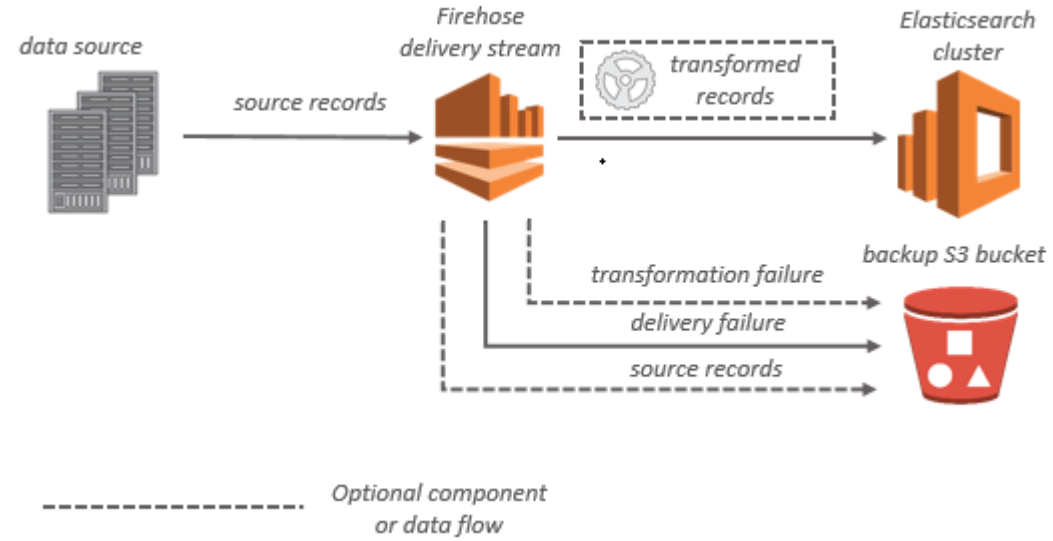
Amazon S3 の送信先の場合、ストリーミングデータは S3 バケットに配信されます。データ変換が有効な場合は、オプションで、送信元データを別の Amazon S3 バケットにバックアップすることもできます。



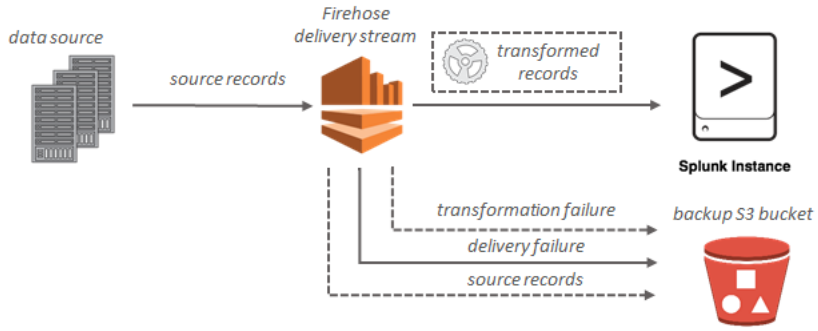
Amazon Redshift の送信先の場合、ストリーミングデータは S3 バケットに配信されます。次に、Amazon Data Firehose は Amazon Redshift COPY コマンドを発行して、S3 バケットから Amazon Redshift クラスターにデータをロードします。データ変換が有効な場合は、オプションで、送信元データを別の Amazon S3 バケットにバックアップすることもできます。



OpenSearch サービス送信先の場合、ストリーミングデータはサービスクラスターに OpenSearch配信され、オプションで S3 バケットに同時にバックアップできます。



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。



Amazon Data Firehose のセットアップ

Amazon Data Firehose を初めて使用する場合は、事前に以下のタスクを完了してください。

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。料金は、使用するサービスの料金のみが請求されます。

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、通話呼び出しを受け取り、電話キーパッドで検証コードを入力するように求められます。

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

(オプション) ライブラリとツールをダウンロードする

以下のライブラリとツールは、Amazon Data Firehose をプログラムでコマンドラインから操作するのに役立ちます。

- Firehose API オペレーションは、Amazon Data Firehose がサポートするオペレーションの基本セットです。

- Go 、Java 、.NET 、Node.js 、Python 、および Ruby 用の 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。 https://aws.amazon.com/developers/getting-started/python/

  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、から最新の AWS SDK をダウンロードすることもできますGitHub。
- は Amazon Data Firehose AWS Command Line Interfaceをサポートしています。 AWS CLI を使用すると、コマンドラインから複数の AWS サービスを制御したり、スクリプトを使用して自動化したりできます。

Firehose ストリームを作成する

AWS Management Console 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 を使用して、作成後にいつでも更新できますUpdateDestination。Firehose ストリームは、設定の更新中も `Active`状態のままで、引き続きデータを送信できます。通常、更新された設定は数分以内に有効になります。Firehose ストリームのバージョン番号は、設定を更新1した後、 の値だけ増加します。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。

以下のトピックでは、Firehose ストリームを作成する方法について説明します。

トピック

- 送信元と送信先を設定する
- レコード変換と形式変換を設定する
- 送信先設定を構成する
- バックアップと詳細設定を構成する
- バッファリングヒントを理解する

送信元と送信先を設定する

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/firehose で Amazon Data Firehose コンソールを開きます。
2. Firehose ストリームの作成 を選択します。
3. 以下のフィールドに値を入力します。

   Source (送信元)

   - 直接 PUT: プロデューサーアプリケーションが直接 に書き込む Firehose ストリームを作成するには、このオプションを選択します。現在、Amazon Data Firehose の Direct PUT と統合されている AWS のサービス、エージェント、オープンソースサービスは次のとおりです。
     - AWS SDK

- AWS Lambda
- AWS CloudWatch ログ
- AWS CloudWatch イベント
- AWS クラウドメトリクスストリーム
- AWS IOT
- AWS イベントブリッジ
- 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

- 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。その後、Amazon Data 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、送信先にロードできます。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、「Kinesis Data Streams を使用した Amazon Data Firehose への書き込み」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 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。その後、Firehose を使用して既存の Amazon MSK クラスターからデータを簡単に読み取り、指定された S3 バケットにロードできま

す。Amazon MSK をデータソースとして使用する方法の詳細については、「Amazon MSK を使用した Amazon Data Firehose への書き込み」を参照してください。

Firehose ストリームの送信先

Firehose ストリームの送信先。Amazon Data Firehose は、Amazon Simple Storage Service (Amazon S3)、Amazon Redshift、Amazon OpenSearch Service、およびユーザーまたはサードパーティーサービスプロバイダーが所有する HTTP エンドポイントなど、さまざまな宛先にデータレコードを送信できます。サポートされている送信先は次のとおりです。

- Amazon OpenSearch サービス
- Amazon OpenSearch Serverless
- Amazon Redshift
- Amazon S3
- 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

Firehose ストリーム名

Firehose ストリームの名前。

レコード変換と形式変換を設定する

レコードデータを変換および変換するように Amazon Data Firehose を設定します。

- Firehose ストリームのソースとして Amazon MSK を選択した場合。

  1. AWS「Lambda でソースレコードを変換する」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

     受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をオンにしないでください。

     Firehose が受信データを配信する前に呼び出して変換するために使用する Lambda 関数を指定するには、データ変換を有効にするチェックボックスをオンにします。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ている必要があります。詳細については、「Amazon Data Firehose データ変換」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

     受信データレコードの形式を変換しない Firehose ストリームを作成するには、無効化を選択します。

     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、「レコード形式の変換」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 AWS 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 Managed Service for Apache Flink または Direct PUT を選択した場合は、ソース設定セクションで次のようにします。

  1. 「レコードの変換」で、次のいずれかを選択します。

     a. 送信先が Amazon S3 または Splunk の場合、ソースレコードの解凍 Amazon CloudWatch Logs セクションで、解凍を有効にする を選択します。

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

データ変換

受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をオンにしないでください。

Amazon Data Firehose が受信データを配信する前に呼び出して変換するために使用する Lambda 関数を指定するには、データ変換を有効にするチェックボックスをオンにします。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ている必要があります。詳細については、「Amazon Data Firehose データ変換」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

受信データレコードの形式を変換しない Firehose ストリームを作成するには、無効化を選択します。

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、「レコード形式の変換」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 AWS CloudFormation、AWS「:::KinesisFirehose：DeliveryStream」を参照してください。

送信先設定を構成する

このトピックでは、選択した送信先に基づいて Firehose ストリームの送信先設定について説明します。バッファリングヒントの詳細については、「」を参照してくださいバッファリングヒントを理解する。

トピック

- Amazon S3 の送信先設定を構成する
- Amazon Redshift の送信先設定を構成する

- OpenSearch サービスの送信先設定を構成する
- OpenSearch Serverless の送信先設定を構成する
- HTTP エンドポイントの送信先設定を構成する
- Datadog の送信先設定を構成する
- Honeycomb の送信先設定を構成する
- Coralogix の送信先設定を構成する
- Dynatrace の送信先設定を構成する
- の送信先設定を構成する LogicMonitor
- Logz.io の送信先設定を構成する
- MongoDB クラウドの送信先設定を構成する
- New Relic の送信先設定を構成する
- Snowflake の送信先設定を構成する
- Splunk の送信先設定を構成する
- Splunk Observability Cloud の送信先設定を構成する
- Sumo Logic の送信先設定を構成する
- Elastic の送信先設定を構成する

Amazon S3 の送信先設定を構成する

Amazon S3 を Firehose ストリームの送信先として使用するには、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

  Firehose ストリームを設定して、Amazon S3 に配信されるオブジェクト内のレコード間に新しい行区切り文字を追加できます。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。Athena を使用して集約レコードを含む S3 オブジェクトをクエリする場合は、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 または指定された新しい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 回の PutRecord および PutRecordBatch API コールに集約しても、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 Kinesis Data Stream をソースとして設定されている場合は、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その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 または区切り記号の集約解除、Lambda 処理、データパーティショニング、データ形式変換、Amazon S3 配信の順に処理ステップを実行します。

マルチレコードの集約解除タイプ

マルチレコードの集約解除を有効にした場合は、Firehose が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できます。また、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することもできます。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 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 Lambda 関数を指定 AWS した場合は、partitionKeyFrom「Lambda:keyID」の形式を使用して S3 バケットプレフィック

ス値 (複数可) 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、partitionKeyFrom「Query:keyID」の形式を使用して S3 バケットプレビュー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選択して、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 のいずれかです。

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 Simple Storage Service オブジェクトのカスタムプレフィックス で、日付と時刻に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 に追加します。別の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。

バッファリングヒント

Firehose は、受信データをバッファしてから、指定された宛先に配信します。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。Amazon Redshift を送信先とする Firehose ストリームでは、Snappy、Zip、Hadoop 互換の Snappy 圧縮は使用できません。

S3 ファイル拡張子形式 (オプション)

Amazon S3 送信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形式を指定します。この機能を有効にすると、指定したファイル拡張子は、Data Format Conversion または .parquet や .gz などの S3 圧縮機能によって追加されたデフォルトのファイル拡張子を上書きします。Data Format Conversion または S3 圧縮でこの機能を使用するときに、適切なファイル拡張子が設定されていることを確認してください。ファイル拡張子はピリオド (.) で始まり、0 ~ 9a ~ z! ~ _.*'() の文字を使用できます。ファイル拡張子は 128 文字を超えることはできません。

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 (SSE-KMS) を使用したデータの保護」を参照してください。

Amazon Redshift の送信先設定を構成する

このセクションでは、Amazon Redshift を Firehose ストリームの送信先として使用する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 ワークグループの送信先設定を構成する

Amazon Redshift プロビジョンドクラスター

このセクションでは、Amazon Redshift でプロビジョニングされたクラスターを Firehose ストリームの送信先として使用する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 IP アドレスのブロックを解除します。詳細については、「Amazon Redshift 送信先へのアクセス権を Amazon Data 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 クラスターへのアクセス許可を持つ 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 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]

  クラスターへのアクセス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 認証情報用に に AWS Secrets Manager 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、まずデータを S3 バケットに配信し、次に Amazon Redshift COPY コマンドを発行して Amazon Redshift クラスターにデータをロードします。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、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、配信された Amazon YYYY/MM/dd/HHS3 オブジェクトに「」UTC 時間形式のプレフィックスを自動的に使用します。Amazon S3 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が失敗した場合に Firehose が再試行COPYする時間 (0 ~ 7200 秒) 。Firehose は、再試行期間が終了するまで 5 分ごとに再試行します。再試行時間を 0 (ゼロ) 秒に設定した場合、Firehose はCOPYコマンドの失敗時に再試行しません。

バッファリングヒント

Firehose は、受信データをバッファしてから、指定された宛先に配信します。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。Amazon Redshift を送信先とする Firehose ストリームでは、Snappy、Zip、Hadoop 互換の Snappy 圧縮は使用できません。

S3 ファイル拡張子形式 (オプション)

S3 ファイル拡張子形式 (オプション) — Amazon S3 送信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形式を指定します。この機能を有効にすると、指定したファイル拡張子は、Data Format Conversion または .parquet や .gz などの S3 圧縮機能によって追加され

たデフォルトのファイル拡張子を上書きします。Data Format Conversion または S3 圧縮でこの機能を使用するときに、適切なファイル拡張子が設定されていることを確認してください。ファイル拡張子はピリオド (.) で始まり、0 ~ 9a ~ z! ~ _.*'() の文字を使用できます。ファイル拡張子は 128 文字を超えることはできません。

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 (SSE-KMS) を使用したデータの保護」を参照してください。

Amazon Redshift Serverless ワークグループの送信先設定を構成する

このセクションでは、Amazon Redshift Serverless ワークグループを Firehose ストリームの送信先として使用する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 IP アドレスのブロックを解除します。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 送信先へのアクセス権を Amazon Data 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アクセス許可を持つ 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 ワークグループへのアクセス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 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 認証情報 AWS Secrets Manager 用に 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。

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、次に Amazon Redshift COPY コマンドを発行して Amazon Redshift Serverless ワークグループにデータをロードします。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、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、配信された Amazon YYYY/MM/dd/HHS3 オブジェクトに「」UTC 時間形式のプレフィックスを自動的に使用します。Amazon S3 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が失敗した場合に Firehose が再試行COPYする時間 (0 ~ 7200 秒) 。Firehose は、再試行期間が終了するまで 5 分ごとに再試行します。再試行時間を 0 (ゼロ) 秒に設定した場合、Firehose はCOPYコマンドの失敗時に再試行しません。

バッファリングヒント

Firehose は、受信データをバッファしてから、指定された宛先に配信します。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。Amazon Redshift を送信先とする Firehose ストリームでは、Snappy、Zip、Hadoop 互換の Snappy 圧縮は使用できません。

S3 ファイル拡張子形式 (オプション)

S3 ファイル拡張子形式 (オプション) — Amazon S3 送信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形式を指定します。この機能を有効にすると、指定したファイル拡張

子は、Data Format Conversion または .parquet や .gz などの S3 圧縮機能によって追加されたデフォルトのファイル拡張子を上書きします。Data Format Conversion または S3 圧縮でこの機能を使用するときに、適切なファイル拡張子が設定されていることを確認してください。ファイル拡張子はピリオド (.) で始まり、0 ~ 9a ~ z! ~ _.*'() の文字を使用できます。ファイル拡張子は 128 文字を超えることはできません。

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 (SSE-KMS) を使用したデータの保護」を参照してください。

OpenSearch サービスの送信先設定を構成する

このセクションでは、送信先に OpenSearch サービス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 サービスドメイン

  データが配信される OpenSearch サービスドメイン。

  [Index] (インデックス)

  OpenSearch サービスクラスターへのデータのインデックス作成時に使用する OpenSearch サービスインデックス名。

  インデックスのローテーション

  OpenSearch サービスインデックスをローテーションするかどうかと頻度を選択します。インデックス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てローテーションします。詳細については、「サービスのインデックスロー OpenSearchテーションを設定する」を参照してください。

タイプ

OpenSearch サービスクラスターにデータのインデックスを作成するときに使用する OpenSearch サービスタイプ名。Elasticsearch 7.x および OpenSearch 1.x では、インデックスごとに 1 つのタイプしか使用できません。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 はランタイム中にエラーを返します。

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

再試行の期間

インデックスリクエストが OpenSearch 失敗した場合に Firehose が再試行する時間。この場合、Firehose は再試行期間が終了するまで 5 分ごとに再試行します。再試行期間については、0 ~ 7200 秒の任意の値を設定できます。

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送信 OpenSearch 先にデータを再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が原因で Firehose ストリームが DLQ にデータを配信するのをブロックする場合は、再試行時間を秒単位でより高い値に設定できます。AWS サポート に連絡することで、再試行時間を 7200 秒以上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メソッドは、Firehose が生成するドキュメント ID と OpenSearch サービスが生成するドキュメント ID です。Firehose が生成するドキュメント ID は、ドキュメント ID 値が設定されていない場合のデフォルトのオプションです。 OpenSearch サービスが生成するドキュメント ID は、ログ分析やオブザーバビリティなどの書き込み負荷の高いオペレーションをサポートし、OpenSearch サービスドメインで消費する CPU リソースが少ないため、パフォーマンスが向上します。

送信先 VPC 接続

OpenSearch サービスドメインがプライベート VPC 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、Amazon Data Firehose が OpenSearch サービスドメインにデータを送信するときに使用するサブネットとサブグループを指定します。 OpenSearch サービスドメインが使用しているのと同じセキュリティグループを使用できます。異なるセキュリティグループを指定する場合は、 OpenSearch サービス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 トラフィックが許可されていることを確認してください。また、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 トラフィックが OpenSearch サービスドメインのセキュリティグループで許可されていることを確認してください。Firehose ストリームと OpenSearch サービス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 トラフィックが許可されていることを確認してください。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 の送信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 内のデータ配信用の ENIs を作成または追加できず、配信は低下または失敗します。

バッファのヒント

Amazon Data Firehose は、受信データをバッファしてから、指定された宛先に配信します。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 の送信先設定を構成する

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 サーバーレスコレクション

  データが配信される OpenSearch サーバーレス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 コレクションへのデータのインデックス作成時に使用する OpenSearch Serverless 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 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、Amazon Data Firehose が OpenSearch Serverless コレクションにデータを送信するときに使用するサブネットとサブグループを指定します。

> **Important**
>
> プライベート VPC の送信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 内のデータ配信用の ENIs を作成または追加できず、配信は低下または失敗します。

再試行の期間

Serverless へのインデックスリクエストが失敗した場合に Firehose OpenSearch が再試行する時間。この場合、Firehose は再試行期間が終了するまで 5 分ごとに再試行します。再試行期間については、0 ~ 7200 秒の任意の値を設定できます。

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の場合、設定された S3 エラーバケットから OpenSearch サーバーレス送信先にデータを再ドライブする必要があります。

Serverless クラスターのダウンタイムまたはメンテナンスにより、Firehose ストリームが DLQ OpenSearch にデータを配信するのをブロックする場合は、再試行時間を秒単位でより高い値に設定できます。AWS サポート に連絡することで、再試行時間を 7200 秒以上に増やすことができます。

バッファのヒント

Amazon Data Firehose は、受信データをバッファしてから、指定された宛先に配信します。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 エンドポイントの送信先設定を構成する

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、付録 - HTTP エンドポイント配信リクエストとレスポンスの仕様 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、エンドポイント所有者にお問い合わせください。

  - 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、コンテンツエンコーディングを使用してリクエストの本文を圧縮してから送信先に送信します。[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 HTTP エンドポイント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確認を待ちます。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 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、Amazon Data Firehose はそれをデータ配信の失敗と見なし、データを Amazon S3 バケットにバックアップします。

Amazon Data Firehose が HTTP エンドポイントにデータを送信するたびに (最初の試行または再試行のいずれか)、確認応答タイムアウトカウンターが再起動し、HTTP エンドポイントからの確認応答を待ちます。

再試行期間が終了しても、Amazon Data Firehose は確認を受け取るか、確認タイムアウト期間に達するまで確認を待機します。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターに残り時間があるかどうかを判断します。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 は、各 HTTP 呼び出しにこれらのキーと値のペアを含めます。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。

バッファリングヒント

Amazon Data Firehose は、受信データをバッファしてから、指定された宛先に配信します。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 エンドポイントの送信先で、 CloudWatch ログの送信先エンドポイントから 413 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 の送信先設定を構成する

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指定します。

  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 設定 - US1
  - Datadog 設定 - US3
  - Datadog 設定 - US5
  - Datadog 設定 - AP1

- Datadog 設定 - EU
- Datadog 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 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、 でシークレットを作成します AWS Secrets Manager。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、コンテンツエンコーディングを使用してリクエストの本文を圧縮してから送信先に送信します。[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 HTTP エンドポイント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確認を待ちます。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 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、Amazon Data Firehose はそれをデータ配信の失敗と見なし、データを Amazon S3 バケットにバックアップします。

Amazon Data Firehose が HTTP エンドポイントにデータを送信するたびに (最初の試行または再試行のいずれか)、確認応答タイムアウトカウンターが再起動し、HTTP エンドポイントからの確認応答を待ちます。

再試行期間が終了しても、Amazon Data Firehose は確認を受け取るか、確認タイムアウト期間に達するまで確認を待機します。確認がタイムアウトすると、Amazon Data Firehose は再

試行カウンターに残り時間があるかどうかを判断します。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 は、各 HTTP 呼び出しにこれらのキーと値のペアを含めます。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。

バッファリングヒント

Amazon Data Firehose は、受信データをバッファしてから、指定された宛先に配信します。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 の送信先設定を構成する

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 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、 でシークレットを作成し

ます AWS Secrets Manager。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、コンテンツエンコーディングを使用してリクエストの本文を圧縮してから送信先に送信します。[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

Amazon Data Firehose が選択した HTTP エンドポイント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確認を待ちます。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 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、Amazon Data Firehose はそれをデータ配信の失敗と見なし、データを Amazon S3 バケットにバックアップします。

Amazon Data Firehose が HTTP エンドポイントにデータを送信するたびに (最初の試行または再試行) 、確認タイムアウトカウンターが再起動され、HTTP エンドポイントからの確認が待機します。

再試行期間が終了しても、Amazon Data Firehose は確認応答を受信するか、確認タイムアウト期間に達するまで確認応答を待機します。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターに残り時間があるかどうかを判断します。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 は、各 HTTP 呼び出しにこれらのキーと値のペアを含めます。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。

バッファリングヒント

Amazon Data Firehose は、受信データをバッファしてから、指定された宛先に配信します。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 の送信先設定を構成する

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、https://coralogix.com/integrations/aws-firehose 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 にアクセスするかを選択できます。

  - プライベートキー

    Coralogix に連絡して、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、 でシークレットを作成します AWS Secrets Manager。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、コンテンツエンコーディングを使用してリクエストの本文を圧縮してから送信先に送信します。[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

Amazon Data Firehose が選択した HTTP エンドポイント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確認を待ちます。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 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、Amazon Data Firehose はそれをデータ配信の失敗と見なし、データを Amazon S3 バケットにバックアップします。

Amazon Data Firehose が HTTP エンドポイントにデータを送信するたびに (最初の試行または再試行) 、確認タイムアウトカウンターが再起動され、HTTP エンドポイントからの確認が待機します。

再試行期間が終了しても、Amazon Data Firehose は確認応答を受信するか、確認タイムアウト期間に達するまで確認応答を待機します。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターに残り時間があるかどうかを判断します。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 は、各 HTTP 呼び出しにこれらのキーと値のペアを含めます。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

バッファリングヒント

Amazon Data Firehose は、受信データをバッファしてから、指定された宛先に配信します。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。

Dynatrace の送信先設定を構成する

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 ストリームの送信先として Dynatrace を使用するオプションを選択します。

  取り込みタイプ

  詳細な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 - トークンと認証」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、 でシークレットを作成します AWS Secrets Manager。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、コンテンツエンコーディングを使用してリクエストの

本文を圧縮してから送信先に送信します。有効にすると、GZIP 形式で圧縮されたコンテンツ。

再試行の期間

Firehose が選択した HTTP エンドポイント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、Firehose はまず HTTP エンドポイントからの確認を待ちます。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 はそれをデータ配信の失敗と見なし、Amazon S3 バケットにデータをバックアップします。

Firehose が最初の試行中または再試行後に HTTP エンドポイントにデータを送信するたびに、確認タイムアウトカウンターが再起動され、HTTP エンドポイントからの確認が待機します。

再試行期間が終了しても、Firehose は確認を受け取るか、確認タイムアウト期間に達するまで確認を待機します。確認がタイムアウトすると、Firehose は再試行カウンターに残り時間があるかどうかを判断します。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。

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 は、各 HTTP 呼び出しにこれらのキーと値のペアを含めます。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。

バッファリングヒント

Amazon Data Firehose は、受信データをバッファしてから、指定された宛先に配信します。バッファヒントには、ストリームのバッファサイズと間隔が含まれます。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。

の送信先設定を構成する LogicMonitor

このセクションでは、送信LogicMonitor先に 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 にアクセスするかを選択できます LogicMonitor。

- API キー

  LogicMonitor に連絡して、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 API キーを取得します。

- シークレット

  の API キー AWS Secrets Manager を含むシークレットを から選択します LogicMonitor。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、 でシークレットを作成します AWS Secrets Manager。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、コンテンツエンコーディングを使用してリクエストの本文を圧縮してから送信先に送信します。[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 HTTP エンドポイント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確認を待ちます。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 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、Amazon Data Firehose はそれをデータ配信の失敗と見なし、データを Amazon S3 バケットにバックアップします。

Amazon Data Firehose が HTTP エンドポイントにデータを送信するたびに (最初の試行または再試行) 、確認タイムアウトカウンターが再起動され、HTTP エンドポイントからの確認が待機します。

再試行期間が終了しても、Amazon Data Firehose は確認応答を受信するか、確認タイムアウト期間に達するまで確認応答を待機します。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターに残り時間があるかどうかを判断します。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 は、各 HTTP 呼び出しにこれらのキーと値のペアを含めます。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。

バッファリングヒント

Amazon Data Firehose は、受信データをバッファしてから、指定された宛先に配信します。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 の送信先設定を構成する

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 の送信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

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 AWS Secrets Manager して Logz.io にアクセスするかを選択できます。

- 配送トークン

  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配送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、 でシークレットを作成します AWS Secrets Manager。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確認を待ちます。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 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、Amazon Data Firehose はそれをデータ配信の失敗と見なし、データを Amazon S3 バケットにバックアップします。

Amazon Data Firehose が HTTP エンドポイントにデータを送信するたびに (最初の試行または再試行) 、確認タイムアウトカウンターが再起動し、HTTP エンドポイントからの確認応答を待ちます。

再試行期間が終了しても、Amazon Data Firehose は確認応答を受信するか、確認タイムアウト期間に達するまで確認応答を待機します。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターに残り時間があるかどうかを判断します。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 は、各 HTTP 呼び出しにこれらのキーと値のペアを含めます。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。

バッファリングヒント

Amazon Data Firehose は、受信データをバッファしてから、指定された宛先に配信します。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 クラウドの送信先設定を構成する

このセクションでは、送信先に MongoDB Cloud 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.mongodb.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。

  MongoDB Realm webhook URL

  HTTP エンドポイントの URL を次の形式で指定します。

  ```
  https://webhooks.mongodb-realm.com
  ```

  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 クラウドにアクセスするかを選択できます。

  - API キー

    MongoDB クラウドに連絡して、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 API キーを取得します。

  - シークレット

MongoDB クラウドの 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、 でシークレットを作成します AWS Secrets Manager。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、コンテンツエンコーディングを使用してリクエストの本文を圧縮してから送信先に送信します。[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確認を待ちます。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 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、Amazon Data Firehose はそれをデータ配信の失敗と見なし、データを Amazon S3 バケットにバックアップします。

Amazon Data Firehose が HTTP エンドポイントにデータを送信するたびに (最初の試行または再試行) 、確認タイムアウトカウンターが再起動し、HTTP エンドポイントからの確認応答を待ちます。

再試行期間が終了しても、Amazon Data Firehose は確認応答を受信するか、確認タイムアウト期間に達するまで確認応答を待機します。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターに残り時間があるかどうかを判断します。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。

バッファリングヒント

Amazon Data Firehose は、受信データをバッファしてから、指定された宛先に配信します。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 は、各 HTTP 呼び出しにこれらのキーと値のペアを含めます。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 の送信先設定を構成する

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から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 進数の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、 でシークレットを作成します AWS Secrets Manager。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、コンテンツエンコーディングを使用してリクエストの本文を圧縮してから送信先に送信します。[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確認を待ちます。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 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、Amazon Data Firehose はそれをデータ配信の失敗と見なし、データを Amazon S3 バケットにバックアップします。

Amazon Data Firehose が HTTP エンドポイントにデータを送信するたびに (最初の試行または再試行) 、確認タイムアウトカウンターが再起動し、HTTP エンドポイントからの確認応答を待ちます。

再試行期間が終了しても、Amazon Data Firehose は確認応答を受信するか、確認タイムアウト期間に達するまで確認応答を待機します。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターに残り時間があるかどうかを判断します。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 は、各 HTTP 呼び出しにこれらのキーと値のペアを含めます。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。

バッファリングヒント

Amazon Data Firehose は、受信データをバッファしてから、指定された宛先に配信します。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 の送信先設定を構成する

このセクションでは、送信先に 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) 、およびアジアパシフィック (シドニー) で利用できます AWS リージョン。

接続設定

- 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 URL を指定します。例: `xy12345.us-east-1.aws.snowflakecomputing.com`。アカウント URL を確認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。ポート番号は指定しないでくださいが、プロトコル (https://) はオプションです。

  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。ユーザーが Snowflake テーブルにデータを挿入するためのアクセス権を持っていることを確認します。

  - プライベートキー

    Snowflake での認証に使用されるユーザーのプライベートキーを指定します。プライベートキーが PKCS8形式であることを確認します。このキーの一部として 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。

  - 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しておくことができます。詳細については、「キーペア認証とキーローテーションの使用」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、 でシークレットを作成します AWS Secrets Manager。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。

ロール設定

デフォルトの Snowflake ロールを使用する – このオプションを選択すると、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ことを想定しています。デフォルトのロールに Snowflake テーブルにデータを挿入するアクセス許可があることを確認してください。

カスタム Snowflake ロールを使用する – Snowflake テーブルにデータをロードするときに 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 形式は com.amazonaws.vpce.[region].vpce-svc-*[id]* です。詳細については、AWS PrivateLink「」および「Snowflake」を参照してください。

> **Note**
>
> Snowflake ネットワークが Firehose へのアクセスを許可していることを確認します。使用できる VPCE IDs「」を参照してくださいVPC の Snowflake へのアクセス。

データベース設定

- Snowflake を Firehose ストリームの送信先として使用するには、次の設定を指定する必要があります。
  - Snowflake データベース — Snowflake 内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 内のすべてのデータは、列と行のコレクションとして論理的に構造化されたデータベーステーブルに保存されます。

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 – テーブルで、raw データをロードする必要がある列名を指定します。
  - メタデータ列名 (オプション) — メタデータ情報をロードする必要がある列名をテーブルに指定します。

再試行の期間

Snowflake サービスの問題により、チャネルを開くか Snowflake への配信が失敗した場合に 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。再試行時間を 0 (ゼロ) 秒に設定すると、Firehose は Snowflake の障害時に再試行せず、データを Amazon S3 エラーバケットにルーティングします。

Splunk の送信先設定を構成する

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 で設定された 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「Splunk プラットフォームにデータを送信するように Amazon Data Firehose を設定する」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し AWS Lambda 、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した`Event endpoint`かどうかを選択します。使用するエンドポイントの詳細については、Splunk ドキュメントの「Splunk プラットフォームにデータを送信するように Amazon Data Firehose を設定する」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 エンドポイントを設定するには、Splunk ドキュメントの「Amazon Data Firehose 用 Splunk アドオンのインストールと設定の概要」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セットアップするときに Splunk から取得したトークンを保存し、ここに追加します。

- 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、 でシークレットを作成します AWS Secrets Manager。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を待機する時間を指定します。タイムアウトに達する前に Splunk が確認を送信しない場合、Amazon Data Firehose はそれをデータ配信の失敗と見なします。次に、Amazon Data Firehose は、設定した再試行期間の値に応じて、データを再試行するか、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 からの承認を待ちます。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 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、Amazon Data Firehose はそれをデータ配信の失敗と見なし、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 (最初の試行または再試行)、確認タイムアウトカウンターが再起動し、Splunk からの確認応答を待ちます。

再試行期間が終了しても、Amazon Data Firehose は確認応答を受信するか、確認タイムアウト期間に達するまで確認応答を待機します。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターに残り時間があるかどうかを判断します。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。

バッファリングヒント

Amazon Data Firehose は、受信データをバッファしてから、指定された宛先に配信します。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。

Splunk Observability Cloud の送信先設定を構成する

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 コンソールの設定で、アクセストークンから INGEST 認証スコープを持つ 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、 でシークレットを作成します AWS Secrets Manager。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、コンテンツエンコーディングを使用してリクエストの本文を圧縮してから送信先に送信します。[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 HTTP エンドポイント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確認を待ちます。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 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、Amazon Data Firehose はそれをデータ配信の失敗と見なし、データを Amazon S3 バケットにバックアップします。

Amazon Data Firehose が HTTP エンドポイントにデータを送信するたびに (最初の試行または再試行) 、確認タイムアウトカウンターが再起動され、HTTP エンドポイントからの確認が待機します。

再試行期間が終了しても、Amazon Data Firehose は確認応答を受信するか、確認タイムアウト期間に達するまで確認応答を待機します。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターに残り時間があるかどうかを判断します。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 は、各 HTTP 呼び出しにこれらのキーと値のペアを含めます。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。

バッファリングヒント

Amazon Data Firehose は、受信データをバッファしてから、指定された宛先に配信します。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 の送信先設定を構成する

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、コンテンツエンコーディングを使用してリクエストの本文を圧縮してから送信先に送信します。[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確認を待ちます。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 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、Amazon Data Firehose はそれをデータ配信の失敗と見なし、データを Amazon S3 バケットにバックアップします。

Amazon Data Firehose が HTTP エンドポイントにデータを送信するたびに (最初の試行または再試行) 、確認タイムアウトカウンターが再起動され、HTTP エンドポイントからの確認が待機します。

再試行期間が終了しても、Amazon Data Firehose は確認応答を受信するか、確認タイムアウト期間に達するまで確認応答を待機します。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターに残り時間があるかどうかを判断します。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 は、各 HTTP 呼び出しにこれらのキーと値のペアを含めます。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。

バッファリングヒント

Amazon Data Firehose は、受信データをバッファしてから、指定された宛先に配信します。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 の送信先設定を構成する

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 に連絡して、Firehose から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、 でシークレットを作成します AWS Secrets Manager。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、コンテンツエンコーディングを使用してリクエストの本文を圧縮してから送信先に送信します。[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確認を待ちます。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 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、Amazon Data Firehose はそれをデータ配信の失敗と見なし、データを Amazon S3 バケットにバックアップします。

Amazon Data Firehose が HTTP エンドポイントにデータを送信するたびに (最初の試行または再試行) 、確認タイムアウトカウンターが再起動され、HTTP エンドポイントからの確認が待機します。

再試行期間が終了しても、Amazon Data Firehose は確認応答を受信するか、確認タイムアウト期間に達するまで確認応答を待機します。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターに残り時間があるかどうかを判断します。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 は、各 HTTP 呼び出しにこれらのキーと値のペアを含めます。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。

バッファリングヒント

Amazon Data Firehose は、受信データをバッファしてから、指定された宛先に配信します。Elastic の送信先の推奨バッファサイズは 1 MiB です。

バックアップと詳細設定を構成する

このトピックでは、Firehose ストリームのバックアップと詳細設定を構成する方法について説明します。

バックアップ設定を構成する

Amazon Data Firehose は Amazon S3 を使用して、選択した送信先に配信しようとした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**
>
> - バックアップ設定は、Firehose ストリームのソースが Direct PUT または Kinesis Data Streams の場合に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送信先でのみ使用でき、Amazon S3 バックアップの送信先では使用できません。

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。

- Firehose ストリームの送信先として次のいずれかのサービスを設定した場合: Amazon OpenSearch Service、Datadog、Dynatrace、HTTP Endpoint、、MongoDB Cloud LogicMonitor、New Relic、Splunk、または Sumo Logic。

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 - 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。
- バックアップのバッファリングヒント、圧縮、暗号化 - Amazon Data Firehose は Amazon S3 を使用して、選択した送信先に配信しようとした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。Amazon Data Firehose は、受信データをバッファしてから Amazon S3 に配信 (バックアップ) します。バッファサイズは 1 ~ 128、バッファ間隔 MiBs は 60 ~ 900 秒を選択できます。最初に満たした条件によって、Amazon S3 へのデータ配信がトリガーされます。データ変換を有効にすると、バッファ間隔は、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 へのデータ配信に適用されます。送信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遅れた場合、Amazon Data Firehose はバッファサイズを動的に上げて追いつきます。このアクションにより、すべてのデータが送信先に適切に配信されます。
- S3 圧縮 - GZIP、Snappy、Zip、Hadoop 互換の Snappy データ圧縮を選択するか、データ圧縮なしを選択します。Amazon Redshift を送信先とする Firehose ストリームでは、Snappy、Zip、Hadoop 互換の Snappy 圧縮は使用できません。
- S3 ファイル拡張子形式 (オプション) — Amazon S3 送信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形式を指定します。この機能を有効にすると、指定したファイル拡張子は、Data Format Conversion または .parquet や .gz などの S3 圧縮機能によって追加されたデフォルトのファイル拡張子を上書きします。Data Format Conversion または S3 圧縮でこの機能を使用するときは、適切なファイル拡張子が設定されていることを確認してください。ファイル拡張子はピリオ

ド (.) で始まり、0 ~ 9a ~ z! ~ _.*'() の文字を使用できます。ファイル拡張子は 128 文字を超えることはできません。

- 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 (SSE-KMS) を使用したデータの保護」を参照してください。

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の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 (SSE-KMS) を使用したデータの保護」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ログに記録します。さらに、データ変換を有効にすると、Lambda 呼び出しをログに記録し、データ配信エラーを CloudWatch Logs に送信できます。詳細については、 CloudWatch 「ログを使用した Amazon Data Firehose のモニタリング」を参照してください。

  > **Important**
  >
  > オプションですが、Firehose ストリームの作成中に Amazon Data Firehose エラーログ記録を有効にすることを強くお勧めします。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。

- アクセス許可 - Amazon Data Firehose は、Firehose ストリームに必要なすべてのアクセス許可に IAM ロールを使用します。必要なアクセス許可が自動的に割り当てられた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するかを選択できます。このロールは、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を Firehose に許可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使って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。

- 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証を実行して、ユーザーがタグを作成するアクセス許可を持っているかどうかを確認します。このアクセス許可を指定しない場合、IAM リソース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、AccessDeniedException次のような 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、タグを適用できるようにするポリシーを示しています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "firehose:CreateDeliveryStream",
            "Resource": "*",
            }
        },
        {
            "Effect": "Allow",
            "Action": "firehose:TagDeliveryStream",
            "Resource": "*",
            }
        }
    ]
}
```

バックアップ設定と詳細設定を選択したら、選択内容を確認し、Firehose ストリームの作成 を選択します。

新しい Firehose ストリームが使用可能になるまでに、Creating 状態で少し時間がかかります。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ストリームにデータの送信を開始できます。

バッファリングヒントを理解する

Amazon Data Firehose は、メモリ内の受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) および一定期間 (バッファリング間隔) バッファしてから、指定された宛先に配信します。バッファリングヒントは、最適なサイズのファイルを 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向上させる場合や、Firehose の配信速度を宛先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間隔を更新したりできます。バッファリングサイズは MBs 単位で測定され、バッファリング間隔は秒単位で測定されます。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条件は、Firehose がデータを配信するようにトリガーします。バッファリング値を設定しない場合、デフォルト値が使用されます。

Firehose バッファリングヒントは AWS Management Console、、 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 Edit 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ヒントを再設定できます。新しいストリームの場合、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 の送信先固有の`DestinationConfiguration`パラメータ`IntervalInSeconds`で `SizeInMBs`と を設定します。

> **Note**
>
> - リアルタイムユースケースの低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。バッファリングヒントを低い値に変更する前に、送信先の Firehose の推奨バッファリング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送信先でのみ使用でき、Amazon S3 バックアップの送信先では使用できません。

> **Note**
>
> Firehose は、60 秒未満のバッファ時間間隔を設定してレイテンシーを低減する場合、S3 送信先にマルチパートアップロードを使用します。S3 送信先のマルチパートアップロードにより、60 秒未満のバッファ時間間隔を選択すると、S3 PUT API コストがいくらか増加します。

送信先固有のバッファリングヒントの範囲とデフォルト値については、次の表を参照してください。

デスティネーション	バッファリングサイズ (デフォルトは括弧内)	秒単位のバッファ間隔 (括弧内のデフォルト)
S3	1 ~ 128 (5)	0 ~ 900 (300)
Redshift	1 ~ 128 (5)	0 ~ 900 (300)
OpenSearch サーバーレス	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)

デスティネーション	バッファリングサイズ (デフォルトは括弧内)	秒単位のバッファ間隔 (括弧内のデフォルト)
ログジオ	1 ~ 64 (5)	0 ~ 900 (60)
mongoDB	1 ~ 16 (5)	0 ~ 900 (60)
newRelic	1 ~ 64 (5)	0 ~ 900 (60)
sumoLogic	1 ~ 64 (1)	0 ~ 900 (60)
Splunk Observability Cloud	1 ~ 64 (1)	0 ~ 900 (60)

サンプルデータを使用して Firehose ストリームをテストする

を使用して AWS Management Console 、シミュレートされた株価データを取り込むことができます。コンソールはブラウザでスクリプトを実行して、サンプルレコードを Firehose ストリームに配置します。これにより、独自のテストデータを生成しなくても、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 ストリームがデータを送信するときは標準の Amazon Data Firehose 料金が適用されますが、データの生成時には料金は発生しないことに注意してください。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。

内容

- 前提条件
- 送信先として Amazon S3 を使用してテストする
- 送信先として Amazon Redshift を使用してテストする
- OpenSearch サービスを送信先として使用してテストする
- 送信先として Splunk を使用してテストする

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、「Firehose ストリームを作成する」を参照してください。

送信先として Amazon S3 を使用してテストする

Amazon Simple Storage Service (Amazon S3) を送信先として使用して Firehose ストリームをテストするには、次の手順に従います。

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

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

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 ストリームがアクティブステータス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。

送信先として Amazon Redshift を使用してテストする

Amazon Redshift を送信先として使用して Firehose ストリームをテストするには、次の手順に従います。

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

1. Firehose ストリームは、Amazon Redshift クラスターにテーブルが存在することを想定しています。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. https://console.aws.amazon.com/firehose/ で Firehose コンソールを開きます。
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 ストリームがアクティブステータスになっている必要があります。
4. Firehose ストリームの送信先の詳細を編集して、新しく作成された`firehose_test_table`テーブルを参照します。
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. Firehose ストリームの送信先の詳細を編集して、別のテーブルをポイントします。
9. (オプション) `firehose_test_table` テーブルを削除します。

OpenSearch サービスを送信先として使用してテストする

Amazon OpenSearch Service を送信先として使用して Firehose ストリームをテストするには、次の手順に従います。

OpenSearch サービスを使用して Firehose ストリームをテストするには

1. https://console.aws.amazon.com/firehose/ で Firehose コンソールを開きます。
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 ストリームがアクティブステータス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 OpenSearch サービスドメインに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「サービスドメインでのドキュメントの検索」を参照してください。 OpenSearch
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。

送信先として Splunk を使用してテストする

Splunk を送信先として使用して Firehose ストリームをテストするには、次の手順に従います。

Splunk を使用して Firehose ストリームをテストするには

1. https://console.aws.amazon.com/firehose/ で Firehose コンソールを開きます。
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 ストリームがアクティブステータス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。

Firehose ストリームにデータを送信する

AWS SDK を使用して、Kinesis データストリーム、Amazon MSK、Kinesis Agent、Amazon Data Firehose API などのソースから Firehose ストリームにデータを送信できます。Amazon CloudWatch Logs、 CloudWatch Events、または AWS IoT をデータソースとして使用することもできます。Amazon Data Firehose を初めて使用する場合は、「」で説明されている概念と用語に慣れてくださいAmazon Data Firehose とは。

> **Note**
>
> 一部の AWS サービスでは、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。

トピック

- Kinesis Data Streams を使用した Amazon Data Firehose への書き込み
- Amazon MSK を使用した Amazon Data Firehose への書き込み
- Kinesis Agent を使用した Amazon Data Firehose への書き込み
- AWS SDK を使用した Amazon Data Firehose への書き込み
- CloudWatch ログを使用した Amazon Data Firehose への書き込み
- CloudWatch イベントを使用した Amazon Data Firehose への書き込み
- を使用した Amazon Data Firehose への書き込み AWS IoT

Kinesis Data Streams を使用した Amazon Data Firehose への書き込み

Firehose ストリームに情報を送信するように Amazon Kinesis Data Streams を設定できます。

> **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。次に、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と、Amazon Data

Firehose はレコードを宛先に配信する前にレコードの集約を解除します。データを変換するように Firehose ストリームを設定すると、Amazon Data Firehose はレコードを に配信する前にレコードの集約を解除します AWS Lambda。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。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/firehose/ で Amazon Data Firehose コンソールを開きます。
2. Firehose ストリームの作成 を選択します。[Name and source (名前とソース)] ページで、次のフィールドに値を入力します。

   Firehose ストリーム名

   Firehose ストリームの名前。

   ソース

   Kinesis ストリームを選択して、Kinesis データストリームをデータソースとして使用する Firehose ストリームを設定します。その後、Amazon Data Firehose を使用して、既存のデータストリームからデータを簡単に読み取り、送信先にロードできます。

   ソースとして Kinesis データストリームを使用するには、[Kinesis ストリーム] リストで既存のストリームを選択するか、[新規作成] を選択して新しい Kinesis データストリームを作成します。新しいストリームを作成した後に、[更新] を選択して [Kinesis ストリーム] 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 `PutRecord`および `PutRecordBatch`オペレーションは無効になります。この場合に Firehose ストリームにデータを追加するには、Kinesis Data Streams `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`LATEST`の位置からデータの読み取りを開始します。Kinesis Data Streams の位置の詳細については、「」を参照してくださいGetShardIterator。

Amazon Data Firehose は、シャードごとに Kinesis Data Streams GetRecordsオペレーションを 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 はシャードごとに 1 秒に 2 回 Kinesis Data Streams GetRecordsオペレーションを呼び出します。1 つはプライマリ配信先、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。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各呼び出しは、シャードの全体的なスロットリング制限にカウントされます。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。

3. [Next] を選択して [レコード変換と形式変換を設定する] ページに進みます。

Amazon MSK を使用した Amazon Data Firehose への書き込み

Firehose ストリームに情報を送信するように Amazon MSK を設定できます。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/firehose/ で Amazon Data Firehose コンソールを開きます。
2. Firehose ストリームの作成 を選択します。

   このページにある [ソースと送信先を選択] セクションで、次のフィールドに値を入力します。

   ソース

   Amazon MSK を選択して、Amazon MSK をデータソースとして使用する Firehose ストリームを設定します。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のどちらかを選択できます。その後、Amazon Data Firehose を使用して、特定の Amazon MSK クラスターとトピックからデータを簡単に読み取り、指定された S3 送信先にロードできます。

   送信先

   Firehose ストリームの送信先として Amazon S3 を選択します。

   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 の外部からのパブリックアクセスを目的としており AWS、プライベートブートストラップブローカーは 内からのアクセスを目的としています AWS。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。
- このクラスターには、Amazon Data Firehose サービスプリンシパルに Amazon MSK CreateVpcConnection API を呼び出すアクセス許可を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。

Amazon MSK クラスター

同じアカウントシナリオで、Firehose ストリームがデータを読み取る Amazon MSK クラスターの ARN を指定します。

クロスアカウントのシナリオについては、「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 ストリームがデータを取り込む Apache Kafka トピックを指定します。Firehose ストリームが作成されると、このトピックを更新することはできません。

ページの Firehose ストリーム名 セクションで、次のフィールドの値を指定します。

Firehose ストリーム名

Firehose ストリームの名前を指定します。

3. 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、「レコード変換と形式変換を設定する」を参照してください。

Kinesis Agent を使用した Amazon Data Firehose への書き込み

Amazon Kinesis エージェントは、データを収集して Firehose に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。エージェントは一連のファイルを継続的にモニタリングし、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、ファイルローテーション、チェックポイント、および障害発生時の再試行を処理する方法を示します。これは、信頼性が高く、タイムリーでシンプルな方法でデータを配信する方法を示しています。また、ストリーミングプロセスのモニタリングとトラブルシューティングを改善するためにメトリクスを出力 CloudWatch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 (エージェントの設定を参照)。

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ら、モニタリングするファイルと Firehose ストリームを指定してデータを設定します。エージェントを設定すると、ファイルから永続的にデータを収集し、Firehose ストリームに確実に送信します。

トピック

- 前提条件
- 認証情報
- カスタム認証情報プロバイダー

- エージェントのダウンロードとインストール
- エージェントの設定と開始
- エージェントの設定
- 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書き込み
- 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 CLI コマンド
- よくある質問

前提条件

- 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 CloudWatch のモニタリングを有効にする場合は、 オペレーションを実行する CloudWatch PutMetricDataアクセス許可も必要です。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のモニタリング「」、および「Amazon の認証とアクセスコントロール CloudWatch」を参照してください。

認証情報

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 “カスタム認証情報プロバイダー”」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 (「」の設定表awsSecretAccessKeyのawsAccessKeyId「」と「」のエントリを参照してくださいthe section called “エージェントの設定”)。
- 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 and assumeRoleExternalId」を参照) 。前述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。

カスタム認証情報プロバイダー

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 “エージェントの設定”」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、`AWSCredentialsProvider` 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。

エージェントのダウンロードとインストール

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「インスタンスに接続する」を参照してください。 Amazon EC2 接続に問題がある場合は、Amazon EC2 ユーザーガイド」の「インスタンスへの接続のトラブルシューティング」を参照してください。

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
```

最新のエージェント v1.x.x は、次のコマンドを使用して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.amzn1.noarch.rpm
```

- リポジトリから GitHubエージェントを設定するには

  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 repo からエージェントをダウンロードします。
  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"]
```

エージェントの設定と開始

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収集するファイル ("filePattern") と、エージェントがデータを送信する Firehose ストリームの名前 ( ) "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、Firehose ストリームを追跡してテールするファイルを決定します。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、「エージェントの設定」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。指定されたファイルを継続的にモニタリングし、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。

エージェントの設定

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。
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	設定されている場合、エージェントがメトリクスを に出力 CloudWatch できるようにします (true)。

構成設定	説明
	デフォルト: true
cloudwatch.endpoint	のリージョンエンドポイント CloudWatch。 デフォルト: monitoring.us-east-1.amazonaws.com
firehose.endpoint	Amazon Data Firehose のリージョンエンドポイント。 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。 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	エージェントにレコードを集約させ、1 回のオペレーションで 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 Firehose ストリームに配置する前に、集約レコードに保持させたいサイズに設定します。 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処理オプションのリスト。処理オプションは指定した順序で実行さ

構成設定	説明
	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。 **Important** 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。 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する最大時間をミリ秒単位で指定します。 値の範囲: 1,000 ~ 900,000 (1 秒 ~ 15 分) デフォルト: 60,000 (1 分)
maxBufferSizeBytes	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するバイト単位の最大サイズ。 値の範囲: 1 ~ 4,194,304 (4 MB) デフォルト: 4,194,304 (4 MB)

構成設定	説明
maxBuffer SizeRecords	Firehose ストリームに送信する前にエージェントがデータをバッファするレコードの最大数。 値の範囲: 1 ~ 500 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)。 値の範囲: 1 以上 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。 値の範囲: 0 以上 デフォルト: 0 (ゼロ)
truncated RecordTer minator	レコードサイズが Amazon Data Firehose レコードサイズ制限を超えたときに、エージェントが解析されたレコードを切り捨てるために使用する文字列。(1,000 KB) デフォルト: '\n' (改行)

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書き込み

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 データストリームと 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して、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がないようにすることができます。

```
{
    "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、モニタリング対象ファイルから解析されたレコードを Firehose ストリームに送信する前に事前処理できます。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"}
```

エージェント CLI コマンド

システムスタートアップ時のエージェントの自動的開始:

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

エージェントのステータスの確認:

```
sudo service aws-kinesis-agent status
```

エージェントの停止:

```
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む:

```
/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
```

よくある質問

Windows 用の Kinesis Agent はありますか?

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`メトリクスを確認します。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 の制限」と「Firehose Streams」を参照してください。

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 の例外が発生するのはなぜですか?

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ためです。`/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 の例外が発生するのはなぜですか?

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値は、使用可能なランタイムプロセッサの 12 倍です。エージェント設定の詳細については、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 を使用した Amazon Data Firehose への書き込み

Amazon Data Firehose API を使用して、AWS SDK for Java 、.NET 、Node.js 、Python 、または Ruby を使用して Firehose ストリームにデータを送信できます。 https://aws.amazon.com/sdk-for-net/ https://aws.amazon.com/sdk-for-python/Amazon Data Firehose を初めて使用する場合は、「」で説明されている概念と用語に慣れてくださいAmazon Data Firehose とは。詳細については、「アマゾン ウェブ サービス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である PutRecordと があります`PutRecordBatch`。 `PutRecord()`は 1 回の呼び出しで 1 つのデータレコードを送信し、1 回の呼び出しで複数のデータレコードを送信`PutRecordBatch()`できます。

トピック

- を使用した単一書き込みオペレーション PutRecord
- を使用したバッチ書き込みオペレーション PutRecordBatch

を使用した単一書き込みオペレーション PutRecord

データを配置するには、Firehose ストリーム名とバイトバッファ (<=1000 KB) のみが必要です。Amazon Data Firehose 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、レコード区切り文字を追加することもできます。Firehose ストリームに一度に 1 つのレコードのデータを配置するには、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については、Firehose API Operations の関連トピックを参照してください。

を使用したバッチ書き込みオペレーション PutRecordBatch

データを配置するには、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 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、レコード区切り文字を追加することもできます。データレコードを Firehose ストリームにバッチで入力するには、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については、Firehose API Operations の関連トピックを参照してください。

CloudWatch ログを使用した Amazon Data Firehose への書き込み

CloudWatch ログイベントは、 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、「Amazon Data Firehose を使用したサブスクリプションフィルター」を参照してください。

CloudWatch ログイベントは圧縮された 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 の送信先に配信する場合は、Firehose の解凍機能を使用して自動的に CloudWatch ログを解凍できます。

> **Important**
>
> 現在、Amazon は CloudWatch 複数の CloudWatch ログイベントを 1 つの Firehose レコードに結合し、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、Firehose は Amazon OpenSearch Service の送信先へのログの配信をサポートしていません。別の方法として、 CloudWatch ログで 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を検討することもできます。

CloudWatch ログの解凍

Firehose を使用して CloudWatch ログを配信し、解凍したデータを Firehose ストリームの送信先に配信する場合は、Firehose データ形式変換 (Parquet、ORC) または動的パーティショニング を使用します。Firehose ストリームの解凍を有効にする必要があります。

、 AWS Management Console、 AWS Command Line Interface または AWS SDKs を使用して解凍を有効にできます。

> **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 ログサブスクリプションフィルターにのみ使用し、提供されたログには使用しません。 CloudWatch ログと提供されたログ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すると、Firehose への提供されたログの取り込みは失敗します。この解凍機能は CloudWatch ログ専用です。

CloudWatch ログの解凍後のメッセージ抽出

解凍を有効にすると、メッセージ抽出を有効にすることもできます。メッセージ抽出を使用する場合、Firehose は、所有者、ロググループ、ログストリームなどのすべてのメタデータを解凍された CloudWatch ログレコードから除外し、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。Splunk の送信先にデータを配信する場合は、Splunk がデータを解析するためのメッセージ抽出

を有効にする必要があります。メッセージ抽出の有無にかかわらず、解凍後の出力例を次に示します。

図 1: メッセージ抽出なしで解凍後の出力例：

```
{
 "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: メッセージ抽出による解凍後の出力例：

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

解凍の有効化と無効化

、、 AWS Command Line Interface または AWS SDKs を使用して AWS Management Console、解凍を有効または無効にできます。

を使用して新しいデータストリームで解凍を有効にする AWS Management Console

を使用して新しいデータストリームで解凍を有効にするには AWS Management Console

1. にサインイン AWS Management Console 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 ストリームの作成 を選択します。
4. 「送信元と送信先の選択」

   ソース

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

   - 直接 PUT – プロデューサーアプリケーションが直接 に書き込む Firehose ストリームを作成するには、このオプションを選択します。Firehose の AWS Direct PUT と統合されている のサービス、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。その後、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、送信先にロードできます。詳細については、「Kinesis Data Streams を使用した Firehose への書き込み」を参照してください。

   送信先

   Firehose ストリームの送信先。以下のうちのひとつを選択します。

   - Amazon S3
   - Splunk
5. Firehose ストリーム名 に、ストリームの名前を入力します。
6. (オプション)「レコードの変換」：

   - 「Amazon CloudWatch Logs からソースレコードを解凍する」セクションで、「解凍を有効にする」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、「メッセージ抽出を有効にする」を選択します。

を使用して既存のデータストリームで解凍を有効にする AWS Management Console

解凍を実行する Lambda 関数を持つ Firehose ストリームがある場合は、Firehose 解凍機能に置き換えることができます。続行する前に、Lambda 関数コードを確認して、解凍またはメッセージ抽出のみを実行することを確認します。Lambda 関数の出力は、前のセクションの図 1 または図 2 に示す例のようになります。出力が似ている場合は、次のステップを使用して Lambda 関数を置き換えることができます。

1. 現在の Lambda 関数をこの設計図 に置き換えます。新しい設計図の Lambda 関数は、受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を実行します。
2. 解凍用の組み込み Firehose オプションを使用して解凍を有効にします。
3. Firehose ストリームの CloudWatch メトリクスがまだ有効になっていない場合は、有効にします。メトリクス CloudWatchProcessorLambda_ をモニタリングIncomingCompressedData し、このメトリクスがゼロに変わるまで待ちます。これにより、Lambda 関数に送信されるすべての入力データが解凍され、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。

を使用した解凍の無効化 AWS Management Console

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 Management Console

1. にサインイン AWS Management Console 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. 「レコードの変換と変換」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する」で、「解凍を有効にする」を選択し、「変更を保存」を選択します。

よくある質問

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。レコードに加えて、配信された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、これらのオブジェクトは という S3 バケットプレフィックスに配信されますdecompression-failed。Firehose は、レコードの解凍に失敗した後も他のレコードを処理し続けます。

解凍が成功した後に処理パイプラインで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変換などの解凍後に Amazon Data Firehose が処理ステップでエラーが発生した場合、レコードは圧縮形式で 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。レコードに加えて、配信された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に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 ログを設定すると、Firehose はエラーメッセージを CloudWatch ログに記録します。さらに、Firehose はモニタリングできるメトリクスに CloudWatch メトリクスを送信します。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 から行われない場合はどうなりますか？

顧客が CloudWatch Logs から来putsない場合、次のエラーメッセージが返されます。

```
Put to Firehose failed for AccountId: <accountID>, FirehoseName: <firehosename> because
 the request is not originating from allowed source types.
```

Firehose は解凍機能に対してどのようなメトリクスを出力しますか？

Firehose は、すべてのレコードを解凍するためのメトリクスを出力します。期間 (1 分) 、統計 (合計) 、日付範囲を選択して、DecompressedRecords失敗した、成功した、DecompressedBytes失敗した、成功した の数を取得する必要があります。詳細については、「CloudWatch ログの解凍メトリクス」を参照してください。

CloudWatch イベントを使用した Amazon Data Firehose への書き込み

イベントルールにターゲットを追加することで、Firehose ストリームに CloudWatch イベント CloudWatch を送信するように Amazon 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 CloudWatch イベントルールのターゲットを作成するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. [Create rule (ルールの作成)] を選択します。
3. ステップ 1: ルールの作成ページで、ターゲット でターゲット を追加 を選択し、Firehose ストリーム を選択します。
4. 既存の Firehose ストリーム を選択します。

CloudWatch イベント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。

を使用した Amazon Data Firehose への書き込み AWS IoT

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際、[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 のセキュリティ

のクラウドセキュリティが最優先事項 AWS です。 AWS のお客様は、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャの恩恵を受けることができます。

セキュリティは、 AWS とユーザーの間で共有される責任です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティと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の詳細については、「コンプライアンスAWS 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。
- クラウドのセキュリティ — お客様の責任は、使用する AWS サービスによって決まります。また、お客様は、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 を使用する際の責任共有モデルの適用方法を理解するのに役立ちます。以下のトピックでは、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 を設定する方法を示します。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 でのデータ保護
- Amazon Data Firehose によるアクセスの制御
- Amazon Data Firehose AWS Secrets Manager で を使用して認証する
- Amazon Data Firehose コンソールを使用して IAM ロールを管理する
- Amazon Data Firehose のモニタリング
- Amazon Data Firehose のコンプライアンス検証
- Amazon Data Firehose の耐障害性
- Amazon Data Firehose のインフラストラクチャセキュリティ
- Amazon Data Firehose のセキュリティのベストプラクティス

Amazon Data Firehose でのデータ保護

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るデータの場合、Amazon Data Firehose は を使用してデータを暗号化AWS Key Management Service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は、Amazon Data Firehose の使用時にサーバー側のデータ暗号化を有効にできます。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。

データソースとして 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、Kinesis 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ヒントに基づいてメモリ内のデータをバッファします。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 を使用して Firehose ストリームにデータを送信する場合PutRecordBatch、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。

を停止するには server-side-encryption、 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。そのためには、 を呼び出すDeliveryStreamEncryptionConfigurationInputときに を指定しますCreateDeliveryStream。

CMK がタイプ の場合CUSTOMER_MANAGED_CMK、、、KMSNotFoundException、KMSInvalidStateExceptionKMSDisabたは が原因で Amazon Data Firehose サービスがレコードを復号化できない場合KMSAccessDeniedException、サービスは問題を解決するまで最大 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 CMKsを使用します。Amazon Data Firehose は非対称 CMKsをサポートしていません。対称および非対称 CMKs「対称および非対称 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るシステムです。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仮定しないでください。

Amazon Data Firehose によるアクセスの制御

以下のセクションでは、Amazon Data Firehose リソースへのアクセスとリソースからのアクセスを制御する方法について説明します。対象となる情報には、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許可する方法が含まれます。また、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権を Amazon Data Firehose に付与する方法と、Datadog、Dynatrace、 LogicMonitorMongoDB、New Relic、Splunk、または Sumo Logic を送信先として使用する場合に必要となるアクセス許可についても説明します。最後に、このトピックのガイダンスでは、別の AWS アカウントに属する送信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

- アプリケーションに Amazon Data Firehose リソースへのアクセスを許可する
- Amazon Data Firehose にプライベート Amazon MSK クラスターへのアクセスを許可する
- Amazon Data Firehose に IAM ロールの引き受けを許可する
- Amazon Data Firehose にデータ形式変換 AWS Glue 用の へのアクセス権を付与する
- Amazon Data Firehose に Amazon S3 送信先へのアクセス権を付与する
- Amazon Redshift 送信先へのアクセス権を Amazon Data Firehose に付与する
- Amazon Data Firehose にパブリック OpenSearch サービスの送信先へのアクセス権を付与する
- Amazon Data Firehose に VPC のサービス OpenSearch 送信先へのアクセスを許可する
- Amazon Data Firehose にパブリック OpenSearchサーバーレス送信先へのアクセスを許可する
- Amazon Data Firehose に VPC OpenSearch内のサーバーレス送信先へのアクセスを許可する
- Amazon Data Firehose に Splunk 送信先へのアクセス権を付与する
- VPC の Splunk へのアクセス
- Snowflake または HTTP エンドポイントへのアクセス
- Amazon Data Firehose に Snowflake の送信先へのアクセス権を付与する
- VPC の Snowflake へのアクセス
- Amazon Data Firehose に HTTP エンドポイントの送信先へのアクセス権を付与する
- Amazon MSK からのクロスアカウント配信
- Amazon S3 の送信先へのクロスアカウント間の配信

- サービス宛先への OpenSearchクロスアカウント配信
- タグを使用したアクセスの制御

アプリケーションに Amazon Data Firehose リソースへのアクセスを許可する

Firehose ストリームへのアクセス権をアプリケーションに付与するには、こ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。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "firehose:DeleteDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch",
                "firehose:UpdateDestination"
            ],
            "Resource": [
                "arn:aws:firehose:region:account-id:deliverystream/delivery-stream-
name"
            ]
        }
    ]
}
```

Amazon Data Firehose にプライベート Amazon MSK クラスターへのアクセスを許可する

Firehose ストリームのソースがプライベート Amazon MSK クラスターである場合は、この例のようなポリシーを使用します。

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

```
    {
      "Principal": {
        "Service": [
          "firehose.amazonaws.com"
        ]
    },
    "Effect": "Allow",
    "Action": [
      "kafka:CreateVpcConnection"
    ],
    "Resource": "cluster-arn"
    }
  ]
}
```

Amazon Data Firehose に IAM ロールの引き受けを許可する

このセクションでは、Amazon Data Firehose にデータの取り込み、処理、送信元から送信先への配信を行うアクセス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すると、 は必要な信頼ポリシーをロールにア AWS タッチします。Amazon Data Firehose で既存の IAM ロールを使用する場合、またはロールを自分で作成する場合は、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* をアカウント AWS 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ために必要なすべてのアクセス許可に IAM ロールを使用します。Amazon Data Firehose がそのロールを引き受けられるように、次の信頼ポリシーがそのロールにアタッチされていることを確認します。

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Sid": "",
  "Effect": "Allow",
  "Principal": {
```

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

このポリシーは、 sts:ExternalId条件コンテキストキーを使用して、 AWS アカウントから発信された Amazon Data Firehose アクティビティのみがこの IAM ロールを引き受けられるようにします。詳細については、「IAM ユーザーガイド」の「混乱する代理問題」を参照してください。

Firehose ストリームのソースとして Amazon MSK を選択した場合は、指定された Amazon MSK クラスターからソースデータを取り込むためのアクセス許可を 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 がそのロールを引き受けられるように、次の信頼ポリシーがそのロールにアタッチされていることを確認します。

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

指定された Amazon MSK クラスターからソースデータを取り込むためのアクセス許可を Amazon Data Firehose に付与するこのロールが、次のアクセス許可を付与していることを確認します。

```
{
```

```
   "Version": "2012-10-17",
   "Statement": [{
        "Effect":"Allow",
        "Action": [
           "kafka:GetBootstrapBrokers",
           "kafka:DescribeCluster",
           "kafka:DescribeClusterV2",
           "kafka-cluster:Connect"
         ],
         "Resource": "CLUSTER-ARN"
       },
       {
         "Effect":"Allow",
         "Action": [
           "kafka-cluster:DescribeTopic",
           "kafka-cluster:DescribeTopicDynamicConfiguration",
           "kafka-cluster:ReadData"
         ],
         "Resource": "TOPIC-ARN"
       }]
}
```

Amazon Data Firehose にデータ形式変換 AWS Glue 用の へのアクセス権を付与する

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 に保存されているテーブル定義を参照します AWS Glue。Amazon Data Firehose に必要な へのアクセスを許可するには AWS Glue、ポリシーに次のステートメントを追加します。テーブルの ARN を検索する方法については、「 AWS Glue リソース ARNs」を参照してください。

```
[{
    "Effect": "Allow",
    "Action": [
        "glue:GetTable",
        "glue:GetTableVersion",
        "glue:GetTableVersions"
    ],
    "Resource": "table-arn"
}, {
    "Sid": "GetSchemaVersion",
    "Effect": "Allow",
```

```
    "Action": [
        "glue:GetSchemaVersion"
    ],
    "Resource": ["*"]
}]
```

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「デシリアライザの IAM の例」を参照してください。

> **Note**
>
> 現在、 AWS Glue はイスラエル (テルアビブ) 、アジアパシフィック (ジャカルタ) 、中東 (アラブ首長国連邦) の各リージョンではサポートされていません。アジアパシフィック (ジャカルタ) リージョンまたは中東 (アラブ首長国連邦) リージョンで Amazon Data Firehose を使用している場合 AWS Glue は、 が現在サポートされているリージョンのいずれか AWS Glue で、Amazon Data Firehose に へのアクセスを許可してください。Data Firehose と の間のクロスリージョン相互運用性 AWS Glue がサポートされています。 AWS Glue がサポートされているリージョンの詳細については、https://docs.aws.amazon.com/general/latest/gr/glue.html を参照してください。

Amazon Data Firehose に Amazon S3 送信先へのアクセス権を付与する

Amazon S3 の送信先を使用している場合、Amazon Data Firehose は S3 バケットにデータを配信し、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 CloudWatch ロググループとストリームにもデータ配信エラーを送信します。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 IAM ロールを引き受け、指定されたバケット、キー、 CloudWatch ロググループ、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 バケットと 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 によって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。

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

```
    [
        {
            "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/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:region:account-id:stream/stream-name"
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:region:account-id:key/key-id"
           ],
           "Condition": {
               "StringEquals": {
                   "kms:ViaService": "s3.region.amazonaws.com"
               },
               "StringLike": {
                   "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::bucket-name/
prefix*"
               }
           }
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:log-group-name:log-stream:log-
stream-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:region:account-id: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、そのステートメントを以下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 の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 の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 送信先へのアクセス権を Amazon Data Firehose に付与する方法については、「」を参照してくださいthe section called “Amazon S3 の送信先へのクロスアカウント間の配信”。

Amazon Redshift 送信先へのアクセス権を Amazon Data Firehose に付与する

Amazon Redshift 送信先を使用するときに Amazon Data Firehose へのアクセスを許可する場合は、以下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 送信先を使用している場合、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 でプロビジョニングされたクラスターまたは 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 CloudWatch ロググループとストリームにもデータ配信エラーを送信します。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ニングされたクラスターまたは Amazon Redshift Serverless ワークグループにアクセスし、IAM ロールを使用し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 バケットと 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`s3:PutObjectAcl`に を追加します。これにより、バケット所有者は Amazon Data Firehose によって配信されるオブジェクトにフルアクセスできるようになります。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。

```
{
"Version": "2012-10-17",
    "Statement":
    [
        {
            "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/*"
            ]
        },
        {
           "Effect": "Allow",
```

```
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:region:account-id:key/key-id"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.region.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::bucket-name/
prefix*"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:region:account-id:stream/stream-name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:log-group-name:log-stream:log-
stream-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
```

```
                "arn:aws:lambda:region:account-id:function:function-name:function-
version"
            ]
        }
    ]
}
```

他の AWS の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 の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 の IP アドレスのブロックを解除して、Amazon Redshift でプロビジョニングされたクラスターまたは Amazon Redshift Serverless ワークグループへのアクセス権を Amazon Data Firehose に付与します。Amazon Data Firehose は現在、利用可能なリージョンごとに 1 つの CIDR ブロックを使用しています。

- 米国東部 (オハイオ) 用の 13.58.135.96/27
- 米国東部 (バージニア北部) 用の 52.70.63.192/27
- 米国西部 (北カリフォルニア) 用の 13.57.135.192/27
- 米国西部 (オレゴン) 用の 52.89.255.224/27
- 18.253.138.96/27 AWS GovCloud （ 米国東部) 用の
- 52.61.204.160/27 AWS GovCloud （ 米国西部) 用の
- カナダ (中部) 用の 35.183.92.128/27
- 40.176.98.192/27 カナダ西部 (カルガリー) 用の
- アジアパシフィック (香港) 用の 18.162.221.32/27
- アジアパシフィック (ムンバイ) 用の 13.232.67.32/27
- アジアパシフィック (ハイデラバード) 用の 18.60.192.128/27
- アジアパシフィック (ソウル) 用の 13.209.1.64/27
- アジアパシフィック (シンガポール) 用の 13.228.64.192/27
- アジアパシフィック (シドニー) 用の 13.210.67.224/27

- アジアパシフィック (ジャカルタ) 用の 108.136.221.64/27
- アジアパシフィック (東京) 用の 13.113.196.224/27
- アジアパシフィック (大阪) 用の 13.208.177.192/27
- 中国 (北京) 用の 52.81.151.32/27
- 中国 (寧夏) 用の 161.189.23.64/27
- 欧州 (チューリッヒ) 用の 16.62.183.32/27
- 欧州 (フランクフルト) 用の 35.158.127.160/27
- 欧州 (アイルランド) 用の 52.19.239.192/27
- 欧州 (ロンドン) 用の 18.130.1.96/27
- 欧州 (パリ) 用の 35.180.1.96/27
- 欧州 (ストックホルム) 用の 13.53.63.224/27
- 中東 (バーレーン) 用の 15.185.91.0/27
- 南米 (サンパウロ) 用の 18.228.1.128/27
- 欧州 (ミラノ) 用の 15.161.135.128/27
- アフリカ (ケープタウン) 用の 13.244.121.224/27
- 中東 (UAE) 用の 3.28.159.32/27
- イスラエル (テルアビブ) 用の 51.16.102.0/27
- アジアパシフィック (メルボルン) 用の 16.50.161.128/27

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。

Amazon Data Firehose にパブリック OpenSearch サービスの送信先へのアクセス権を付与する

OpenSearch サービスの送信先を使用している場合、Amazon Data Firehose はデータを OpenSearch サービスクラスターに配信し、障害が発生したドキュメントまたはすべてのドキュメントを同時に 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 はロググループとストリームにもデータ配信エラー CloudWatchを送信します。Amazon Data Firehose は IAM ロールを使用して、指定された OpenSearchサービスドメイン、S3 バケット、 AWS KMS 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、 OpenSearchサービスドメイン、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`s3:PutObjectAcl`に を追加します。これにより、バケット所有者は Amazon Data Firehose によって配信されるオブジェクトにフルアクセスできるようになります。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。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "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/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:region:account-id:key/key-id"
           ],
           "Condition": {
               "StringEquals": {
                   "kms:ViaService": "s3.region.amazonaws.com"
               },
               "StringLike": {
                   "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::bucket-name/
prefix*"
```

```
                }
            }
         },
         {
            "Effect": "Allow",
            "Action": [
                "es:DescribeDomain",
                "es:DescribeDomains",
                "es:DescribeDomainConfig",
                "es:ESHttpPost",
                "es:ESHttpPut"
            ],
            "Resource": [
                "arn:aws:es:region:account-id:domain/domain-name",
                "arn:aws:es:region:account-id:domain/domain-name/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "es:ESHttpGet"
            ],
            "Resource": [
                "arn:aws:es:region:account-id:domain/domain-name/_all/_settings",
                "arn:aws:es:region:account-id:domain/domain-name/_cluster/stats",
                "arn:aws:es:region:account-id:domain/domain-name/index-name*/
_mapping/type-name",
                "arn:aws:es:region:account-id:domain/domain-name/_nodes",
                "arn:aws:es:region:account-id:domain/domain-name/_nodes/stats",
                "arn:aws:es:region:account-id:domain/domain-name/_nodes/*/stats",
                "arn:aws:es:region:account-id:domain/domain-name/_stats",
                "arn:aws:es:region:account-id:domain/domain-name/index-name*/_stats",
                "arn:aws:es:region:account-id:domain/domain-name/"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:region:account-id:stream/stream-name"
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:log-group-name:log-stream:log-
stream-name"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:region:account-id:function:function-name:function-
version"
            ]
        }
    ]
}
```

他の AWS の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 の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 Data Firehose に別のアカウントの OpenSearch サービスクラスターへのアクセスを許可する方法については、「」を参照してくださいthe section called “サービス宛先への OpenSearchクロスアカウント配信”。

Amazon Data Firehose に VPC のサービス OpenSearch 送信先へのアクセスを許可する

OpenSearch サービスドメインが VPC 内にある場合は、前のセクションで説明したアクセス許可を Amazon Data Firehose に付与してください。さらに、 OpenSearch サービスドメインの VPC にアクセスできるようにするには、Amazon Data Firehose に次のアクセス許可を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 ストリームを作成した後は、これらのアクセス許可を取り消さないでください。これらのアクセス許可を取り消すと、 OpenSearch サービスが ENIsストリームのパフォーマンスが低下したり、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 の送信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 内のデータ配信用の ENIs を作成または追加できず、配信は低下または失敗します。

Firehose ストリームを作成または更新するときは、Firehose が OpenSearch サービスドメインにデータを送信するときに使用するセキュリティグループを指定します。 OpenSearch サービスドメインが使用す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、サービスドメインのセキュリティグループへの OpenSearchアウトバウンド HTTPS トラフィックが許可されていることを確認します。また、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 トラフィックが OpenSearch サービスドメインのセキュリティグループで許可されていることを確認してください。Firehose ストリームと OpenSearch サービス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 トラフィックが許可されていることを確認してください。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。

Amazon Data Firehose にパブリック OpenSearchサーバーレス送信先へのアクセスを許可する

OpenSearch Serverless 送信先を使用している場合、Amazon Data Firehose はデータを OpenSearch Serverless コレクションに配信し、失敗したドキュメントまたはすべてのドキュメントを同時に 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 CloudWatch ロググループとストリームにもデータ配信エラーを送信します。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、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サーバーレスドメイン、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`s3:PutObjectAcl`に を追加します。これにより、バケット所有者は Amazon Data Firehose によって配信されるオブジェクトにフルアクセスできるようになります。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。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "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/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
```

```
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:region:account-id:key/key-id"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.region.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::bucket-name/
prefix*"
                }
            }
        },
        {
           "Effect": "Allow",
           "Action": [
               "kinesis:DescribeStream",
               "kinesis:GetShardIterator",
               "kinesis:GetRecords",
               "kinesis:ListShards"
           ],
           "Resource": "arn:aws:kinesis:region:account-id:stream/stream-name"
        },
        {
           "Effect": "Allow",
           "Action": [
               "logs:PutLogEvents"
           ],
           "Resource": [
               "arn:aws:logs:region:account-id:log-group:log-group-name:log-stream:log-
stream-name"
           ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "lambda:InvokeFunction",
               "lambda:GetFunctionConfiguration"
           ],
           "Resource": [
```

```
                "arn:aws:lambda:region:account-id:function:function-name:function-
version"
            ]
         },
         {
          "Effect": "Allow",
          "Action": "aoss:APIAccessAll",
          "Resource": "arn:aws:aoss:region:account-id:collection/collection-id"
        }
     ]
}
```

上記のポリシーに加えて、Amazon Data Firehose を設定して、データアクセスポリシーに次の最小限のアクセス許可を割り当てる必要があります。

```
[
   {
      "Rules":[
         {
            "ResourceType":"index",
            "Resource":[
               "index/target-collection/target-index"
            ],
            "Permission":[
               "aoss:WriteDocument",
               "aoss:UpdateIndex",
               "aoss:CreateIndex"
            ]
         }
      ],
      "Principal":[
         "arn:aws:sts::account-id:assumed-role/firehose-delivery-role-name/*"
      ]
   }
]
```

他の AWS の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 の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 Data Firehose に VPC OpenSearch内のサーバーレス送信先へのアクセスを許可する

OpenSearch Serverless コレクションが VPC にある場合は、前のセクションで説明したアクセス許可を 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するには、Amazon Data Firehose に次のアクセス許可を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 ストリームを作成した後は、これらのアクセス許可を取り消しないでください。これらのアクセス許可を取り消すと、 OpenSearch サービスが ENIsストリームのパフォーマンスが低下したり、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 の送信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 内のデータ配信用の ENIs を作成または追加できず、配信は低下または失敗します。

Firehose ストリームを作成または更新するときは、Firehose が OpenSearch Serverless コレクションにデータを送信するときに使用するセキュリティグループを指定します。 OpenSearch Serverless コレクションが使用するのと同じセキュリティグループを使用するか、別の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、 OpenSearch サーバーレス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 トラフィックが許可されていることを確認してください。また、 OpenSearchServerless コレクションのセキュリティグループが、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 トラフィックを許可していることを確認します。Firehose ストリームと OpenSearch Serverless コレクショ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 トラフィックが許可されていることを確認してください。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。

Amazon Data Firehose に Splunk 送信先へのアクセス権を付与する

Splunk 送信先を使用している場合、Amazon Data Firehose は Splunk HTTP Event Collector (HEC) エンドポイントにデータを配信します。また、そのデータを指定した Amazon S3 バケットにバックアップし、オプションで Amazon S3 サーバー側の暗号化に所有している 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 はログストリームに CloudWatchデータ配信エラーを送信します。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 で Cookie の有効期限が無効になっている期間ベースのスティッキーセッションを有効にし、有効期限は Application Load Balancer の最大 (7 日間) に設定されます。これを行う方法については、「Classic Load Balancer または Application Load Balancer の所要時間ベースのセッション維持」を参照してください。

Firehose ストリームを作成するときは、IAM ロールが必要です。Firehose は IAM ロールを引き受け、指定されたバケット、キー、 CloudWatch ロググループ、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`s3:PutObjectAcl`に を追加します。これにより、Amazon Data Firehose によって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、エラーログ CloudWatch 記録用の およびデータ変換 AWS Lambda 用の へのアクセス権を Amazon Data 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。Amazon Data Firehose は IAM を使用して Splunk にアクセスしません。Splunk へのアクセスには、HEC トークンが使用されます。

```
{
```

```
    "Version": "2012-10-17",
    "Statement":
    [
        {
            "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/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:region:account-id:key/key-id"
           ],
           "Condition": {
               "StringEquals": {
                   "kms:ViaService": "s3.region.amazonaws.com"
               },
               "StringLike": {
                   "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::bucket-name/
prefix*"
               }
           }
        },
        {
           "Effect": "Allow",
           "Action": [
               "kinesis:DescribeStream",
               "kinesis:GetShardIterator",
               "kinesis:GetRecords",
               "kinesis:ListShards"
```

```
            ],
            "Resource": "arn:aws:kinesis:region:account-id:stream/stream-name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:log-group-name:log-stream:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:region:account-id:function:function-name:function-
version"
            ]
        }
    ]
}
```

他の AWS の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 の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 IP アドレスのブロックを解除して、Splunk プラットフォームへのアクセスを Amazon Data Firehose に許可します。Amazon Data Firehose は現在、次の CIDR ブロックを使用しています。

- 米国東部 (オハイオ) 用の `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) 用の `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) 用の `13.57.180.0/26`

- 米国西部 (オレゴン) 用の 34.216.24.32/27, 34.216.24.192/27, 34.216.24.224/27
- 18.253.138.192/26 AWS GovCloud （ 米国東部) 用の
- 52.61.204.192/26 AWS GovCloud （ 米国西部) 用の
- アジアパシフィック (香港) 用の 18.162.221.64/26
- アジアパシフィック (ムンバイ) 用の 13.232.67.64/26
- アジアパシフィック (ソウル) 用の 13.209.71.0/26
- アジアパシフィック (シンガポール) 用の 13.229.187.128/26
- アジアパシフィック (シドニー) 用の 13.211.12.0/26
- アジアパシフィック (東京) 用の 13.230.21.0/27, 13.230.21.32/27
- イスラエル (テルアビブ) 用の 51.16.102.64/26
- カナダ (中部) 用の 35.183.92.64/26
- 40.176.98.128/26 カナダ西部 (カルガリー) 用の
- 欧州 (フランクフルト) 用の 18.194.95.192/27, 18.194.95.224/27, 18.195.48.0/27
- 欧州 (アイルランド) 用の 34.241.197.32/27, 34.241.197.64/27, 34.241.197.96/27
- 欧州 (ロンドン) 用の 18.130.91.0/26
- 欧州 (パリ) 用の 35.180.112.0/26
- 欧州 (ストックホルム) 用の 13.53.191.0/26
- 中東 (バーレーン) 用の 15.185.91.64/26
- 南米 (サンパウロ) 用の 18.228.1.192/26
- 欧州 (ミラノ) 用の 15.161.135.192/26
- アフリカ (ケープタウン) 用の 13.244.165.128/26
- アジアパシフィック (大阪) 用の 13.208.217.0/26
- 中国 (北京) 用の 52.81.151.64/26
- 中国 (寧夏) 用の 161.189.23.128/26
- アジアパシフィック (ジャカルタ) 用の 108.136.221.128/26
- 中東 (UAE) 用の 3.28.159.64/26
- イスラエル (テルアビブ) 用の 51.16.102.64/26
- 欧州 (チューリッヒ) 用の 16.62.183.64/26
- アジアパシフィック (ハイデラバード) 用の 18.60.192.192/26
- アジアパシフィック (メルボルン) 用の 16.50.161.192/26

Snowflake または HTTP エンドポイントへのアクセス

送信先が HTTP エンドポイントまたは Snowflake パブリッククラスターの場合、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 または 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進入ルールに追加します。

> **Note**
>
> 通知は、関連するトピックと同じ AWS リージョンの IP アドレスから常に送信されるわけではありません。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。

Amazon Data Firehose に Snowflake の送信先へのアクセス権を付与する

Snowflake を送信先として使用すると、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ます。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、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`s3:PutObjectAcl`に を追加します。これにより、バケット所有者に Firehose によって配信されるオブジェクトへのフルアクセスが付与されます。このポリシーは、エラーログ CloudWatch 記録のための 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 IAM を使用して Snowflake にアクセスしません。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。

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

```
"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/*"
            ]
        },
        {
"Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:region:account-id:key/key-id"
           ],
           "Condition": {
"StringEquals": {
"kms:ViaService": "s3.region.amazonaws.com"
               },
               "StringLike": {
"kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::bucket-name/prefix*"
               }
           }
        },
        {
"Effect": "Allow",
           "Action": [
               "kinesis:DescribeStream",
               "kinesis:GetShardIterator",
               "kinesis:GetRecords",
               "kinesis:ListShards"
           ],
           "Resource": "arn:aws:kinesis:region:account-id:stream/stream-name"
        },
        {
"Effect": "Allow",
```

```
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:log-group-name:log-stream:*"
            ]
        }
    ]
}
```

他の AWS のサービスが 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 の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 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 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、クラスターが属`AwsVpceIds`AWS リージョン する次の 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「ネットワークルールの作成」を参照してください。

クラスターがあ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 vpce-0cec34343d48f537b
米国東部 (バージニア北部)	vpce-0b4d7e8478e141ba8 vpce-0b75cd681fb507352 vpce-01c03e63820ec00d8 vpce-0c2cfc51dc2882422 vpce-06ca862f019e4e056 vpce-020cda0cfa63f8d1c

AWS リージョン	**VPCE IDs**
	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
	vpce-066b7e13cbfca6f6e
	vpce-0674541252d9ccc26

AWS リージョン	VPCE IDs
	vpce-03540b88dedb4b000 vpce-0b1828e79ad394b95 vpce-0dc0e6f001fb1a60d vpce-0d8f82e71a244098a vpce-00e374d9e3f1af5ce vpce-0c1e3d6631ddb442f
米国西部 (オレゴン)	vpce-0f60f72da4cd1e4e7 vpce-0c60d21eb8b1669fd vpce-01c4e3e29afdafbef vpce-0cc6bf2a88da139de vpce-0797e08e169e50662 vpce-033cbe480381b5c0e vpce-00debbdd8f9eb10a5 vpce-08ec2f386c809e889 vpce-0856d14310857b545
欧州 (フランクフルト)	vpce-068dbb7d71c9460fb vpce-0a7a7f095942d4ec9
欧州 (アイルランド)	vpce-06857e59c005a6276 vpce-04390f4f8778b75f2 vpce-011fd2b1f0aa172fd

AWS リージョン	VPCE IDs
アジアパシフィック (東京)	vpce-06369e5258144e68a vpce-0f2363cdb8926fbe8
アジアパシフィック (シンガポール)	vpce-049cd46cce7a12d52 vpce-0e8965a1a4bdb8941
アジアパシフィック (ソウル)	vpce-0aaa444d9001e1faa1 vpce-04a49d4dcfd02b884
アジアパシフィック (シドニー)	vpce-048a60a182c52be63 vpce-03c19949787fd1859

Amazon Data Firehose に HTTP エンドポイントの送信先へのアクセス権を付与する

Amazon Data Firehose を使用して、任意の HTTP エンドポイントの送信先にデータを配信できます。Amazon Data Firehose は、指定した Amazon S3 バケットにもそのデータをバックアップします。オプションで AWS KMS 、Amazon S3 サーバー側の暗号化用に所有している 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 CloudWatch ログストリームにデータ配信エラーを送信します。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 IAM ロールを引き受け、指定されたバケット、キー、 CloudWatch ロググループ、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`s3:PutObjectAcl`に を追加します。これにより、バケット所有者は Amazon Data Firehose によって配信されるオブジェクトにフルアクセスできるようになります。このポリシーは、エラーログ CloudWatch 記録用の およびデータ変換 AWS Lambda 用の へのアクセス権を Amazon Data 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。

> **Important**
>
> Amazon Data Firehose は、Datadog、Dynatrace、MongoDB、New Relic、Splunk、Sumo Logic など LogicMonitor、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 エンドポイントの送信先にアクセスするために IAM を使用しません。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 エンドポイントの送信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。

```
{
    "Version": "2012-10-17",
    "Statement":
    [
        {
            "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/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": [
               "arn:aws:kms:region:account-id:key/key-id"
           ],
           "Condition": {
               "StringEquals": {
```

```
                    "kms:ViaService": "s3.region.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::bucket-name/
prefix*"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:region:account-id:stream/stream-name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:log-group-name:log-stream:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:region:account-id:function:function-name:function-
version"
            ]
        }
    ]
}
```

他の AWS の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 の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、ソースが別のアカウント (AWS 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 Management Console 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::arn:role/mskaasTestDeliveryRole"
      },
      "Action": [
        "kafka:GetBootstrapBrokers",
        "kafka:DescribeCluster",
        "kafka:DescribeClusterV2",
        "kafka-cluster:Connect"
```

```
      ],
      "Resource": "arn:aws:kafka:us-east-1:arn:cluster/DO-NOT-TOUCH-mskaas-
provisioned-privateLink/xxxxxxxxx-2f3a-462a-ba09-xxxxxxxxxx-20"  // ARN of the
 cluster
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::arn: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::233450236687: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 ストリームがデータを取り込む Apache Kafka トピックを指定します。Firehose ストリームが作成されると、このトピックを更新することはできません。
6. [Save changes] (変更の保存) 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 ストリームを作成するを選択します。
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 ストリームがデータを取り込む Apache Kafka トピックを指定します。Firehose ストリームが作成されると、このトピックを更新することはできません。
5. 配信ストリーム名で、Firehose ストリームの名前を指定します。

Firehose ストリームを作成するときのアカウント B には、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 ストリームに付与する IAM ロール (の使用時にデフォルトで作成 AWS Management Console) が必要です。

AWS Management Console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: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: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: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、「レコード変換と形式変換を設定する」を参照してください。

Amazon S3 の送信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、1 つの 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 Amazon S3 の送信先を設定できます。次の手順は、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ています。

1. Amazon S3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。アクセスポリシーの Amazon S3 アクションのリスト`s3:PutObjectAcl`にを追加し、Amazon Data Firehose によって配信されるオブジェクトへのフルアクセスをアカウント B に付与します。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。

   ```
   {

       "Version": "2012-10-17",
       "Id": "PolicyID",
       "Statement": [
           {
               "Sid": "StmtID",
               "Effect": "Allow",
               "Principal": {
   ```

```
                "AWS": "arn:aws:iam::accountA-id:role/iam-role-name"
            },
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

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 の下に Firehose ストリームを作成します。

サービス宛先への OpenSearchクロスアカウント配信

AWS CLI または Amazon Data Firehose APIs を使用して、1 つの 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 サービスを送信先として指定できます。次の手順は、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 サービス宛先にデータを配信するように設定する方法の例を示しています。

1. the section called “Amazon Data Firehose にパブリック OpenSearch サービスの送信先へのアクセス権を付与する” 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 サービスポリシーを作成します。次の JSON は例です。

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

```
      "Principal": {
        "AWS": "arn:aws:iam::Account-A-ID:role/firehose_delivery_role "
      },
      "Action": "es:ESHttpGet",
      "Resource": [
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/_all/
_settings",
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/_cluster/
stats",
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/roletest*/
_mapping/roletest",
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/_nodes",
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/_nodes/
stats",
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/_nodes/*/
stats",
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/_stats",
        "arn:aws:es:us-east-1:Account-B-ID:domain/cross-account-cluster/roletest*/
_stats",
        "arn:aws:es:us-east-1:Account-B-ID: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 の下に 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、 for DomainARN OpenSearch Service ではなく `ClusterEndpoint`フィールドを指定します。

> **Note**
>
> 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 サービス送信先を作成するには、 AWS CLI または Amazon Data Firehose APIsを使用する必要があります。 AWS Management Console を使用して、この種のクロスアカウント設定を作成することはできません。

タグを使用したアクセスの制御

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、タグの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微調整できます。以下のサブセクションでは、さまざまな Amazon Data Firehose オペレーションに対して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」を参照してください。

```
{
    "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。

```
{
    "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。

```
{
    "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 Amazon Data Firehose オペレーション

CreateDeliveryStream、、、UntagDeliveryStreamおよび 以外のすべての Amazon Data Firehose TagDeliveryStreamオペレーションにはListDeliveryStreams、

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、 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制御します。

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 の という名前のタグを持つ Data Firehose MyKey ストリームにのみ適用されますMyValue。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。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
            "Effect": "Deny",
            "Action": "firehose:DescribeDeliveryStream",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                      "firehose:ResourceTag/MyKey": "MyValue"
                  }
            }
        }
    ]
}
```

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送信先に接続し、データストリームを配信できます。これにより、 AWS Management Console または API パラメータのいずれかで、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、パスワードの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されます。

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。

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。

送信先ごとに、次のセクションに示すように、シークレットのキーと値のペアを正しい JSON 形式で指定する必要があります。シークレットの JSON 形式が送信先に従って正しくない場合、Amazon Data Firehose は送信先への接続に失敗します。

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 のシークレット形式

```
{
    "hec_token":  "<hec token>"
}
```

Snowflake のシークレットの形式

```
{
    "user":  "<user>",
    "private_key":  "<private_key>",
    "key_passphrase":  "<passphrase>" // optional
}
```

HTTP エンドポイント、Coralogix、Datadog、Dynatrace、Elastic、Honeycomb LogicMonitor、Logz.io、MongoDB Cloud、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、「 ユーザーガイド」の「 AWS Secrets Manager シークレットの作成AWS Secrets Manager 」のステップに従います。

シークレットを使用する

AWS Secrets Manager を使用して、Amazon Redshift、HTTP エンドポイント、Snowflake、Splunk、Coralogix、Datadog、Dynatrace、Elastic、Honeycomb、Logz.io、MongoDB Cloud、New Relic LogicMonitorなどのストリーミング送信先に接続するための認証情報またはキーを保存することをお勧めします。

これらの送信先の認証は、Firehose ストリームの作成時に マネジメントコンソールを使用して AWS Secrets Manager で設定できます。詳細については、「送信先設定を構成する」を参照してください。または、 CreateDeliveryStreamおよび UpdateDestination API オペレーションを使用して、Secrets Manager で認証を設定することもできます。

Firehose は、シークレットを暗号化でキャッシュし、送信先への接続のたびにシークレットを使用します。キャッシュは 10 分ごとに更新され、最新の認証情報が使用されるようになります。

ストリームのライフサイクル中いつでも 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ができます。Secrets Manager を使用してシークレットを取得しない場合は、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 ではこの機能に追加料金はかかりませんが、Secrets Manager へのアクセスとメンテナンスには課金されます。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。

Firehose へのアクセスを許可してシークレットを取得する

Firehose が からシークレットを取得するには AWS Secrets Manager、その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、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、「 のアクセス許可ポリシーの例 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と の両方のkms:Decryptアクセス許可が必要です。CMK ポリシーでは、IAM ロールに の実行を許可する必要がありますkms:Decrypt。

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "Secret ARN"
        },
        {
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "KMSKeyARN"
        }
    ]
}
```

- 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、この設定は実行できません。

- シークレットが別のアカウントに保存され、CMK で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` アクセス許可と CMK に対する アクセス`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 ロールに必要なアクセス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。

シークレットのローテーション

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 AWS Secrets Manager を自動的にローテーションするように を設定できます。これにより、長期シークレットを短期シークレットに置き換えることができます。これにより、侵害のリスクを軽減できます。詳細については、「 AWS Secrets Manager ユーザーガイド」のAWS Secrets Manager「シークレットのローテーション」を参照してください。

Amazon Data Firehose コンソールを使用して IAM ロールを管理する

Amazon Data Firehose は、リアルタイムの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および変換するように 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、Firehose ストリームを作成または編集するときに Firehose にアクセス許可を付与する IAM ロールを最初に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アクセス許可にこの 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 機能を有効にしてソースレコードを変換するシナリオを考えてみましょう。この場合、次に示すように、S3 バケットにアクセスして Lambda 関数を呼び出すアクセス許可を Firehose に付与する IAM ロールを指定する必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "lambdaProcessing",
        "Effect": "Allow",
        "Action": ["lambda:InvokeFunction", "lambda:GetFunctionConfiguration"],
        "Resource": "arn:aws:lambda:us-east-1:<account id>: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、これらのロールの提供方法を選択できます。次のいずれかのオプションから選択できます。

- 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、送信元と送信先に必要な適切な信頼ポリシーと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。

コンソールから新しい IAM ロールを作成する

または、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することもできます。

Firehose がユーザーに代わって IAM ロールを作成すると、そのロールには、Firehose ストリーム設定に基づいて必要なアクセス許可を付与するすべてのアクセス許可と信頼ポリシーが自動的に含まれます。

例えば、 機能を使用してソースレコードの変換 AWS Lambdaを有効にしなかった場合、コンソールは アクセス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-1:<account id>:function:
%FIREHOSE_POLICY_TEMPLATE_PLACEHOLDER%"
}
```

> **Note**
>
> リソースに対するアクセス許可を付与しない%FIREHOSE_POLICY_TEMPLATE_PLACEHOLDER%ため、 を含むポリシーステートメントは無視しても安全です。

コンソールで 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。

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

> **Important**
>
> - 複数の Firehose ストリームに同じコンソール管理の IAM ロールを使用しないでください。そうしないと、IAM ロールが過度に許容されたり、エラーが発生する可能性があります。
> - コンソール管理の IAM ロールからアクセス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、そのポリシーステートメントを新しいロールにアタッチされたアクセス許可ポリシーにコピーします。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ときは、コンソールが変更を加えないように、ARN に service-role 文字列を含まない IAM ロールを必ず選択してください。

コンソールから IAM ロールを作成する手順

1. https://console.aws.amazon.com/firehose/ で Firehose コンソールを開きます。
2. Firehose ストリームの作成 を選択します。
3. 送信元と送信先を選択します。詳細については、「Firehose ストリームを作成する」を参照してください。
4. 送信先設定を選択します。詳細については、「送信先設定を構成する」を参照してください。
5. 詳細設定の「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 ロールの選択 オプションを選択します。Firehose コンソールは、自分のロールを変更しません。

6. Firehose ストリームの作成 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アクセス許可ポリシーを更新し、設定とアクセス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 として使用して 機能を使用してソースレコードの変換 AWS Lambdaを有効にすると`exampleLambdaFunction`、 アクセス許可ポリシーに次のポリシーステートメントが表示されます。

```
{
  "Sid": "lambdaProcessing",
  "Effect": "Allow",
  "Action": [
    "lambda:InvokeFunction",
    "lambda:GetFunctionConfiguration"
```

```
  ],
  "Resource": "arn:aws:lambda:us-east-1:<account id>:function:exampleLambdaFunction:
$LATEST"
}
```

> **Important**
>
> コンソール管理の IAM ロールは、自律型であるように設計されています。コンソールの外部でアクセス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

1. https://console.aws.amazon.com/firehose/ で Firehose コンソールを開きます。
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 IAM ロールを ARN のパターン service-role で更新します。既存の IAM ロールオプションを選択するときは、コンソールが変更を加えないように、ARN に service-role 文字列を含まない IAM ロールを必ず選択してください。

5. [変更の保存] を選択します。

Amazon Data Firehose のモニタリング

Amazon Data Firehose は、Firehose ストリームのモニタリング機能を提供します。詳細については、「モニタリング」を参照してください。

Amazon Data Firehose のコンプライアンス検証

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、「コンプライアンスAWS 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 「Artifact 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、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まります。Data Firehose の使用が HIPAA、PCI、FedRAMP などの標準に準拠していることを前提としている場合、 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 にデプロイする手順について説明します AWS。
- 「HIPAA セキュリティとコンプライアンスの設計」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、リソース設定が社内プラクティス、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 – 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 の耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョンとアベイラビリティーゾーンを中心に構築されています。 AWS リージョンは、低レイテンシー、高スループット、および高度に冗長なネットワークで接続された、物理的に分離された複数の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。

災害対策

Amazon Data Firehose はサーバーレスモードで実行され、自動移行を実行することで、ホストの劣化、アベイラビリティーゾーンの可用性、その他のインフラストラクチャ関連の問題に対応します。この場合、Amazon Data Firehose は、Firehose ストリームがデータを失うことなく移行されるようにします。

Amazon Data Firehose のインフラストラクチャセキュリティ

マネージドサービスである Amazon 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 AWS 環境を設計するには、「セキュリティの柱 AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

が AWS 公開した API コールを使用して、ネットワーク経由で Firehose にアクセスします。クライアントは以下をサポートする必要があります:

- Transport Layer Security (TLS)。TLS 1.2 は必須で TLS 1.3 がお勧めです。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは、Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

> **Note**
>
> 送信 HTTPS リクエストの場合、Amazon Data Firehose は、送信先側でサポートされている最高の TLS プロトコルバージョンを自動的に選択する HTTP ライブラリを使用します。

VPC エンドポイント (PrivateLink)

Amazon Data Firehose は VPC エンドポイント () をサポートしていますPrivateLink。詳細については、「での Amazon Data Firehose の使用 AWS PrivateLink」を参照してください。

Amazon Data Firehose のセキュリティのベストプラクティス

Amazon Data Firehose には、独自のセキュリティポリシーを開発および実装する際に考慮すべきいくつかの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、これらは処方箋ではなく、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

アクセス許可を付与するときは、Amazon Data Firehose リソースに対するアクセス許可を取得するユーザーを決定します。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

プロデューサーアプリケーションとクライアントアプリケーションは、Firehose ストリームにアクセスするために有効な認証情報を持っている必要があり、Firehose ストリームは送信先にアクセスするために有効な認証情報を持っている必要があります。 AWS 認証情報は、クライアントアプリケーションまたは 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、プロデューサーおよびクライアントアプリケーションが Firehose ストリームにアクセスするための一時的な認証情報を管理する必要があります。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 でのデータ保護」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。

で収集された情報を使用して CloudTrail、Amazon Data Firehose に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

詳細については、「the section called “を使用した Amazon Data Firehose API コールの口グ記録 AWS CloudTrail”」を参照してください。

Amazon Data Firehose データ変換

Amazon Data Firehose は Lambda 関数を呼び出して、受信ソースデータを変換し、変換されたデータを宛先に配信できます。Firehose ストリームを作成するときに、Amazon Data Firehose データ変換を有効にできます。

データ変換フロー

Firehose データ変換を有効にすると、Firehose は受信データをバッファします。バッファリングサイズのヒントの範囲は 0.2 MB ~ 3MB。デフォルトの Lambda バッファリングサイズヒントは、Splunk と Snowflake を除くすべての送信先で 1 MB です。Splunk および Snowflake の場合、デフォルトのバッファリングヒントは 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ヒントは、Snowflake を除くすべての送信先で 60 秒です。Snowflake の場合、デフォルトのバッファリングヒント間隔は 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 というProcessorParameter名前の で設定します`IntervalInSeconds`。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で指定された Lambda 関数を AWS Lambda 非同期的に呼び出します。変換されたデータは Lambda から Firehose に送信されます。Firehose は、指定された送信先バッファリングサイズまたはバッファリング間隔のいずれか早い方に達したときに送信先に送信します。

> **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。

データ変換とステータスモデル

Lambda から変換されたすべてのレコードには、次のパラメータが含まれている必要があります。含まれていない場合、Amazon Data Firehose はそれらを拒否し、データ変換の失敗として扱います。

Kinesis Data Streams と Direct PUT の場合:

recordId

レコード ID は、呼び出し中に 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またはの場合Dropped、Amazon Data Firehose はレコードが正常に処理されたと見なします。それ以外の場合、Amazon Data Firehose は処理に失敗したと見なします。

データ

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>"
}
```

Amazon MSK の場合

recordId

レコード ID は、呼び出し中に 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 の場合Dropped、Firehose はレコードが正常に処理されたと見なします。それ以外の場合、Firehose は処理に失敗したと見なします。

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>"
}
```

Lambda の設計図

これらのブループリントは、Lambda AWS 関数を作成して使用して 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 Management Console 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、 キーワードを検索`firehose`して Amazon Data Firehose Lambda ブループリントを検索します。

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 に送信されるプロセス CloudWatch ログ

  このブループリントは廃止されました。Firehose に送信された CloudWatch ログの処理については、「ログを使用した Firehose への書き込み CloudWatch 」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。

データ変換失敗の処理

ネットワークタイムアウトまたは Lambda 呼び出しの制限に達したために Lambda 関数の呼び出しが失敗した場合、Amazon Data Firehose はデフォルトで呼び出しを 3 回再試行します。呼び出しが成功しない場合、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。このタイプの障害では、呼び出しエラーを Amazon CloudWatch Logs にログ記録できます。詳細については、「CloudWatch ログを使用した Amazon Data Firehose のモニタリング」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 の場合`ProcessingFailed`、Amazon Data Firehose はレコードを処理に失敗したものとして扱います。このタイプの障害については、Lambda 関数から Amazon CloudWatch Logs にエラーログを発行できます。詳細については、「 デベロッパーガイド」の「 の Amazon CloudWatch Logs へのアクセス AWS Lambda」を参照してください。 AWS Lambda

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刻。

`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刻。

`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)。

Lambda の呼び出し時間

Amazon Data Firehose は、最大 5 分の Lambda 呼び出し時間をサポートします。Lambda 関数の完了に 5 分以上かかる場合、次のエラーが発生します。Firehose は AWS Lambda の呼び出し時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 の動作については、「」を参照してくださいthe section called “データ変換失敗の処理”。

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送信先に配信しながら、変換されていないすべてのレコードを S3 バケットに同時にバックアップできます。Firehose ストリームを作成または更新するときに、ソースレコードのバックアップを有効にできます。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 での動的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 (`customer_id`や などtransaction_id) を使用して Firehose でストリーミングデータを継続的にパーティション化し、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 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ます QuickSight。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、従来、データをキャプチャして Amazon S3 にロードするために使用されていました。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転送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、数分または数時間短縮 time-to-insight 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

- パーティショニングキー
- 動的パーティショニングの Amazon S3 バケットプレフィックス
- 集約データの動的パーティショニング
- S3 にデータを配信するときに新しい行区切り文字を追加する
- 動的パーティショニングを有効にする方法
- 動的パーティショニングのエラー処理
- データバッファリングと動的パーティショニング

パーティショニングキー

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の作成でサポートされているメソッドは次のとおりです。

- インライン解析 - このメソッドは、Firehose 組み込みサポートメカニズム jq パーサー を使用して、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を抽出します。現在、`jq 1.6`バージョンのみがサポートされています。
- 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。

インライン解析によるパーティショニングキーの作成

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 形式でエンコードする必要があることに注意してください。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 は上記の右側の列を使用して、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 関数を用い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、統合された AWS Lambda 関数と独自のカスタムコードを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 で現在利用可能な既存の変換 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。

以下は、入力から出力へ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 で Lambda 関数を処理する Firehose ストリームの例です。

```
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'),
                          "date": 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
```

以下は、入力から出力へ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 で Lambda 関数を処理する Firehose ストリームの例です。

```
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)
}
```

動的パーティショニングの Amazon S3 バケットプレフィックス

Amazon S3 を送信先として使用する Firehose ストリームを作成するときは、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有効にするすべての Firehose ストリームでは、S3 バケットプレフィックス値は、その Firehose ストリームの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。次に、Firehose は各データセットを評価済みの S3 プレフィックスに配信します。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 スタイル形式を使用して S3 バケットプレフィックス値を指定することもできます。例: customer_id=!{partitionKeyFromクエリ:customer_id}。

詳細については、Amazon S3 Firehose ストリームの作成」の「送信先に Amazon S3 を選択する」とAmazon S3 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。

集約データの動的パーティショニング

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。つまり、レコードの分離は連続する JSON オブジェクトに基づいています。デアグリゲーションは タイプでもかまいません。つまりDelimited、レコードの分離は、指定されたカスタム区切り文字に基づいて実行されます。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、次の文字列をカスタム区切り文字 として使用する場合は####、base-64 でエンコードされた形式で指定し、 に変換する必要がありますIyMjIw==。

> **Note**
>
> JSON レコードを集約解除するときは、入力がサポートされている JSON 形式で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 1 行にする必要があります。JSON オブジェクトの配列は有効な入力ではありません。
> 正しい入力の例を次に示します。 {"a":1}{"a":2} and {"a":1}\n{"a":2}

正しくない入力の例を次に示します。 `[{"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、有効な JSON オブジェクトまたは各 API コール内の区切り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 でデータ変換機能を使用すると、データ変換の前に集約解除が適用されます。Firehose に送信されるデータは、集約解除 → Lambda によるデータ変換 → パーティショニングキーの順に処理されます。

S3 にデータを配信するときに新しい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 Amazon S3 に配信されるオブジェクト内のレコード間に新しい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、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る) が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ため、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。

動的パーティショニングを有効にする方法

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

> **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有効になっていない既存の Firehose ストリームに対して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手順については、「Amazon Firehose ストリームの作成」を参照してください。Firehose ストリームの送信先を指定するタスクに到達するときは、送信先として Amazon S3 を使用する Firehose ストリームでのみ動的パーティショニングがサポートされるため、「送信先として Amazon S3 を選択」セクションのステップに従ってください。

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、削除、または更新することで、設定を更新できます。更新すると、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 ストリームで無効にすることはできません。

動的パーティショニングのエラー処理

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、または指定されたパーティショニングキーの抽出に失敗した場合、または S3 プレフィックス値に含まれる式を評価する場合、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を有効にする Firehose ストリームの作成時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 送信先に配信できないすべてのレコードが含まれます。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。

この Firehose ストリームの動的パーティショニングを有効にする場合は、Firehose ストリームの 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。

データバッファリングと動的パーティショニング

Amazon Data Firehose は、受信ストリーミングデータを特定のサイズにバッファしてから、指定された宛先に配信します。新しい Firehose ストリームの作成時にバッファサイズとバッファ間隔を設定したり、既存の Firehose ストリームのバッファサイズとバッファ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。

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に、Firehose は内部的にマルチステージバッファリングを使用します。したがって、レコードのバッチの end-to-end 遅延は、設定されたバッファリングヒント時間の 1.5 倍になる可能性があります。これは、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できるようになるまで配信バッファ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サイズ制限またはバッファ時間間隔に達すると、Firehose はバッファデータを含むオブジェクトを作成し、指定された 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ると、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削除されます。

Firehose は、各パーティションのバッファサイズまたは間隔が個別に満たされると、各バッファデータを 1 つのオブジェクトとして配信します。アクティブなパーティションの数が Firehose ストリームあたり 500 の制限に達すると、Firehose ストリーム内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 () に配信されますactivePartitionExceeded。Amazon Data Firehose の制限フォームを使用して、特定の Firehose ストリームごとに最大 5000 個のアクティブなパーティションのクォータの引き上げをリクエストできます。さらに多くの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し、アクティブなパーティションをそれらに分散できます。

Firehose での入力レコードフォーマットの変換

Amazon Data Firehose は、データを Amazon S3 に保存する前に、入力データの形式を JSON から Apache Parquet または Apache ORC に変換できます。Parquet と ORC は、容量を節約し、JSON のような行指向の形式に比べ、より高速なクエリを可能にするカラム型のデータ形式です。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、まず JSON に変換できます。 AWS Lambda 詳細については、「データ変換」を参照してください。

トピック

- レコード形式の変換の要件
- JSON デシリアライザーの選択
- シリアライザーの選択
- 入力レコードの形式の変換 (コンソール)
- 入力レコードの形式の変換 (API)
- レコード形式変換のエラー処理
- レコード形式の変換例

レコード形式の変換の要件

Amazon Data Firehose では、レコードデータのフォーマットを変換するために次の 3 つの要素が必要です。

- 入力データの JSON を読み取るデシリアライザー — Apache Hive JSON と OpenX JSON 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。 SerDe SerDe

  > **Note**
  >
  > 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。
  > たとえば、これは正しい入力です: `{"a":1}{"a":2}`
  > そして、これは間違った入力です。`[{"a":1}, {"a":2}]`

- データの解釈方法を決定するスキーマ – AWS Glue を使用して AWS Glue Data Catalogにスキーマを作成します。次に Amazon Data Firehose はそのスキーマを参照し、それを使用して入力デー

タを解釈します。同じスキーマを使用して Amazon Data Firehose 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 開発者ガイド』の「 AWS Glue データカタログへの入力」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。

- 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) に変換するシリアライザー — ORC または Parquet の 2 種類のシリアライザーのいずれかを選択できます。 SerDe SerDe

> **Important**
>
> レコード形式の変換を有効にすると、Amazon Data Firehose の送信先を Amazon OpenSearch サービス、Amazon Redshift、または Splunk に設定することはできません。フォーマット変換が有効になっていると、Firehose ストリームに使用できる送信先は Amazon S3 だけです。

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

JSON デシリアライザーの選択

入力 JSON SerDe に以下の形式のタイムスタンプが含まれている場合は、OpenX JSON 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 を通じてこのデシリアライザーで使用できるオプションの詳細については、「OpenX」を参照してください。JsonSerDe

どのデシリアライザーを選択すればよい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 SerDe、OpenX JSON を使用してください。

上記以外の形式のタイムスタンプがある場合は、Apache Hive JSON を使用してください。 SerDe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、「クラス」を参照してください。 DateTimeFormat

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。フォーマットを指定しない場合、Amazon Data Firehose java.sql.Timestamp::valueOf はデフォルトで使用します。

Hive JSON SerDe では次の操作は許可されません。

- 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 がの場合{"a":"123"}、Hive SerDe はエラーを返します。

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。

シリアライザーの選択

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。2 つのシリアライザーオプションについて詳しくは、ORC と Parquet をご覧ください。 SerDe SerDe

入力レコードの形式の変換 (コンソール)

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変換を有効にできます。データ形式の変換が有効になっている場合、Firehose ストリームに設定できる送信先

は Amazon S3 だけです。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 Amazon Data Firehose が使用している 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 が採用している Snappy フレーミング形式については、.java を参照してください。BlockCompressorStream

Firehose データストリームのデータ形式変換を有効にするには

1. にサインインし AWS Management Console、https://console.aws.amazon.com/firehose/ にある 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、の手順に従って新しい Firehose ストリームを作成します。Firehose ストリームを作成する
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。

入力レコードの形式の変換 (API)

Amazon Data Firehose で入力データの形式を JSON から Parquet または ORC に変換させたい場合は、extendeDS3 または DestinationConfigurationextendeDS3 DataFormatConversionConfigurationでオプションの要素を指定してください。DestinationUpdate指定する場合、以下の制限が適用されます。DataFormatConversionConfiguration

- ではBufferingHints、レコード形式の変換を有効にした場合、64 `SizeInMBs` 未満の値には設定できません。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。
- extendeDS3 DestinationConfiguration または extendeDS3 `CompressionFormat` ではをに設定する必要があります。DestinationUpdate `UNCOMPRESSEDCompressionFormat` のデフォルト値は `UNCOMPRESSED` です。そのため、extendeDS3 ではこれを未指定のままにしておくこともできます。DestinationConfiguration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 Amazon Data Firehose が使用している 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 が採用している Snappy フレーミング形式については、.java を参照してください。BlockCompressorStream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。

レコード形式変換のエラー処理

Amazon Data Firehose はレコードを解析または逆シリアル化できない場合 (データがスキーマと一致しない場合など)、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 は書き込みを永久に再試行し、それ以降の配信をブロックします。Amazon Data Firehose は、失敗したレコードごとに、次のスキーマを使用して JSON ドキュメントを書き込みます。

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

レコード形式の変換例

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

Amazon Managed Service for Apache Flink を使用する

Amazon Managed Service for Apache Flink を使用すると、Java、Scala、SQL のいずれかを使用してストリーミングデータを処理および分析できます。このサービスを使用すると、ストリーミングソースに対するコードを作成して実行し、時系列分析の実行、ダッシュボードへのリアルタイムフィード、メトリクスのリアルタイム作成を行うことができます。

Amazon Managed Service for Apache Flink との統合の例については、「例: Amazon Data Firehose への書き込み」を参照してください。

この演習では、ソースとして Kinesis データストリーム、シンクとして Firehose ストリームを持つ Apache Flink アプリケーションを作成します。シンクを使用すると、Amazon S3 バケット内のアプリケーションの出力を検証できます。

開始前に、次の前提条件を準備します。

- Managed Service for Apache Flink アプリケーションのコンポーネント
- この演習を実行するための前提条件

Amazon Data Firehose データ配信を理解する

Firehose ストリームにデータが送信されると、選択した送信先にデータが自動的に配信されます。

> **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。次に、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と、Amazon Data Firehose はレコードを宛先に配信する前にレコードの集約を解除します。データを変換するように Firehose ストリームを設定すると、Amazon Data Firehose はレコードを に配信する前にレコードの集約を解除します AWS Lambda。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 Amazon Kinesis Data Streams 開発者ガイドの「集約」を参照してください。

トピック

- データ配信形式を設定する
- データ配信の頻度を理解する
- 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
- サービスのインデックスロー OpenSearchテーションを設定する
- AWS アカウントとリージョン間の配信を理解する
- 重複レコード
- Firehose ストリームの一時停止と再開

データ配信形式を設定する

Amazon Simple Storage Service (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受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。レコード間に新しい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、新しい行区切り文字を追加できます。

Amazon Redshift へのデータ配信の場合、Firehose はまず、前述の形式で受信データを 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 でプロビジョニングされたクラスターまたは Amazon Redshift Serverless ワークグループにデータをロードします。Amazon Data Firehose が複数の受信レコードを Amazon S3 オブジェクトに連結した後、Amazon S3 オブジェクトを Amazon Redshift でプロビジョニングされたクラスターまたは 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 サービスおよび OpenSearch サーバーレスへのデータ配信の場合、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受信レコードをバッファリングします。次に、 OpenSearch サービスクラスターまたは OpenSearch サーバーレスコレクションに複数のレコードをインデックスするための OpenSearch サービスまたは OpenSearch サーバーレス一括リクエストを生成します。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、単一行の JSON オブジェクトにフラット化されていることを確認してください。また、サービスクラスターの OpenSearch `rest.action.multi.allow_explicit_index`オプションを true (デフォルト) に設定して、レコードごとに設定された明示的なインデックスを持つ一括リクエストを受け取る必要があります。詳細については、Amazon OpenSearch Service デベロッパーガイドの「サービス設定の詳細オプション」を参照してください。 OpenSearch

Splunk へのデータ配信の場合、Amazon Data Firehose は送信するバイトを連結します。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。

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 エンドポイントにデータを配信する場合、統合された Amazon Lambda サービスを使用して、受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。

Snowflake へのデータ配信の場合、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。挿入呼び出しを行うと、Firehose は Snowflake にデータがコミットされるまでにかかった時間を測定する 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 配列はサポートされていません。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。

データ配信の頻度を理解する

Firehose の各送信先には、独自のデータ配信頻度があります。詳細については、「バッファリングヒントを理解する」を参照してください。

データ配信の失敗を処理する

各 Amazon Data Firehose 送信先には、独自のデータ配信失敗処理があります。

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場合、Amazon Data Firehose が引き受ける IAM ロールがバケットにアクセスできない、ネットワークが失敗する、または同様のイベントが発生する可能性があります。このような条件下では、Amazon Data Firehose は配信が成功するまで最大 24 時間再試行し続けます。Amazon Data Firehose の最大データストレージ時間は 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。

Amazon Redshift

Amazon Redshift の送信先では、Firehose ストリームの作成時に再試行時間 (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、メンテナンス中のクラスターまたはワークグループ、ネットワーク障害などが考えられます。これらの条件下では、Amazon Data Firehose は指定された期間再試行し、Amazon S3 オブジェクトの特定のバッチをスキップします。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サーバーレス

OpenSearch サービスと OpenSearch サーバーレスの送信先では、Firehose ストリームの作成中に再試行時間 (0 ~ 7200 秒) を指定できます。

OpenSearch サービスクラスターまたは OpenSearch サーバーレス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ことがあります。例えば、Firehose ストリームのサービス OpenSearch クラスターまたは OpenSearch サーバーレスコレクション設定が正しくない、メ

ンテナンス中の OpenSearch サービスクラスターまたは OpenSearch サーバーレスコレクション、ネットワーク障害、または同様のイベントがある可能性があります。これらの条件下では、Amazon Data Firehose は指定された期間再試行し、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。

Service の場合 OpenSearch 、各ドキュメントの JSON 形式は次のとおりです。

```
{
    "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 の場合、各ドキュメントの JSON 形式は次のとおりです。

```
{
    "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 Splunk にデータを送信すると、Splunk からの確認応答を待機します。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 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、Amazon Data Firehose はそれをデータ配信の失敗と見なし、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、最初の試行でも再試行でも、確認タイムアウトカウンターが再起動されます。Splunk から送達確認が来るのを待機します。再試行期間が終了しても、Amazon Data Firehose は確認応答を受信するか、確認タイムアウトに達するまで確認応答を待機します。確認がタイムアウトすると、Amazon Data Firehose は再試行カウンターに残り時間があるかどうかを確認します。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口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":
"MiAyNTE2MjAyNzIyMDkgZW5pLTA1ZjMyMmQ1IDIxOC45Mi4xODguMjE0IDE3Mi4xNi4xLjE2NyAyNTIzMyAxNDMzID
  "EventId": "49577193928114147339600778471082492393164139877200035842.0"
}
```

HTTP エンドポイント送信先

Amazon Data Firehose が HTTP エンドポイントの送信先にデータを送信すると、この送信先からの応答を待機します。エラーが発生した場合、または応答タイムアウト期間内に応答が到着しない場合、Amazon Data 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、Amazon Data Firehose はそれをデータ配信の失敗と見なし、データを Amazon S3 バケットにバックアップします。

Amazon Data Firehose は、最初の試行でも再試行でも、データを HTTP エンドポイントの送信先に送信するたびに、応答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。再試行期間が終了しても、Amazon Data Firehose は応答を受信するか、応答タイムアウトに達するまで応答を待機します。レスポンスがタイムアウトした場合、Amazon Data Firehose は再試行カウンターに残り時間があるかどうかを確認します。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 の送信先

Snowflake 送信先の場合、Firehose ストリームを作成するときに、オプションの再試行時間 (0 ~ 7200 秒) を指定できます。再試行時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 の送信先設定の誤り、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ことがあります。再試行ポリシーは、再試行不可能なエラーには適用されません。例えば、Snowflake が JSON ペイロードをテーブルに追加の列がないために拒否した場合、Firehose は再度配信しようとしません。代わりに、S3 エラーバケットへの JSON ペイロードの問題によるすべての挿入失敗のバックアップが作成されます。

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が失敗した場合、Firehose はデータを再試行せず、S3 バケットに書き込みます。再試行期間は、Snowflake サービスの問題、一時的なネットワーク障害などの障害にのみ適用されます。これらの条件下では、Firehose は指定された期間再試行してから 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、手動バックフィルに使用できます。

以下は、S3 に配信する各レコードの JSON の例です。

```
{
    "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 < 評価プレフィックス><suffix> の形式に従います。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>-<year>-<month>-<day>-<hour>-<minute>-<second>-<uuid><file 拡張> <Firehose ストリームバージョン > は 1 で始まり、Firehose ストリームの設定変更ごとに 1 ずつ増加します。Firehose ストリームの設定 (S3 バケットの名前、バッファリングヒント、圧縮、暗号化など) を変更できます。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 はデフォルトの時間プレフィックスを の形式で追加しますYYYY/MM/dd/HH。このプレフィックスはバケットに論理階層を作成し、各スラッシュ (/) は階層にレベルを作成します。この構造を変更するには、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します。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、タイムプレフィックスとサフィックスに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。例えば、UTC の代わりに日本標準時を使用するには、 AWS Management Console 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でサポートするタイムゾーンが含まれています。

タイムゾーン

以下は、Firehose が S3 プレフィックス設定でサポートするタイムゾーンのリストです。

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
```

<file extension> 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。データ形式の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追加します。次の表に、Firehose によって追加されたデフォルトのファイル拡張子を示します。

構成	ファイル拡張子
データ形式の変換: Parquet	.parquet
データ形式の変換: ORC	.orc
圧縮: Gzip	.gz
圧縮: Zip	.zip
圧縮: スナップ	.snappy
圧縮: Hadoop-Snappy	.hsnappy

Firehose コンソールまたは API で必要な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 (.) で始まり、0 ~ 9a ~ z! ~ _.*'() の文字を使用できます。ファイル拡張子は 128 文字を超えることはできません。

> **Note**
>
> ファイル拡張子を指定すると、データ形式の変換または圧縮が有効になっているときに 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。

サービスのインデックスロー OpenSearchテーションを設定する

OpenSearch サービス送信先では、、、NoRotation、、OneHourOneDayOneWeekまたは の 5 つのオプションのいずれかから時間ベースのインデックスローテーションオプションを指定できますOneMonth。

選択したローテーションオプションに応じて、Amazon Data Firehose は指定したインデックス名に UTC 到着タイムスタンプの一部を追加します。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。次の例は、各インデックスローテーションオプションの結果のインデックス名を OpenSearch Service で示myindexしています。指定されたインデックス名は で、到着タイムスタンプは です2016-02-25T13:00:00Z。

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。

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 、アカウント間で HTTP エンドポイントの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 アカウントに属している可能性があります。

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 AWS 。Firehose ストリームから AWS リージョン外の HTTP エンドポイントの送信先にデータを配信することもできます。例えば、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することもできます。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。

重複レコード

Amazon Data Firehose は、データ配信に at-least-once 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、Amazon Data Firehose による配信の再試行によって、元のデータ配信リクエストが最終的に処理された場合に重複が発生することがあります。これは、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。

Firehose ストリームの一時停止と再開

Firehose ストリームを設定すると、ストリームソースで利用可能なデータが送信先に継続的に配信されます。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。この方法については次のセクションで紹介します。

> **Important**
>
> 以下に説明する方法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送信先に配信されます。これは、このアプローチの既知の制限であり、複数回の再試行が失敗として追跡された後に送信先に以前に配信できなかった少数のレコードが原因で発生します。

Firehose が配信失敗を処理する方法を理解する

Firehose ストリームを設定する場合、 OpenSearch、Splunk、HTTP エンドポイントなどの多くの送信先に対して、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、「データ配信の失敗処理」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の詳細については、Amazon S3先 へのアクセス権を Firehose に付与する」を参照してください。Firehose は、(a) ストリームの送信先にデータを配信できず、(b) 配信に失敗したデータをバックアップ S3 バケットに書き込めない場合、データが送信先に配信されるか、バックアップ S3 の場所に書き込まれるまで、ストリーム配信を効果的に一時停止します。

Firehose ストリームの一時停止

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 S3 バックアップ場所に書き込むためのアクセス許可を削除します。例えば、 OpenSearch 送信先で Firehose ストリームを一時停止する場合は、アクセス許可を更新することでこれを行うことができます。詳細については、「パブリック OpenSearch サービスの送信先 へのアクセスを Firehose に許可する」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、ストリームの送信先をオフにするか (送信先 OpenSearch ドメインをオフにするなど)、Firehose が送信先に書き込むためのアクセス許可を削除します。他の送信先のアクセス許可を更新するには、「Amazon Data Firehose によるアクセスの制御」で送信先の セクションを確認してください。これら 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、CloudWatch Firehose の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場合、ストリームのソース (Kinesis Data Streams や Managed Service for Kafka など) が、ストリーム配信が再開され、データが送信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 の場合、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 ストリームの再開

配信を再開するには、まず送信先をオンにし、Firehose が送信先にストリームを配信するアクセス許可を持っていることを確認して、ストリームの送信先に以前に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、CloudWatch Firehose のメトリクスを使用してモニタリングし、ストリームが送信先に配信されていることを確認します。エラーを表示およびトラブルシューティングするには、Firehose の Amazon CloudWatch Logs モニタリングを使用します。

Amazon Data Firehose のモニタリング

Amazon Data Firehose は、次の機能を使用してモニタリングできます。

トピック

- CloudWatch アラームに関するベストプラクティス
- CloudWatch メトリクスを使用した Amazon Data Firehose のモニタリング
- Amazon Data Firehose の CloudWatch メトリクスへのアクセス
- CloudWatch ログを使用した Amazon Data Firehose のモニタリング
- Amazon Data Firehose の CloudWatch ログへのアクセス
- Kinesis エージェントの状態のモニタリング
- を使用した Amazon Data Firehose API コールのログ記録 AWS CloudTrail

CloudWatch アラームに関するベストプラクティス

次のメトリクスがバッファリング制限 (最大 15 分) を超えた場合の CloudWatch アラームを追加します。

- `DeliveryToS3.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、「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た Amazon Data Firehose のモニタリング

> **Important**
>
> エラーをタイムリーに特定するために、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てください。

Amazon Data Firehose は Amazon CloudWatch メトリクスと統合されているため、Firehose ストリームのメトリクスを収集、表示、分析 CloudWatch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 に取り込まれたデータを追跡できます。

Amazon Data Firehose は 1 分ごとに CloudWatch メトリクスを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 1 分間隔で Amazon Data Firehose から集計されるためです。

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のモニタリング」を参照してください。

トピック

- 動的パーティショニング CloudWatch メトリクス
- データ配信 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 ログの解凍メトリクス
- 変換 CloudWatch メトリクスのフォーマット
- サーバー側の暗号化 (SSE) CloudWatch メトリクス

- Amazon Data Firehose のディメンション
- Amazon Data Firehose 使用状況メトリクス

動的パーティショニング 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	Firehose ストリームがエラーバケットにデータを送信する前に処理するアクティブなパーティションの最大数。 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。 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。 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。 単位： StandardUnitBytesSecond
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。 単位: カウント

データ配信 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、レコードが設定された送信先またはバックアップ S3 バケットに正常に配信されるまで転送されません。

トピック

- OpenSearch サービスへの配信
- サーバーレスへの OpenSearch配信
- Amazon Redshift への配信
- Amazon S3 への配信
- Snowflake への配送
- 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。 単位: バイト
DeliveryToAmazonOpenSearchService.DataFreshness	Amazon Data Firehose で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 OpenSearch サービスに配信されました。 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の数。

メトリクス	説明
	単位: カウント
DeliveryToAmazonOpenSearchService.Success	インデックス作成が試みられたレコードの合計に対する正常にインデックス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。 単位: カウント
DeliveryToS3.DataFreshness	Amazon Data Firehose で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。 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。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。 単位: カウント
DeliveryToS3.Success	すべての Amazon S3 put コマンドの合計に対する正常に実行された Amazon S3 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みに対して有効になっているか、すべてのドキュメントに対して有効になっているかに関係なく、常にこのメトリクスを発行します。

メトリクス	説明
DeliveryToAmazonOpenSearchService.AuthFailure	認証/承認のエラー。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。 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。 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。

サーバーレスへの OpenSearch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。 単位: バイト
DeliveryToAmazonOpenSearchServerless.DataFreshness	Amazon Data Firehose で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)。この期間より古いレコードは Serverless OpenSearch に配信されました。 単位: 秒
DeliveryToAmazonOpenSearchServerless.Records	指定された期間に OpenSearch サーバーレスにインデックス作成されたレコードの数。 単位: カウント
DeliveryToAmazonOpenSearchServerless.Success	インデックス作成が試みられたレコードの合計に対する正常にインデックス作成されたレコードの合計。

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。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。 単位: カウント
DeliveryToS3.DataFreshness	Amazon Data Firehose で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。 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。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。 単位: カウント
DeliveryToS3.Success	すべての Amazon S3 put コマンドの合計に対する正常に実行された Amazon S3 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みに対して有効になっているか、すべてのドキュメントに対して有効になっているかに関係なく、常にこのメトリクスを発行します。
DeliveryToAmazonOpenSearchServerless.AuthFailure	認証/承認のエラー。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。 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。

メトリクス	説明
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。 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。 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。 単位: カウント
DeliveryToRedshift.Success	すべての Amazon Redshift COPY コマンドの合計に対する正常に実行された Amazon Redshift 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。 単位: バイト
DeliveryToS3.DataFreshness	Amazon Data Firehose で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。 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。

メトリクス	説明
	単位: カウント
DeliveryToS3.Success	すべての Amazon S3 put コマンドの合計に対する正常に実行された Amazon S3 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、Amazon S3 へのバックアップが有効になっている場合、このメトリクスを出力します。 単位: カウント
BackupToS3.DataFreshness	Amazon Data Firehose で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場合、このメトリクスを出力します。 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場合、このメトリクスを出力します。 単位: カウント
BackupToS3.Success	すべての Amazon S3 バックアップ put コマンドの合計に対する、成功したバックアップの Amazon S3 put コマンドの合計。Amazon Data Firehose は、Amazon S3 へのバックアップが有効になっている場合、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 ストリームの主な送信先である場合の Amazon S3 への配信に関連しています。

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。 単位: バイト
DeliveryToS3.DataFreshness	Amazon Data Firehose で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。 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。 単位: カウント
DeliveryToS3.Success	すべての Amazon S3 put コマンドの合計に対する正常に実行された Amazon S3 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場合 (データ変換も有効になっている場合にのみ可能) 、このメトリクスを出力します。 単位: カウント
BackupToS3.DataFreshness	Amazon Data Firehose で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場合

メトリクス	説明
	(データ変換も有効になっている場合にのみ可能)、このメトリクスを出力します。 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場合 (データ変換も有効になっている場合にのみ可能)、このメトリクスを出力します。 単位: カウント
BackupToS3.Success	すべての Amazon S3 バックアップ put コマンドの合計に対する、成功したバックアップの Amazon S3 put コマンドの合計。Amazon Data Firehose は、バックアップが有効になっている場合 (データ変換も有効になっている場合にのみ可能)、このメトリクスを出力します。

Snowflake への配送

メトリクス	説明
DeliveryToSnowflake.Bytes	指定した期間に Snowflake に配信されたバイト数。 単位: バイト
DeliveryToSnowflake.DataFreshness	Firehose で最も古いレコードの経過時間 (Firehose に入ってから現在まで)。この期間より古いレコードは Snowflake に配信されました。Firehose の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注意してください。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。

メトリクス	説明
	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。 単位: 秒
DeliveryToSnowflake.Records	指定した期間に Snowflake に配信されたレコードの数。 単位: カウント
DeliveryToSnowflake.Success	試行された挿入呼び出しの合計に対して Snowflake に対して行われた成功した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、Firehose が失敗したデータを S3 にバックアップしようとしたときにのみ使用できます。 単位: バイト
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、Firehose が失敗したデータを S3 にバックアップしようとしたときにのみ使用できます。 単位: カウント
DeliveryToS3.Success	すべての Amazon S3 put コマンドの合計に対する正常に実行された Amazon S3 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、Firehose が失敗したデータを S3 にバックアップしようとしたときにのみ使用できます。

メトリクス	説明
BackupToS3.DataFreshness	Firehose で最も古いレコードの経過時間 (Firehose から現在まで) 。この経過時間より古いレコードは 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、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使用できます。 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、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使用できます。 単位: カウント
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、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使用できます。 単位: カウント
BackupToS3.Success	すべての Amazon S3 バックアップ put コマンドの合計に対するバックアップの正常な Amazon S3 put コマンドの合計。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、Firehose は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。 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のにかかるおおよその時間。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。 単位: 秒
DeliveryToSplunk.DataFreshness	Amazon Data Firehose で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。 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。 単位: カウント
DeliveryToSplunk.Success	インデックス作成が試みられたレコードの合計に対する正常にインデックス作成されたレコードの合計。
DeliveryToS3.Success	すべての Amazon S3 put コマンドの合計に対する正常に実行された Amazon S3 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。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、Amazon Data Firehose はこのメトリクスを出力します。 単位: カウント

メトリクス	説明
BackupToS3.DataFreshness	Amazon Data Firehose で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。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、Amazon Data Firehose はこのメトリクスを出力します。 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。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、Amazon Data Firehose はこのメトリクスを出力します。 単位: カウント
BackupToS3.Success	すべての Amazon S3 バックアップ put コマンドの合計に対する、成功したバックアップの Amazon S3 put コマンドの合計。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、Amazon Data Firehose は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。 単位: バイト
DeliveryToHttpEndpoint.Records	HTTP エンドポイントに正常に配信されたレコード数。 単位: カウント

メトリクス	説明
DeliveryToHttpEndpoint.DataFreshness	Amazon Data Firehose で最も古いレコードの経過時間。 単位: 秒
DeliveryToHttpEndpoint.Success	HTTP エンドポイントへの成功したデータ配信リクエストの合計 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。 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

メトリクス	説明
	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。 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。 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。 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。 単位: カウント

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 Data Firehose は、Amazon S3 または Amazon Redshift の送信先でデータ変換が有効になっている場合、このメトリクスを出力します。 単位: バイト
BackupToS3.DataFreshness	Amazon Data Firehose で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

メトリクス	説明
	に Amazon S3 バケットに配信済みです。Amazon Data Firehose は、Amazon S3 または Amazon Redshift の送信先でデータ変換が有効になっている場合、このメトリクスを出力します。 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 または Amazon Redshift の送信先でデータ変換が有効になっている場合、このメトリクスを出力します。 単位: カウント
BackupToS3.Success	すべての Amazon S3 バックアップ put コマンドの合計に対する、成功したバックアップの Amazon S3 put コマンドの合計。Amazon Data Firehose は、Amazon S3 または Amazon Redshift の送信先でデータ変換が有効になっている場合、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する前に取り込むことができる 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。
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。 単位: バイト

メトリクス	説明
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。 単位: カウント
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。 単位: バイト
DeliveryToAmazonOpenSearchService.DataFreshness	Amazon Data Firehose で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 OpenSearch Service に配信されました。 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の数。 単位: カウント
DeliveryToAmazonOpenSearchService.Success	インデックス作成が試みられたレコードの合計に対する正常にインデックス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。 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。 単位: カウント

メトリクス	説明
DeliveryToRedshift .Success	すべての Amazon Redshift COPY コマンドの合計に対する正常に実行された Amazon Redshift 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。 単位: バイト
DeliveryToS3.DataF reshness	Amazon Data Firehose で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。 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。 単位: カウント
DeliveryToS3.Success	すべての Amazon S3 put コマンドの合計に対する正常に実行された Amazon S3 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。 単位: バイト
DeliveryToSplunk.D ataAckLatency	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のにかかるおおよその時間。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。 単位: 秒

メトリクス	説明
DeliveryToSplunk.DataFreshness	Amazon Data Firehose で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。 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。 単位: カウント
DeliveryToSplunk.Success	インデックス作成が試みられたレコードの合計に対する正常にインデックス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 ストリームの制限の 1 つ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 にはカウントされませんIncomingBytes 。 単位: バイト
IncomingPutRequests	指定した期間に成功した PutRecord および PutRecord Batch リクエストの数。 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の数。データインジェストは、Firehose ストリームの制限の 1 つ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 にはカウントされませんIncomingRecords 。 単位: カウント

メトリクス	説明
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。 単位: ミリ秒
RecordsPerSecondLimit	Firehose ストリームがスロットリング前に取り込むことができる 1 秒あたりの現在の最大レコード数。 単位: カウント
ThrottledRecords	データ取り込みが Firehose ストリーム制限の 1 つを超えたためにスロットリングされたレコードの数。 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。 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。 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。 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。

メトリクス	説明
	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。 単位: カウント
FailedValidation.Records	レコード検証に失敗したレコード数。 単位: カウント
FailedValidation.Bytes	レコード検証に失敗したバイト数。 単位: バイト
DataReadFromSource .Backpressured	パーティション BytesPerSecondLimit ごとの配信フローが超過しているか、通常の配信フローが遅いか停止しているため、Firehose ストリームがソースパーティションからのレコードの読み取りに遅延していることを示します 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
DescribeDeliverySt ream.Latency	指定された期間に測定された DescribeDeliverySt ream オペレーションごとにかかった時間。 単位: ミリ秒
DescribeDeliverySt ream.Requests	DescribeDeliveryStream リクエストの総数。 単位: カウント

メトリクス	説明
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。 単位: ミリ秒
ListDeliveryStreams.Requests	ListFirehose リクエストの総数。 単位: カウント
PutRecord.Bytes	指定した期間に を使用して Firehose ストリームPutRecord に送信されたバイト数。 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。 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。 単位: カウント
PutRecordBatch.Bytes	指定した期間に を使用して Firehose ストリームPutRecordBatch に送信されたバイト数。 単位: バイト
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。 単位: ミリ秒
PutRecordBatch.Records	PutRecordBatch オペレーションのレコード総数。 単位: カウント

メトリクス	説明
PutRecordBatch.Requests	PutRecordBatch リクエストの総数。 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 put リクエストの最大数。この番号には、PutRecord および PutRecordBatch リクエストが含まれます。 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。 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。 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。 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。 単位: ミリ秒
UpdateDeliveryStream.Requests	UpdateDeliveryStream リクエストの総数。 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。 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。 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。 単位: バイト

CloudWatch ログの解凍メトリクス

CloudWatch ログ配信で解凍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
OutputDecompressed Bytes.Success	バイト単位の正常な解凍データ 単位: バイト
OutputDecompressed Bytes.Failed	バイト単位の解凍に失敗したデータ 単位: バイト

メトリクス	説明
OutputDecompressedRecords.Success	解凍に成功したレコードの数 単位: カウント
OutputDecompressedRecords.Failed	解凍に失敗したレコードの数 単位: カウント

変換 CloudWatch メトリクスのフォーマット

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。 単位: カウント
SucceedConversion.Bytes	正常に変換されたレコードのサイズ。 単位: バイト
FailedConversion.Records	変換できなかったレコードの数。 単位: カウント
FailedConversion.Bytes	変換できなかったレコードのサイズ。 単位: バイト

サーバー側の暗号化 (SSE) CloudWatch 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	Firehose ストリームKMSAccessDeniedException の にサービスが遭遇した回数。 単位: カウント
KMSKeyDisabled	Firehose ストリームKMSDisabledException でサービスが に遭遇した回数。 単位: カウント
KMSKeyInvalidState	Firehose ストリームKMSInvalidStateException でサービスが に遭遇した回数。 単位: カウント
KMSKeyNotFound	Firehose ストリームKMSNotFoundException の にサービスが遭遇した回数。 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リングするには、 DeliveryStreamNameディメンションを使用します。

Amazon Data Firehose 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使用状況を可視化できます。これらのメトリクスを使用して、 CloudWatch グラフとダッシュボードで現在のサービス使用状況を視覚化します。

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、1 分ごとに収集されます。

現在、この名前空間で CloudWatch 公開されるメトリクス名は のみですResourceCount。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。 このメトリクスで最も役に立つ統計は MAXIMUM です。これは、1 分間の期間中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 によって公開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 ですFirehose。
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は、このディメンションを の値で使用しますNone。
Type	追跡されるリソースのタイプ。現在、Service ディメンションが 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 メトリクスへのアクセス

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. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションバーで、リージョンを選択します。
3. ナビゲーションペインで [メトリクス]を選択します。
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 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 と 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 ログを使用した Amazon Data Firehose のモニタリング

Amazon Data Firehose は Amazon CloudWatch Logs と統合されているため、データ変換またはデータ配信の Lambda 呼び出しが失敗したときに特定のエラーログを表示できます。Firehose ストリームを作成するときに、Amazon Data Firehose エラーログ記録を有効にできます。

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、ユーザーに代わって Firehose ストリームのロググループと対応するログストリームが作成されます。ロググループ名の形式は です。ここで/aws/kinesisfirehose/*delivery-stream-name*、 *delivery-stream-name*は対応する Firehose ストリームの名前です。 は、プライマリ送信先への配信に関連するエラーをログに記録するために作成および使用されるログストリームDestinationDelivery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、MyStreamAmazon Redshift を送信先として Firehose ストリーム「」を作成し、Amazon Data Firehose エラーログ記録を有効にすると、 という名前のロググループ`aws/kinesisfirehose/MyStream`と `DestinationDelivery`および という名前の 2 つのログストリームがユーザーに代わって作成されますBackupDelivery。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 AWS 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。Amazon Data Firehose エラーログ記録専用のロググループとログストリームを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 は、すべての配信エラーログが CloudWatch Logs に送信されることを保証するものではないことに注意してください。配信失敗率が高い状況では、Amazon Data Firehose は配信エラーログを CloudWatch ログに送信する前にサンプリングします。

CloudWatch ログに送信されるエラーログには少額の料金がかかります。詳細については、「Amazon CloudWatch の料金」を参照してください。

内容

- データ配信エラー

データ配信エラー

以下は、Amazon Data Firehose の送信先ごとのデータ配信エラーコードと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

- Amazon S3 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 データ配信エラー

- HTTPS エンドポイントデータ配信エラー
- Amazon OpenSearch サービスデータ配信エラー
- 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-relatedエラーを CloudWatch ログに送信できます。

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。" 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。提供された IAM ロールの信頼ポリシーで Amazon Data Firehose がロールを引き受けることが許可され、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ていることを確認します。」
`S3.AccountProblem`	AWS 「アカウントに問題があるため、オペレーションが正常に完了しません。 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。」

エラーコード	エラーメッセージおよび情報
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.Method NotAllowed	「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。」
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ます。」

エラーコード	エラーメッセージおよび情報
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。KMS MasterKeyID を設定するか、S3 バケットの暗号化を無効にします。」
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するリージョンで 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。」

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。」 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 Data Firehose が存在しない場合、Amazon Redshift テーブルは作成され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。提供された IAM ロールの信頼ポリシーで、Amazon Data Firehose がロールを引き受けることが許可されてい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。」
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。」

エラーコード	エラーメッセージおよび情報
	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。セキュリティ設定で Amazon Data Firehose 接続が許可されていること、Amazon Redshift 送信先設定または 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いこと、およびクラスターが使用可能であることを確認します。」
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。」
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。」

エラーコード	エラーメッセージおよび情報
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 送信先設定 DataTableName の 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。」
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。」

エラーコード	エラーメッセージおよび情報
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。」
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。」

エラーコード	エラーメッセージおよび情報
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。」
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。」

エラーコード	エラーメッセージおよび情報
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 からQIException 観測された PS」
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。」

エラーコード	エラーメッセージおよび情報
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、以下の Snowflake 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
Snowflake.InvalidUrl	「Firehose は Snowflake に接続できません。Snowflake の送信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 の送信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、承認されていません。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、承認されていません」
Snowflake.InvalidSchema	「指定されたスキーマが存在しないか、承認されていません」
Snowflake.InvalidDatabase	「指定されたデータベースが存在しないか、承認されていません」
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。提供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注意してください。」

エラーコード	エラーメッセージおよび情報
`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。null 以外のすべての列に値が指定されていることを確認してください」
`Snowflake.ExtraColumns`	「追加の列のため、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 JSON 形式で許容でき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 値が Snowflake テーブル定義で宣言されたデータ型に従っ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、以下の Splunk 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
`Splunk.ProxyWithoutStickySessions`	「Amazon Data Firehose と HEC ノードの間にプロキシ (ELB またはその他の) がある場合は、スティッキーセッションを有効にして HEC ACKs."」
`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。」
`Splunk.InvalidToken`	"The HEC token is invalid。Amazon Data Firehose を有効な HEC トークンで更新します。」
`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。」

エラーコード	エラーメッセージおよび情報
`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 の再試行時間が 0 より大きい場合、Amazon Data Firehose はデータの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、確認タイムアウトの有効期限が切れた Amazon S3 データを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.InvalidEndpoint`	"Could not connect to the HEC endpoint。HEC 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。」
`Splunk.ConnectionClosed`	"Unable to send data to Splunk due to a connection failure。This might be a transient error。Amazon Data Firehose 設定の再試行期間を長くすると、このような一時的な障害を防ぐことができます。」

エラーコード	エラーメッセージおよび情報
`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 データ配信エラー

Amazon Data Firehose は、次の ElasticSearch エラーを CloudWatch ログに送信できます。

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 AWS Elasticsearch 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 ログに送信できます。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.DestinationException	「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
HttpEndpoint.ConnectionFailed	「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。」
HttpEndpoint.ConnectionReset	「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。」
HttpEndpoint.ConnectionReset	「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。」
HttpEndpoint.ResponseReasonPhraseExceededLimit	「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。」
HttpEndpoint.InvalidResponseFromDestination	「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:「
HttpEndpoint.DestinationException	「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。」
HttpEndpoint.InvalidStatusCode	「無効なレスポンスステータスコードを受信しました。」
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。」

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.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	HttpEndpoint 「URI の入力が無効であるため、リクエストに失敗しました。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。」
HttpEndpoint.MakeRequestFailure.IllegalCharacterInHeaderValue	HttpEndpoint 「不正なレスポンスエラーのためリクエストに失敗しました。ヘッダー値に不正な文字 '\n' が含まれています。」

エラーコード	エラーメッセージおよび情報
`HttpEndpoint.IllegalResponseFailure`	HttpEndpoint 「不正なレスポンスエラーのためリクエストに失敗しました。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`	HttpEndpoint 「不正なレスポンスエラーのためリクエストに失敗しました。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 サービスデータ配信エラー

OpenSearch サービスの送信先の場合、Amazon Data Firehose は OpenSearch 、サービスによって返されるときにエラーを CloudWatch ログに送信します。

OpenSearch クラスターから返される可能性のあるエラーに加えて、次の 2 つのエラーが発生することがあります。

- 認証/承認エラーは、送信先 OpenSearch サービスクラスターにデータを配信しようとしたときに発生します。これは、アクセス許可の問題や、Amazon Data Firehose ターゲット OpenSearch サービスドメインの設定が変更されたときに断続的に発生する可能性があります。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/承認の失敗により、データを宛先 OpenSearch サービスクラスターに配信できませんでした。これは、アクセス許可の問題や、Amazon Data Firehose ターゲット OpenSearch サービスドメインの設定が変更されたときに断続的に発生する可能性があります。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。提供された IAM ロールの信頼ポリシーで Firehose がロールを引き受けることが許可され、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ます。」

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。提供された IAM ロールの信頼ポリシーで Firehose がロールを引き受けることが許可され、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ます。」
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。提供された IAM ロールの信頼ポリシーで Firehose がロールを引き受けることが許可され、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ます。」
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しました。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 クラスターが未指定エラーを返しました」
OS.RequestTimeout	「Amazon OpenSearch Service クラスターへのリクエストがタイムアウトしました。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。」

エラーコード	エラーメッセージおよび情報
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ができません。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 コレクションと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 コレクションとの接続を維持するのに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 コレクションとの接続を維持するのに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が、設定された IAM ロールへのアクセスを許可していることを確認します。」
OS.ValidationException	OpenSearch 「クラスターは ES を返しましたServiceException。その理由の 1 つは、クラスターが OS 2.x 以降にアップグレードされたが、それでも TypeName パラメータが設定されていることです。を TypeName 空の文字列に設定するか、 エンドポイントを タイプパラメータをサポートするクラスターに変更して、Hose 設定を更新します。」
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 を返しました JsonParseException。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。」

エラーコード	エラーメッセージおよび情報
OS.AmazonOpenSearchServiceParseException	「Amazon OpenSearch Service クラスターが を返しました AmazonOpenSearchServiceParseException。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ます。」
OS.ClusterError	「Amazon OpenSearch Service クラスターまたは OpenSearch Serverless コレクションが未指定エラーを返しました」
OS.ClusterBlockException	「クラスターは を返しました ClusterBlockException。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 サービス 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。」
OS.UnsupportedVersion	「Amazon OpenSearch Service 6.0 は現在、Amazon Data Firehose では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。」

エラーコード	エラーメッセージおよび情報
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	VPCs 内の Amazon OpenSearch サービス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閉じました。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、許可されたよりも大きいフィールドデータが含まれているため、リクエストを中止しました。」
OS.BadGateway	「Amazon OpenSearch Service クラスターまたは OpenSearch Serverless コレクションは、レスポンス: 502 Bad Gateway でリクエストを中止しました」
OS.ServiceException	「Amazon OpenSearch Service クラスターまたは OpenSearch サーバーレスコレクションから受信したエラー。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	「Amazon OpenSearch Service クラスターまたは OpenSearch Serverless コレクションに接続するときに、Firehose でタイムアウトエラーが発生しました。」
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」

エラーコード	エラーメッセージおよび情報
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 ログに送信できます。

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。提供された IAM ロールの信頼ポリシーで、Amazon Data Firehose がロールを引き受けることが許可されてい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 に必要なステータスモデルに従っていることを確認します。」 詳細については、「データ変換とステータスモデル」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。」 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。
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。」

エラーコード	エラーメッセージおよび情報
	詳細については、「データ変換とステータスモデル」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。」 詳細については、「データ変換とステータスモデル」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 「1 つ以上の設定済み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。」 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。
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。」 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。

エラーコード	エラーメッセージおよび情報
`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。 • 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。 • 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。 • 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タイムアウト設定エラーが発生する前にリクエストが完了しませんでした。Lambda コードを再確認して、Lambda コードが設定されたタイムアウトを超えて実行されるかどうかを確認します。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 でエラーが発生しました。 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。
`Lambda.InvalidZipFileException`	Lambda 関数の呼び出し中に Amazon Data Firehose エラーが発生しました InvalidZipFileException。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。

エラーコード	エラーメッセージおよび情報
`Lambda.InternalServerError`	「Lambda サービスを呼び出すときに Amazon Data Firehose AWS エラーが発生しました InternalServerError。Amazon Data Firehose は、一定の回数だけデータの送信を再試行します。`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 Lambda AWS サービスを呼び出す ServiceUnavailableException ときに発生しました。Amazon Data Firehose は、一定の回数だけデータの送信を再試行します。`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。
`Lambda.InvocationFailure`	これは次のいずれかのエラーが原因で発生します。 • 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。」 • Amazon Data Firehose で Lambda InvalidStateException からの KMS 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。 • Amazon Data Firehose が Lambda AWS LambdaException から を検出しました。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。 • 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。

エラーコード	エラーメッセージおよび情報
`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ります。Firehose が DirectPut タイプの場合、Firehose はもう存在し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 AWS Kinesis サブスクリプションがあることを確認してください。」
`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 APIs を呼び出すこと、またはソースと同じ Kinesis ストリームを使用して作成した Firehose ストリームが多すぎる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 APIs を呼び出すこと、またはソースと同じ Kinesis ストリームを使用して作成した Firehose ストリームが多すぎることが原因である可能性があります。」

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.AccessDenied	「基盤となる Kinesis Streams で API オペレーションを呼び出そうとして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ます。」
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。」
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 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.Acces	「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。」

エラーコード	エラーメッセージおよび情報
sDeniedEx ception	
Firehose. KMS.Inval idStateEx ception	「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.NotFo undException	「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.KeyDi sabled	「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、次の AWS Glue 呼び出しエラーを CloudWatch ログに送信できます。

エラーコード	エラーメッセージおよび情報
DataForma tConversi on.Invali dSchema	「スキーマが無効です。」
DataForma tConversi on.Entity NotFound	「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。」
DataForma tConversi on.Invali dInput	「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。」

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。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。」
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。」

エラーコード	エラーメッセージおよび情報
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。」
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」アクセス許可を追加するか (推奨) 、未使用のテーブルバージョンを削除してください。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」アクセス許可を追加するか (推奨) 、未使用のテーブルバージョンを削除してください。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.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 呼び出しエラーを CloudWatch ログに送信できます。

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.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。」
DataFormatConversi	「JSON キーの長さは 262144 を超えることはできません。」

エラーコード	エラーメッセージおよび情報
on.Malfor medData	
DataForma tConversi on.Malfor medData	「データを UTF-8 としてデコードすることはできません。」
DataForma tConversi on.Malfor medData	「トークンの間に不正な文字が見つかりました。」
DataForma tConversi on.Invali dTypeFormat	「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。」
DataForma tConversi on.Invali dSchema	「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。」
DataForma tConversi on.Invali dRecord	「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。」
DataForma tConversi on.Malfor medData	「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。」

エラーコード	エラーメッセージおよび情報
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。」
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。」

エラーコード	エラーメッセージおよび情報
DataFormatConversion.ConversionFailureException	"ConversionFailureException"
DataFormatConversion.DataFormatConversionCustomerErrorException	"DataFormatConversionCustomerErrorException"
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。」

エラーコード	エラーメッセージおよび情報
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。」
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.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。」

エラーコード	エラーメッセージおよび情報
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」アクセス許可を追加するか (推奨)、未使用のテーブルバージョンを削除してください。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」アクセス許可を追加するか (推奨)、未使用のテーブルバージョンを削除してください。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.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 ログへのアクセス

Amazon Data Firehose コンソールまたは コンソールを使用して、Amazon Data Firehose データ配信の失敗に関連するエラーログを表示できます CloudWatch 。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 Management Console 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 ストリーム名を選択して、Firehose ストリームの詳細ページに移動します。
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. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。

2. ナビゲーションバーで、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のモニタリング

Kinesis Agent は、 の名前空間でカスタム CloudWatch メトリクスを発行しますAWS KinesisAgent。これは、エージェントが正常かどうかを評価し、指定されたとおりに Amazon Data Firehose にデータを送信し、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するのに役立ちます。

レコード数や送信された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理解するのに役立ちます。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、「エージェントの設定」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 の CloudWatch 料金」を参照してください。

によるモニタリング CloudWatch

Kinesis Agent は、次のメトリクスを に送信します CloudWatch。

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。 単位: バイト

メトリクス	説明
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)。 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。 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。 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 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。 は、Amazon Data Firehose のすべての API コールをイベントとして CloudTrail キャプチャします。キャプチャされた呼び出しには、Amazon Data Firehose コンソールからの呼び出しと、Amazon Data Firehose API オペレーションへのコード呼び出しが含まれます。証跡を作成する場合は、Amazon Data Firehose の CloudTrail イベントなど、Amazon S3 バケットへのイベントの継続的な配信を有効にすることができます。 Amazon S3 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。で収集された情報を使用して CloudTrail、Amazon Data Firehose に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

の設定と有効化の方法など CloudTrail、 の詳細については、AWS CloudTrail 「 ユーザーガイド」を参照してください。

の Amazon Data Firehose 情報 CloudTrail

CloudTrail AWS アカウントを作成すると、 がアカウントで有効になります。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

CloudTrailの他の AWS サービスイベントとともにイベントに記録されます。 AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「イベント履歴を使用したCloudTrailイベントの表示」を参照してください。

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、それに基づいて行動するように他の 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 サポートされているサービスと統合
- の Amazon SNS 通知の設定 CloudTrail
- 複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからの CloudTrail ログファイルの受信

Amazon Data Firehose では、以下のアクションをイベントとして CloudTrail ログファイルに記録できます。

- 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 root または AWS Identity and Access Management (IAM) ユーザーの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、CloudTrailuserIdentity Element」を参照してください。

例: Amazon Data Firehose ログファイルエントリ

証跡は、指定した Amazon S3 バケットにイベントをログファイルとして配信できるようにする設定です。 CloudTrail ログファイルには 1 つ以上のログエントリが含まれます。イベントは任意のソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。 CloudTrail ログファイルはパブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次の例
は、、CreateDeliveryStream、、DescribeDeliveryStream、ListDeliveryStreamsUpdateDesよび DeleteDeliveryStreamアクションを示す CloudTrail ログエントリを示しています。


```
{
  "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:::firehose-cloudtrail-test-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:::firehose-cloudtrail-test-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:::firehose-cloudtrail-test-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"
    }
  ]
}
```

Amazon S3 オブジェクトのカスタムプレフィックス

Amazon S3 に配信されるオブジェクトは、 < 評価済みプレフィックス><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、 のデフォルトのプレフィックスを上書きしますYYYY/MM/dd/HH。

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 S3 オブジェクト名形式」を参照してください。 FirehoseDeveloper

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際、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着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。例えば、UTC の代わりに日本標準時を使用する場合は、 AWS Management Console 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 オブジェクト名形式」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	メモ
`error-output-type`	Firehose ストリームの設定と失敗の理由に応じて、次のいずれかの文字列に評価されます: {processing-failed, AmazonOpenSearchService-failed, splunk-failed, format-conversion-failed，http-endpoint-failed}。 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 フィールドでのみ使用できます ErrorOutputPrefix。
`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字列として評価されます。	`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。 形式の文字列の先頭にこれを配置すると、ランダム化されたプレフィックスを

変換	説明	入力例	出力例	メモ
				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、「Amazon Firehose ストリームの作成」の「送信先に Amazon S3 を選択する」を参照してください。 https://docs.aws.amazon.com/firehose/latest/dev/basic-create.html

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を含まないプレフィックスを指定すると、Firehose は `Prefix` フィールドの値`!{timestamp:yyyy/MM/dd/HH/}`に式を追加します。
- 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。
- 送信先が Amazon OpenSearch Service または Splunk で、 が指定されていない場合、Firehose ErrorOutputPrefix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 Management Console 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。プレフィックスを指定してもエラープレフィックスを指定しない場合、Firehose は自動的にエラープレフィックスを に設定します!{firehose:error-output-type}/。
- を AWS CLI、API、または ExtendedS3DestinationConfigurationで使用する場合 AWS CloudFormation、 を指定してもS3BackupConfiguration、Firehose はデフォルトの を提供しませんErrorOutputPrefix。
- ErrorOutputPrefix 式の作成時に 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

入力	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

入力	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}/	ErrorOutputPrefix : myFirehoseFailures/processing-failed/
Prefix: !{timestamp:yyyy/MM/dd} 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} ErrorOutputPrefix : myFirehoseFailures/!{firehose:error-output-type}/!{timestamp:yyyy}/anyMonth/!{timestamp:dd}	Prefix: myFirehose/DeliveredYear=2018/anyMonth/rand=5abf82daaa5 ErrorOutputPrefix : myFirehoseFailures/processing-failed/2018/anyMonth/10
Prefix: myPrefix/year=!{timestamp:yyyy}/month=!{timestamp:MM}/day=!{timestamp:dd}/hour=!{timestamp:HH}/ 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/ ErrorOutputPrefix : myErrorPrefix/year=2018/month=07/day=06/hour=23/processing-failed
Prefix: myFirehosePrefix/ ErrorOutputPrefix : 未指定	Prefix: myFirehosePrefix/2018/08/27/ ErrorOutputPrefix : myFirehosePrefix/processing-failed/2018/08/27/

での Amazon Data Firehose の使用 AWS PrivateLink

Amazon Data Firehose のインターフェイス VPC エンドポイント (AWS PrivateLink)

インターフェイス VPC エンドポイントを使用して、Amazon VPC と Amazon Data Firehose 間のトラフィックが Amazon ネットワークから離れないようにすることができます。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 AWS 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 AWS PrivateLink、Amazon VPC 内のプライベート IP を備えた 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 を利用しています。 IPs 詳細については、「Amazon Virtual Private Cloud」を参照してください。

Amazon Data 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 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始めるために、インターフェイス VPC エンドポイントを作成します。エンドポイントを作成するときに、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アタッチできます。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の詳細については、「VPC エンドポイントによる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 関数が現在のリージョンの Firehose ストリームを一覧表示するが、Firehose ストリームを記述しないことを許可する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 Management Console 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 (次へ: アクセス許可)] 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、Runtime 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. 関数を作成 を選択します。
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。成功すると、リージョン内の Firehose ストリームのリストと、次の出力が表示されます。

```
Calling describe_delivery_stream.

AccessDeniedException
```

```
Access denied test succeeded.
```

可用性

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 (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)
- 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)

Amazon Data Firehose での Firehose ストリームのタグ付け

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

トピック

- タグの基本
- タグ付けを使用したコストの追跡
- タグの制限
- Amazon Data Firehose API を使用した Firehose ストリームのタグ付け

タグの基本

Amazon Data Firehose API を使用して、次のタスクを完了できます。

- 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します。
- Firehose ストリームからタグを削除します。

タグを使用して Firehose ストリームを分類できます。例えば、Firehose ストリームを目的、所有者、環境別に分類できます。タグごとにキーと値を定義するため、特定のニーズを満たすためのカテゴリのカスタムセットを作成できます。例えば、所有者および関連するアプリケーションごとに Firehose ストリームを追跡するのに役立つ一連のタグを定義できます。

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証を実行して、ユーザーがタグを作成するアクセス許可を持っているかどうかを確認します。このアクセス許可を指定しない場合、IAM リソース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、AccessDeniedException次のような 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、タグを適用できるようにするポリシーを示しています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "firehose:CreateDeliveryStream",
            "Resource": "*",
            }
        },
        {
            "Effect": "Allow",
            "Action": "firehose:TagDeliveryStream",
            "Resource": "*",
            }
        }
    ]
}
```

タグ付けを使用したコストの追跡

タグを使用して、 AWS コストを分類および追跡できます。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

Amazon Data Firehose のタグには、次の制限が適用されます。

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

タグ値の制限

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

Amazon Data Firehose API を使用した Firehose ストリームのタグ付け

を呼び出して新しい Firehose ストリームCreateDeliveryStreamを作成するときに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、一覧表示、削除できます。

- TagDeliveryStream

- ListTagsForDeliveryStream
- UntagDeliveryStream

チュートリアル: Amazon Data Firehose を使用して VPC フローログを Splunk に取り込む

チュートリアルについては、「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。

Amazon Data Firehose のトラブルシューティング

Firehose は、データの配信または処理中にエラーが発生した場合、設定された再試行期間が終了するまで再試行します。データの配信が成功す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。送信先が Amazon S3 で、配信が失敗した場合、またはバックアップ S3 バケットへの配信が失敗した場合、Firehose は保持期間が終了するまで再試行を続けます。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。

データソースが Kinesis データストリームの場合、Firehose は次のオペレーションを無期限に再試行します: DescribeStream、GetRecords、GetShardIterator。

Firehose ストリームが を使用している場合はDirectPut、 IncomingBytesおよび IncomingRecordsメトリクスをチェックして、受信トラフィックがあるかどうかを確認します。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、コードがレスポンスFailedPutCountの の値をチェックしていることを確認してください。

Firehose ストリームがソースとして Kinesis データストリームを使用している場合は、ソースデータストリームの IncomingBytesおよび IncomingRecordsメトリクスを確認します。さらに、Firehose ストリームに対して DataReadFromKinesisStream.Bytesおよび DataReadFromKinesisStream.Recordsメトリクスが出力されていることを確認します。

を使用した配信エラーの追跡については CloudWatch、「」を参照してくださいthe section called “CloudWatch ログによるモニタリング”。

一般的な問題

一般的な問題と ypu による解決方法をいくつか紹介します。

- Firehose ストリームは、 CloudWatch ログ、 CloudWatch イベント、または AWS IoT アクションのターゲットとして使用できません – 一部の AWS 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ます 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。
- 適切なメトリクスにもかかわらず送信先にデータがない – データインジェストの問題がなく、Firehose ストリームに対して出力されたメトリクスが良好に見えるが、送信先にデータが表

示されない場合は、リーダーロジックを確認してください。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。

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 `IncomingBytes`と `IncomingRecords`メトリクスをチェックして、データが Firehose ストリームに正常に送信され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た Amazon Data Firehose のモニタリング」を参照してください。
- Lambda によるデータ変換が有効になっている場合は、Firehose `ExecuteProcessing.Success`メトリクスをチェックして、Firehose が Lambda 関数の呼び出しを試み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た Amazon Data Firehose のモニタリング」を参照してください。
- Firehose `DeliveryToS3.Success`メトリクスをチェックして、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た Amazon Data Firehose のモニタリング」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 ログを使用した Amazon Data Firehose のモニタリング」を参照してください。
- ログに「Amazon S3 サービスを呼び出す InternalServerError ときにFirehose が検出された」というエラーメッセージが表示された場合。 Amazon S3 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、S3 にお問い合わせください。「」、S3 の 1 つのパーティションでのリクエストレートの大幅な増加が原因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 ストリームで指定された 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になっている場合は、Firehose ストリームで指定された Lambda 関数がまだ存在することを確認してください。
- Firehose ストリームで指定された IAM ロールが S3 バケットと Lambda 関数にアクセスできることを確認します (データ変換が有効になっている場合) 。また、IAM ロールが CloudWatch ロググループとログストリームにアクセスしてエラーログをチェックしていることを確認します。詳細に

ついては、「Amazon Data Firehose に Amazon S3 送信先へのアクセス権を付与する」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 データ FirehoseData 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 `DeliveryToRedshift.Success`メトリクスをチェックして、Firehose が S3 バケットから Amazon Redshift プロビジョニングクラスターまたは 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た Amazon Data Firehose のモニタリング」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 ログを使用した Amazon Data Firehose のモニタリング」を参照してください。
- Amazon Redshift `STL_CONNECTION_LOG`テーブルで、Firehose が接続を正常に実行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 ストリームで指定された IAM ロールが、Amazon Redshift がデータをコピーする S3 バケットと、データ変換用の Lambda 関数にアクセスできることを確認します (データ変換が有効になっている場合) 。また、IAM ロールが CloudWatch ロググループとログストリームにアクセスしてエラーログをチェックしていることを確認します。詳細については、「Amazon Redshift 送信先へのアクセス権を Amazon Data Firehose に付与する 」を参照してください。

- Amazon Redshift でプロビジョニングされたクラスターまたは Amazon Redshift Serverless ワークグループが Virtual Private Cloud (VPC) にある場合は、クラスターが Firehose IP アドレスからのアクセスを許可していることを確認してください。詳細については、「Amazon Redshift 送信先へのアクセス権を Amazon Data 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 データ FirehoseData 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

データが OpenSearch サービスドメインに配信されない場合は、以下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 `IncomingBytes`と `IncomingRecords`メトリクスをチェックして、データが Firehose ストリームに正常に送信され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た Amazon Data Firehose のモニタリング」を参照してください。
- Lambda によるデータ変換が有効になっている場合は、Firehose `ExecuteProcessing.Success`メトリクスをチェックして、Firehose が Lambda 関数の呼び出しを試み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た Amazon Data Firehose のモニタリング」を参照してください。
- Firehose `DeliveryToAmazonOpenSearchService.Success`メトリクスをチェックして、Firehose が OpenSearch サービス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た Amazon Data Firehose のモニタリング」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 ログを使用した Amazon Data Firehose のモニタリング」を参照してください。
- Firehose ストリームのサービス OpenSearch 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になっている場合は、Firehose ストリームで指定された Lambda 関数がまだ存在することを確認してください。また、IAM ロールが CloudWatch ロググループと

ログストリームにアクセスしてエラーログをチェックしてい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 ストリームで指定された IAM ロールが、 OpenSearch サービスクラスター、S3 バックアップバケット、および Lambda 関数にアクセスできることを確認します (データ変換が有効になっている場合) 。また、IAM ロールが CloudWatch ロググループとログストリームにアクセスしてエラーログをチェックしてい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。
- 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 データ FirehoseData 変換」を参照してください。
- Amazon Data Firehose は、現在、Amazon OpenSearch Service の送信先への CloudWatch ログの配信をサポートしていません。Amazon CloudWatch は複数のログイベントを 1 つの Firehose レコードに結合し、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。別の方法として、 CloudWatch ログで 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を検討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 内にある場合は、Firehose がそれにアクセスできることを確認してください。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 で Cookie の有効期限が無効になっている期間ベースのスティッキーセッションを有効にし、有効期限は Application Load Balancer の最大 (7 日間) に設定されます。これを行う方法については、「Classic Load Balancer または Application Load Balancer の所要時間ベースのセッション維持」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 またはその他の) 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 (ACKs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。Enable and disable Event Collector tokens を参照してください。
- 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。詳細については、「Enable indexer acknowledgement」を参照してください。
- Firehose ストリームの `HECAcknowledgmentTimeoutInSeconds` Splunk 送信先設定で の値を増やします。
- Firehose ストリームの Splunk 送信先設定`DurationInSecondsRetryOptions`で、 の下の の値を増やします。
- HEC のヘルスを確認します。
- 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 データ FirehoseData 変換」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た IAM ロールが、データ変換 (データ変換が有効になっている場合) のために S3 バックアップバケットと Lambda 関数にアクセスできることを確認します。また、IAM ロールが CloudWatch ロググループとログストリームにアクセスしてエラーログをチェックしていることを確認します。詳細については、「Splunk 送信先 FirehoseAccess への付与」を参照してください。
- 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 を送信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手順について説明します。

Firehose ストリームの作成が失敗する

PrivateLinkが有効な Snowflake クラスターにデータを配信するストリームに対して Firehose ストリームの作成が失敗した場合、VPCE-ID に Firehose が到達できないことを示します。これは、次のいずれかの理由が原因である可能性があります。

- 正しくない VPCE-ID。タイプミスがないことを確認します。
- Firehose は、プレビューでリージョンレス Snowflake URLsをサポートしていません。Snowflake Account Locator を使用して URL を指定します。詳細については、Snowflake の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

Snowflake テーブル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。Snowflake の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。以下は、一般的なエラーシナリオです。エラーコードのリスト全体については、「」を参照してくださいSnowflake データ配信エラー。

- 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、デフォルトのロールを指定された Snowflake ユーザーに設定してください。
- エラーコード: Snowflake.ExtraColumns: 入力ペイロードの余分な列が原因で Snowflake への挿入が拒否されたことを示します。テーブルに存在しない列は指定しないでください。Snowflake の列名では大文字と小文字が区別されることに注意してください。テーブルに列が存在するにもかかわらず配信がこのエラーで失敗する場合は、入力ペイロードの列名の大文字と小文字が、テーブル定義で宣言された列名と一致することを確認してください。
- エラーコード: Snowflake.MissingColumns: 入力ペイロードに列がないため、Snowflake への挿入が拒否されたことを示します。NULL 不可能なすべての列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有効な JSON 形式で解析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、余分な二重引用符、引用符、エスケープ文字などが含まれていないことを確認します。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ために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従っ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、「制限事項」) の「スノーパイプストリーミング」の制限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 の送信先でテーブル定義またはロールのアクセス許可が変更された場合、Firehose がそ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。このために配信エラーが表示される場合は、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、次の手順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 内のホストから行われている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、「 での Firehose の使用 AWS PrivateLink」を参照してください。
- トラフィックが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 エンドポイントは、Firehose エンドポイントとクォータ にあります。
  - traceroute や などのツールを使用してtcping、ネットワーク設定が正しいかどうかを確認します。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:

    ```
    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:

    ```
    curl firehose.us-east-2.amazonaws.com
    ```

  上記のコマンドが成功した場合は、API から返されたレスポンスがあるかどうかを API でもう一度試してください。

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、一般的な HTTP エンドポイントの送信先と、Datadog、Dynatrace、LogicMonitorMongoDBの一般的なトラブルシューティング手順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た IAM ロールが、データ変換 (データ変換が有効になっている場合) のために S3 バックアップバケットと Lambda 関数にアクセスできることを確認します。また、IAM ロールが CloudWatch ロググループとログストリームにアクセスしてエラーログをチェックしていることを

確認します。詳細については、「HTTP エンドポイントの送信先 へのアクセスを Firehose に付与する」を参照してください。

> **Note**
>
> このセクションの情報は、Splunk、 OpenSearch Service、S3、Redshift の送信先には適用されません。

CloudWatch ログ

CloudWatch Firehose の口グ記録を有効にすることを強くお勧めします。口グは、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 ログに記録します。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 の設定ミスが原因で不正なリクエストを送信していることを示します。送信先について、正しい url、一般的な属性、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あることを確認します 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 ストリーム用に設定したアクセスキーが正しくないか、欠落していることを示します。
- 403: Firehose ストリーム用に設定したアクセスキーに、設定されたエンドポイントにデータを配信するアクセス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が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 が送信先が処理できるよりも高いレート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"
}
```

無効なレスポンス例外は、Amazon Data Firehose がエンドポイントの送信先から無効なレスポンスを受信したことを示します。レスポンスはレスポンス仕様に準拠している必要があります。そうしないと、Amazon Data Firehose は配信試行が失敗と見なし、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、レスポンスのステータスが 200 であっても、レスポンス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、応答仕様に従って、データが正常に配信されるようにします。

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。
- 欠落 RequestId: レスポンスに requestId が含まれていないことを示します。
- RequestId が と一致しない: レスポンスの requestId が送信 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 リクエストおよびレスポンスの仕様 に従った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送信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、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: HttpEndpointRequestTimeout- エンドポイントが応答するのに 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 のタイプミス、エンドポイントが Amazon Data Firehose 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ることが原因である可能性があります。
- エラーコード: HttpEndpointConnectionReset- 接続は行われたが、エンドポイントによってリセット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 を使用する 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であ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であ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。
- ソース設定のロールに、クラスターのトピックからデータを取り込むアクセス許可があ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が使用するポートで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。トピックが削除されて再作成された場合は、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 BytesPerSecondLimit ごとの が超過した場合、または通常の配信フローが遅くなったり停止したりすると 1 になります。

- ヒットした場合は BytesPerSecondLimit 、 DataReadFromSource.Bytes メトリクスをチェックし、制限の引き上げをリクエストしてください。
- CloudWatch ログ、送信先メトリクス、データ変換メトリクス、フォーマット変換メトリクスをチェックして、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「Amazon MSK の使用開始」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。Amazon EC2
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. を Firehose で使用したロール ARN awsRoleArnに置き換えSourceConfiguration、証明書の場所を確認します。 AWS クライアントユーザーがロールを引き受けることを許可しますawsRoleArn。 AWS クライアントユーザーは、ここで指定したロールを引き受けようとします。

```
[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. 次の Kafka コマンドを実行して、トピックを一覧表示します。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
```

リクエストが成功すると、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、指定したポートで指定したブートストラップサーバーにアクセスできることを確認します。これを行うには、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。つまり、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
```

前のステップに示すように、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 接続がまだ失敗している場合は、 AWS サポートにお問い合わせください。

データ鮮度メトリクスの増加または出力なし

データの鮮度は、Firehose ストリーム内のデータの現在の状態を示す尺度です。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 がデータを取り込んだ時点から現在までの間に測定されます。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。

data-freshness メトリクスが出力されていない場合、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。Firehose でトラフィックが多いために一時的なエラーが発生した場合、配信が遅れる可能性があります。これは、Amazon S3 以外の送信先で発生する可能性があります (サービス、Amazon Redshift、または Splunk で OpenSearch発生する可能性があります) 。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 で高いレイテンシーが発生すると、データ配信が遅れる可能性があります。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、Firehose ストリームのデータ取り込みレートよりも低いデータ配信レートが発生する可能性があります。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、「」を参照してくださいthe section called “ CloudWatch ログによるモニタリング”。
- Firehose ストリームのデータソースが Kinesis データストリームの場合、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` が高い場合は、 で設定されたロール に アクセス`kinesis:listshards`許可を追加することをお勧めしますKinesisStreamSourceConfiguration。
- 配信先のバッファリングヒントが低い。これにより、Firehose が送信先に対して行う必要がある往復回数が増加し、配信が遅れる可能性があります。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、「」を参照してくださいBufferingHints。
- 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、「」を参照してくださいthe section called “ CloudWatch ログによるモニタリング”。
- 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、 サポートに連絡して AWS 、Firehose が Splunk クラスターとの通信に使用しているチャネル数の増加をリクエストしてください。

レコード形式の Apache Parquet への変換が失敗する

これは、 `Set`タイプを含む DynamoDB データを取得し、Lambda 経由で Firehose ストリームにストリーミングし、 を使用してレコード形式を Apache Parquet AWS Glue Data Catalog に変換した場合に発生します。

AWS Glue クローラーが DynamoDB セットのデータ型 (`StringSet`、`NumberSet`、および `BinarySet`) にインデックスを作成すると、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`SET<BINARY>`。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 Management Console 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 フィールドに、シナリオに適したタイプのデータに応じてarray<binary>、、array<string>array<int>、または を入力します。
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。

Amazon Data Firehose クォータ

Amazon Data Firehose には次のクォータがあります。

- 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のクォータは、パーティションあたり 10 MB/秒の読み取りスループットと最大レコードサイズ 10MB です。サービスクォータの引き上げを使用して、パーティションあたりの読み取りスループットのデフォルトクォータである 10 MB/秒の引き上げをリクエストできます。
- 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 6Mb、Lambda が無効になっている場合は最大レコードサイズが 10Mb 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 は受信レコードを 10 MB に制限します。Amazon Data Firehose が 10MB を超えるレコードサイズを Amazon MSK から受け取った場合、Amazon Data Firehose はこのレコードを S3 エラーバケットに配信し、Cloudwatch メトリクスをアカウントに出力します。 AWS Lambda の制限の詳細については、https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html を参照してください。
- Firehose ストリームの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 ストリーム用に作成できるアクティブなパーティションは 500 個というデフォルトのクォータがあります。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。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。Amazon Data Firehose の制限フォームを使用して、特定の Firehose ストリームごとに最大 5000 個のアクティブなパーティションのクォータの引き上げをリクエストできます。さらに多くの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し、アクティブなパーティションをそれらに分散できます。
- Firehose ストリームの動的パーティショニングが有効になっている場合、アクティブな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。
- 各アカウントには、リージョンあたりの Firehose ストリーム数に対して次のクォータがあります。
  - 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 (アラブ首長国連邦)、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): 100 Firehose ストリーム
- この数を超えると、 を呼び出すと`LimitExceededException`例外CreateDeliveryStream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。Service Quotas がお客様のリージョンで利用できない場合は、Amazon Data Firehose の制限フォームを使用して引き上げをリクエストできます。

- 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および PutRecordBatchリクエストに対して次の組み合わせ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。
  - 米国東部 (オハイオ) の場合、 米国西部 (北カリフォルニア)、 AWS GovCloud (米国東部)、 AWS GovCloud (米国西部)、 アジアパシフィック (香港)、 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)、 中国 (北京)、 中国 (寧夏)、 アジアパシフィック (シドニー)、 アジアパシフィック (東京)、 カナダ (中部)、 カナダ西部 (カルガリー)、 欧州 (フランクフルト)、 欧州 (ロンドン)、 欧州 (パリ)、 欧州 (ストックホルム)、 中東 (バーレーン)、 南米 (サンパウロ)、 アフリカ (ケープタウン)、 および欧州 (ミラノ): 100、000 レコード/秒、 1、000 リクエスト/秒、 および 1 MiB/秒。

  クォータの引き上げをリクエストするには、Amazon Data Firehose の制限フォーム を使用します。3 つのクォータは比例してスケールされます。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。

> **Important**
>
> 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。

Important

データレコードが小さくなると、コストが高くなる可能性があることに注意してください。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に、最も近い 5KB (5,120 バイト) に切り上げられた各レコードのサイズを掛けたものです。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 計算ツール」の「Amazon Data Firehose」を参照してください。

Note

Kinesis Data Streams がデータソースとして設定されている場合、このクォータは適用されません。Amazon Data Firehose は制限なしでスケールアップおよびスケールダウンします。

- 各 Firehose ストリームは、配信先が利用できない場合、およびソースが の場合に、最大 24 時間データレコードを保存します DirectPut。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 エンコードの前に 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 オペレーションには、呼び出しごとに最大 500 レコード、または呼び出しごとに 4 MiB のいずれか小さい方かかる場合があります。このクォータは変更できません。
- 次のオペレーションは 1 秒あたり最大 5 つ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`、`ListDelivery`
- 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、パブリックにアクセス可能な Amazon Redshift クラスターのみがサポートされます。

- Amazon Redshift および OpenSearch サービス配信の再試行期間の範囲は 0 秒から 7,200 秒です。
- Firehose は、Elasticsearch バージョン 1.5、2.3、5.1、5.3、5.5、5.6、およびすべての 6.* および 7.* バージョンと Amazon OpenSearch Service 2.x から 2.11 までをサポートしています。
- 送信先が Amazon S3、Amazon Redshift、または OpenSearch Service の場合、Amazon Data Firehose はシャードごとに最大 5 つの未処理の Lambda 呼び出しを許可します。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。
- CUSTOMER_MANAGED_CMK タイプの CMK を使用して、最大 500 の Firehose ストリームを暗号化できます。

付録 - HTTP エンドポイント配信リクエストとレスポンスの仕様

Amazon Data Firehose がカスタム HTTP エンドポイントにデータを正常に配信するには、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 リクエストのフォーマット仕様と、Amazon Data Firehose サービスが想定する HTTP レスポンスのフォーマット仕様について説明します。HTTP エンドポイントは、Amazon Data Firehose がリクエストをタイムアウトするまで、3 分以内にリクエストに応答する必要があります。Amazon Data Firehose は、適切な形式に従っていないレスポンスを配信エラーとして扱います。

トピック

- リクエストの形式
- レスポンスの形式
- 例

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 はそれらを変更せずに設定どおりに送信します。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:

Firehose ストリームの ARN は ASCII 文字列形式で表現されています。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、認証トークン) 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は 4096 バイトに制限されています。Amazon Data Firehose は、このキーをいかなる方法でも解釈しようとはしません。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。

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またはエンコードせず、内容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 ストリームを作成するときに直接設定されます。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

応答が以下の要件を満たしていない場合、Firehose サーバーはその応答を本文のない 500 ステータスコードであるかのように扱います。

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) には従いません。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

どのようなエラーでも、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バッチのレコードの配信を再試行します。再試行は、ジッター係数 (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、以降の再試行はそれぞれ、ジッターを加えた式 (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。たとえば、「n」回目のリトライでは、バックオフ時間は = 最大 (120, 2^n) * ランダム (0.85, 1.15) です。

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。指数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、ジッター率については、 AWS Firehose のドキュメントを参照してください。

その後再試行するたびに、Firehose ストリームの更新された構成に基づいて、レコードの配信先のアクセスキーや宛先が変更される可能性があります。Amazon Data Firehose サービスは、ベストエフォート方式で再試行しても同じリクエスト ID を使用します。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。最大許容時間 (Firehose ストリーム設定に基づく) を過ぎても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"
        }
      ]
    }
   }
  ]
}
```

ドキュメント履歴

次の表に、Amazon Data Firehose ドキュメントの重要な変更点を示します。

変更	説明	変更日
新しいリージョンの送信先としての 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) 、およびアジアパシフィック (シドニー) の目的地として利用可能になりました。「the section called “Snowflake の送信先設定を構成する”」を参照してください。	2024 年 6 月 19 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 “で認証する AWS Secrets Manager”」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	Dynatrace にログとイベントを送信して、さらに分析できるようになりました。「the section called “Dynatrace の送信先設定を構成する”」を参照してください。	2024 年 4 月 18 日
Snowflake を送信先とする一般提供 (GA) リリース	Snowflake が送信先として一般利用可能になりました。the section called “Snowflake の送信先設定を構成する” を参照してください。	2024 年 4 月 17 日
Amazon Kinesis Data Firehose が Amazon Data Firehose として知られるようになりました	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	2024 年 2 月 9 日
Snowflake を送信先として追加 (パブリックプレビュー)	Snowflake を送信先として Firehose ストリームを作成できます。the section called “Snowflake の送信先設定を構成する” を参照してください。	2024 年 1 月 19 日

変更	説明	変更日
CloudWatch ログの自動解凍を追加	新規または既存のストリームで解凍を有効にして、解凍された CloudWatch ログデータを Firehose の送信先に送信できます。the section called “ CloudWatch ログを使用した書き込み” 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 を送信先として Firehose ストリームを作成できます。the section called “Splunk Observability Cloud の送信先設定を構成する” 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 を使用した書き込み” を参照してください。	2023 年 9 月 26 日
OpenSearch サービス送信先の DocumentID タイプのサポートを追加	OpenSearch サービスが Firehose ストリームの送信先である場合、DocumentID タイプはドキュメント ID を設定する方法を示します。サポートされているメソッドは、Firehose が生成するドキュメント ID と OpenSearch サービスが生成するドキュメント ID です。the section called “送信先設定を構成する” を参照してください。	2023 年 5 月 10 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 での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。動的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。 カスタム Amazon S3 プレフィックス を参照してください。	2018 年 12 月 20 日

変更	説明	変更日
新しい Amazon Data Firehose チュートリアルを追加	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。チュートリアル: 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 リージョンを追加	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 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 リージョンを追加	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 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 ストリームのレコードの潜在的なソースとして Kinesis Streams を追加しました。詳細については、「送信元と送信先を設定する」を参照してください。	2017 年 8 月 18 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、「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	データ配信前にデータを変換するように Amazon Data Firehose を設定できます。詳細については、「Amazon Data Firehose データ変換」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	失敗した場合は、Amazon Redshift クラスターに対して COPY コマンドを再試行するように Amazon Data Firehose を設定できます。詳細については、Firehose ストリームを作成する、Amazon Data Firehose データ配信を理解する、および Amazon Data Firehose クォータを参照してください。	2016 年 5 月 18 日

変更	説明	変更日
新しい Amazon Data Firehose 送信先、Amazon OpenSearch Service	Amazon OpenSearch Service を送信先として Firehose ストリームを作成できます。詳細については、Firehose ストリームを作成する、Amazon Data Firehose データ配信を理解する、および Amazon Data Firehose にパブリック OpenSearch サービスの送信先へのアクセス権を付与するを参照してください。	2016 年 4 月 19 日
新しい拡張 CloudWatch メトリクスとトラブルシューティング機能	「Amazon Data Firehose のモニタリング」および「Amazon Data Firehose のトラブルシューティング」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	更新済み Kinesis Agent を使用した Amazon Data Firehose への書き込み。	2016 年 4 月 11 日
新しい Kinesis エージェント	Kinesis Agent を使用した Amazon Data Firehose への書き込みが追加されました。	2015 年 10 月 2 日
初回リリース	Amazon Data Firehose デベロッパーガイド の初回リリース。	2015 年 10 月 4 日

AWS 用語集

AWS の最新の用語については、「AWS の用語集リファレンス」の「AWS 用語集」を参照してください。